

# When Homomorphic Cryptosystem Meets Differential Privacy: Training Machine Learning Classifier with Privacy Protection

Xiangyun Tang

Beijing Institute of Technology  
xiangyunt@bit.edu.cn

Liehuang Zhu

Beijing Institute of Technology  
liehuangz@bit.edu.cn

Meng Shen

Beijing Institute of Technology  
shenmeng@bit.edu.cn

Xiaojiang Du

Temple University  
dxj@ieee.org

**Abstract**—Machine learning (ML) classifiers are invaluable building blocks that have been used in many fields. High quality training dataset collected from multiple data providers is essential to train accurate classifiers. However, it raises concern about data privacy due to potential leakage of sensitive information in training dataset. Existing studies have proposed many solutions to privacy-preserving training of ML classifiers, but it remains a challenging task to strike a balance among accuracy, computational efficiency, and security.

In this paper, we propose **Heda**, an efficient privacy-preserving scheme for training ML classifiers. By combining homomorphic cryptosystem (HC) with differential privacy (DP), **Heda** obtains the tradeoffs between efficiency and accuracy, and enables flexible switch among different tradeoffs by parameter tuning. In order to make such combination efficient and feasible, we present novel designs based on both HC and DP: A library of building blocks based on partially HC are proposed to construct complex training algorithms without introducing a trusted third-party or computational relaxation; A set of theoretical methods are proposed to determine appropriate privacy budget and to reduce sensitivity. Security analysis demonstrates that our solution can construct complex ML training algorithm securely. Extensive experimental results show the effectiveness and efficiency of the proposed scheme.

## I. INTRODUCTION

Machine learning (ML) classifiers are widely used in many fields, such as spam detection, image classification, and natural language processing. Many studies have modeled user data and obtained satisfactory classifiers that meet accuracy requirements [19], [36]. The accuracy of a classifier obtained from supervised learning is closely related to the quality of the training dataset, in addition to well-designed ML algorithms. An experimental study with a dataset of 300 million images at Google [29] demonstrates that the performance of classifiers increases as the order of magnitude of training data grows. However, training dataset is usually held by multiple data providers and may contain sensitive information, so it is important to protect data privacy in training of ML classifiers.

Consider the typical training process depicted in Figure 1. There are multiple data providers and a single data user. Upon receiving the request of dataset from the data user, each data provider applies privacy-preserving mechanisms (e.g., encryption or perturbation) to its own dataset. Then, the data user trains an ML classifier based on the dataset gathered from multiple data providers. During this process, each data provider

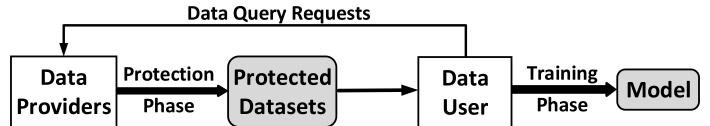


Fig. 1. Application Scenario. Each non-shaded rectangle represents a type of role. Each shaded box indicates private data that should be accessible to only one party: a protected dataset to a data provider, and the model to the data user. Each solid arrow indicates an algorithm or a process.

cannot know the classifier, while the data user cannot learn any sensitive information of the shared data.

More specifically, consider the following example of an advertisement recommendation task: In order to attract more consumers, a company wants to build a classifier to discern the most appropriate time for advertising. The training dataset used for constructing the classifier is extracted from the consumer purchase behavior data recorded by several online shopping sites. The consumer data is confidential because it contains sensitive information about consumers. Online shopping sites agree to share their data with companies, but refuse to reveal any privacy of the consumers. The company wants to construct a classifier based on the consumer data, but is unwilling to reveal the classifier to online shopping sites. Ideally, online shopping sites and the company run a privacy-preserving training algorithm, at the end of which the company learns the classifier parameters, and neither party learns anything else about the other party's input.

In general, supervised ML classifiers consist of two phases: the training phase and the classification phase. A series of secure schemes for the classification phase have been proposed [6], [10]. In this paper, we focus on the training phase, that is, privacy-preserving training of ML classifiers<sup>1</sup>.

Existing solutions to training ML classifier securely roughly depend on three types of techniques, namely secure multi-party computing (SMC), homomorphic cryptosystem (HC), and differential privacy (DP). SMC can construct many classifiers theoretically. But it relies on a trusted third-party for providing random number, and results in a large number of interactions and redundant computations for protecting data

<sup>1</sup>In this paper, ML classifiers and ML models are used interchangeably.

privacy [15], [23]. HC<sup>2</sup> allows the operation on ciphertext to be mapped to the corresponding plaintext. The secure training solutions based on HC [11], [17] may suffer from low efficiency. In addition, since partially HC only enables a single type of operation (e.g., addition or multiplication), HC-based solutions for training complex ML classifiers usually introduce a trusted third-party (e.g., the authorization server [10], [17]) or use an approximate equation that simplifies the complex iteration formula [2], [3]. DP can resist the attacker with the largest background knowledge [5], which ensures the security of the published data by adding noises. The computational efficiency of operations on perturbed data is significantly higher than those on ciphertext [5], [9]. Nevertheless, the quality of the published dataset is reduced due to the introduction of noises, and thereby the accuracy of the resulting classifiers is decreased inevitably.

As discussed above, HC is low efficient due to ciphertext-based computation, but can obtain a classifier with lossless accuracy. DP has high computational efficiency but leads to an inevitable loss of accuracy. Intuitively, we can take the strengths of HC and DP by adopting them simultaneously.

However, HC and DP are completely different systems: one for data encryption, and the other for data perturbation. It is a challenging task to combine them together. In particular, partially HC only supports one type of operation, which sets a barrier to the training of ML classifiers with complex operations such as power function, division, and square root. Furthermore, the noises added to sensitive data in DP determines the accuracy of classifiers and privacy of published data. The third challenge is how to archive high accuracy while ensuring privacy in DP.

In this paper, we propose Heda, an efficient privacy-preserving scheme for training ML classifiers. By combining HC with DP, Heda obtains the tradeoffs between efficiency and accuracy and enables flexible switch among different tradeoffs by parameter tuning. Security analysis demonstrates that our building blocks can construct complex ML training algorithms. Extensive experimental results show the effectiveness and efficiency of the proposed scheme.

We address the above challenges by developing a set of key techniques.

We make an observation that different features<sup>3</sup> in a dataset usually contribute differently to the accuracy of classifiers [21], [31]. For the features with high contributions, we apply HC to these features such that the model parameters obtained from them are as accurate as possible. We apply DP to the rest features to improve the computational efficiency. The contribution of each feature in training ML classifiers can be evaluated using readily available techniques [21].

To address the second challenge, we employ two homomorphic encryption primitives: a multiplicative homomorphic encryption RSA and an additively homomorphic encryption Paillier. We carefully design a library of building blocks

---

<sup>2</sup>In this paper, we only consider partially HC due to the computational inefficiency of fully HC.

<sup>3</sup>Without loss of generality, when facing the same training task, we assume that all the dataset has been locally preprocessed and represented with the same feature vectors [17], [32].

supporting for complex operations such as power function and dot product, which can handle ML classifiers with complex training operations. We take Logical Regression (LR) as an example to illustrate the power of our building blocks. The sigmoid function in the iterative formula of LR makes it difficult to construct a secure LR training algorithm based on HC. It is the first time that constructing a secure LR training algorithm by HC without an authorization server or any approximation. (Section VI)

In the face of the third challenge, we develop a formal method to determine the reasonable privacy budget, and we reduce the sensitivity by using insensitive microaggregation. We reduce the added noise and improve the usability of the noise dataset published by DP reasonably. (Section V)

To the best of our knowledge, it is the first study that achieves privacy-preserving training of ML classifiers by jointly applying HC and DP in an individual scheme. The rest of our paper is organized as follows. Section II describes related work, and Section III provides the background. Section IV describes the problem statement. Section V and Section VI present the special designs with DP and HC, respectively. Section VII describes the construction of Heda in detail. The security analysis is exhibited in Section VIII, and the evaluation results are provided in Section IX. Section X concludes this paper.

## II. RELATED WORK

Since our work is related to secure ML classifiers algorithms which can be broadly divided into two categories: privacy-preserving classification and privacy-preserving training. We give the literature review of both subjects. Because Heda jointly applying HC and DP, and there are some studies about combining HC with DP but not about secure classifiers training, we present a discussion about these works. We give an analysis of our novelty at last.

### A. Privacy-Preserving ML Classification

A series of techniques have been developed for privacy-preserving ML Classification. Wang et al. [33] proposed an encrypted image classification algorithm based on multi-layer extreme learning machine that is able to directly classify encrypted images without decryption. They assumed the classifier had been trained, and the classifier not confidential. Grapell et al. [18] constructed several secure classification algorithms by HC, while the parameters of trained classifiers are not confidential for classifiers users. Zhu et al. [37] proposed a secure nonlinear kernel SVM classification algorithm, which is able to keep users' health information and healthcare provider's prediction model confidential.

Several works have designed general (non-application specific) privacy-preserving protocols and explored a set of common classifiers by HC [6], [10]. Usually, classification algorithms are simpler than training algorithms, building blocks that are able to build classification algorithms can be powerless for complex training algorithms.

## B. Privacy-Preserving ML Classifier Training

Three techniques have been applied to privacy-preserving ML classifier training, they are SMC, HC, and DP. Constructing secure classifier training algorithms based on SMC relies on a large number of interactions and many redundant calculations for protect privacy, and it generally needs to introduce authoritative third parties to provide random number distribution services as well. In addition, SMC protocols for generic functions existing in practice rely on heavy cryptographic machinery. Applying them directly to model training algorithms would be inefficient [4], [23], [35].

HC is able to compute using only encrypted values. Employing HC, many secure algorithms have been developed for different specialized ML training algorithms such as Support Vector Machine (SVM) [17], [27], LR [11], [18], decision trees [30] and Naive Bayes [24]. However, partially HC only enables a single type of operation (e.g., addition or multiplication). In order to construct complex training algorithms, HC-based schemes usually need to rely on trusted third parties such as the Authorization Server [10], [17], or use an approximate equation to simplify the original complex iteration formula into a simple one [2], [3]. Gonzlez et al. [17] developed secure addition protocol and secure subtractions protocol to construct the secure SVM training algorithm by employing Paillier, while some operations that are not supported by Paillier have to be implemented with the assistance of the Authorization Server in their scheme. Secure LR training algorithms existing implemented by HC are actually the linear regression [11], [18], because the sigmoid function contains power function and division operation, which makes LR training algorithms harder to be implemented by HC than other ML training algorithms. Several works solved the sigmoid function by an approximate equation<sup>4</sup> [2], [3].

Many secure ML classifier training algorithms have been explored in DP area such as decision tree [5], LR [9] and deep learning [1]. Blum et al. [5] proposed the first DP based decision tree training algorithm on the SuLQ platform. Abadi et al. [1] applied DP objective perturbation in a deep learning algorithm, where the noise was added to every step of the stochastic gradient descent. Due to the introduction of noise under DP mechanisms, the quality of the datasets were reduced, and the accuracy of these trained models was decreased inevitably. So the essential challenge for DP based frameworks is guaranteeing the accuracy by reducing the added noise, especially for the operation has high sensitivities [38]. According to the Laplace mechanism (cf. Definition 4), privacy budget  $\epsilon$  and the sensitivity  $\Delta f$  are two important factors affecting noise addition. In many papers, the value of  $\epsilon$  is merely chosen arbitrarily or assumed to be given [1], [5]. Lee et al. [22] explored the selection rules of  $\epsilon$ , but they have not given a way to determine the value of the privacy budget. Soria et al. [28] proposed a insensitive microaggregation-based DP mechanism, they found the amount of noise required to fulfill  $\epsilon$ -DP can be reduced in insensitive microaggregation. Heda develops the insensitive microaggregation-based DP mechanism and decreases the amount of noise required to fulfill  $\epsilon$ -DP again.

---

<sup>4</sup> $\log\left(\frac{1}{1+\exp(u)}\right) \approx \sum_{j=0}^k a_j \cdot u$

## C. Homomorphic Cryptosystem Combine Differential Privacy

Several works have studied combining HC with DP to solve a special security problem. Pathak et al. [26] proposed a scheme for composing a DP aggregate classifier using classifiers trained locally by separate mutually untrusting parties, where HC was used for composing the trained classifiers. Yilmaz et al. [34] proposed a scheme for optimal location selection utilizing HC as the building block and employing DP to formalize privacy in statistical databases. Aono et al. [3] constructed a secure LR training algorithm via HC and achieved DP to protect the model parameters. These works general constructed a secure algorithm via HC and used DP to protect the algorithm results. As we have discussed above, constructing a secure algorithm via HC is low efficient, and secure algorithm based on DP has inevitable loss in accuracy. We aim of constructing a secure classifier training algorithm jointly applying HC and DP in an individual scheme to obtain a tradeoff between efficiency and accuracy.

