Privacy-Preserving PCA on Horizontally-Partitioned Data

Mohammad Al-Rubaie*, Pei-yuan Wu†, J. Morris Chang‡ and Sun-Yuan Kung§
* Iowa State University, Ames, Iowa.
Email: mti@iastate.edu
† Princeton University, Princeton, New Jersey.
Email: peiyuanwu1987@gmail.com
‡ University of South Florida, Tampa, Florida.
Email: chang5@usf.edu
§ Princeton University, Princeton, New Jersey.
Email: kung@princeton.edu

***Abstract*—Private data is used on daily basis by a variety of applications where machine learning algorithms predict our shopping patterns and movie preferences among other things. Principal component analysis (PCA) is a widely used method to reduce the dimensionality of data. Reducing the data dimension is essential for data visualization, preventing overfitting and resisting reconstruction attacks. In this paper, we propose methods that would enable the PCA computation to be performed on horizontally-partitioned data among multiple data owners without requiring them to stay online for the execution of the protocol. To address this problem, we propose a new protocol for computing the total scatter matrix using additive homomorphic encryption, and performing the Eigen decomposition using Garbled circuits. Our hybrid protocol does not reveal any of the data owner's input; thus protecting their privacy. We implemented our protocols using Java and Obliv-C, and conducted experiments using public datasets. We show that our protocols are efficient, and preserve the privacy while maintaining the accuracy.**

I. Introduction

Machine learning tasks include supervised learning like regression and classification, and unsupervised learning like clustering. These techniques are widely used in modern applications including supporting identity and access management [1], [2], detecting email spam [3], identifying fraudulent credit card transactions [4], or building clinical decision support systems [5]. Very often, these applications use personal data (e.g. biometrics, health care records, and financial datasets) during the training (enrollment) and testing (prediction) phases of machine learning.

Dimensionality reduction is an important machine learning tool that was traditionally used to overcome issues like: (a) over-fitting when the features dimensions far exceed the number of training samples, (b) performance degradation due to suboptimal search, and (c) higher computational cost and power consumption resulting from high dimensionality in the feature space. Principal Component Analysis (PCA) is a widely used method for dimensionlity reduction. PCA aims to project the data on the principal components with the highest variance; thus preserving most of the information in the data while reducing the data dimensions. From looking at fig. 1, it can be noticed that most of the variability happens along the black axis (denoted Principal Component 1). Hence, projecting all the points on that new axis could reduce the dimensions without sacrificing much of the data variability.

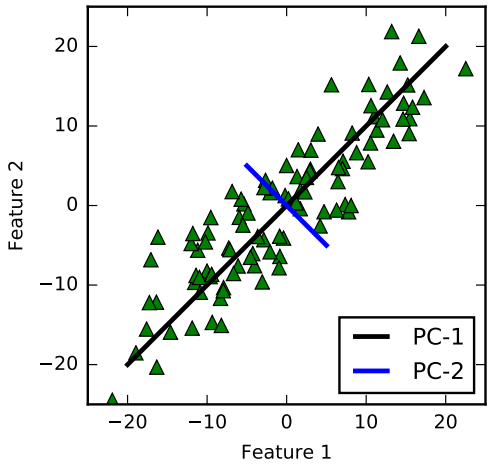


Fig. 1: Principal Component Analysis (PCA)

In this paper, we address the problem of performing PCA by the data user wihtout compromising the privacy of the data owners. We consider the case of collaborative learning on a joint dataset formed of samples held by different data owners, where each sample contains the same attributes (features). Such data is described as horizontally-partitioned since the data is represented as rows of similar features (columns), and each data owner holds a different set of rows in the joint data matrix. Many practical applications fall into this data sharing model (e.g. detecting email spams or continuous authentication). While PCA was performed traditionally by gathering the data in a centralized location, it is critical to develop systems that can enable data owners to protect their data privacy while computing the PCA projection matrix.

Previous approaches to distributed PCA either did not con-

978-1-5090-5569-2/17/$31.00 2017 IEEE

sider privacy preservation [6], [7] or required all data owners to remain online throughout the execution of their protocols [8]. Due to this, some approaches for privacy-preserving continuous authentication included performing PCA on data from one data owner and only sharing the PCA projection matrix [9]. Such method would not utilize the data from all data owners; hence, possibly compromising the utility of data. In our work, we provide a solution to such issues by proposing and implementing a practical method to perform PCA in a privacy-preserving way. Our protocols could be utilized as a privacy-preserving data preprocessing stage that comes before applying other privacy-preserving machine learning algorithms for the purpose of classification [9] or regression [10]. Thus, ensuring privacy and utility throughout the machine learning process on private data.

