RASP-Boost: Confidential Boosting-Model Learning with Perturbed Data in the Cloud

Keke Chen and Shumin Guo
Data Intensive Analysis and Computing Lab, Kno.e.sis Center
Department of Computer Science and Engineering
Wright State University, Dayton, OH 45435
Email: {keke.chen, guo.18}@wright.edu

Abstract—Mining large data requires intensive computing resources and data mining expertise, which might be unavailable for many users. With widely available cloud computing resources, data mining tasks can now be moved to the cloud or outsourced to third parties to save costs. In this new paradigm, data and model confidentiality becomes the major concern to the data owner. Data owners have to understand the potential trade-offs among client-side costs, model quality, and confidentiality to justify outsourcing solutions. In this paper, we propose the RASP-Boost framework to address these problems in confidential cloud-based learning. The RASP-Boost approach works with our previous developed Random Space Data Perturbation (RASP) method to protect data confidentiality and uses the boosting framework to overcome the difficulty of learning high-quality classifiers from RASP perturbed data. We develop several cloud-client collaborative boosting algorithms. These algorithms require low client-side computation and communication costs. The client does not need to stay online in the process of learning models. We have thoroughly studied the confidentiality of data, model, and learning process under a practical security model. Experiments on public datasets show that the RASP-Boost approach can provide high-quality classifiers, while preserving high data and model confidentiality and requiring low client-side costs.

1 INTRODUCTION

Most data mining tasks require a good understanding of the mining techniques, time-consuming parameter tuning, algorithm tweaking, and sometimes algorithm innovation. They are normally resource-intensive and often need the expertise of applying data-mining techniques in a large-scale parallel processing cluster. As a result, many data owners, who have no sufficient computing resources or data-mining expertise, cannot mine their data.

The development of cloud computing enables at least two solutions. First, if data owners have the data-mining expertise but not the computing resources, they can rent public cloud resources to process their data with low costs and in a short time. A 100-node cluster built with small virtual machine instances in Amazon EC2 costs less than 10 dollars per hour and is ready in a few minutes. Second, if data owners do not have the expertise, they can also outsource their data-mining tasks to data-mining service providers, known as *Mining as a Service* (MaaS) in the context of cloud computing. For example, Kaggle (kaggle.com) provides such a service and also crowd-sources data mining tasks to the public.

In spite of all these benefits of cost saving and elasticity of resource provisioning, the unprotected mining-with-the-cloud approach has at least three drawbacks.

- The exported data may contain private information. The Netflix prize, as a successful example of outsourced mining, has to be suspended due to privacy breach of the shared data [1].
- The data ownership is not protected. Once published, the dataset can be accessed and used by anyone. As data has become an important property for many companies, protecting data ownership is in top priority.
- The ownership of the mined models is not protected. Curious cloud service providers can easily learn the unprotected models, and make profits by sharing them with other parties, which may seriously damage the data owner's interests.

As a result, data owners have realized that without protecting data and model confidentiality they will have to keep their data mining tasks in-house.

The previous studies on privacy preserving data mining (PPDM) [2], data anonymization [3], and differential privacy [4] have distinct system settings and privacy requirements from our study (see Related Work for details). They normally aim to *share data and mined models* with untrusted parties without leaking the individual's private information in the data. Our goal is to *prevent sharing* data and models with untrusted parties.

While confidentiality is the major concern in outsourced mining, the expense of a confidential solution needs to justify the cost saving purpose of outsourced computing. Among all the related costs, the data owner is more conscious about the in-house processing costs required by a confidential learning solution. Specifically, such client-side costs include in-house pre- and post- processing, and the communication between the client and the cloud/the service provider. A practical solution has to minimize these client-side costs.

Model quality is as important as good confidentiality and low client-side costs. Depending on the specific applications, sacrificing a small amount of model quality to achieve other benefits could be quite acceptable to users. An important study is to understand how this trade-off happens and what the optimal trade-off is.

The Scope of Research. We propose the RASP-Boost framework to address the key aspects of confidential cloud-based learning: data confidentiality, model confidentiality, model quality, and the client-side costs. Our current work focuses on confidentially learning boosting classification models with RASP perturbed training data exported to untrusted clouds or data-mining service providers. The data in the cloud is protected by the RASP (RAndom Space Perturbation) data perturbation [5]. We propose several client-cloud collaborative learning methods, aiming to find the best trade-off among the key aspects.

The key problem is to compute with the "protected data". The current well-known approaches include Fully Homomorphic Encryption (FHE) [6] and Garbled Circuits (GC) [7]. FHE enables homomorphic additions and multiplications on encrypted data without decrypting them. This approach theoretically allows any function to be implemented with an FHE scheme. However, the current most efficient schemes are still far from practical use [6], [8]. In particular, the current schemes will generate large-size ciphertext, (e.g., one plaintext value is turned to about 100K ciphertext), making the cost of outsourced mining difficult to justify. Garbled Circuits allow the data owner to encode a computation circuit based on secure gates and then outsource it to the cloud for evaluation. The data can be encrypted by simply adding uniformly random noises, and the circuits are used to de-noise and conduct data mining operations [7], [9]. However, all the gates are based on bit operations, and each gate evaluation involves two-party communication. Specifically, for each gate the data owner needs to send four encrypted 80-bit values, and thus the communication costs are very high.

We notice that in the outsourced mining context (Figure 1), the client side can take a small amount of computational responsibility with low communication costs. This factor might be used to simplify the protocol design. The key is to minimize the client-side costs to justify the benefits of cost saving for outsourced computing. We will design different algorithms to address this issue.

Another key idea is using the boosting method to address the difficulties brought by the perturbed data. Specifically, RASP perturbation only allows low-quality models (i.e., non-optimal linear classifiers) to be learned from perturbed data. The boosting model is an ensemble of a set of weighted weak classifiers, each of which can give prediction accuracy barely higher than a random guess. In existing boosting algorithms, such weak classifiers are selected from a large hypothesis space to be locally optimal. Our approach studies the effect of a relatively small pool of *randomly* generated weak classifiers for boosting. This strategy is applied to minimize the client side's work and preserve the confidentiality of the weak classifiers. Our result shows that boosting with such randomly generated weak classifiers can still produce high-quality models.

With these key ideas in mind, we studied the collaborative learning algorithms under the RASP-Boost framework. Our research has several unique contributions.

1) We identify the urgent need and the challenges of confidential classifier learning in the cloud or with data-mining service providers. We also propose four measures: data confidentiality, model confidentiality, model accuracy, and client-side costs to holistically evaluate a confidential learning method.
2) The proposed approach can utilize the RASP perturbation method to protect data and model confidentiality, which provides much stronger confidentiality guarantee than other perturbation methods such as geometric data perturbation [10] and random projection perturbation [11].
3) As we originally design the RASP perturbation method for outsourced database services (i.e., range queries and kNN queries) [5], it was unknown whether it can be used for confidential cloud mining or not. We design algorithms to generate weak classifiers with random half-space range queries, and then use the boosting framework to construct high-quality models from these weak classifiers. Our experiments show that two of the four candidate methods can generate high-quality models with accuracy very close to the optimal boosting models.
4) Due to the unique properties of RASP perturbation, we can prove that the confidentiality of data, model, and learning process is satisfactorily guaranteed. In particular, we develop a new method to evaluate model confidentiality that directly links to model quality and attack evaluation.

The remaining sections are organized as follows. First, we present the related work in Section 2. Next, we give the notations and theoretical background in Section 3. Then, we present the RASP-Boost framework and the security model in Section 4. In Section 5, we discuss the four algorithms for the client to generate base classifiers and for the cloud to search and find optimal one in boosting iterations. We also analyze the related costs and the confidentiality guarantee of the proposed approach. Section 6 focuses on the experimental evaluation of the costs and performance of these algorithms to find the best candidate. It also shows the model confidentiality of the proposed methods for several carefully selected real datasets.

2 Related Work

Previous studies on privacy-preserving data mining (PPDM) are often based in the context of sharing data and mined models without leaking individual data records. Thus, the confidentiality of data and model is not the major concern, but the private information hidden in the shared data is. Three groups of techniques have been developed in PPDM. (1) Additive perturbation techniques that hide the real values by adding noises [2]. Because the resultant models are not protected, they are not appropriate for outsourced mining. (2) Cryptographic protocols that enable multi-party collaborative mining without leaking either party's private information [12]. These protocols typically do not move the original data to another party, but exchange intermediate results, which are proved not breaching data privacy. However, the learned models are shared by the participants. (3) Data anonymization [3] that disguises personal identities or virtual identifiers in the shared data. It only protects personal identities, while the sensitive attributes and the resultant models are not protected.

Some of the PPDM work also targets on outsourced mining, such as geometric data perturbation [10] and random projection perturbation [11], which can be applied to cloud-based or data-mining-service based mining. They typically transform both the data and the learned models, trying to protect the confidentiality of both. The RASP-Boost framework also works for these methods. However, these two perturbation methods are subject to distance-based attacks [13] and the ICA attack [10], providing much weaker protection than the RASP perturbation.

Fully Homomorphic Encryption [14] envisions an ideal scenario for confidential cloud computing. It allows basic operations: addition and multiplication to be done on the encrypted data without the need of decryption. Theoretically, once the basic homomorphic operations are implemented, any functions can be built up on top of them. However, the current solutions are way too expensive to be used in practice [8], [15]. Some operations such as comparison and division are very expensive. The ciphertext is also too large to be practical for encrypting large datasets with the current solution - one value will be encoded as a 100KB ciphertext [6]. ML Confidential [16] uses a building block of the current best FHE implementation: the Ring-LWE based encryption to encrypt data for computing functions that require a small number of multiplications. It also suffers from most problems with the current FHE implementation.