## D. Novelty of Our Construction

Secure training algorithms based on HC have to handle datasets in ciphertext case, where the time consumption is considerable, while the accuracy is able to be guaranteed. Noise datasets published by DP mechanism are in plaintext case, it is efficient to train a model in plaintext case, while using the noise dataset may lead to a low accuracy. HC and DP have drawbacks as well as merits.

Heda takes the strengths of HC and DP to get a high-efficiency and high-accuracy privacy-preserving ML classifier training algorithm. Heda is the first to combine these two techniques and construct a privacy-preserving ML classifier training algorithm in a multi-party setting, where feature evaluation techniques are employed to give the way of combination. By combining HC with DP, Heda obtains the tradeoffs between efficiency and accuracy, and enables flexible switch among different tradeoffs by parameter tuning. What's more, we develop a library of building blocks by HC that is able to construct complex training algorithms, and by using our building blocks this is the first time that solving the sigmoid function in secure LR training based on HC without any approximate equation. We develop the works of Lee et al. [22] and Soria et al. [28] giving a formula to determine the appropriate privacy budget and another lower sensitive solution.

## III. PRILIMINARIES

### A. Notation

A dataset  $D$  is an unordered set of  $n$  records with the size of  $|D|$ .  $x_i \in \mathbb{R}^d$ ,  $x_i = (x_{i1}, x_{i2}, \dots, x_{id})$  is the  $i$ -th record in dataset  $D$ , and  $y_i$  is a class label correspond to  $x_i$ .  $X = (x_1, x_2, \dots, x_m)$ ,  $Y = (y_1, y_2, \dots, y_m)$ .  $\beta$ ,  $\omega$ , and  $b$  are the relevant parameters of the model trained by a ML algorithm. The subset  $A_i$  corresponding to the  $i$ -th attribute in  $D$ .  $S$  is the scores assign to the features.

Cryptosystems define a plaintext space  $\mathbb{M}$ , and a ciphertext space  $\mathbb{C}$ . In Heda, we employ two public-key cryptosystems, Paillier and RSA.  $[[m]]$  and  $||m||$  are represented as the ciphertext of  $m$  under Paillier or RSA respectively.

DP is generally achieved by a randomized algorithm  $\mathcal{M}$ .

TABLE I. NOTATIONS

Notations	Explanation	Notations	Explanation
$\mathbb{R}$	Set of real numbers	$\mathbb{R}^d$	d-dimension $\mathbb{R}$
$D$	Dataset	$ D $	The size of D
$m$	Size of dataset	$D'$	Neighbour dataset
$X$	The record set in D	$Y$	The label set in D
$x_i$	i-th Record in dataset	$y_i$	Class label
$o, \sigma$	Functional operation	$d$	Dataset dimension
$\mathbb{M}$	Plaintext space	$\beta, b$	Parameters of models
$\mathcal{M}$	Mechanism	$\mathbb{C}$	Ciphertext space
$k$	The cluster size	$N$	n-bit Primes
$f$	Query	$\epsilon$	Privacy budget
$\gamma$	Noise	$\Delta f$	Sensitivity
$[[m]]$	Ciphertext under Paillier	$  m  $	Ciphertext under RSA
$\iota$	The number of encrypted features in D	$S$	The scores of features
$[m]$	The encryption of m under a certain cryptosystems	$A_i$	The subset of i-th attribute in D

$\epsilon$  is the privacy budget in a DP mechanism. A query  $f$  maps dataset  $D$  to an abstract range  $f : D \rightarrow R$ . The maximal difference in the results of query  $f$  is defined as the sensitivity  $\Delta f$ .  $D'$  is a neighboring dataset of  $D$ .

Table I summarizes the notations used in the following sections.

### B. Homomorphic Cryptosystem

Cryptosystems are composed of three algorithms: key generation (Gen) to generate the key, encryption (Enc) encrypting secret message and decryption (Dec) for decrypting ciphertext. Public-key cryptosystems employ a pair of keys (PK, SK), the public key (PK, the encryption key) and the private key (SK, the decryption key). Some cryptosystems are gifted with a property of homomorphic that makes cryptosystems perform a set of operations on encrypted data without knowledge of the decryption key. Formalized definition is given in Definition 1.

**Definition 1:** (homomorphic) [20]. A public-key encryption scheme (Gen, Enc, Dec) is homomorphic if for all  $n$  and all (PK, SK) output by Gen( $1^n$ ), it is possible to define groups  $\mathbb{M}, \mathbb{C}$  (depending on PK only) such that:

(i) The message space is  $\mathbb{M}$ , and all ciphertexts output by  $Enc_{pk}$  are elements of  $\mathbb{C}$ .

(ii) For any  $m_1, m_2 \in \mathbb{M}$ , any  $c_1$  output by  $Enc_{pk}(m_1)$ , and any  $c_2$  output by  $Enc_{pk}(m_2)$ , it holds that  $Dec_{sk}(o(c_1, c_2)) = \sigma(m_1, m_1)$ .

In Heda, we employ two public-key cryptosystems, Paillier and RSA. Paillier possesses additively homomorphic property, and RSA possesses multiplicative. For more details about Paillier or RSA, we refer the reader to [20].

**Paillier.** The security of Paillier is based on an assumption related to the hardness of factoring. Assuming a pair of ciphertext  $(c_1, c_2)$  is  $(m_1, m_2)$  under the same Paillier encryption scheme where the public key is  $N$ , we have:  $c_1 \times c_2 = [(1 + N)^{m_1+m_2} r^N \text{mod } N^2]$ , where  $(m_1 + m_2) < N$ . The additively homomorphic property in Paillier can be described as  $[[m_1 + m_2]] = [[m_1]] \times [[m_2]] (\text{mod } N^2)$ .

**RSA.** Based on the definition of a one-way trapdoor function, RSA gives the actual implementation of the first public key cryptosystem. RSA is a multiplicative HC, because that:  $Enc_{RSA}(m_1) \times Enc_{RSA}(m_2) = |(m_1 \times m_2)| \text{mod } N$ , where  $(m_1 \times m_2) < N$ . The multiplicative homomorphic property in RSA can be described as  $||m_1 \times m_2|| = ||m_1|| \times ||m_2|| (\text{mod } N)$ .

### C. Differential Privacy

**Definition 2:** (Neighbor Dataset) [5]. The datasets  $D$  and  $D'$  have the same attribute structure, and the symmetry difference between them is denoted as  $|D \Delta D'|$ . We call  $D$  and  $D'$  neighbour datasets if  $|D \Delta D'| = 1$ .

**Definition 3:** ( $\epsilon$ -Differential Privacy) [5]. A randomized mechanism  $\mathcal{M}$  gives  $\epsilon$ -DP for every set of outputs  $\mathcal{R}$ , and for any neighbor dataset of  $D$  and  $D'$ , if  $\mathcal{M}$  satisfies:  $\Pr[\mathcal{M}(D) \in \mathcal{R}] \leq \exp(\epsilon) \times \Pr[\mathcal{M}(D') \in \mathcal{R}]$ .

A smaller  $\epsilon$  represents a stronger privacy level [38]. While  $\epsilon$  is equal to 0, for any neighbour dataset, the randomized mechanism  $\mathcal{M}$  will output two identical results of the same probability distribution which cannot reflect any useful information. If  $\epsilon$  is selected as a too large value in a DP mechanism, it does not mean that privacy is actually enforced by the mechanism. A composition theorem for  $\epsilon$  named parallel composition (Theorem 1) is widely used.

**Theorem 1:** (Parallel Composition) [25]. Suppose we have a set of privacy mechanisms  $\mathcal{M} = \{\mathcal{M}_1, \mathcal{M}_2, \dots, \mathcal{M}_m\}$ . If each  $\mathcal{M}_i$  provides a  $\epsilon_i$ -DP guaranteed on a disjointed subset of the entire dataset,  $\mathcal{M}$  will provide  $(\max\{\epsilon_1, \epsilon_2, \dots, \epsilon_m\})$ -DP.

Laplace Mechanism (Definition 4) is the basic DP implementation mechanism and is suitable for the numerical data, which adds independent noise following the Laplace distribution to the true answer.

**Definition 4:** (Laplace mechanism) [14]. For a dataset  $D$  and a query function  $f : D \rightarrow R$  with sensitive  $\Delta f$ . Privacy mechanisms  $\mathcal{M}(D) = f(D) + \gamma$  providers  $\epsilon$ -DP, where  $\gamma \sim \text{lap}\left(\frac{\Delta f}{\epsilon}\right)$  represents the noise sampled from a Laplace distribution with a scaling of  $\left(\frac{\Delta f}{\epsilon}\right)$ .

**Definition 5:** (Sensitivity) [5]. For a query  $f : D \rightarrow R$ , and a pair of neighbor datasets  $(D, D')$ , the sensitivity of  $f$  is defined as:  $\Delta f = \max_{D, D'} \|f(D) - f(D')\|_1$ . Sensitivity  $\Delta f$  is only related to the type of query  $f$ . It considers the maximal difference between the query results.

## IV. PROBLEM STATEMENT

We are devoted to addressing the problem on the secure training of ML classifier using private protected data gathered from different data providers. In this section, we introduce the overview of the system model and the roles involved in Heda. Then, we formally define the threat model and the security goal.

### A. System Model

We target at the system application scenario which has been illustrated in Figure 1. There are  $n$  data providers  $\mathcal{P}$  and a data user  $\mathcal{U}$  in our model. Each  $\mathcal{P}$  holds their own sensitive dataset  $D_i$  and a pair of keys (PK, SK).  $\mathcal{P}$  protects their sensitive data by applying privacy-preserving mechanisms (e.g., DP mechanism and HC).  $\mathcal{U}$  holds his own keys (PK, SK). After obtaining the permission,  $\mathcal{U}$  requests the sensitive data from  $\mathcal{P}$ , and  $\mathcal{P}$  returns the protected data. By running a sequence of secure interactive protocols with several  $\mathcal{P}$ ,  $\mathcal{U}$  obtains the classifier parameters of being encrypted by  $\mathcal{U}$ 's keys.

As discussed in Section I and II, HC is able to construct accurate secure training algorithms, and DP mechanism providers high efficient secure training algorithms. However, it is low efficient that constructing a secure ML training algorithm by HC, and the model may poor in accuracy if the training data is under DP mechanism. We thereby desire to take the strengths of HC and DP, and feature evaluation techniques is used for providing a right combination method. We describe the overall idea of *Heda* as follows:

- 1)  $\mathcal{P}$  scores all features by feature evaluation techniques and divides the dataset into two parts according to the scores (see Section VII-A).
- 2)  $\mathcal{P}$  applies privacy-preserving mechanisms to the two parts respectively: the low scores part published by DP mechanism (see Section V); the high scores part encrypted by HC (see Section VI).
- 3) Upon receiving the query requests,  $\mathcal{P}$  sends the protected data to  $\mathcal{U}$ .
- 4)  $\mathcal{U}$  trains a ML classifier under these two protected sub-datasets (see Section VII-C).

### B. Threat Model

$\mathcal{U}$  interacts with several  $\mathcal{P}$  to obtain the protected data and performs training algorithms on the data. Each  $\mathcal{P}$  tries to learn as much other  $\mathcal{P}$ 's sensitive data and  $\mathcal{U}$ 's trained classifier as possible by honestly executing pre-defined protocols.  $\mathcal{U}$  follows the protocol honestly, but it tries to infer  $\mathcal{P}$ 's sensitive data as much as possible from the values he learns. As discussed above, we assume each participant is a passive (or honest-but-curious) adversary [16], that is, it does follow the protocols but tries to infer others' privacy as much as possible from the values they learn.

### C. Security Goal

In *Heda*, we allow any two or more parties conspire to steal the privacy of other participants. We make the following assumptions: Each participant as a honest-but-curious adversary performs protocol honestly but may have interest in the private information of other domains. Any two or more participants may collude with each other. As passive adversaries, they do follow the protocol but try to infer other's privacy as much as possible from the values they learn.

The aim of *Heda* is achieving keeping privacy of each participant and computing model parameters securely when facing honest-but-curious adversaries or any collusion. To be

specific, the privacy of  $\mathcal{U}$  is model parameters, and each  $\mathcal{P}$  is their sensitive data. We specify our security goals as follows:

- 1) When facing honest-but-curious adversaries,  $\mathcal{U}$  and each  $\mathcal{P}$ 's privacy are confidential.
- 2) when facing any two or more parties collude with each other,  $\mathcal{U}$  and each  $\mathcal{P}$ 's privacy are confidential.

## V. ACCURACY AND PRIVACY DESIGN WITH DIFFERENTIAL PRIVACY

DP ensures the security of the published data by adding noise. Insufficient noise leads to the security of the published data cannot be guaranteed, while excess noise causes the data unusable. Obviously, the key to using DP in the secure classifier training is to reduce the added noise while ensuring the security of the published data.

The two important parameters that determine the added noise are  $\epsilon$  and  $\Delta f$  (cf. Definition 4). A bigger  $\epsilon$  or a smaller  $\Delta f$  are able to reduce the added noise. However, if  $\epsilon$  is selected as a too large value, although the system has been built upon DP framework, it dose not mean that privacy is actually enforced by the system. Therefore,  $\epsilon$  must be combined with specific requirements to achieve the balance of security and usability of output results. On the other hand,  $\Delta f$  is only determined by the type of query function (cf. Definition 5).