Recently, efficient implementations of Garbled Circuits (e.g., FastGC [7]) are also applied to confidential cloud-based computing. The data owner encodes both data and the circuits that securely compute certain functions on the encoded data, and exports them to the cloud. The cloud then interacts with the data owner to execute the circuits. The whole process does not leak additional information. However, since each gate of the circuit has to be encoded and each gate evaluation has to involve both parties, the client-side computation and communication costs are extremely high. Nikolaenko et al. [9] implemented a garbled circuit for matrix factorization, where the communication cost is about 40GB for a small 4096×4096 matrix.

Bost et al. [17] address a related problem in cloud-based learning: privately evaluating an already learned model in the cloud. Assume the model is already learned, encoded in a secure form, and hosted in the cloud. The users of the model will submit encrypted input data and privately interact with the cloud via the proposed protocols to find the result of the prediction. The protocols do not leak additional information to a curious cloud provider. However, privately evaluating a function on a dataset has much lower computation and storage complexity than privately learning a model from a dataset. It would be interesting to study the possibility of using their building blocks to construct algorithms learning models from datasets.

Secure database outsourcing has similar security assumptions to confidential cloud mining. The major database components are moved to the cloud to save costs. Typical techniques include order preserving encryption (OPE) [18], crypto-index [19], and secure kNN [20]. However, if the dimensional data distributions are known, OPE and crypto-index cannot protect data confidentiality. CryptDB [21] uses OPE for some database query operations and thus provides weak security guarantee on distributional attacks. Note that although OPE is one component of the RASP perturbation method, RASP does not preserve dimensional order after the A transformation, which invalidates the distributional attacks on OPE [5].

3 Theoretical Background

First, we will give the notations and basic concepts used in this paper. Then, we will briefly introduce the RASP perturbation method and its important properties to make the paper self-contained.

3.1 Notations and Definitions

Our work will be focused on classifier learning on numeric datasets. Classifier learning is to learn a model $t = f(x)$ from a set of training examples $R = \{(x_i, t_i), i = 1 \dots N\}$, where N is the number of examples, $x_i \in \mathbb{R}^d$ is a d-dimensional feature vectors describing an example, and t_i is the label (or target value) for the example - if we use '+1' and '-1' to indicate two classes, $t_i \in \{-1, +1\}$. The learning result is a function $t = f(x)$, i.e., given any known feature vector x, we can predict the label t for the example x. The quality of the model is defined as the accuracy of prediction on the testing set T that has the same structure.

We will use the boosting framework [22] in our approach to preserving model quality in cloud mining. A boosted model is a weighted sum of n base classifiers, $H(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$, which has the following features. (1) The base models $h_i(x)$ can be any *weak learner*, of which the accuracy is slightly higher than a random guess. For example, it has >50% accuracy for the two-class problem. (2) α_i, $\alpha_i \in \mathbb{R}$, are the weights of the base models, which are learned by using algorithms such as AdaBoost [22].

In general, a weak learner can be treated as a simple decision rule, such as:

```
if $h(x) < 0$ then t=-1, otherwise t=+1,
```

for the two-class case. One of the simple weak learners is **linear classifier (LC)**: $h(x) = w^T x + b$, where $w \in \mathbb{R}^k$ and $b \in \mathbb{R}$ are to be learned from examples. With linear classifiers, $h(x)$ is a hyperplane and $f(x) < 0$ defines a **half space**.

An even simpler weak learner is **decision stump (DS)**. Let X_j denote the j-th dimension of the feature vectors and a be some constant in X_j's domain. A condition $X_j < a$ can be used as a decision rule, for example,

```
if $X_j < a$ then t=-1; otherwise, t=1.
```

Note that $X_j < a$ can be represented in the form of a hyperplane $w^T x + b < 0$ as well, by setting $b = -a$ and all dimensions of w to 0 except for dimension j, w_j to 1. Thus, decision stump is also a linear classifier.

3.2 RASP perturbation

The random space perturbation method (RASP) works on vector datasets. For each d-dimensional original vector x_i, the RASP perturbation can be described in three steps.

1) The vector x_i is transformed by applying an order preserving encryption[1] (OPE) [18], denoted as $E_{OPE}(K_{OPE}, x_i)$ (and $E(x_i)$ for simplicity), where K_{OPE} is the OPE key. The OPE scheme is used to transform the distribution of j-th dimension X_j to a normal distribution $N(\mu_j, \sigma_j^2)$ with dimensional order preserved, where the distribution parameters, the mean μ_j and the standard deviation σ, are selected as a part of the OPE key.
2) The transformed vector is extended to $d+2$ dimensions as $z_i = ([E_{OPE}(K_{OPE}, x_i)]^T, 1, v_i)^T$: the $(d+1)$-th dimension is always 1; the $(d+2)$-th dimension, v_i, is drawn from a random number generator RG that generates values from the normal distribution $N(\mu_{d+2}, \sigma_{d+2}^2)$, with $v_i > v_0$, where v_0 is set to such a constant that the probability of having $v_i < v_0$ is negligible so that the distribution of the selected values keeps normal.
3) The $(d+2)$-dimensional vector is further transformed to

$$\begin{aligned} y_i &= RASP(x_i; A, K_{OPE}) \\ &= A(E_{OPE}(K_{OPE}, x_i)^T, 1, v_i)^T, \end{aligned} \quad (1)$$

where A is a $(d+2) \times (d+2)$ randomly generated invertible matrix.

Note that A is the secret key matrix shared by all vectors, but v_i is randomly generated for each individual vector. As a result, the same original vector x_i can be mapped to different y_i in the perturbed space due to the randomly chosen v_i, which provides the desired indeterministic feature. Because of the dimensional OPE transformations, the y vectors makes approximately normal distribution with the major population around the origin, which is resilient to a certain kind of attack as discussed later.

Secure Half-Space Query. The RASP perturbation approach enables a secure query transformation and processing method for half-space queries [5]. A simple half-space query is like $X_i < a$, where X_i represents the dimension i and a is a scalar. It can be transformed to an encrypted half-space query in the perturbed space: $y^T Q y < 0$, where y is the perturbed vector, and Q is the $(d+2) \times (d+2)$ query matrix, which can be done as follows. Note that $X_i - a < 0$ is equivalent to $(X_i - a)(V - v_0) < 0$, where V is the appended noise dimension in z_i which guarantees $V - v_0 > 0$. Let $z_i = (E(x_i)^T, 1, v_i)^T$ be the intermediate extended vector, i.e., $z_i = A^{-1} y_i$. The two parts $X_i - a$ and $V - v_0$ are transformed to $z^T u$ and $v^T z$, respectively, where $u = (w^T, -E(a), 0)^T$, w is the dimension indication vector: all entries are zero except for the dimension i set to 1, and $v = (0, \ldots, -v_0, 1)^T$ is a vector with all entries zero except for the last two. Plug in $z = A^{-1} y$, we get the quadratic form $y^T (A^{-1})^T u v^T A^{-1} y < 0$. Thus, we have

$$Q = (A^{-1})^T u v^T A^{-1}. \quad (2)$$

Under the security assumptions, which will be described later, as long as the matrix A keeps confidential, there is no effective method to recover the condition $X_i < a$ from the exposed matrix Q.

The half-space queries can be generalized to *general linear queries*. However, it would be difficult to convert a linear query, say $w^T x + b < 0$, in the original space to a query in the RASP-perturbed space, because of the non-linear OPE transformation. Instead, we can design linear queries based on the intermediate space and derive the query matrix Q, where each record $s_i = E(x_i)$. A linear function $f(z) = u^T z$ with $u = (w^T, -b, 0)^T$ is $g(s) = w^T s + b$ in the OPE transformed d dimensional space. $f(z)$ can be readily mapped to the final perturbed space, giving the query matrix Q. This method will be used in our design of general linear classifier.

4 The RASP-Boost Framework

We will briefly describe the procedure of learning with the cloud or the mining service provider, and the security model in this context.

Using cloud infrastructure services for mining and using data mining services are slightly different in terms of the level of user involvement in the mining process. Figure 1) shows the interactions between the cloud and the client. The data owner prepares protected data (e.g., perturbed datasets), $P = F(D)$, and auxiliary data (e.g., queries), exporting them to the cloud. Then, they use the cloud resources to mine models. Data owner needs to take care of all the mining steps, which may include multiple interactions between the client and the cloud. This two-party framework gives more flexibility for the design of confidential mining algorithms as you can expect the client side to be an integrated part of the mining process. In contrast, with data mining services, the data owner only provides the perturbed data and auxiliary data in the beginning and is notified when the models are ready. In this setting, the data owner prefers not staying online in the process of mining, and thus, ideally, the intermediate interactions should be minimized or eliminated. Our design will consider eliminating the interactions in the middle of learning so that they can be applied to both application scenarios.

4.1 Major Procedures

Preparing Training Data. The data owner uses the RASP perturbation to prepare the training data for outsourcing. To protect the confidentiality of training data, we assume it is sufficient to protect the confidentiality of the feature vectors x_i of each training record $\{x_i, t_i\}$ while leaving t_i unchanged. This procedure exposes very limited information to the attackers. Thus, the problem becomes learning from the perturbed data $\{(Perturb(x_i), t_i)\}$.