In this section, we develop a formula for reasonably determining the appropriate  $\epsilon$  in DP mechanism, and we reduce the  $\Delta f$  by using insensitive microaggregation.

### A. Selection of Appropriate $\epsilon$

In many papers,  $\epsilon$  is chosen arbitrarily or assumed to be given, while decision on  $\epsilon$  should be made carefully with considerations of the domain and the acceptable ranges of risk of disclosure. Lee et al. [22] explored the rule of  $\epsilon$ , but they did not give a specific method for determining  $\epsilon$ . We give a method for determining  $\epsilon$ . It is worth noting that based on different criteria and backgrounds,  $\epsilon$  can have different values, and we are trying to give a general one.

We follow some notations of Lee et al. [22]: If an adversary knows all the background knowledge, he tries to guess which one is the different values between  $D'$  and  $D$ . Let  $\mathbb{W}$  denotes the set of all possible combinations  $\omega$  of  $D'$ ,  $\omega \in \mathbb{W}$ . For each possible  $\omega$ , the adversary maintains a set of tuples  $\langle \alpha, \mu \rangle$ . For a given query response,  $\alpha$  and  $\mu$  are the adversary's prior belief and posterior belief on  $D'$ , i.e.,  $\forall \omega \in \mathbb{W}, \alpha(\omega) = \frac{1}{m}$ . For each possible  $\omega$ , the adversary's posterior belief on  $\omega$  is defined as  $\mu(\omega) = P(D' = \omega | \gamma) = \frac{P(M(\omega) = \gamma)}{\sum_{\omega \in \mathbb{W}} P(M(\omega) = \gamma)}$ . Lee et al. [22] obtain the upper bound of  $\epsilon$  through a series of derivations as Formula 1 (cf. Section V in [22])

$$\epsilon \leq \frac{\Delta f}{\Delta v} \ln \left( \frac{(m-1)\rho}{1-\rho} \right) \quad (1)$$

where  $\Delta v = \max_{1 \leq i, j \leq n, i \neq j} |f(\omega_i) - f(\omega_j)|$ ,  $\rho$  is the probability that the adversary guessing success. Nevertheless, Lee et al. [22] did not give a method for setting  $\rho$ . We give a method for determining the upper bound of  $\rho$  (Proposition 1).

*Proposition 1 (the upper bound of  $\rho$  for  $D'$ ):* Let  $A_j$  is the subset of the  $j$ -th attribute in dataset  $D$ .  $Count_{max}$  is

---

**Algorithm 1** Generating Appropriate Value of  $\epsilon$ 


---

**Input :**  $D = \{(x_i, y_i)\}_{i=1}^m$ .  
**Output :** The appropriate  $\epsilon$  on dataset  $D$ .

- 1: **for**  $j = 1$  to  $d$  **do**
- 2:   **for**  $i = 1$  to  $m$  **do**
- 3:     Computing  $\Delta f$  and  $\frac{Count_{max}}{|A_j|}$  in  $A_j$ .
- 4:     Obtaining  $\epsilon_j$  by Formula 1.
- 5: **return**  $\epsilon = \{\epsilon_1, \epsilon_2, \dots, \epsilon_d\}$ .

---

the occurrences number of the record which has the highest frequency in  $A_j$ . Then  $\frac{Count_{max}}{|A_j|}$  is the upper bound of  $\rho$ .

*Proof 1 (Proof of Proposition 1):*  $\rho$  is the probability that the adversary successfully guesses which instance is the different one between  $D'$  and  $D$ . DP mechanism assumes that the adversary has a strong background knowledge, that is, he knows the value of each instance in  $D$ .  $x_{ij}$  is the highest frequency instance in  $A_j$ , so the adversary guesses  $x_{ij}$  will get the highest probability of success. After DP mechanism, the adversary's probability of success should not be greater than the highest probability of random guessing and success, so the upper bound of  $\rho$  is  $\frac{Count_{max}}{|A_j|}$ .  $\square$

The upper bound of  $\epsilon_i$  is obtained from each subset  $A_i$  by Formula 1, then the dataset  $D$  provides  $\max(\epsilon_i)$ -DP according to Theorem 1. Algorithm 1 details the steps for generating the appropriate  $\epsilon$  on dataset  $D$ .

### B. Reducing $\Delta f$ by In-sensitive Microaggregation

According to the Definition 4, the smaller the  $\Delta f$ , the less noise is added, and thereby the more usable the data is. In this subsection, we detail the solution of reducing the  $\Delta f$  in Heda. The amount of noise required to fulfill  $\epsilon$ -DP can be greatly reduced if the query is run on a insensitive microaggregation version of all attributes instead of running it on the raw input data [28].

*1) What is in-sensitive microaggregation:* Microaggregation is used to protect microdata releases and works by clustering groups of individuals and replacing them by the group centroid. DP makes no assumptions about the adversary's background knowledge. Microaggregation with DP can help increasing the utility of DP query outputs while making as few assumptions on the type of queries as microaggregation does [28]. However, if we modify one record in  $D$ , more than one clusters will differ from the original clusters generally. According to the Definition 3, we expect that we modify one record in  $D$ , each pair of corresponding clusters differs at most in single record. Microaggregation that satisfies this property is named in-sensitive microaggregation (IMA). Soria et al. [28] give a formal definition of IMA. Microaggregation is in-sensitive to input data if and only if the distance function  $Dist(x, y)$  is a fixed sequence of total order relations defined over the domain of  $D$  [28].

The sequence of total orders is determined by a sequence of reference points. The reference points are the two boundary points  $P$  and  $P'$ , i.e.  $P = (p_1, p_2, \dots, p_d)$ ,  $p_i = \max(A_i)$ , and  $P' = (p'_1, p'_2, \dots, p'_d)$ ,  $p'_i = \min(A_i)$ . The total order relations between two points in Heda is:  $Dist(x, y) =$

---

**Algorithm 2** Generating an IMA Dataset

---

**Input :**  $D = \{(x_i, y_i)\}_{i=1}^m$ ,  $k$  is the cluster size,  $m \geq 2k$ .  
**Output :** A IMA dataset  $D_{IMA}$  that can perform DP.

- 1: Set  $i := 0$
- 2: **while**  $|D| \geq 2k$  **do**
- 3:     Computing the boundary point  $P$  and  $P'$ .
- 4:      $C_i \leftarrow k$  nearest instances to  $P$  from  $D$  according to  $Dist(x, y)$ ,  $D := D \setminus C_i$ .
- 5:      $C_{i+1} \leftarrow k$  nearest instances to  $P'$  from  $D$  according to  $Dist(x, y)$ ,  $D := D \setminus C_{i+1}$ .
- 6:      $i := i + 2$
- 7:      $C_i \leftarrow$  remaining records.
- 8: Computing each centroid of  $C_i$  and use it to replace the records in each cluster.
- 9: **return**  $D_{IMA}$ .

---

$\sqrt{\sum_{i=0}^d \frac{(x_i - y_i)^2}{(P_i - P'_i)^2}}$ . Generating a IMA dataset is detailed in Algorithm 2.

*2) Determining the sensitivity:* As Definition 5,  $\Delta f_j = \max(A_j)$  in dataset  $D$ , and the sensitivity in IMA is  $\frac{\Delta f_j}{k} \times \frac{m}{2k}$ , which is formalized in the Proposition 2. We detail Algorithm 3 constructing our DP mechanism.

*Proposition 2 ( $\Delta f$  in IMA):*  $f_j(D)$  is a query function with  $\epsilon$ -DP mechanism returning the noised values corresponding to the  $j$ -th attribute of  $D$ . After obtaining  $D_{IMA}$  by Algorithm 2, the sensitivity of  $D_{IMA}$  with cluster size  $k$  is  $\Delta f'_j(D) = \frac{\Delta f_j}{k} \times \frac{m}{2k}$ , where  $\Delta f_j = \max(A_j)$ .

*Proof 2 (Proof of Proposition 2):* If  $M$  is an IMA algorithm, for every pair of datasets  $D$  and  $D'$  differing in a single record, there is a bijection between the set of clusters  $\{C_1, \dots, C_n\}$  and  $\{C'_1, \dots, C'_n\}$  such that each pair of corresponding clusters differs at most in a single record. So if the centroid is computed as the mean of the records in the same cluster, then the maximum change in any centroid is, at most,  $\frac{\Delta f_j}{k}$ . The modification of single record may lead to multiple modifications of the centroid of clusters, and there are  $\lceil \frac{m}{k} \rceil^5$  different clusters in  $D$ .

According to a distance function  $Dist()$  with an total order relation, IMA algorithm iteratively takes sets with cardinality  $k$  from the extreme points until less than  $2k$  records are left. The less than  $2k$  records are formed the last cluster  $C_r$  that is the cluster at the center of the total order relation sequence. Every  $x_i$  in  $D$  is ordered by  $Dist()$ . A pair of databases  $(D, D')$  differing only in one instance  $x_i$  means the larger database contains just one additional row [13]. The number of clusters on the left and the right of  $C_r$  is equal, as shown in the Figure 2. If the different record in  $(D, D')$  is  $x_i$  on the left of  $C_r$  and  $x_i$  is located to  $C_i$ . Then the changed clusters are the clusters from  $C_i$  to  $C_r$ , and the maximum change for each changed cluster is  $\frac{\Delta f_j}{k}$ . Other clusters on the right side of  $C_r$  will not be changed. The worst scenario is when  $x_i$  is located to  $C_1$ , there is the maximum number  $\lceil \frac{m}{2k} \rceil$  of changed clusters. The scenario on the left and the right sides of  $C_r$  is symmetrical, so the number of changed clusters is at most  $\lceil \frac{m}{2k} \rceil$ .  $\square$

---

<sup>5</sup> $\lceil \cdot \rceil$  denotes a ceiling functions.

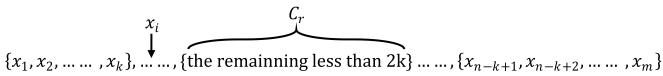


Fig. 2. Clusters in IMA

### Algorithm 3 IMA $\epsilon$ -DP Mechanism

---

**Input :**  $D = \{(x_i, y_i)\}_{i=1}^m$ ,  $k$  is the cluster size,  $m \geq 2k$ .  
**Output :** An IMA  $\epsilon$ -DP dataset  $D_{IMA-\epsilon}$ .

- 1: Generating the appropriate  $\epsilon$  on dataset  $D$  by Algorithm 1.
- 2: Obtaining an IMA dataset  $D_{IMA}$  from  $D$  by Algorithm 2.
- 3: Obtaining noise by using  $\epsilon$  and  $\Delta f_j$  (cf. the definition of Laplace mechanism 4).
- 4: Adding  $x_i^\epsilon = x_i + \text{noise}$  to  $D_{IMA-\epsilon}$ .
- 5: **return**  $D_{IMA-\epsilon}$ .

---

To make the sensitivity of  $D_{IMA}$  smaller than the original dataset, we let  $\frac{\Delta f_j}{k} \times \frac{m}{2k} \leq \Delta f_j$ , then we can get the best cluster size:  $k = \sqrt{\frac{m}{2}}$ . Soria et al. [13] thought that  $D'$  and  $D$  differ in a “modified” record  $x_i$ . The modification causes the whole sequence originally obtained by  $Dist()$  is changed from the position of  $x_i$  in turn. So they considered the sensitive is  $\Delta f'_j(D) = \frac{\Delta f_j}{k} \times \frac{m}{k}$ . However, Dwork er al. [13] give that: “On pairs of Adjacent Dataset ( $D, D'$ ) differing only in one row, meaning one is a subset of the other and the larger database contains just one additional row.” Their sensitive method causes a greater sensitivity than ours, which reduces the usability of the dataset.

## VI. PRIVACY DESIGN WITH HOMOMORPHIC CRYPTOSYSTEM

A homomorphic encryption algorithm can only support one type of operation (e.g., Addition or Multiplication). Existing HC-based secure training algorithms need to rely on trusted third parties such as the Authorization Server [10], [17], or use an approximate equation to simplify the original complex iteration formula [2], [3].

We elaborately design a library of building blocks by multiplicative homomorphic encryption RSA and additively homomorphic encryption Paillier, which is able to construct complex secure ML training algorithms needing no the Authorization Server or any approximate equation. In order to illustrate the power of our building blocks, we construct a secure LR training algorithm (see Section VII-B). It is the first solving the sigmoid function based on HC. In this section, we detail our building blocks. The security proofs for each building block are given in Section VIII.

ML training algorithms are computationally complex, so the building blocks need to support a range of choices including which party gets the input, which party gets the output, and whether the input or output are encrypted or not. Table II shows the different conditions for building blocks. For all conditions, both parties Alice and Bob cannot obtain other useful information except for the legal information, the input and output of other parties are confidential.