Privately Learning Models. To make sure the models learned from the perturbed data useful, we introduce the definition of ϵ-effective learning. Let H be the classifier learned from the original data $D = \{(x_i, t_i)\}$ and H_P be the

1. For any value pairs in the original space, if they have an order, say $a < b$, then we also have the same order $E_{OPE}(a) < E_{OPE}(b)$.

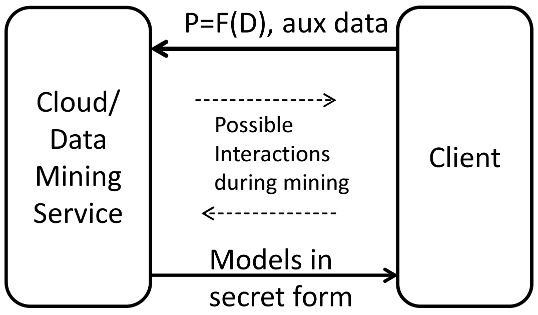
P=F(D), aux data
Cloud/ Data Mining Service
Possible Interactions during mining
Client
Models in secret form

Fig. 1. The RASP-Boost framework works with the cloud infrastructure or data mining service providers.

one learned from $D' = \{(P(x_i), t_i)\}$, where $P()$ is a specific perturbation method.

Definition 1. *Let $Error(H, D)$ represent the classification evaluation function that applies the classifier H to the data D and returns the error rate. For any set of testing data D, if $|Error(H, D) - Error(H_P, D')| < \epsilon$, where ϵ is a user-defined small positive number, we say that learning from the perturbed data is* ϵ-effective.

In practice, because of the downgraded data quality (e.g., noise addition) or the specific way transforming the data, the available learning methods are quite limited and learning from perturbed data often results in sub-optimal models (i.e., ϵ has to be large). To find ϵ-effective classifiers for small ϵ, we propose to use the boosting idea in our framework. This framework extends the existing boosting algorithm such as AdaBoost [22], and generates models in the following form.

$$H_P = \sum_{i=1}^{n} \alpha_i h_P^{(i)}, \tag{3}$$

where $h_P^{(i)}, i = 1..n$, are the models *learned from the perturbed data with special base learners*. Note that the parameters α_i are exposed in our approach, which, however, does not affect the confidentiality of the model H_P - without knowing $h_P^{(i)}$, H_P cannot be used.

Applying Learned Models. There are two methods to apply the learned models that are in a protected form H_P. Let D_{new} be the new dataset. (1) Transforming the new data with the same parameters: $P_{new} = P(D_{new})$ and then applying $H_P(P_{new})$. This method is useful when the testing procedure has to be done in the cloud. (2) Recovering the model in the original space: $Transform(H_P) \rightarrow H'$, H' works in the original data space, and then applying $H'(D_{new})$. The model recovering approach seems more cost-effective for the data owner. It is done once for applying all new data later. However, sometimes the model cannot be easily recovered. For example, a linear model in the RASP-perturbed space corresponds to a non-linear model in the original space, which is difficult to recover due to the nonlinear OPE transformation.

4.2 Security Modeling

Clearly, this outsourced learning scenario involves with two parties: the data owner and the service provider who is either the cloud infrastructure provider or the data mining service provider. Because we consider only the confidentiality of data and model, it is appropriate to assume that the service provider is a honest-but-curious party [23], who aims to breach the data owner's privacy but honestly provides services. This assumption is practical for real-world service providers.

There are two levels of adversarial prior knowledge. (1) If the user only uses the cloud infrastructure for mining, we can safely assume the adversaries know only the perturbed data, corresponding to the ciphertext-only attack in crypto-analysis [23]. (2) In the case of data-mining services, we also assume the adversaries know the feature distributions, as such information might be provided for model analysis or exposed via other channels to the service provider. We exclude the case of insider attacks, e.g., an insider on the client side colludes with the adversary and provides perturbation parameters or original unperturbed data records. In general, we consider the level-2 knowledge to allow the results applicable to both scenarios of outsourced mining.

The curious service provider can see the outsourced data, each execution step of the mining algorithm, and the generated model in the protected form. However, the adversary should not have any statistically meaningful estimation of the original data and actual models.

Data Confidentiality Data confidentiality refers the confidentiality of each value in the dataset. We model an attack as a data estimation procedure using the prior knowledge and the perturbed data to reconstruct the original data. We define data confidentiality as the accuracy of estimation for any specific original value. It can be defined quantitatively by the existing statistic measures such as mean-squared-error (MSE) method between the original values and the estimated values. Let $\{\hat{v}_i, i = 1..N\}$ is a series of estimated values for the *normalized* original values $\{v_i, i = 1..N\}$. MSE $= 1/N \sum_{i=1}^{N} (\hat{v}_i - v_i)^2$. Let e_t be an estimation method in the set of all possible methods $\mathcal{E}$. Then, the level of preserved confidentiality can be evaluated by the measure $\zeta = \min_{e_t \in \mathcal{G}} MSE_{e_t}$. Normalization allows the metric to be used crossing dimensions and datasets.

The $\{v_i\}$ and $\{\hat{v}_i\}$ series can be considered as samples from the random variables X and $\hat{X}$, respectively. Thus, MSE is the mean $E(X - \hat{X})^2$, with which we have

Proposition 1. *A random guess attack with the level-two prior knowledge gives $2var(X)$ to the MSE-based confidential measure.*

Proof: Note that the variance of $X - \hat{X}$ is $var(X - \hat{X}) = E(X - \hat{X})^2 - E^2(X - \hat{X})$ by definition of variance. Thus, we have $E(X - \hat{X})^2 = var(X - \hat{X}) + E^2(X - \hat{X})$. Since $\hat{X}$ and X have the same distribution and random guess makes them independent, we have $var(X - \hat{X}) = 2var(X)$ and $E(X - \hat{X}) = 0$. □

This gives the meaningful upper bound of the measure, or the *inherent confidentiality* for a given distribution. It is quite intuitive - for a known distribution with small variance, the attacker can always get good estimation by random guess.

Let ϵ_0 be some user-defined small positive threshold so that the user can tolerate the confidentiality loss in the sense that the best possible attack can only result in $2var(x) - \epsilon < \zeta$. We consider data confidentiality is satisfactorily protected: if any estimation attack in polynomial time complexity will result in $\zeta > 2var(x) - \epsilon$ for each dimension or any possible attack will take exponential time complexity, which is computationally impractical.

Model Confidentiality. The goal of protecting model confidentiality is to protect from unauthorized use of models. Thus, it is directly related to model utility. We define model confidentiality as a function of two factors: the *unknown* model parameters and the impact of the *estimated* unknown parameters on model utility. We assume the type of model is always known since the learning procedure is exposed to the adversary.

Let p represent the m unknown parameters, and $\mathcal{U}(f)$ be the utility of the model f, which is accuracy in classification modeling. Let $\mathcal{U}(f|p, \mathcal{D})$ be the average model utility if p is randomly drawn from a distribution $\mathcal{D}$. The impact of parameter is defined as the reduction of model utility $c(f, p, \mathcal{D}) = (\mathcal{U}(f) - \mathcal{U}(f|p, \mathcal{D}))/\mathcal{U}(f)$ under the estimated distribution $\mathcal{D}$ (i.e., the attack).

We use $c(f, p) = \min\{c(f, p, \mathcal{D}_i)\}$ for all possible attacks $\mathcal{D}_i$ to define model confidentiality with the following intuitive understanding. (1) The unknown well-protected key parameters should be enough to make the whole model useless. (2) The insignificant parameters, known or not, may not affect overall model utility. The upper bound of this measure is given by a random-guess based model, which gives $\mathcal{U}(f|p, \mathcal{D}) \approx 0.5$ for a two-class problem. The specific upper bound should be determined by the optimal model utility $\mathcal{U}(f)$, which is specific to each dataset. Let η be this upper bound, and ϵ be some user-defined tolerance threshold for confidentiality breach. If $c > \eta - \epsilon$, we consider the model confidentiality is well protected.

Process Privacy. In addition to the data and model confidentiality, the learning process (the cloud-client protocols) should not affect data and model confidentiality. We define process privacy as the learning process not providing any additional information to what the adversary have already known.

5 Core RASP-Boost Algorithms

In the RASP-Boost framework, the key is the algorithms that can learn base learners from the perturbed data. We categorize the algorithms into two categories: the pool based and the seed based. For each category, we will investigate two types of base classifiers: random decision stumps and random linear classifiers.

To understand the basic working mechanism of the developed algorithms, we start with the basic boosting framework and map it to the setting of cloud-client collaborative learning. Algorithm 1 shows the boosting procedure in our approach. We map each step to the cloud or the client as shown in the comment. $I(p_i == t_i)$ is the indicator function, which returns 1 if the condition $p_i == t_i$ is true, otherwise returns 0. This framework enables two key features: (1) the perturbed data allows model evaluation and comparisons to be done independently by the cloud; (2) the testing procedure is also independently done by the cloud. These features maximize the use of cloud and eliminate the cloud-client interactions in the iterations.