**1) Secure addition and secure subtraction:** Relying on Paillier’s additive homomorphic property, it is straightforward

### Protocol 1 Secure Addition Protocol

---

**Input Alice:**  $a = \{a_1, a_2, \dots, a_d\}$ ,  $(PK_{Alice}, SK_{Alice})$ .  
**Input Bob:**  $PK_{Alice}$ ,  $[[b]]_{Alice} = [[\{b_1, b_2, \dots, b_d\}]]_{Alice}$  or  $b = \{b_1, b_2, \dots, b_d\}$ .  
**Output Bob:**  $[[f(a, b)]]_{Alice} = [[a + b]]_{Alice}$ .  
1: Alice sends  $[[a]]_{Alice}$  to Bob.  
2: **for** i to d **do**  
3:     Bob computes  $[[f(a, b)_i]]_{Alice} = [[b_i]]_{Alice} \times [[a_i]]_{Alice}$ .  
4: **return**  $[[f(a, b)]]_{Alice}$  to Bob.

---

### Protocol 2 Secure Subtraction Protocol

---

**Input Alice:**  $a = \{a_1, a_2, \dots, a_d\}$ ,  $(PK_{Alice}, SK_{Alice})$ .  
**Input Bob:**  $PK_{Alice}$ ,  $[[b^{-1}]]_{Alice} = [[\{b_1^{-1}, b_2^{-1}, \dots, b_d^{-1}\}]]_{Alice}$  or  $b^{-1} = \{b_1^{-1}, b_2^{-1}, \dots, b_d^{-1}\}$ .  
**Output Bob:**  $[[f(a, b)]]_{Alice} = [[a - b]]_{Alice}$ .  
1: Alice sends  $[[a]]_{Alice}$  to Bob.  
2: **for** i to d **do**  
3:     Bob computes  $[[f(a, b)_i]]_{Alice} = [[b_i^{-1}]]_{Alice} \times [[a_i]]_{Alice}$ .  
4: **return**  $[[f(a, b)]]_{Alice}$  to Bob.

---

### Protocol 3 Secure Dot Product Protocol

---

**Input Alice:**  $a = \{a_1, a_2, \dots, a_d\}$ ,  $(PK_{Alice}, SK_{Alice})$ .  
**Input Bob:**  $b = \{b_1, b_2, \dots, b_d\}$  and  $PK_{Alice}$ .  
**Output Bob:**  $[[f(a, b)]]_{Alice} = [[a \times b]]_{Alice}$ .  
1: Alice sends  $[[a]]_{Alice}$  to Bob.  
2: Bob computes  $[[f(a, b)]]_{Alice} = \prod_{i=1}^d [[a_i]]_{Alice}^{b_i}$ .  
3: **return**  $[[f(a, b)]]_{Alice}$  to Bob.

---

### Protocol 4 Secure Multiplication Protocol

---

**Input Alice:**  $a$ ,  $(PK_{Alice}, SK_{Alice})$ .  
**Input Bob:**  $||b||_{Alice}$  and  $PK_{Alice}$ .  
**Output Bob:**  $||f(a, b)||_{Alice} = ||a \times b||_{Alice}$ .  
1: Alice sends  $||a||_{Alice}$  to Bob.  
2: Bob computes  $||f(a, b)||_{Alice} = ||a||_{Alice} \times ||b||_{Alice}$ .  
3: **return**  $||f(a, b)||_{Alice}$  to Bob.

---

to obtain the secure addition protocol (Protocol 1) and secure subtraction protocol (Protocol 2).

**2) Secure dot product and secure multiplication:** Using Paillier’s additive homomorphism property, we can construct a secure dot product protocol (Protocol 3) that satisfies Condition 1 easy (i.e.,  $[[a \times b]] = [[a]]^b$ ). However, when Bob only has ciphertext  $[[b]]_{Alice}$  who is unable to perform  $[[a]]^b_{Alice}$ . Paillier fails to construct a secure dot product protocol that satisfies Condition 2. Because the length of Paillier’s ciphertext is 1024 bits or longer usually (We will discuss the key length setting in detail in Section IX.), so the computational complexity of  $[[a]]^{[[b]]}$  is awfully large. Therefor, when faced with Condition 2, we use RSA’s multiplicative homomorphism property to construct secure multiplication protocol (Protocol 4).

**3) Secure power function:** In order to cope with more complex training algorithms, we design protocol 5 satisfying Condition 1 under RSA to obtain  $||e^{ab}||$  securely.

TABLE II. THE CONDITIONS OF BUILDING BLOCKS

Conditions	Input		Output		Protocols
	Alice	Bob	Alice	Bob	
Condition 1	$SK_A, PK_B, a$	$SK_B, PK_A, b$	-	$[f(a, b)]_A$	1, 2, 3, 5
Condition 2	$SK_A, PK_B, a$	$SK_B, PK_A, [b]_A$	-	$[f(a, b)]_A$	1, 2, 4
Condition 3	$SK_A, PK_B$	$SK_B, PK_A,   f(a, b)  _A$	-	$[[f(a, b)]]_A$	6
Condition 4	$SK_A, PK_B$	$SK_B, PK_A, [[f(a, b)]]_A$	-	$[[f(a, b)]]_B$	7

**Protocol 5** Secure Power Function Protocol

**Input Alice:**  $a = \{a_1, a_2, \dots, a_d\}$ ,  $(PK_{Alice}, SK_{Alice})$ .  
**Input Bob:**  $b = \{b_1, b_2, \dots, b_d\}$  and  $PK_{Alice}$ .  
**Output Bob:**  $\|e^{ab}\|_{Alice}$ .  
1: Alice sends  $\|e^a\|_{Alice} = \{\|e^{a_1}, e^{a_2}, \dots, e^{a_d}\|\}_{Alice}$  to Bob.  
2: Bob initializes  $W$ .  
3: In Bob:  
4: **for**  $i$  to  $d$  **do**  
5:     Letting  $w_i = \|e^{x_i}\|_{Alice}$ .  
6:     **for**  $t$  to  $b_i - 1$  **do**  
7:          $w_i = w_i \times \|e^{a_i}\|_{Alice}$  by protocol 4.  
8:      $W = W \times w_i$  by protocol 4.  
9: **return**  $\|W\|_{Alice} = \|e^{ab}\|_{Alice}$  to Bob.

**Protocol 6** Converting Ciphertext:  $\|e^{ab}\|_{Alice}$  to  $[[e^{ab}]]_{Alice}$ 

**Input Alice:**  $(PK_{Alice-Paillier}, SK_{Alice-Paillier})$  and  $(PK_{Alice-RSA}, SK_{Alice-RSA})$ .  
**Input Bob:**  $\|e^{ab}\|_{Alice}$ .  
**Output Bob:**  $[[e^{ab}]]_{Alice}$ .  
1: Bob uniformly picks  $r$  and computes  $\|e^{ab+r}\|_{Alice}$  by Protocol 5.  
2: Bob sends  $\|e^{ab+r}\|_{Alice}$  to Alice.  
3: Alice decrypts  $\|e^{ab+r}\|_{Alice}$ , obtains and sends  $[[e^{ab+r}]]_{Alice}$  to Bob.  
4: Bob computes  $[[e^{ab}]]_{Alice} = [[e^{ab+r}]]_{Alice} \times (e^r)^{-1}$  by protocol 3.  
5: **return**  $[[e^{ab}]]_{Alice}$  to Bob.

**4) Secure changing the encryption cryptosystem:** There are multiple participants in Heda. Different participants have their own encryption scheme (i.e., the certain plaintext space and a pair of keys (PK,SK)). Homomorphic operations can only be operated in the same plaintext space. For completeness, we design two protocols converting ciphertext from one encryption scheme to another while maintaining the underlying plaintext. The first (Protocol 6) switches  $\|e^{ab}\|_{Alice}$  to  $[[e^{ab}]]_{Alice}$  satisfying Condition 3, and the other (Protocol 7) switches  $[[b]]_{Alice}$  to  $[[b]]_{Bob}$  satisfying Condition 4.

*Proposition 3 (The security of building blocks):* Protocol 1 to 7 is secure in the honest-but-curious model.

## VII. CONSTRUCTION OF HEDA

In this section, we introduce the overall framework of Heda. In Heda,  $\mathcal{U}$  is able to learn a model without learning anything about the sensitive data of  $\mathcal{P}$ , and in addition to  $\mathcal{U}$ , others should learn nothing about the model. The security

**Protocol 7** Converting Ciphertext:  $[[b]]_{Alice}$  to  $[[b]]_{Bob}$ 

**Input Alice:**  $(PK_{Alice}, SK_{Alice})$  and  $PK_{Bob}$ .  
**Input Bob:**  $[[b]]_{Alice}$ .  
**Output Bob:**  $[[b]]_{Bob}$ .  
1: Bob uniformly picks  $r$  and computes  $[[b+r]]_{Alice}$  by Protocol 1.  
2: Bob sends  $[[b+r]]_{Alice}$  to Alice.  
3: Alice decrypts  $[[b+r]]_{Alice}$ , obtains and sends  $[[b+r]]_{Bob}$  to Bob.  
4: **return**  $[[b]]_{Bob}$  to Bob.

**Algorithm 4** Privacy-Preserving Training

**Each  $\mathcal{P}$ 's Input:**  $D_i = \{(x_i, y_i)\}_{i=1}^m$ ,  $(PK_{\mathcal{P}_i-Paillier}, SK_{\mathcal{P}_i-Paillier})$  and  $(PK_{\mathcal{P}_i-RSA}, SK_{\mathcal{P}_i-RSA})$ .  
 **$\mathcal{U}$ 's Input:**  $(PK_{\mathcal{U}-Paillier}, SK_{\mathcal{U}-Paillier})$ .  
 **$\mathcal{U}$ 's Output:**  $\beta$ .  
1:  $\mathcal{U}$  initializes  $\beta$ .  
2: According to  $S$ , each  $\mathcal{P}$  divides all features in two part: the high scores part and the low scores part.  
3: each  $\mathcal{P}$  obtains the noise dataset from the low scores part subdataset by Algorithm 3 and sends it to  $\mathcal{U}$ .  
4:  $\mathcal{U}$  trains a model by building blocks (Algorithm 5) combine with noised datasets.  
5: **return**  $\beta$  to  $\mathcal{U}$ .

proof will be given in Section VIII.

Heda is exhibited in Algorithm 4. We introduce the following details in this section: how to use feature evaluation technologies to divide a dataset into two parts (Algorithm 4 Step 2), how to construct a specific training algorithm using building blocks (Algorithm 4 Step 4), and how to combine DP mechanism with the building blocks (Algorithm 4 Step 4).

*Proposition 4 (The Security of Heda):* Algorithm 4 is secure in the honest-but-curious model.

## A. Feature Partitioning

Obviously, It is best that each  $\mathcal{P}$  conducts the feature evaluation locally. The locally computation does not require interaction with any other party which guarantee the privacy of each  $\mathcal{P}$ . In addition,  $\mathcal{P}$  can perform arbitrary computations on their sensitive data in plaintext case with high efficiency. After Several  $\mathcal{P}$  who join in Heda locally implement feature evaluation, they communicate with each other negotiating the final scores  $S$ . Feature scores do not reveal the privacy of datasets, so it is feasible that several  $\mathcal{P}$  share the scores of their datasets and negotiate the final feature scores.

According to the feature scores  $S$ , each  $\mathcal{P}$  processes the dataset into an ordered dataset. Let an ordered dataset  $D = \{A_1, A_2, \dots, A_d\}$  ordered by feature scores, i.e.  $S_i$  is the scores of  $A_i$ ,  $S_i > S_j, 0 < i < j < d$ . Let the high scores part has  $\iota$  features, then  $D = \{D_1, D_2\}$ ,  $D_1 = \{A_1, A_2, \dots, A_\iota\}$ ,  $D_2 = \{A_{\iota+1}, A_{\iota+2}, \dots, A_d\}$ .

We assume that  $\mathcal{U}$  spends  $t_1$  in learning a classifier parameters on the low scores part (the noise dataset) and  $t_2$  on the high scores part (the encrypted dataset), then the total time is  $T = t_1 + t_2$ . Training a model in plaintext case usually takes time in milliseconds but usually takes thousands of seconds or longer in ciphertext case [10], [11]. There is a linear relationship between  $t_2$  and  $\iota$ , i.e.  $t_2 \approx \tau\iota + b$ , where  $\tau > 0$ ,  $\tau$  and  $b$  are two constants. The training time on noise dataset is much less than the time training a model on the encrypted dataset. Formula 2 shows the total time consumption.

$$T \leq \tau(\iota + 1) + b \quad (2)$$

Heda enables flexible switch among different tradeoffs between efficiency and accuracy by parameter tuning. With parameter  $\iota$ , one is able to obtain the tradeoff between efficiency and accuracy. As the decreasing number of the  $\iota$ , the total time consumption is consequent reduction. When the number of dimensions assigned to the high scores part is small, the accuracy is relatively low. According to the specific situations,  $\iota$  is set appropriately.

As for the selection of feature evaluation techniques, many excellent feature evaluation techniques have been studied [8], [21]. When facing a dataset with different types, different backgrounds or different magnitude, different methods have their drawbacks as well as merits. We evaluate six widely used methods in our experiments. The methods we use are: Chi-square test, Kruskal-Wallis H (KW), Pearson correlation, Spearman correlation, Random forest and minimal Redundancy Maximal Relevance (mRMR). We are committed to finding a feature evaluation technique with the best robustness. After extensive experiments, we find that KW has the most stable effect when facing with different datasets (see Section IX-D).

### B. Constructing Specific Training Algorithms using Building Blocks

There are rich variety of ML algorithms. Describing the implementation of building blocks towards each ML training algorithm naturally requires space beyond the page limit. We use LR<sup>6</sup> as an example to illustrate how to construct secure model training algorithms by our building blocks.

LR training algorithm is not the most complicated one compared to other ML classifier training algorithms. However, the iterative process of LR involves sigmoid function (Sigmoid ( $\beta^T x_i$ ) =  $\frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}}$ ) which makes it difficult to implement in ciphertext case. Most studies claimed they had constructed a secure LR training algorithm by HC which were the secure linear regression training algorithms actually, or they solved the sigmoid function by an approximate equation (cf. Section II-B). To best of our knowledge, it is the first

<sup>6</sup>In order to maintain the continuity of our description, LR is also used as the example in the following

---

### Algorithm 5 privacy-preserving training of LR

---

**Each  $\mathcal{P}$ 's Input:**  $D_i = \{(x_i, y_i)\}_{i=1}^m$ ,  $(\text{PK}_{\mathcal{P}_i-\text{Paillier}}, \text{SK}_{\mathcal{P}_i-\text{Paillier}})$  and  $(\text{PK}_{\mathcal{P}_i-\text{RSA}}, \text{SK}_{\mathcal{P}_i-\text{RSA}})$ .

**$\mathcal{U}$ 's Input:**  $(\text{PK}_{\mathcal{U}-\text{Paillier}}, \text{SK}_{\mathcal{U}-\text{Paillier}})$ .

**$\mathcal{U}$ 's Output:**  $\beta$ .

- 1:  $\mathcal{U}$  initializes a learning rate  $\alpha$ , a fixed number of iterations  $Cycles$  and  $\beta$ .
- 2: **while**  $t$  in  $Cycles$  or minimum learning rate is not reached **do**
- 3:      $\mathcal{P}_i$  sends  $(\left[ \sum_{i=1}^m XY \right]_{\mathcal{P}_i})$  to  $\mathcal{U}$ .
- 4:     **for**  $i = 1$  to  $n$  **do**
- 5:          $\mathcal{P}_i$  sends  $(\|e^X\|_{\mathcal{P}_i}, \left[ \sum_{i=1}^m XY \right]_{\mathcal{P}_i})$  to  $\mathcal{U}$ .
- 6:          $\mathcal{U}$  obtains  $[[e^{\beta X}]]_{\mathcal{P}_i}$  by Protocol 5 and Protocol 6 sequentially.
- 7:          $\mathcal{U}$  uniformly picks  $r$  and computes  $[[e^{\beta X+r} + e^r]]_{\mathcal{P}_i}$  by Protocol 1 and Protocol 4 sequentially.
- 8:          $\mathcal{U}$  sends  $[[e^{\beta X+r} + e^r]]_{\mathcal{P}_i}$  to  $\mathcal{P}$ .
- 9:          $\mathcal{P}$  decrypts  $[[e^{\beta X+r} + e^r]]_{\mathcal{P}_i}$  and sends  $\left[ \left[ \frac{X}{e^{\beta X+r} + e^r} \right] \right]_{\mathcal{P}_i}$  to  $\mathcal{U}$ .
- 10:         $\mathcal{U}$  obtains  $\left[ \left[ \frac{X}{e^{-\beta X+1}} \right] \right]_{\mathcal{P}_i}$  by Protocol 3.
- 11:         $\mathcal{U}$  obtains  $\left[ \left[ \sum_{i=1}^m \left( X \frac{e^{\beta X}}{1+e^{\beta X}} - XY \right) \right] \right]_{\mathcal{P}_i}$  by Protocol 1 and Protocol 2 sequentially.
- 12:         $\mathcal{U}$  obtains  $\left[ \left[ \sum_{i=1}^m \left( X \frac{e^{\beta X}}{1+e^{\beta X}} - XY \right) \right] \right]_{\mathcal{U}}$  by Protocol 7.
- 13:         $\mathcal{U}$  updates  $\beta$  by  $\sum_{i=1}^m \left( X \frac{e^{\beta X}}{1+e^{\beta X}} - XY \right)$ .
- 14: **return**  $\beta$  to  $\mathcal{U}$ .

---

constructing a secure LR training algorithm by HC. Our HC-based secure LR training algorithm only needs 3 interactions (i.e. interactions between  $\mathcal{U}$  and  $\mathcal{P}$ ) throughout each iteration process.

LR is a binary classifier and try to learn a pair of parameters  $w$  and  $b$ , where  $w = (w_1, w_2, \dots, w_d)$  to satisfy  $f\{x_i\} = w^T x_i + b$  and  $f\{x_i\} \cong y_i$ . LR uses Sigmoid Function to associate the true label  $y_i$  with the prediction label  $f\{x_i\}$ . Let  $\beta = (w, b)$ ,  $X = (x, 1)$ , the iteration formula of LR is shown in Formula (3). The steps of LR training algorithm are as follows: (i) Initializing learning rate  $\alpha$ , a fixed number of iterations and model parameters  $\beta$ . (ii) Updating  $\beta$  by Formula (3). (iii) If the maximum number of iterations or minimum learning rate is reached, output  $\beta$ ; otherwise go to step 2.

$$\beta_j^{t+1} = \beta_j^t - \frac{\alpha}{m} \sum_{i=1}^m x_{ij} \left( \frac{e^{\beta^T x_i}}{1+e^{\beta^T x_i}} - y_i \right) \quad (3)$$

Each building block is designed in a modular way, so carrying out secure LR training algorithm come down to invoking the right module. Suppose there are  $n$  data providers  $\mathcal{P}$ , Algorithm 5 specifies our secure LR training algorithm. In all execution steps of Algorithm 5, when protocols are called,  $\mathcal{P}$  is the role of Alice, and  $\mathcal{U}$  is the role of Bob.

*Proposition 5:* Algorithm 5 is secure in the honest-but-curious model.

### C. Combining Differential Privacy Mechanism with Our Building Blocks

In this sub-section we present our solution of combining DP mechanism with building blocks, that is, how to train a ML classifier on the mixed dataset that combine an encrypted dataset and a noise dataset.

The idea is simple but effective: We still let the high scores part has  $\iota$  features. After requesting data from  $\mathcal{P}, \mathcal{U}$  obtains a dataset mixed with the encrypted dataset and the noise dataset, as depict in Figure 3. In the process of iteratively updating parameters, learning rate in the high scores part is computed by Algorithm 5, and learning rate in the low scores part is computed by Formula 3 just as in normal plaintext case.

Here, we present a simplified example to understand: Suppose we are computing  $w = X + Y$ .  $X = \{[x_1], [x_2], \dots, [x_\iota], x_{\iota+1}, x_{\iota+2}, \dots, x_d\}$ ,  $Y = \{[y_1], [y_2], \dots, [y_\iota], y_{\iota+1}, y_{\iota+2}, \dots, y_d\}$ . Then,  $w = \{[x_1 + y_1], [x_2 + y_2], \dots, [x_\iota + y_\iota], x_{\iota+1} + y_{\iota+1}, x_{\iota+2} + y_{\iota+2}, \dots, x_d + y_d\}$ . Analogous, when the training algorithm updates parameters, the learning rate of the encrypted dataset is computed by Algorithm 5, and the learning rate of the noise dataset is computed as normal.

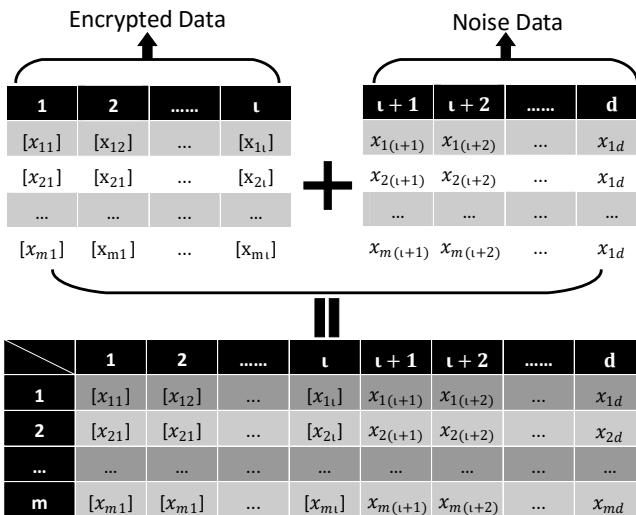


Fig. 3. A Mixed Dataset: the Encrypted Dataset Combine with the Noise Dataset.

## VIII. SECURITY ANALYSIS

When facing the honest-but-curious adversaries, we follow a commonly used definition – secure two-party computation (cf. Appendix A) and a useful theorem – modular sequential composition (cf. Appendix B). We present our security proofs according to the ideas of these two definitions. For more details, we refer the reader to [16] for secure two-party computation and [7] for modular sequential composition.

*Proof 3 (Proof of Proposition 3):*

*Security proof for Protocol 1, 2, 3 and 4:* Protocol 1, 2 and 3 construct secure two-party computation using Paillier's additively homomorphic property. Protocol 4 constructs secure two-party computation using RSA's multiplicative homomorphic property. In these four protocols, Alice does not receive any message whose view only consists in its input, and Alice

does not call to any other protocols. The security proofs of these four protocols can be summarized as follows.

The input of Alice is  $(a, \text{PK}_A, \text{SK}_A)$ , Bob is  $(b \text{ or } [b]_A, \text{PK}_B, \text{SK}_B)$ .

As Alice does not receive any message, Alice's view only consists in its input. Hence the simulator  $S_A^\pi(a, F(a, b)) = \text{view}_A^\pi(a, \text{PK}_A, \text{SK}_A)$ .

Bob's view is  $\text{view}_B^\pi(b \text{ or } [b]_A, \text{PK}_A, [a]_A, \text{Output}_B)$ .  $a$  is encrypted by  $\text{PK}_A$ , and the confidentiality of  $[a]_A$  is equivalent to the cryptosystem, thereby Bob cannot infer the value of  $[a]_A$  straightforward. The simulator  $S_B^\pi$  does the following: (i) Generates  $l$  random coins, obtains  $[c]_A = \{c_1, c_2, \dots, c_l\}_A$  by  $\text{PK}_A$ , where  $l$  is the length of  $a$ . (ii) Outputs  $([c]_A, \text{PK}_A, b \text{ or } [b]_A, [f(c, b)]_A)$ . By semantic security of the used cryptosystem:  $([c]_A, \text{PK}_A, b \text{ or } [b]_A, [F(c, b)]_A) \approx ([a]_A, \text{PK}_A, b \text{ or } [b]_A, [F(a, b)]_A)$ .

*Security proof for Protocol 5:* The input of Alice is  $(a, \text{PK}_A, \text{SK}_A)$ , Bob is  $(b, \text{PK}_B)$ .

As Alice does not receive any message in this hybrid model, Alice's view only consists in its input. Hence the simulator  $S_A^\pi(a, F(a, b)) = \text{view}_A^\pi(a, \text{PK}_A, \text{SK}_A)$ .

Bob's view is  $\text{view}_B^\pi(b, \text{PK}_A, \|e^a\|_A, F_{\text{protocol-4}}(\|e^a\|_A, \|b\|_A, \text{Output}_B)$ .  $e^a$  is encrypted by  $\text{PK}_A$ , and the confidentiality of  $\|e^a\|_A$  is equivalent to the cryptosystem, So Bob cannot infer its value straightforward. The simulation  $S_B^\pi$ , on input  $(b, \text{PK}_A)$ , dose the following: (i) Generates  $l$  random coins, then obtains  $c = \{c_1, c_2, \dots, c_l\}$  and  $\|e^c\|_A$  by  $\text{PK}_A$ , where  $l$  is the length of  $a$ . (ii) Outputs  $(c, \text{PK}_A, \text{SK}_A, \|e^{cb}\|_A, F_{\text{protocol-4}}(\|e^c\|_A, \|b\|_A))$ . As RSA is semantically secure, the distributions  $S_B^\pi = (b, \text{PK}_A, \|e^c\|_A, \|e^{cb}\|_A)$  and  $\text{view}_B^\pi = (b, \text{PK}_A, \|e^a\|_A, \|e^{ab}\|_A)$  are computationally indistinguishable.

As Protocol 4 is secure in the honest-but-curious model, we obtain the security of the Protocol 5 using Theorem 2.

*Security proof for Protocol 6:* The function is  $F: F(\|e^{ab}\|_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R) = (\phi, [[e^{ab}]]_A)$ .