Algorithm 1 Cloud-Client Boosting on Perturbed Data

```
1: $N$: the number of records, $\omega_{i,k}$: the weight for the record
   $i$ in $k$-th iteration, $n$: the number of iterations, $y_i$: the
   perturbed records in the cloud
2: $\omega_{i,0} \leftarrow 1/N, i = 1..N$; //by cloud
3: prepare and send a set of base classifiers $\mathcal{H}_0$ encoded and
   protected with perturbation parameters; // by client
4: extend $\mathcal{H}_0$ to a large set $\mathcal{H}_1$ with some algorithms (op-
   tional) // by cloud
5: for $k$ from 1 to $n$ do
6:     search $\mathcal{H}_1$ to find a base classifier $h_k(y)$ that minimizes
       the weighted error with weights $\{\omega_{i,k}, i = 1..N\}$; // by
       cloud
7:     apply $p_i = h_k(y_i)$ to each record $y_i$ and generate the
       prediction $\{p_i, i = 1..N\}$; // by cloud
8:     compute the weighted error rate $\epsilon_k =$
       $\sum_{i=1}^{N} \omega_{i,k-1} I(p_i == t_i)$; // by cloud
9:     $\alpha_k \leftarrow 1/2ln\frac{1-\epsilon_k}{\epsilon_k}$; // by cloud
10:    $\omega_{i,k} \leftarrow \omega_{i,k-1} \exp^{-t_i \alpha_k h_k(x_i)}$, and $Z = \sum_{i=1}^{N} \omega_{i,k}$; //
       by cloud
11:    normalize $\omega_{i,k}$ by $\omega_{i,k} \leftarrow \omega_{i,k}/Z$; // by cloud
12: end for
13: repeat the above procedure for different parameter set-
    tings, such as $n$; // by cloud
14: download $\{\alpha_i, h_i(), i = 1..n\}$; //by client
```

The key steps include (1) Step 3: the client prepares and sends a set of encoded base classifiers, (2) the optional Step 4: the cloud extends the set with an algorithm, and (3) Step 6: the cloud works with the pool of encoded base classifiers to find the base classifier h_k that works reasonably well on the weighted examples.

The original AdaBoost algorithm [22] in each iteration will search for one classifier h_k that minimizes the weighted error rate ε for N examples in a family of weak classifiers $\mathcal{H}$. Specifically, it is defined as

$$h_k = \arg\max_{h_j \in \mathcal{H}} \epsilon_j, \text{where } \epsilon_j = 1/N \sum_{i=1}^{N} \omega_{i,k} I(h_j(y_i)! = t_i).$$

The search space $\mathcal{H}$ is often limited, for example, the entire set of decision stumps for all dimensions.

However, in the RASP-Boost framework the client needs to encode and transfer the set $\mathcal{H}$ to the cloud, which is prohibitively expensive for a large set such as the entire set of decision stumps. Instead, we let the client prepare a small pool of base classifiers, and the cloud tries to find an acceptable one from the pool in each iteration. Theoretically, this method still works because the boosting framework requires only weak base classifiers. The major problem is how the candidates should be selected and how large the pool should be to avoid the situation that all candidates in the pool give ≈ 0.5 weighted error rate (for two-class problem) in certain iteration, which will significantly reduce the model quality.

In the following, we will present the key idea of encoding the base classifiers for RASP-perturbed data. Then, we develop several algorithms for the client to generate the pool of base classifiers and for the cloud to (optionally extend and) search the pool.

5.1 Query-based Linear Classifiers.

As we have shown in Section 3, half-space queries can be transformed to the RASP perturbed space and processed on the perturbed data. Specifically, a half-space condition like $X_i < a$ is transformed to the condition $y^T Q y < 0$. This query transformation has to be done by the client, as the perturbation parameters will be used to generate Q. With the transformed query, it is possible to count the number of '+1' and '-1' examples on a half plane as the following query shows.

select count(t=="+1"), count (t=="-1")
from P={y_i=RASP(x_i), t_i}
where $y^T Q y < 0$.

Similarly, with the other half-space condition $y^T Q y \geq 0$ we can get the counts for the other half of the dataset. Based on these numbers, a classifier can be designed, for example, as

$$f(y) = y^T Q y \begin{cases} < 0, \text{return prediction} - 1 \\ \geq 0, \text{return prediction} + 1. \end{cases} \quad (4)$$

It is straightforward to derive the prediction error, based on the number of disagreements between the predictions and the labels. If the prediction error $> 50\%$, the prediction rule is reversed. The classification rule based on a single dimension such as $X_i < a$ is traditionally called *Decision Stump* (DS).

The method for defining our DS classifiers can be extended to general linear classifiers (LC) defined in the space $z_i = (E_{OPE}(K_{OPE}, x_i)^T, 1, v_i)^T$ as we have described in Section 3. A general linear query in OPE transformed d-dimensional space $s_i = E_{OPE}(K_{OPE}, x_i)$, $g(s) = w^T s + b$, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$, can be equivalently represented as $f(z) = u^T z < 0$, with u in the form of $(w^T, -b, 0)^T$, with which we can then derive the query matrix $Q = (A^{-1})^T u v^T A^{-1}$ with the v vector defined previously in Section 3.

In the following, we will design algorithms to generate DS and LC classifiers in the perturbed space, aiming to minimize the client-side costs and maximize model quality.

5.2 Pool-Based Algorithms

In this set of algorithms, the client generates a pool of randomly selected linear classifiers, based on only the dimensional distribution of the training data. The cloud will select one from the pool. We will discuss two methods for the client to generate the pool, and the method for the cloud to utilize the pool.

Random Decision Stump Pool (DSPool). In this method, the client randomly selects a set of decision stumps to encode and transfer. The key problem is to select *effective decision stumps* that shatter the major population of the records. Since a decision stump in the original space is mapped directly to another decision stump in the OPE transformed space, we can work with the OPE space directly to simplify the encoding procedure. The OPE space has each dimension in a normal distribution. For $E(X_i) < E(a)$, if we draw $E(a)$ from the normal distribution $N(\mu_i, \sigma_i^2)$, we have great chance to shatter the major population well. Specifically, the dimension will be randomly selected, and then the splitting value is drawn from the corresponding normal distribution. We will study in experiments how the size of the pool affects the cloud-side learning result.

Random Linear Classifier Pool (LCPool). Similar to decision stumps, we can also generate random linear classifiers $f(z) = u^T z$ in the space $z_i = (E(x_i)^T, 1, v_i)^T$, with randomly generated $u = (w^T, -b, 0)^T$. As there are an unlimited number of linear classifiers, the problem is again to appropriately sample them to get the relevant ones into the pool.

The basic idea is to find the hyperplanes that shatter around the center of the dataset. Because the OPE transformed dimensions have normal distributions, we can imagine the records projected to these d dimensions are distributed in an ellipsoid. Figure 2 shows the two-dimension, with each dimension in standard normal distribution. The majority of the records is inside the hyper-sphere of radius $\gamma = 2$ with the center at (0, 0).

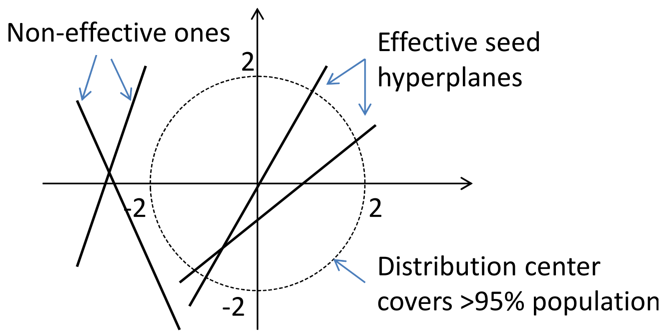
Non-effective ones
Effective seed hyperplanes
2
-2
2
-2
Distribution center covers >95% population

Fig. 2. Effective hyperplanes shatter around the distribution center.

Thus, to shatter in the major population, we need to have the distance between the center o and the hyperplane less than some radius γ, which gives

$$\frac{|w^T o + b|}{\|w\|} < \gamma, \quad (5)$$

where $\|w\|$ is the length of the vector w, and $|b|$ is the absolute value of b. γ can be set to the minimum dimensional standard deviation. To satisfy this condition, we can simply generate a unit-length random vector w, and then choose a random value b so that $|w^T o + b| < \gamma$. Intuitively, the smaller the $|w^T o + b|$ value, the closer the hyperplane will be to the center. The client randomly generates a set of such hyperplanes as the pool and send to the cloud. Again, we will investigate how the pool size affects the learning result.

Cloud-side Processing. The cloud side will search the pool to find the best one that gives the lowest weighted error rate. Depending on the data distribution and the randomly generated candidates, there is a small probability that all the base learners in the pool give weighted error rates $\sim 50\%$. This probability will exponentially decrease with the increasing pool size. We will study the lower bounds of pool size for different datasets in experiments.

5.3 Seed-based Algorithms

In the pool based algorithms, the client needs to generate a set of randomly selected base classifiers, encode them, and send to the cloud. It might be costly, e.g., with hundreds of encoded classifiers. In the following, we consider reducing the client's work further by using the seed-based algorithms. The client will send a few randomly selected "seed classifiers", and the cloud will generate a pool based on these seeds. The following algorithms depend on *the linearity property of the query matrix Q*.

Derived Decision Stumps (DerivedDS). The query matrices on the *same dimension* has the following linearity property.