Alice's view is  $\text{view}_A^\pi = ([[e^{ab+r}]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R, \text{output}_{\text{Protocol-3}})$ , and  $\text{output}_{\text{Protocol-3}} = \|e^{ab+r}\|_A$ .  $S_A^\pi$  runs as follows: (i) Generates  $l$  random coins, then obtains  $c = \{c_1, c_2, \dots, c_l\}$  and  $\|e^c\|_A$  by  $\text{PK}_P$ , where  $l$  is the length of  $(ab+r)$ . (ii) Outputs  $(\|e^c\|_A, [[e^c]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R)$ .  $(ab+r)$  and  $c$  are taken from the same distribution, independently from any other parameter. Paillier and RSA is semantically secure. So  $(\|e^c\|_A, [[e^c]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R) \approx (\|e^{ab+r}\|_A, [[e^{ab+r}]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R)$ .

Bob's view is  $\text{view}_B^\pi = (\|e^{ab}\|_A, \|e^{ab+r}\|_A, r, [[e^{ab+r}]]_A)$ .  $S_B^\pi$  runs as follows: (i) Generates  $l$  random coins, obtains  $c = \{c_1, c_2, \dots, c_l\}$ , where  $l$  is the length of  $ab$ . (ii) obtains  $\|e^c\|_A$  and  $\|e^{c+r}\|_A$  by  $\text{PK}_P$ . (ii) Outputs  $(\|e^c\|_A, \|e^{c+r}\|_A, [[e^{c+r}]]_A, r, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R)$ . The distribution of  $c$  and  $ab$  are identical and RSA is semantically secure, so the real distribution  $\{r, \|e^{ab}\|_A\}$  and the ideal distribution  $\{r, \|e^c\|_A\}$  are statistically indistinguishable.

As Protocol 3 is secure in the honest-but-curious model, we obtain the security of the Protocol 6 using Theorem 2.

*Security proof for Protocol 7:* The function is  $F: F([[b]]_A, \text{PK}_A, \text{SK}_A, \text{PK}_B, \text{SK}_B) = (\phi, [[b]]_B)$ .

Alice's view is  $\text{view}_A^\pi = ([[b+r]]_A, \text{PK}_A, \text{SK}_A, \text{PK}_B)$ .  $S_A^\pi$  runs as follows: (i) Generates  $l$  random coins and obtains  $[c]_A = [[\{c_1, c_2, \dots, c_l\}]]_A$  by  $\text{PK}_A$ , where  $l$  is the length of

$(m+r)$ . (ii) Outputs  $([[c]]_B, [[c]]_A, \text{PK}_A, \text{SK}_A, \text{PK}_B)$ .  $(b+r)$  and  $c$  are taken from the same distribution, independently from any other parameter, and Paillier is semantically secure, so  $([[c]]_B, [[c]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R) \approx ([[b+r]]_B, [[b+r]]_A, \text{PK}_P, \text{SK}_P, \text{PK}_R, \text{SK}_R)$ .

Bob's view is  $\text{view}_B^\pi = ([[b]]_A, [[b+r]]_A, r, \text{PK}_B, \text{SK}_B, \text{PK}_A)$ ,  $S_B^\pi$  runs as follows: (i) Generates  $l$  random coins and obtains  $[[c]]_A = [[\{c_1, c_2, \dots, c_l\}]]_A$  by  $\text{PK}_A$ , where  $l$  is the length of  $m$ . (ii) Outputs  $([[c]]_B, [[c+r]]_A, r, \text{PK}_B, \text{SK}_B, \text{PK}_A)$ . The distribution of  $c$  and  $m$  are identical, and Paillier is semantically secure, so the real distribution  $\{[[m]]_B, [[m+r]]_A, r; [[m+r]]_B\}$  and the ideal distribution  $\{[[c]]_B, [[c+r]]_A, r; [[c+r]]_B\}$  are statistically indistinguishable.

As Protocol 1 is secure in the honest-but-curious model, we obtain the security of the Protocol 7 using Theorem 2.  $\square$

*Proof 4 (Proof of Proposition 5):* Since each  $\mathcal{P}_i$  behaves in the same way, we use  $\mathcal{P}_i$  substitute each  $\mathcal{P}$ 's behavior in this security proof.

$\mathcal{U}$ 's view is  $\text{view}_{\mathcal{U}}^\pi = (\text{PK}_{\mathcal{U}}, \text{SK}_{\mathcal{U}}, [[e^X]]_{\mathcal{P}_i}, \|e^X\|_{\mathcal{P}_i}, [[e^{X\beta}]]_{\mathcal{P}_i}, [[\frac{X}{e^{\beta X} + r + e^r}]]_{\mathcal{P}_i}, [[\sum_{i=1}^m X \frac{e^{\beta X}}{1+e^{\beta X}}]]_{\mathcal{P}_i}, \text{Output}_{\mathcal{U}})$ . As

intermediate results are encrypted by the public key of each  $\mathcal{P}_i$ , and Paillier and RSA is semantically secure, thus the sensitive information of each  $\mathcal{P}_i$  is computationally indistinguishable in intermediate results. What we need to discuss is the confidentiality of  $\text{Output}_{\mathcal{U}} = \left( \beta, \sum_{i=1}^m \left( X \frac{e^{\beta X}}{1+e^{\beta X}} - XY \right) \right)$ , that is, whether  $\mathcal{U}$  can guess the sensitive information of each  $\mathcal{P}_i$  from  $\text{Output}_{\mathcal{U}}$  successfully.

Obviously  $a = \sum_{i=1}^m \left( X \frac{e^{\beta X}}{1+e^{\beta X}} - XY \right)$  is a no-solution equation for the unknown  $x$  and the known  $\beta$ . In addition to brute force cracking, there is no other better way to get the value of  $X$ . We assume a small dataset has 2-dimensional 100 instances, and the length of each dimension is 32 bits<sup>7</sup>. Then the probability that  $\mathcal{U}$  guesses success is  $\frac{1}{2^{32 \times d \times m}} = \frac{1}{2^{6400}}$ . It is a negligible probability of success [20].

$\mathcal{P}_i$ 's view is  $\text{view}_{\mathcal{P}_i}^\pi = (\text{PK}_{\mathcal{P}_i}, \text{SK}_{\mathcal{P}_i}, [[e^{\beta X+r} + e^r]]_{\mathcal{P}_i}, \|e^X\|_{\mathcal{P}_i}, [[e^X]]_{\mathcal{P}_i})$ .  $\mathcal{P}_i$  does not output any message. Hence the simulator  $S_{\mathcal{P}_i}^\pi = \text{view}_{\mathcal{P}_i}^\pi$ .  $P_i$  runs as follows: (i) Generates  $l$  random coins and obtains  $c = \{c_1, c_2, \dots, c_l\}$ , where  $l$  is the length of  $r$ . (ii) Uniformly picking  $m = \{m_1, m_2, \dots, m_d\}$ , where  $m \in \mathbb{M}_{\mathcal{P}_i}$ . (iii) Output  $(\text{PK}_{\mathcal{P}_i}, \text{SK}_{\mathcal{P}_i}, [[e^{mX+c} + e^c]]_{\mathcal{P}_i})$ . The distribution of  $(c, m)$  and  $(r, \beta)$  are identical, so the real distribution  $(\text{PK}_{\mathcal{P}_i}, \text{SK}_{\mathcal{P}_i}, [[e^{\beta X+r} + e^r]]_{\mathcal{P}_i})$  and the ideal distribution  $(\text{PK}_{\mathcal{P}_i}, \text{SK}_{\mathcal{P}_i}, [[e^{mX+c} + e^c]]_{\mathcal{P}_i})$  are statistically indistinguishable.

As those Protocols used in Algorithm 5 are secure in the honest-but-curious model, we obtain the security of Algorithm 5 using Theorem 2.  $\square$

*Proof 5 (Proof of Proposition 4):*

Each  $\mathcal{P}$  computes feature scores by feature evaluation technologies locally, obtains the noise dataset by Algorithm 3, and interacts with  $\mathcal{U}$  to process the encrypted dataset by Algorithm 5.  $\mathcal{U}$  trains a LR classifier with the encrypted datasets by Algorithm 5.

<sup>7</sup>Typically, single-precision floating-point occupies 4 bytes (32-bit) memory space.

As those Protocols or Algorithms used in Algorithm 4 are secure in the honest-but-curious model, we obtain the security using Theorem 2.  $\square$

## IX. PERFORMANCE EVALUATION

We present the evaluation of Heda in this section. We answer the following questions in our evaluations toward Heda: (i) the performance of our DP components, (ii) the performance of our building blocks, and (iii) the performance overhead of Heda.

### A. Preparations

*1) Implementations:* Our experiments were run using a desktop computer with configuration: single Intel i7 (i7-3770 64bit) processor for total 4 cores running at 3.40GHz and 8 GB RAM. We have implemented the building blocks, DP components, Heda, and mRMR in Java Development Kit 1.8. Feature selection algorithms including Chi-square test, Pearson correlation, Spearman correlation, Random forest and KW have been implemented in scikit-learn<sup>8</sup>.

The operations of Paillier and RSA are carried out in finite fields involving modular arithmetic operations on integers, while classifiers training generally use floating point numbers. In order to encrypted data taking real values, it is necessary to previously perform a format conversion into an integer representation. According to the international standard IEEE 754, binary floating point number  $D$  is expressed as  $D = (-1)^s \times M \times 2^E$ , where  $s$  is the sign bit,  $M$  is a significant number, and  $E$  is the exponent bit. We employ it to perform the format conversion toward our implementations. Overlength effective bits lead to inefficient algorithms, while underlength effective bits may low the accuracy. We retained two decimal places empirically.

Plaintext should be guaranteed in the plaintext space of the used cryptosystem. So we must consider the key length to avoid the possibility of overflow. After analyzing all the intermediate results, Paillier's key N is set to 2048-bit and RSA's key N is 2048-bit.

*2) Datasets:* Four datasets from the UCI ML repository [12] originated from several different application domains are used in our evaluation: (i) Breast Cancer Wisconsin (Diagnostic) Dataset (BCWD), (ii) Adult Dataset (Adult), (iii) Credit Approval Dataset (CAD), (iv) Car Evaluation Dataset (Car). The statistics are shown in Table III. In order to avoid overfitting or contingent results, we show the average results of cross-validation of 10 runs. In each cross-validation, we randomly take 80% to train the model, and the remainder for testing.

### B. Evaluation of Differential Privacy Components

In DP mechanism, we employ IMA to reduce the sensitivity  $\Delta f$  and give a formula to determining the appropriate privacy budget  $\epsilon$ . We recall that the best cluster size is  $k = \sqrt{\frac{m}{2}}$ , i.e. the best cluster size of the four datasets – BCWD, Adult, CAD and Car – are 16, 127, 18 and 29 respectively. Standard DP (cf. Definition 4) is the baseline for comparison, where the  $\epsilon$  for

<sup>8</sup><http://scikit-learn.org>

TABLE III. STATISTICS OF DATASETS

Datasets	Instances number	Attributes number	Discrete attributes	Numerical attributes
BCWD	699	9	0	9
Adult	32561	14	8	6
CAD	690	15	9	6
Car	1728	6	0	6

each dataset is obtained by Algorithm 1. The scheme proposed by Soria et al. [28] is employed to serve as a control group named IMDAV, which is the state of the art for reducing the sensitivity in DP mechanisms based on IMA. Sum of Squared Errors (SSE) is used to evaluate the information loss of the dataset, and the percentage of Record Linkages (RL) is used to evaluate the risk of privacy leak.

For a given dataset  $D$  and an  $\epsilon$ -DP dataset  $D_\epsilon$  with microaggregation of  $k$  size, SSE (Formula 4) is defined as the sum of squares of attribute distances between original records in  $D$  and their versions in the  $\epsilon$ -DP dataset  $D_\epsilon$ .

$$SSE = \sum_{A_j \in D, A'_j \in D'} \sum_{x_{ij} \in A_j, x'_{ij} \in A'_j} (x_{ij} - x'_{ij})^2 \quad (4)$$

We implement our DP components (Algorithm 3), Standard DP and IMDAV on the four datasets with different cluster sizes, then record different SSE values. The results are depicted in Figure 4. It is clear that our DP components have low information loss compare to IMDAV. When  $k = \sqrt{\frac{m}{2}}$ , our DP components have lower SSE than Baseline.

Insufficient noise may cause privacy cannot be guaranteed. RL is the percentage of records of the original dataset  $D$  that can be correctly matched from the  $\epsilon$ -DP dataset  $D_\epsilon$ .  $R$  is the set of original records that are at minimum distance from  $D$ , and RL is defined as:  $RL = \frac{\sum_{x_i \in D} \Pr(x'_i)}{n}$ , where  $\Pr(x'_i) = \begin{cases} 0 & \text{if } x_i \in R \\ \frac{1}{|R|} & \text{if } x_i \notin R \end{cases}$ . We record the RL on the four datasets for different cluster sizes and show the results in Figure 5. We can see from the Figure 5 that our RL is roughly the same as IMDAV, though our SSE is much lower than it, and our RL is lower than the Baseline when  $k = \sqrt{\frac{m}{2}}$ . The lines in Figure 5(b) and 5(c) are not smooth, the phenomenon of which may be caused by the discrete attributes in Dataset Adult and Dataset CAD.