Proposition 2. *For half-space conditions on the same dimension, say $X_i < a$ and $X_i < b$, a linear combination of the corresponding query matrix Q_a and Q_b: $Q_a + \tau(Q_b - Q_a)$, $\tau \in \mathbb{R}$, is the query matrix of some condition $X_i < c$ on the same dimension.*

Proof: For the condition $X_i < a$, we have the corresponding query $y^T Q_a y < 0$ in the perturbed space, where $Q_a = (A^{-1})^T u_a v^T A^{-1}$, $u_a^T = (w, -E(a), 0)$ and $v^T = (0, \ldots, v_0, 1)$ (Section 3).

Similarly, $Q_b = (A^{-1})^T u_b v^T A^{-1}$, where $u_b^T = (w, -E(b), 0)$. Therefore, it follows

$$Q_a + \tau(Q_b - Q_a) = (A^{-1})^T (u_a + \tau(u_b - u_a)) v^T A^{-1} \quad (6)$$

where $u_a + \tau(u_b - u_a) = (w, -E(a) - \tau(E(a) - E(b)), 0)$. According to the definition of OPE, the value $E(a)+\tau(E(b)-E(a))$ in the encrypted domain must correspond to a value in the original domain of X_i. Thus, $Q_a + \tau(Q_b - Q_a)$ is the query matrix of some condition $X_i < c$. □

Note that since τ can be any real value, $E(a) + \tau(E(b) - E(a))$ can be any value in the OPE domain if $E(b) \neq E(a)$. Therefore, with two randomly picked seed decision stumps on the same dimension, we can derive all decision stumps on the same dimension. However, not all of these decision stumps are effective for our use. Again, we hope the result will shatter around the center of the population (e.g., in the range $(\mu{-}2\sigma, \mu{+}2\sigma)$ for the normalized data). We can achieve this goal by setting the seeds around the bounds $[-\gamma, \gamma]$ on the OPE domain, where γ can be some value in $(0, \sigma)$ and τ in the range $(0, 1)$. With such a setting, we have $|E(a) + \tau(E(b) - E(a))| \le |(1-\tau)E(a)| + |\tau E(b)| < 2\sigma$ to shatter around the center of the population.

Derived Random Linear Classifier (DrivedLC). The linearity property of query matrices can be extended to general linear classifiers on the OPE space.

Proposition 3. *Assume a set of query matrices $\{Q_i, i = 1..m\}$ encode the general linear functions $f_i(s) = w_i^T s + b_i$ in the OPE space, respectively. Then, $\sum_{i=1}^m \tau_i Q_i$, where $\tau_i \in \mathbb{R}$, represents a valid general linear function in the OPE space.*

Proof: Similarly, we have $\sum_{i=1}^m \tau_i Q_i = (A^{-1})^T (\sum_{i=1}^m \tau_i u_i) v^T A^{-1}$, where $u_i = (w_i^T, -b_i, 0)^T$. Apparently, $\sum_{i=1}^m \tau_i u_i = (\sum_{i=1}^m \tau_i w_i, -\sum_{i=1}^m \tau_i b_i, 0)$ is a valid parameter vector for a general linear function in the OPE space. □

Two key problems are to be addressed. First, we want the generated hyperplane to shatter around the major population. For simplicity of presentation, we assume all the dimensions have the center on 0. As we have discussed, the condition $|\sum_{i=1}^m \tau_i b_i| / \|\sum_{i=1}^m \tau_i w_i\| < \gamma$ should be satisfied. However, $\|\sum_{i=1}^m \tau_i w_i\|$ can be a very small value close to 0, which dissatisfies the condition. Second, the random combination $\sum_{i=1}^m \tau_i w_i$ represents the direction of the generated hyperplane, which should be able to cover as many possible directions as possible. We have the following result to address these problems.

Proposition 4. *Let $\{w_i, i = 1..d\}$ be d random orthonormal vectors and $(\tau_1, \ldots, \tau_d)$ be a random unit vector, i.e., $\sum_{i=1}^d \tau_i^2 = 1$, and $|b_i| < \gamma/d$. This condition*

$$|\sum_{i=1}^m \tau_i b_i| / \|\sum_{i=1}^m \tau_i w_i\| < \gamma$$

is satisfied.

Proof: The proof follows the logic that if $\|\sum_{i=1}^m \tau_i w_i\| = 1$ and $|\sum_{i=1}^m \tau_i b_i| < \gamma$ then the condition is satisfied. Let R be an orthonormal matrix, i.e., $R^T R = I$. $\{w_i\}$ can be considered as the rotational transformation of the standard basis $\{e_i, i = 1..d\}$: $w_i = Re_i$, where all elements of e_i are 0 except for the i-th set to 1. Also, the transformation Rw for any vector w preserves the length of w, i.e., $\|Rw\| = \|w\|$. Therefore, $\|\sum_{i=1}^d \tau_i w_i\| = \|\sum_{i=1}^d R\tau_i e_i\| = \|\sum_{i=1}^d \tau_i e_i\| = \sqrt{\sum_{i=1}^d \tau_i^2} = 1$.

Meanwhile, since $|\tau_i| \le 1$, if $|b_i| < \gamma/d$ we have $|\sum_{i=1}^d \tau_i b_i| \le \sum_{i=1}^d |\tau_i||b_i| \le \gamma$. □

It is also straightforward to prove that any unit vector w can be represented with the above combination method $\sum_{i=1}^d \tau_i w_i$, where $\sum_{i=1}^d \tau_i^2 = 1$. It implies that this combination method can generate hyperplanes in any direction.

Note that the d random orthogonal vectors $\{w_i\}$ can be easily obtained by applying QR decomposition of a random invertible $d \times d$ matrix. A random unit vector τ can be obtained from a random vector r by $\tau \leftarrow r/\|r\|$. Thus, it is easy for both the client to generate the seed vectors and the cloud to generate the random combinations.

Cloud-side Processing. The cloud side will randomly generate a batch derived decision stumps or linear classifiers to find the best one. Similar to the pool based algorithms, the problem is the appropriate number of random trials. We will investigate this problem in experiments.

5.4 Cost Analysis

The cost in the whole learning procedure consists of three parts: the cost of cloud-side processing, the amount of data transferred to the cloud, and the cost of client-side processing. Excluding the initial cost of preparing and uploading the perturbed data, we are more interested in the client-side costs of preparing the base classifiers and transferring them to the cloud.

According to the Equation of computing Q, the cost of preparing one base classifier is about $O((d+2)^2)$ and the size of Q is also $O((d+2)^2)$. So the key factor is really how many base classifiers the specific algorithm needs to encode and transfer. Let p be the pool size for the pool-based algorithms. Table 1 shows the costs of the four algorithms. Note that these costs are only determined by the dimensionality d and the pool sizes p, not by the size of the dataset N or the number of boosting iterations n. Thus, it is favorable to big data with large N. For most datasets, d is less than 1000 and the client-side costs are low. However, these algorithms will be too expensive for very high dimensional datasets to be practical - e.g., a text mining dataset using words as the dimensions often results in $d > 10000$.

The cloud-side processing cost is determined by the number of RASP queries (i.e., the base classifier processing) issued in the learning process. Each RASP query is processed by using the condition $y^Q y < 0$ to scan the whole dataset, resulting in $O((d+2)^2 N)$ complexity. For each of k iterations in boosting learning, the cloud will search the pool of p client-generated queries to find the best one or p cloud-derived random queries to find a valid one. A query is simply a linear combination of queries in the pool, which has $O((d+2)^2)$ complexity for decision stumps, and $O(d(d+2)^2)$ for linear classifiers. Thus, if the pool has a size of p, the first option has the total cloud-side cost $O(k(d+2)^2 N)$, while the second $O(kp(d+2)^2 N)$.

5.5 Confidentiality Analysis

Confidentiality guarantee consists of several parts: the confidentiality of perturbed data in the cloud, the confidentiality of queries and the learning process, and the confidentiality of generated models. We discuss them separately in the following.

5.5.1 Data Confidentiality

Data confidentiality has been discussed in our paper on the RASP approach for outsourced databases [5]. We include the key points here to make the paper self-contained. According to the threat model, the attacker may know only the perturbed data, i.e., the first level of prior knowledge, or the distribution of each dimension, i.e., the second level, which corresponds to the brute-force attack, and the ICA attack, respectively.

To conveniently represent the complexity of attacks, we assume each value in the vector or matrix is encoded with or converted to n-bit integers. Let the perturbed vector y be drawn from a random variable $\mathcal{Y}$, and the original vector x be drawn from a random variable $\mathcal{X}$. The corresponding matrices are X and Y.

Brute-Force Attack. This attack will examine each possible original matrix X according to the known Y. We show that this process is computationally intractable. The goal is to show the number of the valid X dataset in terms of a known perturbed dataset Y. Below we discuss a simplified version that contains no OPE component - the OPE version has at least the same level of security.

Proposition 5. *For a known perturbed dataset Y, there exists $O(2^{(d+1)(d+2)n})$ candidate X datasets in the original space to be examined.*

Proof: For a given perturbation $Y = AZ$, where Z is X with the two extended dimensions, we have $A^{-1}Y = Z$. Let $B = A^{-1}$ and B_{d+1} represent the $(d+1)$-th row of A^{-1}. We have $B_{d+1}Y = [1, \ldots, 1]$, i.e., the appended $(d+1)$-th row of Z. Keeping B_{d+1} unchanged, we randomly generate other rows of B for a candidate $\hat{B}$. The result $\hat{Z} = \hat{B}P$ is a validate estimate of Z if $\hat{B}$ is invertible. Thus, the number of candidate X is the number of invertible $\hat{B}$.

The total number of $\hat{B}$ including non-invertible ones is $2^{(d+1)(d+2)n}$. Based on the theory of invertible random matrix [24], the probability of generating a non-invertible random matrix is less than $\exp^{-c(d+2)}$ for some constant c. Thus, there are about $(1 - \exp^{-c(d+2)})2^{(d+1)(d+2)n}$ invertible $\hat{B}$. Correspondingly, there are a same number of candidate X. $\square$

Thus, examining all possible X is computationally intractable, and the brute-force attack is impractical.

ICA Attack. With the known distributional information, the attacker can do more on estimating the original data than simple. The known most relevant method is called Independent Component Analysis (ICA) [25]. For a multiplicative perturbation $Y = AX$, the fundamental method [25], [26] is to find an optimal projection, wY, where w is a $d+2$ dimension row vector, to result in a row vector with its value distribution close to that of one original attribute. This goal is approximately achieved by examining the *non-gaussianity*[2] characteristics of the original distribution - finding the projections by maximizing the non-gaussianity of the result wY. The non-gaussianity of the original attributions is crucial because any projection of a multidimensional normal distribution is still a normal distribution, which leaves no clue for recovery.

To simplify the proof, we assume the original dimensions are independent[3]. We have the following result.

Proposition 6. *There are $O(2^{dn})$ candidate projection vectors, w, that lead to the same level of non-gaussianity.*

Proof: First, we show that Y has a multidimensional normal distribution. As the d original dimensions are independent, the generated OPE dimensions are independent of each other with high probability. The additional noise dimension is also independently generated. Thus, we consider grouping all the independent $d+1$ dimensions together as the submatrix Z_1. Z_1 contains the sample vectors from a $d+1$-dimensional normal distribution $N(\mu, \Sigma)$, where μ is the mean and Σ is

2. Non-gaussianity means the distribution is not a normal distribution.

3. If not, we can use *rotation transformation* to de-correlate the dimensions. Previous studies [27] show that this does not affect classification modeling results for geometry-based methods including the base classifiers we use.

Method	Client Computation	Client-¿Cloud Transfer
DSPool	$O(p(d+2)^2)$	$O(p(d+2)^2)$
LCPool	$O(p(d+2)^2)$	$O(p(d+2)^2)$
DerivedDS	$O(2d(d+2)^2)$	$O(2d(d+2)^2)$
DerivedLC	$O(d(d+2)^2)$	$O(d(d+2)^2)$

TABLE 1
Client-side costs of the four methods.

the covariance matrix. Thus, the RASP transformation can be represented as $Y = (A_1, A_2)(Z_1^T, 1)^T$, where A_1 is the first $d+1$ columns of the A matrix, A_2 is the last column, and $\not{1}$ is the added constant row of 1. It follows $Y = A_1 Z + A_2 \cdot 1$. According to the basic property of multidimensional normal distribution, $A_1 Z$ contains samples from the multidimensional normal distribution $N(A_1\mu, A_1\Sigma A_1^T)$. $A_2 \cdot 1$ simply adds a constant to each row vector (i.e., a dimension) of $A_1 Z$, which does not change the dimensional distribution. Therefore, Y has a multidimensional normal distribution.

It immediately follows that any projection wY will not change the Gaussianity of the result, and there are $O(2^{dn})$ such candidates of w. □

Thus, enumerating all possible projections and analyzing each is computationally impractical. It shows that any ICA-style estimation that depends on Gaussianity is equally ineffective to the RASP perturbation.

In addition to ICA, Principal Component Analysis (PCA) based attack is another possible distributional attack, which, however, depends on the preservation of covariance matrix [13]. Because the covariance matrix is not preserved in RASP perturbation, the PCA attack cannot be used on RASP perturbed data. It is unknown whether there are other distributional methods for approximately separating X or A from the perturbed data Y, which will be studied in the ongoing work.

5.5.2 Query and Process Privacy

Queries in the proposed algorithms consist of two parts: the original queries generated by the client, and the derived queries by the cloud using the seed-based query derivation algorithms, for which we need to check whether the derivation algorithm gives additional information.

According to the threat model, the attacker does not have any additional prior knowledge about queries except for the known matrix Q. Now, the task is to find the decomposition of a query matrix $Q = (A^{-1})^T uv^T A^{-1}$ to figure out any information about A^{-1} and u (since v is constant, we consider it is known by the public). We show that a stronger attack with additional knowledge of u is still computationally intractable.

Proposition 7. *With known u, there are about $O(2^{(d-1)(d+2)n})$ valid guesses of A that result in the same query matrix Q.*

Proof: Without loss of generality, we can assume that Q encodes a condition $X_i < a$. Let r_i be the i-th row of A^{-1}. Q is represented as $(r_i - E(a)r_{d+1})^T(r_{d+2} - v_0 r_{d+1})$. Note that Q is only determined by the three rows of the matrix A^{-1}. The remaining $d-1$ rows are free to choose, leading to $2^{(d-1)(d+2)n}$ valid candidates of A^{-1}, among which $O(2^{(d-1)(d+2)n})$ are invertible [24]. □

This shows a lower bound of the difficulty in attacking the Q matrix without any additional information.

The next problem is whether the cloud-side algorithms will breach additional information. Obviously, since all the algorithms either simply search the pool or use purely random combinations of existing query matrices, there is no additional information is leaked.

5.5.3 Model Confidentiality

As we have discussed in Section 4, model confidentiality is defined as $c(f, p) = \min\{c(f, p, \mathcal{D}_i)\}$, where f is the learned model, p are the unknown parameters, $\mathcal{D}_i$ is the p distribution estimated by an attacker, $c(f, p, \mathcal{D}_i)$ is the model utility reduction under the attack. The smaller the reduction, the more effective the attack.

For classification modeling, we use the accuracy on the testing data T as the model utility. In a boosting model $f(x) = \sum_{i=1}^{n} \alpha_i h_i(x)$, the model parameters consist of n, $\{\alpha_i\}$, and the parameters in $h_i(x)$. If $h_i(x)$ are decision stumps, the parameters include the selected dimension, the splitting value, and the direction: $X_j < a$ or $X_j \geq a$. For linear classifiers on the OPE space, w, b, and the direction: $w^t x + b < 0$ or $w^t x + b \geq 0$ are the parameters of the base model.

The proposed learning algorithms will expose the parameters n and $\{\alpha_i\}$, but keep all parameters in $h_i(x)$ secret. Because the query privacy is fully preserved, the privacy of the base models $h_i(x)$ is preserved. Thus, the corresponding parameters of decision stump or linear classifier are the unknown parameters. As we have discussed, under our security assumption the only known attack on the query matrix is the brute-force attack, which, however, is computationally intractable. Since $h_i(x)$ (especially for smaller i) are significant to the model, we expect the model confidentiality is preserved well. In experiments, we will further explore the concept of model confidentiality.

6 Experiments

The previous sections have addressed several major aspects: the cloud-client algorithms, the client-side cost analysis, and the confidentiality analysis. The experiments will study how the cloud-client algorithms perform in terms of different settings that may also involve the tradeoff between model quality and client-side costs. Specifically, (1) We will show the scalability of our approach with client-side computation and communication costs on real datasets. (2) We will conduct a set of experiments to understand which of the four private learning methods is the best in terms of costs and model quality. (3) We will evaluate model confidentiality with the proposed method.

Dataset	Records	Dimensions	Link
German Credit	1000	20	https://goo.gl/lVy34O
Ozone Days	2536	73	https://goo.gl/Si6aDh
Spambase	4601	57	https://goo.gl/WPyXTi
Bank Marketing	45211	17	https://goo.gl/vvgj3M
Twitter Buzz	140000	77	https://goo.gl/Yfy80u

TABLE 2
Datasets for experiments.

	DSPool	LCPool	DerivedDS	DerivedLC
German Credit	1.16	1.16	0.15	0.08
Ozone Days	13.50	13.50	6.57	3.29
Spambase	8.35	8.35	3.17	1.59
Bank Marketing	0.87	0.87	0.10	0.05
Twitter Buzz	15.00	15.00	7.69	3.84

TABLE 3
Average Communication Costs (Megabyte).

6.1 Experiment Setup

Datasets. For easier validation and reproducibility of our results, we use a set of public datasets from UCI machine learning repository for evaluation, each of which has only two classes. These datasets have been widely applied in various classification modeling and evaluation. Table 2 lists the statistics of the datasets. They cover different scales and dimensions to make the results more representative. In preprocessing, each dimension of the datasets is normalized with the transformation $(v - \mu_j)/\sigma_j$, where μ_j is the mean and σ_j^2 is the variance of the dimension j.

Implementation. We implement the perturbation methods based on the algorithms in the corresponding papers [5]. The RASP-Boost framework is implemented based on the AdaBoost algorithm [22]. The four learning algorithms are implemented as plugins to the framework. All these implementations use C++ and are thoroughly tested on a Ubuntu Linux server. We also used the Scikit-Learn toolkit to generate the AdaBoost baseline accuracy based on the original datasets for comparison.

6.2 Experimental Result

Client-side Costs of Preparing Base Learners. Data owners in our framework are more concerned with the costs in the client side. Section 5.4 has given a formal analysis on the client-side costs, which are mainly determined by the dimensionality of the dataset. We conduct a simple evaluation to show the real costs for different algorithms in the RASP-Boost framework.