The smaller SSE means the less loss of information, and the smaller RL means the less risk of privacy leak. In conjunction with Figure 4 and Figure 5, comparing to the Baseline and IMDAV, our DP components in Heda have higher security and less information loss when the cluster size is given the best value  $k = \sqrt{\frac{m}{2}}$ .

### C. Evaluation of Building Blocks

The way to use our building blocks constructing a secure ML training algorithm (cf. Section VII-B) and the security analysis of our building blocks (cf. Section VIII) have been given. In this subsection, we evaluate building blocks in terms of time consumption, accuracy and the number of interactions (round trips), taking Algorithm 5 as an example. A widely

used criterion – accuracy ( $\frac{\# \text{correctly classified instances}}{\# \text{total instances}}$ ) is employed to evaluate the accuracy, and standard LR<sup>9</sup> is implemented as a control group. Table IV(a) gives the running time of each building block (Protocol 1 to Protocol 7) with encrypted four datasets on Algorithm 5, where Protocol 6 named Exchange 1, and Protocol 7 named Exchange 2. Table IV(b) gives the total time consumption, accuracy and communication overhead.

As the performance results in Table IV(a) and Table IV(b), training a model by Algorithm 5 not only has almost no loss of accuracy, but also has the acceptable time consumption. In experiments, we linearly simulate several  $\mathcal{P}$ , and the time consumption of  $\mathcal{P}$  in Table IV(b) is the accumulation of time spent by several  $\mathcal{P}$ . In actual application scenarios, several  $\mathcal{P}$  conduct algorithms in parallel, so that the time consumption of  $\mathcal{P}$  and the total time consumption can be decreased sharply. We just show the raw running time to better illustrate the performance of building blocks. We believe our building blocks to be practical for sensitive applications.

When the datasets become very large, the trained models have almost no loss of accuracy as shown in Table IV(b). Results even show an increase in accuracy when evaluating on dataset Car. The increase is because local optimal values are chosen when initializing model parameters  $\beta$ , and it also shows secure training algorithms constructed by our building blocks would not affect the accuracy. It is worth mentioning that our building blocks construct a secure LR training algorithm without use the Authorization Server and any approximate equation and it is the first solving the sigmoid function in secure LR based on HC.

### D. Evaluation of Heda

We want to find a more robust feature evaluation method for serving Heda, i.e., finding a method that is able to divide the dataset into the high scores part and the low scores part more accurately. We implement six widely used methods and evaluate each of them as following steps: (i) Computing the scores of each attribute by the feature evaluation method, (ii) According to the scores, obtaining ordered datasets (cf. Section VII-A), (iii) Obtaining a new sub-dataset by removing a feature with the lowest scores, then conducting an evaluation on this new sub-dataset, (iv) Repeating step (iii) until there is only one dimensional data left in the dataset. We suppose that step (iii) is able to get an ordered sub-dataset that gets the smallest decline in accuracy compared to the previous sub-dataset. We use the standard LR evaluating each sub-dataset obtained from step (iii) and record the results. Because of the layout restrictions, we only show the results of the dataset Adult and Car. Adult is the dataset with a large amount of data. Car represents the datasets that has only numerical attributes. Figure 6 visualizes the results, where accuracy is employed to measure the quality of datasets.

With respect to the six feature selection algorithms in Figure 6, we find that KW (Green lines) has a more stable performance no matter facing with a all numeric attribute dataset (Car as shown in Figure 6(b)) or a large amount of data dataset with numerical attributes and discrete attributes

<sup>9</sup>A LR model that are trained and tested non-privately using scikit-learn (<http://scikit-learn.org>).

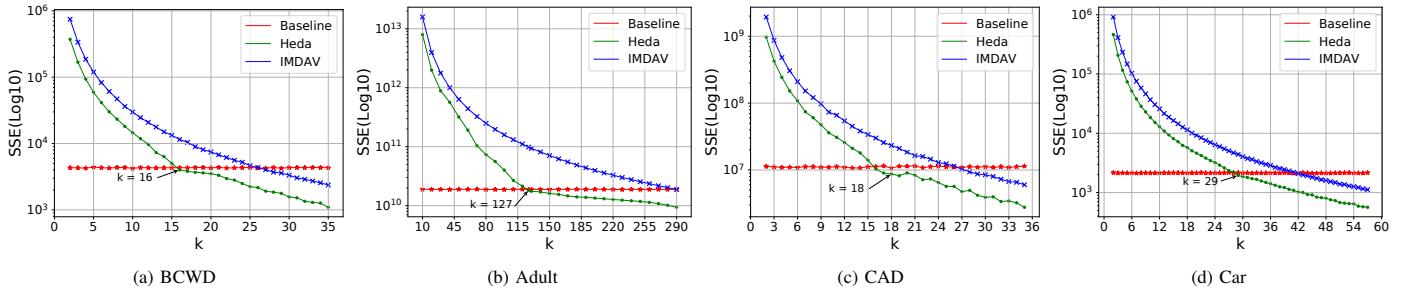


Fig. 4. Differential Privacy Components Performance: Sum of Squared Errors.

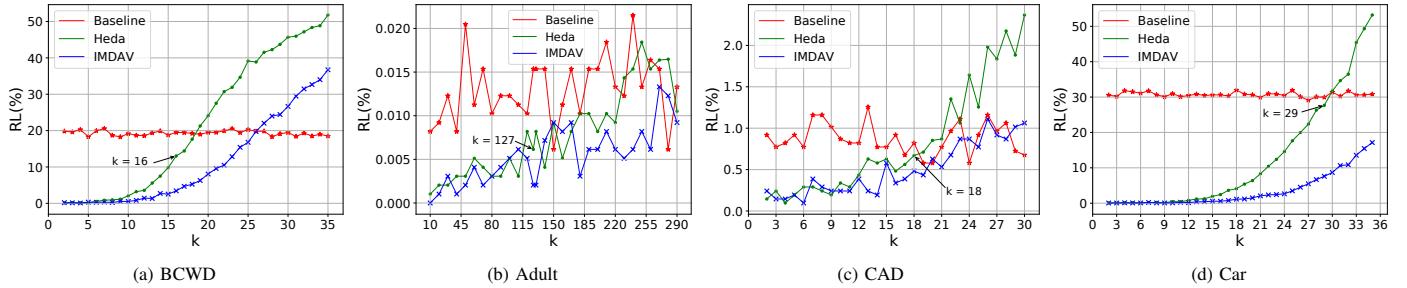


Fig. 5. Differential Privacy Components Performance: Record Linkages.

TABLE IV. BUILDING BLOCKS PERFORMANCE

(a) The running time of each building block

Datasets	Add	Subtraction	Dot Product	Multiplication	Power Function	Exchange 1	Exchange 2
BCWD	1462ms	268ms	372ms	2400ms	2452ms	61510ms	18ms
Adult	17026ms	5710ms	2759ms	47012ms	52291ms	1112344ms	17ms
CAD	8643ms	546ms	1114ms	18882ms	20906ms	411493ms	16ms
Car	2752ms	374ms	890ms	5800ms	6515ms	295934ms	22ms

(b) The total time, accuracy and communication overhead

Datasets	Standard LR	Secure LR	Total Time	$\mathcal{U}$ Time	$\mathcal{P}$ Time	Interactions
BCWD	96.595%	95.422%	2428s	32s	2396s	189
Adult	82.590%	81.578%	41145s	459s	40685s	1500
CAD	85.607%	84.463%	17346s	252s	17094s	1173
Car	72.32%	72.68%	10458s	122s	10335s	1122

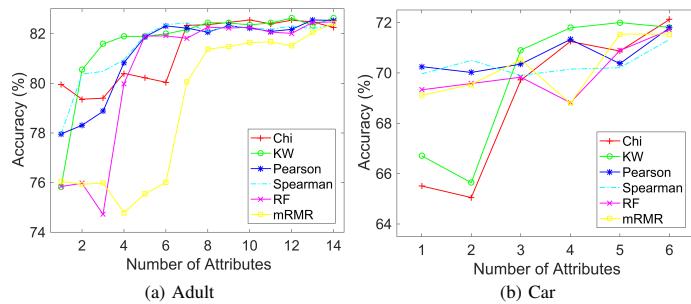


Fig. 6. Feature Evaluation Performance

(Adult as shown in Figure 6(a)). More accurate scores are able to be obtained by KW, so we use KW as our feature evaluation algorithm in the following.

Recalling the steps of Heda, after obtaining the mixed dataset, it computes model parameters on the mixed dataset. Assuming the high scores part has  $\iota$  features. We record and observe the changes in terms of accuracy and time consumption of Algorithm 4 when  $\iota$  changes from 1 to  $d$ . Results are

depicted in Figure 7, where “ $\mathcal{U}$  Time in HC” and “ $\mathcal{U}$  Time in DP” represent the training time spent by  $\mathcal{U}$  on encrypted datasets and noise datasets respectively. After the dataset is published as the noise dataset by DP mechanism, several  $\mathcal{P}$  are no longer involved in the training process, so the “ $\mathcal{P}$  Time” is just the time consumption for several  $\mathcal{P}$  participating in the encrypted dataset training.

*1) Efficiency:* As can be seen from Figure 7, the total time consumption linearly increases as the  $\iota$  grows, which is in line with our preset Formula 2. When  $\iota = d$ , Heda has the lowest efficiency. But when  $\iota$  changes to 1 (i.e., the number of the encrypted attributes are 1), we obtain the highest efficiency, and the total time consumption of Heda spent on each dataset can be reduced by more than 70% compare to when  $\iota = d$ .

When  $\iota$  becomes smaller, the efficiency of Heda becomes higher. The application needs to quickly train a model on sensitive datasets could set  $\iota$  to one to obtain the model parameters as quickly as possible. Results show that Heda is able to train a model within one hour in the face with different scales encrypted datasets. We linearly simulate several  $\mathcal{P}$ , and several  $\mathcal{P}$  could conduct Algorithms in parallel in practical application,

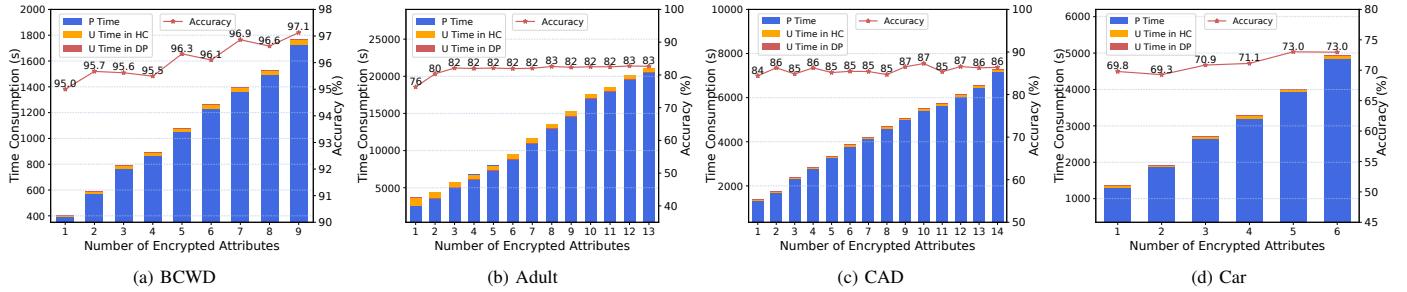


Fig. 7. Heda Performance. The bars represent the time consumption of Heda, and the line represents the accuracy.

so the time consumption of  $\mathcal{P}$  and the total time consumption can be decreased sharply again. The time consumption on noise datasets is less than 1 second, because noise datasets is in the plaintext case, the speed of computing in plaintext case is much faster than computing on the encrypted data.

2) *Accuracy*: We observe from Figure 7 that training speed become faster with smaller  $\iota$ , but the accuracy is reduced, which indicates that the noise dataset does affect the dataset quality to a certain extent. Nevertheless, results show that even when the minimum value of  $\iota$  is taken, the trained classifier accuracy does not be reduced much (not more than 8%). According to the requirements of specific applications, one can adjust  $\iota$  appropriately to obtain a tradeoff between accuracy and time consumption.

Feature evaluation techniques help ML training algorithms remove redundant or noisy attributes, and adding appropriate noise is able to smooth out discrete attributes which has been quantized. So the accuracy is a slightly increased, when  $\iota = 5$  in the mixed dataset CAD, as depicted in Figure 7(c).

3) *Some discussions*: Heda obtains the tradeoffs between efficiency and accuracy by jointly applying HC and DP in an individual scheme, and enables flexible switch among different tradeoffs by parameter tuning. A small  $\iota$  is able to achieve high efficiency and a slight loss in accuracy (In our experiments, the loss is not worse than 8%). Different application scenarios pay different attention to efficiency and accuracy. According to the requirements of specific applications, developers can obtain a balance between accuracy and efficiency by adjusting  $\iota$  appropriately.