The client-side costs include those generating the transformed queries and transferring them to the service provider. The time complexity of the pool-based algorithms is about $O(pd^2)$, where p is the pool size, and d is the dimensionality. The two seed-derived algorithms has $O(d^3)$. Since both d (< 100) and p (a few hundred) are not large for all the datasets, we skip the client-side computation cost.

Table 3 shows the amount of data transferred for different algorithms and different datasets in uncompressed format. Each element in the query matrix is encoded with an 8-byte double type. We assume the pool-based algorithms need 300 base classifiers in the pool (a detailed discussion on this number setting will be given later). The pool-based algorithms typically cost more than the seed-based algorithms. Overall, these costs are pretty low.

Progressive Error Rates in Boosting. We try to understand how these algorithms perform in terms of the model quality. First, we look at the progressive boosting result, which provides valuable information on their convergence rate and final model quality. Spambase is used as it is not very small and with many dimensions. We use each of the four proposed methods to train a boosting model with 500 base classifiers in five-fold cross-validation. Again, the pool size is set to 300 for the two pool-based methods, and the seed-based algorithms will try 300 randomly generated candidates to find the best one. The baseline uses the existing AdaBoost implementation with decision stumps provided by Scikit-Learn. In each iteration, the pool-based algorithms will try to find the classifier in the pool that works best on the weighted examples (i.e., giving the lowest weighted error rate). For every 20 iterations, we test the boosted classifier to get the progressively reduced error rates, which are shown in Figure 3. The error bars are skipped for a clear presentation.

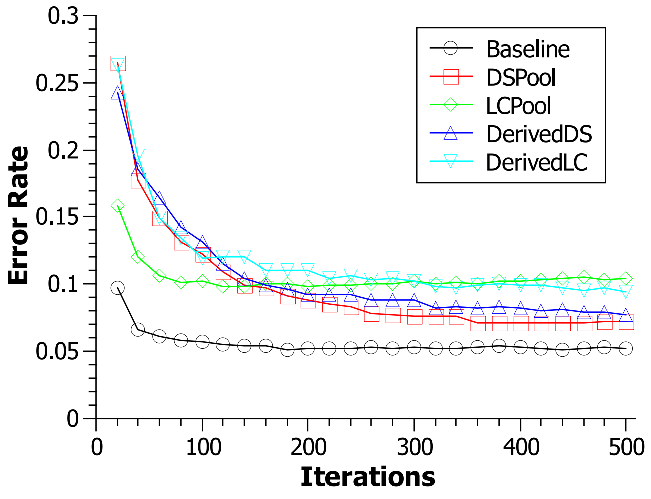

Fig. 3. Progressive testing error rates on Spambase.

The result shows that the decision-stump based algorithms: DSPool and DerivedDS work better than linear-classifier based methods. A possible reason is that linear classifiers have higher degrees of freedom that reduces the chance of shattering the population nicely, especially in high-dimensional spaces, where the examples are sparsely distributed. The two DS algorithms also have results very close to the baseline of original AdaBoost with decision stumps and the differences are not statistically significant.

Pool Size and Number of Random Trials. Another key question is the appropriate pool size for the pool-based algorithms, or the number of randomly generated candidates for the seed based algorithms. In the pool-based algorithms, in each iteration the cloud-side will find the best one with the lowest weighted error rate in the pool. In the seed-based algorithms, the cloud-side will try a number of random combinations of the seeds to find the best one. Since the two work similarly

	DSPool	LCPool	DerivedDS	DerivedLC
German Credit	200	150	50	50
Ozone Days	150	150	150	150
Spambase	150	150	150	150
Bank Marketing	100	150	50	50
Twitter Buzz	200	200	200	200

TABLE 4
The lower bounds of the pool size for the pool-based algorithms, or the number of random trials rated candidates for the seed-based algorithms.

and have similar meaning for the corresponding algorithms, we discuss them together. This problem is studied in two aspects: (1) the lower bound of the size, and (2) the impact of the increasing size.

We consider the lower bound is the size that the boosting framework can find a meaningful base classifier from the pool (or the number of trials) in each iteration. To study this problem, we start with the size 50 and then progressively increase the size by 50 to probe the valid lower bounds for each dataset and each cloud-side learning algorithm. Specifically, in each algorithm, we use weighted error rate 0.49 as the threshold for meaningful weak learners. A learner with an error rate > 0.49 is considered equivalent to a random guess. If all candidates in the pool are equivalent to random guess, we increase the lower bound by 50 to start the next probe.

Table 4 gives the summary of the lower bounds for all datasets. All of them are bounded by a few hundreds. As we have discussed, by carefully designing the pool and the seeds, the probability that all the candidates fail is extremely low.

Next, we try to understand whether increasing the pool size for the pool-based algorithms or the number of random trials for the seed-based algorithms will help the overall performance. This setting of size represents a potential trade-off between the client-side costs and model quality for the pool-based algorithms, and also affect the cloud-side costs in searching the best base classifier.

Again, we use Spambase for example. Table 4 shows that 150 is the lower bound for Spambase to find valid weak classifiers in each iteration. Starting from 150, we gradually increase the size to 350 and observe the performance differences. In Figure 4, we see a significant jump for DSPool from 150 to 200, meaning that 150 is not optimal. However, there is no significant improvement by increasing the size further. The increase also helps LCPool steadily. However, the changes are not significant for the remaining two methods.

Overall Model Quality. Finally, we conduct a comprehensive evaluation on all datasets with 500 boosting iterations and the pool size/random trials set to 300. Five-fold cross-validation is applied. Figure 5 shows the result. Overall, the DS-based algorithms generate results very close to the baseline. They also consistently perform better than the LC-based algorithms, which agrees with our initial observation that LC increases the difficulty to shatter the major population in high-dimensional space.

Model Confidentiality. In Section 5.5.3, we have defined model confidentiality, which is connected to model utility - how much accuracy a specific parameter-estimation based

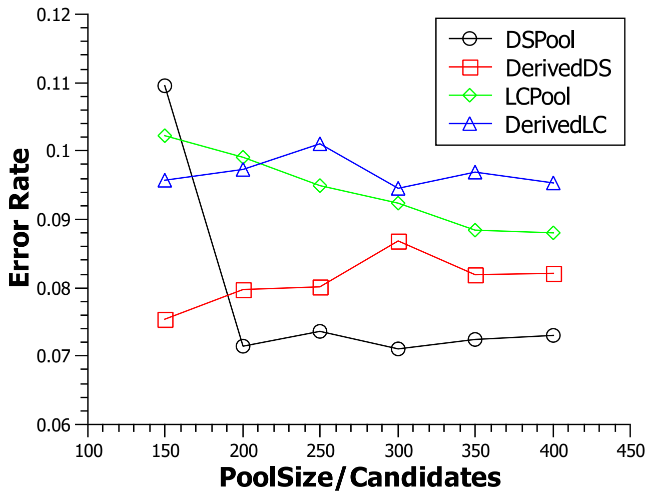

Fig. 4. The effect of pool size for the pool-based algorithms or the number of random trials for the seed-based algorithms.

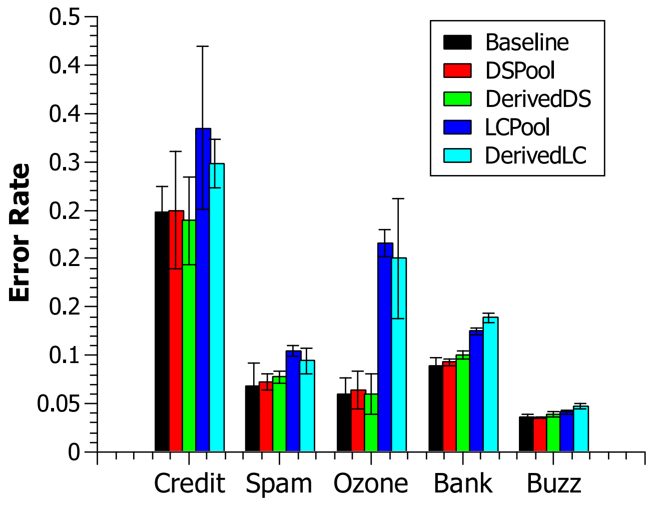

Fig. 5. Model quality for different methods, with the pool size or random trials=300 and the number iterations=500

attack can achieve. Since there is no effective method to estimate the base model parameters (i.e., attacking the query matrices), the only valid attack method is randomly guessing the parameters of base models. Below we show how model confidentiality looks like under the random guess of base models.

We first train models with the setting used in the last experiment, and then replace the base models with the randomly selected DS or LC, corresponding to the type of models. The α weights are kept unchanged. This randomized model is tested on the test data. Five-fold cross-validation is used to estimate the variance of results. Figure 6 shows the model confidentiality, i.e., the percentage of model accuracy reduction. Almost all values are greater than 0.2, which represent more than 20% reduction of the accuracy, basically meaning the estimated models useless.

This result can be better understood compared to the pure random-guess model with the accuracy 50%. Let r be the

	German Credit	Spambase	Ozone Days	Bank Marketing	Twitter Buzz
Optimal Accuracy	0.75	0.93	0.94	0.91	0.96
Upper Bound of Model Confidentiality	**0.33**	**0.46**	**0.47**	**0.45**	**0.48**
DSPool Models	0.32	0.40	0.44	0.40	0.26
DerivedDS Models	0.19	0.43	0.31	0.42	0.38
LCPool Models	0.28	0.39	0.44	0.24	0.46
DerivedLC Models	0.25	0.38	0.38	0.35	0.39

TABLE 5
The theoretical upper bounds of model confidentiality, and the average of model confidentiality for each type of model and dataset.

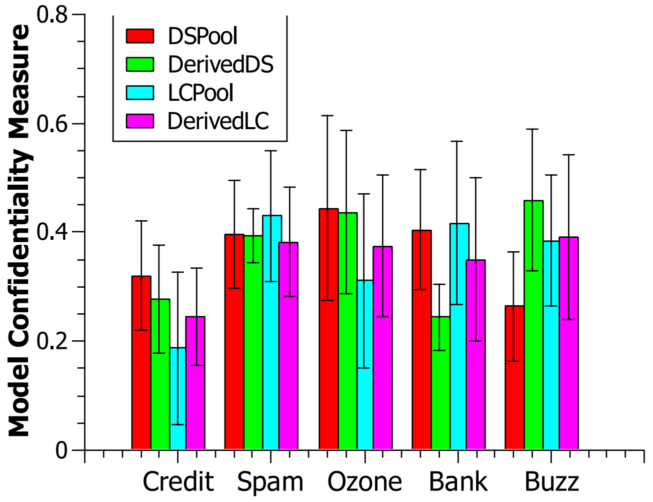

Fig. 6. Model confidentiality under the random guess of the base models.

optimal accuracy of the boosting model. Under the random guess, the accuracy reduction rate is $(r-0.5)/r$, which serves as the theoretical upper bound of the model confidentiality for a specific model. Table 5 gives these upper bounds and compares them to the actual model confidentiality obtained for each model. We found that the most of the actual model confidentiality values are quite close to the upper bounds. It means the random guess of the base classifiers for an RASP-Boosted model is almost as ineffective as a pure random-guess model.

6.3 Discussion

Based on the experimental results, we summarize the features of these methods as follows.

- The DS-based methods are in general better than the LC-based methods. The LC-based methods may work as good as the DS-based methods for some datasets (e.g., the Buzz dataset).
- The two DS-based methods have about the same performance, but the DerivedDS method have advantages of smaller client-side costs, especially for lower dimensional datasets - only two seed queries per dimension.
- Based on the observations on the datasets with dimensionality < 100, $100 \sim 300$ are enough for the pool size for the pool-based algorithms or the number of random trails for the seed-based algorithms. Increasing this size further brings minor benefits.
- Although a part of the model parameters is exposed (i.e., n and $\{\alpha_i\}$), as long as the base models are private, the overall model confidentiality is preserved well.

7 Conclusion

This paper presents the RASP-Boost framework that aims to provide practical confidential classifier learning with the cloud or a third-party mining service provider. Confidential cloud mining should address four aspects: data confidentiality, model confidentiality, model quality, and low client-side costs. We use the RASP perturbation to guarantee the data confidentiality. However, it is difficult to learn a high-quality classifier from the RASP perturbed data, as it only allows linear queries, which can be translated to non-optimal linear classifiers. We develop the boosting based RASP-Boost framework to obtain high-quality classifiers with these non-optimal linear classifiers. The intuition is that boosting requires only weak base classifiers that are slightly better than random guesses.

Four algorithms are developed with the same working pattern: the client provides a set of encoded base classifiers, and the cloud computes a boosting model from the set. This pattern does not require the client to stay online during boosting iterations, which is convenient for the client. We have developed four such algorithms: DSPool, LCPool, DerivedDS, and DerivedLC as the candidates. The confidentiality of data, query, learning process and models is formally analyzed, and we show the confidentiality of data and query is satisfactorily guaranteed.

We have conducted an extensive evaluation of the proposed algorithms and studied the effect of the major factors in the RASP-Boost framework. The result shows that DSPool and DerivedDS can generate high-quality models with accuracy very close to the optimal boosting models. We also evaluate the concept of model confidentiality for real data and models, and show that the model confidentiality is well preserved under the security assumption.

8 Acknowledgment

This material is based upon work partially supported by the National Science Foundation under Grant No. 1245847. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] A. Narayanan and V. Shmatikov, "Robust de-anonymization of large sparse datasets," in *Proceedings of the IEEE Symposium on Security and Privacy*, 2008, pp. 111–125.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of ACM SIGMOD Conference*. Dallas, Texas: ACM, 2000.

[3] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu, "Privacy-preserving data publishing: A survey of recent developments," *ACM Computing Survey*, vol. 42, pp. 14:1–14:53, June 2010.

[4] C. Dwork, "Differential privacy," in *International Colloquium on Automata, Languages andProgramming*. Springer, 2006.

[5] H. Xu, S. Guo, and K. Chen, "Building confidential and efficient query services in the cloud with rasp data perturbation," *IEEE Transactions on Knowledge and Data Engineering*, vol. 26, no. 2, 2014.

[6] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ser. ITCS '12. New York, NY, USA: ACM, 2012, pp. 309–325.

[7] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 35–35.

[8] B. Schneier, "Homomorphic encryption breakthrough," http://tinyurl.com/nlzv4w, 2009.

[9] V. Nikolaenko, S. Ioannidis, U. Weinsberg, M. Joye, N. Taft, and D. Boneh, "Privacy-preserving matrix factorization," in *Proceedings of the 2013 ACM SIGSAC conference on Computer and communications security*. New York, NY, USA: ACM, 2013, pp. 801–812.

[10] K. Chen and L. Liu, "Geometric data perturbation for outsourced data mining," *Knowledge and Information Systems*, vol. 29, no. 3, 2011.

[11] K. Liu, H. Kargupta, and J. Ryan, "Random projection-based multiplicative data perturbation for privacy preserving distributed data mining," *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, vol. 18, no. 1, pp. 92–106, 2006.

[12] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Journal of Cryptology*, vol. 15, no. 3, pp. 177–206, 2000.

[13] K. Liu, C. Giannella, and H. Kargupta, "An attacker's view of distance preserving maps for privacy preserving data mining," in *Proceedings of PKDD*, Berlin, Germany, September 2006.

[14] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Annual ACM Symposium on Theory of Computing*. New York, NY, USA: ACM, 2009, pp. 169–178.

[15] M. Cooney, "Ibm touts encryption innovation," http://tinyurl.com/d7xpkuk, 2009.

[16] T. Graepel, K. Lauter, and M. Naehrig, "Ml confidential: Machine learning on encrypted data," in *Proceedings of the 15th International Conference on Information Security and Cryptology*, ser. ICISC'12. Berlin, Heidelberg: Springer-Verlag, 2013, pp. 1–21.

[17] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data," in *22nd Annual Network and Distributed System Security Symposium, NDSS 2015, San Diego, California, USA, February 8-11, 2014*, 2015.

[18] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu, "Order preserving encryption for numeric data," in *Proceedings of ACM SIGMOD Conference*, 2004.

[19] H. Hacigumus, B. Iyer, C. Li, and S. Mehrotra, "Executing sql over encrypted data in the database-service-provider model," in *Proceedings of ACM SIGMOD Conference*, 2002.

[20] W. K. Wong, D. W.-l. Cheung, B. Kao, and N. Mamoulis, "Secure knn computation on encrypted databases," in *Proceedings of ACM SIGMOD Conference*. New York, NY, USA: ACM, 2009, pp. 139–152.

[21] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, "Cryptdb: Protecting confidentiality with encrypted query processing," in *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, ser. SOSP '11. New York, NY, USA: ACM, 2011, pp. 85–100.

[22] Y. Freund and R. E. Schapire, "A short introduction to boosting," in *International Joint Conferences on Artificial Intelligence*. Morgan Kaufmann, 1999, pp. 1401–1406.

[23] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*. Chapman and Hall/CRC, 2007.

[24] M. Rudelson and R. Vershynin, "Smallest singular value of a random rectangular matrix," *Communications on Pure and Applied Mathematics*, vol. 62, pp. 1707–1739, 2009.

[25] A. Hyvarinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. Wiley, 2001.

[26] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer-Verlag, 2001.

[27] K. Chen and L. Liu, "A random rotation perturbation approach to privacy preserving data classification," in *Proceedings of International Conference on Data Mining (ICDM)*. Houston, TX: IEEE, 2005.

Keke Chen is an associate professor in the Department of Computer Science and Engineering, a member of the Ohio Center of Excellence in Knowledge-Enabled Computing (the Kno.e.sis Center), at Wright State University. He directs the Data Intensive Analysis and Computing (DIAC) Lab at the Kno.e.sis Center. He earned his Ph.D. degree from Georgia Institute of Technology in 2006, his Master's degree from Zhejiang University in China in 1999, and his Bachelor's degree from Tongji University in China in 1996. All degrees are in Computer Science. His current research areas include secure data services and mining of outsourced data, the privacy issues in social computing, cloud computing, and visual exploration of big data. During 2006-2008, he was a senior research scientist at Yahoo! Labs, working on web search ranking, cross-domain ranking, and web-scale data mining. He owns three patents for his work in Yahoo!.

Shumin Guo graduated from the Department of Computer Science and Engineering with a Master's degree in 2013. He was a member of the Data Intensive Analysis and Computing (DIAC) Lab at Wright State University, working with Dr. Keke Chen. He also received his Master's degree in Electronics Engineering from Xidian University, Xi'an China, in 2008. His research interests include privacy preserving data mining, social network analysis, and cloud computing.