## X. CONCLUSION

In this paper, we proposed a novel efficient privacy-preserving ML classifier training algorithm named Heda. By jointly applying HC and DP, Heda obtained the balance between efficiency and accuracy, enabled flexible switch among different tradeoffs by parameter tuning. In order to make Heda more efficient and accurate, we developed a library of building blocks base on HC, gave a formula for determining the appropriate privacy budget, and reduced the sensitivity of the query function by IMA in DP mechanism. We demonstrated the efficiency of Heda and our algorithms. In the future work, we plan to explore a generalized framework which enables constructing a wide range of privacy-preserving ML classifier training algorithms.

## REFERENCES

- [1] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, “Deep learning with differential privacy,” in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’16. New York, NY, USA: ACM, 2016, pp. 308–318.
- [2] Y. AONO, T. Hayashi, L. Trieu PHONG, and L. WANG, “Privacy-preserving logistic regression with distributed data sources via homomorphic encryption,” vol. E99.D, pp. 2079–2089, 08 2016.
- [3] Y. Aono, T. Hayashi, L. Trieu Phong, and L. Wang, “Scalable and secure logistic regression via homomorphic encryption,” in *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY ’16. New York, NY, USA: ACM, 2016, pp. 142–144.
- [4] A. Ben-David, N. Nisan, and B. Pinkas, “Fairplaymp: A system for secure multi-party computation,” in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS ’08. New York, NY, USA: ACM, 2008, pp. 257–266.
- [5] A. Blum, C. Dwork, F. McSherry, and K. Nissim, “Practical privacy: The sulq framework,” in *Proceedings of the Twenty-fourth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, ser. PODS ’05. New York, NY, USA: ACM, 2005, pp. 128–138.
- [6] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, “Machine learning classification over encrypted data,” in *Network and Distributed System Security Symposium*, 2014.
- [7] R. Canetti, “Security and composition of multiparty cryptographic protocols,” *Journal of Cryptology*, vol. 13, no. 1, pp. 143–202, Jan 2000. [Online]. Available: <https://doi.org/10.1007/s001459910006>
- [8] G. Chandrashekhar and F. Sahin, “A survey on feature selection methods,” *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, Jan. 2014.
- [9] K. Chaudhuri and C. Monteleoni, “Privacy-preserving logistic regression,” in *Advances in Neural Information Processing Systems 21*, D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, Eds. Curran Associates, Inc., 2009, pp. 289–296.
- [10] M. D. Cock, R. Dowsley, C. Horst, R. Katti, A. Nascimento, W. S. Poon, and S. Truex, “Efficient and private scoring of decision trees, support vector machines and logistic regression models based on pre-computation,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2017.
- [11] M. d. Cock, R. Dowsley, A. C. Nascimento, and S. C. Newman, “Fast, privacy preserving linear regression over distributed datasets based on pre-distributed data,” in *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*, ser. AISec ’15. New York, NY, USA: ACM, 2015, pp. 3–14.
- [12] D. Dheeru and E. Karra Taniskidou, “UCI machine learning repository,” 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [13] C. Dwork, “A firm foundation for private data analysis,” *Commun. ACM*, vol. 54, no. 1, pp. 86–95, Jan. 2011.
- [14] C. Dwork, F. McSherry, K. Nissim, and A. Smith, “Calibrating noise to sensitivity in private data analysis,” in *Proceedings of the Third Conference on Theory of Cryptography*, ser. TCC’06. Berlin, Heidelberg: Springer-Verlag, 2006, pp. 265–284.
- [15] O. Goldreich, S. Micali, and A. Wigderson, “How to play any mental game,” in *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, ser. STOC ’87. New York, NY, USA: ACM, 1987, pp. 218–229.
- [16] O. Goldreich, *Foundations of cryptography: volume 2, basic applications*. Cambridge university press, 2009.
- [17] F.-J. Gonzlez-Serrano, n. Navia-Vzquez, and A. Amor-Martn, “Training

- support vector machines with privacy-protected data," *Pattern Recognit.*, vol. 72, no. C, pp. 93–107, Dec. 2017.
- [18] T. Graepel, K. Lauter, and M. Naehrig, "ML confidential: Machine learning on encrypted data," in *Information Security and Cryptology – ICISC 2012*, T. Kwon, M.-K. Lee, and D. Kwon, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 1–21.
  - [19] H. Huang, B. Zhao, H. Zhao, Z. Zhuang, Z. Wang, X. Yao, X. Wang, H. Jin, and X. Fu, "A cross-platform consumer behavior analysis of large-scale mobile shopping data," in *Proceedings of the 2018 World Wide Web Conference*, ser. WWW '18. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1785–1794.
  - [20] J. Katz and Y. Lindell, *Introduction to modern cryptography*, ser. CRC Cryptography and Network Security Series. CRC press, 2014.
  - [21] C. Lazar, J. Taminau, S. Meganck, D. Steenhoff, A. Coletta, C. Molter, V. de Schachtzen, R. Duque, H. Bersini, and A. Nowe, "A survey on filter techniques for feature selection in gene expression microarray analysis," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 9, no. 4, pp. 1106–1119, July 2012.
  - [22] J. Lee and C. Clifton, "How much is enough? choosing  $\epsilon$  for differential privacy," in *Proceedings of the 14th International Conference on Information Security*, ser. ISC'11. Berlin, Heidelberg: Springer-Verlag, 2011, pp. 325–340.
  - [23] Y. Lindell and B. Pinkas, "An efficient protocol for secure two-party computation in the presence of malicious adversaries," in *Proceedings of the 26th Annual International Conference on Advances in Cryptology*, ser. EUROCRYPT '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 52–78.
  - [24] X. Liu, R. Lu, J. Ma, L. Chen, and B. Qin, "Privacy-preserving patient-centric clinical decision support system on naive bayesian classification," *IEEE Journal of Biomedical and Health Informatics*, vol. 20, no. 2, pp. 655–668, March 2016.
  - [25] F. D. McSherry, "Privacy integrated queries: An extensible platform for privacy-preserving data analysis," in *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '09. New York, NY, USA: ACM, 2009, pp. 19–30.
  - [26] M. A. Pathak, S. Rane, and B. Raj, "Multiparty differential privacy via aggregation of locally trained classifiers," in *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'10. USA: Curran Associates Inc., 2010, pp. 1876–1884.
  - [27] Y. Rahulamathavan, R. C. W. Phan, S. Veluru, K. Cumanan, and M. Rajarajan, "Privacy-preserving multi-class support vector machine for outsourcing the data classification in cloud," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 467–479, Sept 2014.
  - [28] J. Soria-Comas, J. Domingo-Ferrer, D. Sánchez, and S. Martínez, "Enhancing data utility in differential privacy via microaggregation-based k-anonymity," *The VLDB Journal*, vol. 23, no. 5, pp. 771–794, Oct. 2014.
  - [29] C. Sun, A. Shrivastava, S. Singh, and A. Gupta, "Revisiting unreasonable effectiveness of data in deep learning era," in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 843–852.
  - [30] J. Vaidya, B. Shafiq, W. Fan, D. Mahmood, and D. Lorenzi, "A random decision tree framework for privacy-preserving data mining," *IEEE Transactions on Dependable and Secure Computing*, vol. 11, no. 5, pp. 399–411, Sept 2014.
  - [31] S. Vora and H. Yang, "A comprehensive study of eleven feature selection algorithms and their impact on text classification," in *2017 Computing Conference*, July 2017, pp. 440–449.
  - [32] Q. Wang, S. Hu, M. Du, J. Wang, and K. Ren, "Learning privately: Privacy-preserving canonical correlation analysis for cross-media retrieval," in *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, May 2017, pp. 1–9.
  - [33] W. Wang, C.-M. Vong, Y. Yang, and P.-K. Wong, "Encrypted image classification based on multilayer extreme learning machine," *Multidimensional Syst. Signal Process.*, vol. 28, no. 3, pp. 851–865, Jul. 2017.
  - [34] E. Yilmaz, H. Ferhatosmanoglu, E. Ayday, and R. C. Aksoy, "Privacy-preserving aggregate queries for optimal location selection," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2017.
  - [35] C. Zhang, L. Zhu, C. Xu, and R. Lu, "PPDP: an efficient and privacy-preserving disease prediction scheme in cloud-based e-healthcare system," *Future Generation Comp. Syst.*, vol. 79, pp. 16–25, 2018.
  - [36] L. Zhang, T. Hu, Y. Min, G. Wu, J. Zhang, P. Feng, P. Gong, and J. Ye, "A taxi order dispatch model based on combinatorial optimization," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '17. New York, NY, USA: ACM, 2017, pp. 2151–2159.
  - [37] H. Zhu, X. Liu, R. Lu, and H. Li, "Efficient and privacy-preserving online medical prediagnosis framework using nonlinear svm," *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 3, pp. 838–850, May 2017.
  - [38] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially private data publishing and analysis: A survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 8, pp. 1619–1638, Aug 2017.

## APPENDIX

### A. Secure two-party computation

For all our two-party protocols, to ensure security, we have to show that whatever Alice (Bob) can compute from its interactions with Bob (Alice) can be computed from its input and output, which leads to a commonly used definition secure two-party computation (e.g., [6], [11], [16]).

We follow the notations of Bost et al. [6]: Let  $F = (F_A, F_B)$  be a (probabilistic) polynomial function and  $\pi$  a protocol computing  $F$ ; Alice and Bob want to compute  $F(a, b)$  where  $a$  is Alice's input and  $b$  is Bob's input, using  $\pi$  and with the security parameter  $\lambda$ ; The view of party Alice during the execution of  $\pi$  is the tuple  $\text{view}_\text{Alice}^\pi(\lambda, a, b) = (\lambda; a; m_1, m_2, \dots, m_n)$  where  $m_1, m_2, \dots, m_n$  are the messages received by Alice. We define the view of Bob similarly. Let  $\text{output}_\text{Alice}^\pi(a, b)$  and  $\text{output}_\text{Bob}^\pi(a, b)$  denote Alice's and Bob's outputs respectively. The global output as  $\text{output}^\pi(a, b) = (\text{output}_\text{Alice}^\pi(a, b), \text{output}_\text{Bob}^\pi(a, b))$ .

*Definition 6:* (Secure Two-Party Computation) [6], [16]. A protocol  $\pi$  privately computes  $f$  with statistical security if for all possible inputs  $(a, b)$  and simulators  $S_{\text{Alice}}$  and  $S_{\text{Bob}}$  hold the following properties<sup>10</sup>:

$$\begin{aligned} \{S_{\text{Alice}}, f_2(a, b)\} &\approx \{\text{view}_\text{Alice}^\pi(a, b), \text{output}^\pi(a, b)\} \\ \{f_1(a, b), S_{\text{Bob}}\} &\approx \{\text{output}^\pi(a, b), \text{view}_\text{Bob}^\pi(a, b)\} \end{aligned}$$

### B. Modular sequential composition

For our protocols and algorithms are constructed in a modular way, we use the following useful sequential modular composition (Theorem 2) [7]: The Parties run a protocol  $\pi$  and use calls to an ideal functionality  $F$  in  $\pi$  (e.g. A and B compute  $F$  privately by sending their inputs to a trusted third party and receiving the results); If we can show that  $\pi$  respects privacy in the honest-but-curious model and if we have a protocol  $\rho$  that privately computes  $F$  in the same model, then we can replace the ideal calls for  $F$  by the execution of  $\rho$  in  $\pi$ ; the new protocol, denoted  $\pi^\rho$  is then secure in the honest-but-curious model [6], [7]. We assume an incorruptible trusted party T for evaluating functionalities  $(F_1, F_2, \dots, F_n)$  – hybrid model.

<sup>10</sup> $\approx$  denotes computational indistinguishability against probabilistic polynomial time adversaries with negligible advantage in the security parameter  $\lambda$ .

Parties not communicate until receiving T's output (i.e. sequential composition). Let  $\pi$  be a two-party protocol in the  $(F_1, F_2, \dots, F_n) - hybrid model$ , and  $\rho_1, \rho_2, \dots, \rho_n$  be real protocols (i.e. protocols in the semi-honest model) computing  $F_1, F_2, \dots, F_n$ . All ideals calls of  $\pi$  to the trusted party for  $F_i$  is replaced by a real execution of  $\rho_i$ .

*Theorem 2:* (Modular Sequential Composition) restated as in [6]. Let  $F_1, F_2, \dots, F_n$  be two-party probabilistic polynomial time functionalities and  $\rho_1, \rho_2, \dots, \rho_n$  protocols that compute respectively  $F_1, F_2, \dots, F_n$  in the presence of semi-honest adversaries. Let  $G$  be a two-party probabilistic polynomial time functionality and  $\pi$  a protocol that securely computes  $G$  in the  $(F_1, F_2, \dots, F_n) - hybrid model$  in the presence of semi-honest adversaries. Then  $\pi^{\rho_1, \rho_2, \dots, \rho_n}$  securely computes  $G$  in the presence of semi-honest adversaries.