Our contributions are as follows:

First. Computing the projection matrices for PCA requires computing the scatter matrix in a distributed way. Thus, we developed protocols that use additive homomorphic encryption to compute the scatter matrix. We assume the existence of a crypto service provider (CSP) that would not collude with the data user (similar assumption to [10], [11]). Participating data owners are required to compute their individual shares, encrypt them using homomorphic encryption with the CSP's public key, and send these shares to the data user that would aggregate the shares. The CSP builds a garbled circuit that performs Eigen Decomposition on the scatter matrix computed from the aggregate shares supplied by the data user using obliviuous transfer. While we mainly consider semi-honest parties, we highlight the possibility of extending our approach to resist malicious attacks. The use of our protocols for the scatter matrix distributed computation is not limited to our PCA scenario since such a matrix is necessary for multiple purposes like financial applications [12], [13] and synthetic data generation [13]. It is also worth noting that our distributed PCA technique could be used whenever dimensionality reduction is needed as part of the machine learning process as in [9].

Second. We implement our privacy-preserving protocols using Java and Obliv-C, and perform testing using public datasets. First, we show that our protocols are efficient. We further demonstrate that our approach maintains correctness and does not degrade the accuracy of ML tasks.

II. Related Work

PCA Distributed Computation: There are approaches that considered distributed PCA computation ([6], [7]), however, unlike our paper, they did not consider privacy at all in computing the projection matrices, and their method included sharing private information such as the data mean with other parties. In addition, [7] requires all parties including data owners to remain online for the computation to be completed. The third approach [8] had privacy of the individual shares in mind, however, it required multiple stages of computation, and all the data owners had to remain online during these stages, while in our approach, the data owners do not have to remain online after submitting their individual encrypted shares.

While not directly concerned with computing PCA in a privacy-preserving way, Sedenka *et al.* [9] proposed computing PCA based on a single data owner data, and only revealing the PCA projection matrix to project the data. Our approach is a solution to the privacy problem they were faced with as they needed PCA as a preprocessing stage to improve the utility of their single-class privacy-preserving classification method. Our protocols enable the use of all data owners' data without revealing the original data to the other parties; thus preserving the privacy while maximizing the utility.

Cryptographic Methods: Multiple secure multiparty computation (MPC) techniques have been utilized to solve a variety of problems including creating privacy-preserving versions of many machine learning algorithms. These approaches mainly used additive homomorphic encryption [11], [14], [15], commutative keyed hash function [16] or hybrid approaches such as [10]. Many solutions that only rely on additive homomorphic encryption require all data owners to remain online during the computation stage which does not seem practical for many applications. Some exceptions like [11] relied on two computation parties, each with a different role, to avoid having data owners remain online. It is notable that Garbled Circuit solutions has two main parties: a garbler and an evaluator. Nikolaenko *et al.* [10] utilized this fact to allow performing a complex computation by using a hybrid approach of additive homomorphic encryption and garbled circuits, without requiring data owners to remain online. In both approaches [10], [11], the two computation parties are trusted not to collude as they perform different roles, and could essentially be represented by different companies.

III. Preliminaries

A. Principal Component Analysis (PCA)

PCA is an unsupervised dimensionality reduction technique (i.e. it does not utilize data labels). Consider a dataset with N training samples $\mathbf{x}_1, \cdots, \mathbf{x}_N$, where each sample has M features ($\mathbf{x}_i \in \mathbb{R}^M$). PCA performs spectral decomposition of the center-adjusted scatter matrix $\overline{\mathbf{S}} \in R^{M \times M}$:

$$\overline{\mathbf{S}} = \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \tag{1}$$

where μ is the mean vector $\mu = \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i$, and $\mathbf{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \cdots, \lambda_M)$ is a diagonal matrix of eigenvalues, with the eigenvalues arranged in a monotonically decreasing order (i.e. $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_M$). The matrix $\mathbf{U} = [\mathbf{u}_1\ \mathbf{u}_2\ \cdots \mathbf{u}_M]$ is an $M \times M$ unitary matrix where $\mathbf{u}_j$ denotes the j^{th} eigenvector of the scatter matrix. For PCA, we retain the m principle components that correspond to the m highest eigenvalues in order to obtain the projection matrix $\mathbf{U}_m \in \mathbb{R}^{M \times m}$. We obtain the reduced-dimensions feature vector by:

$$\widetilde{x}_i = \mathbf{U}_m^T \mathbf{x_i} \tag{2}$$

The parameter m determines to which extent the signal power is retained after dimensionality reduction. More precisely, while the original feature vectors have signal power

$\sum_{j=1}^{M} \lambda_j$, the reduced-dimensions' feature vectors have power $\sum_{j=1}^{m} \lambda_j$. In fig. 2, $m = 1$ while $M = 2$, and the red dots $\widetilde{\mathbf{x}}_\mathbf{i}$ are the projections of the blue dots $\mathbf{x_i}$ on the principal component $\mathbf{u^1}$.

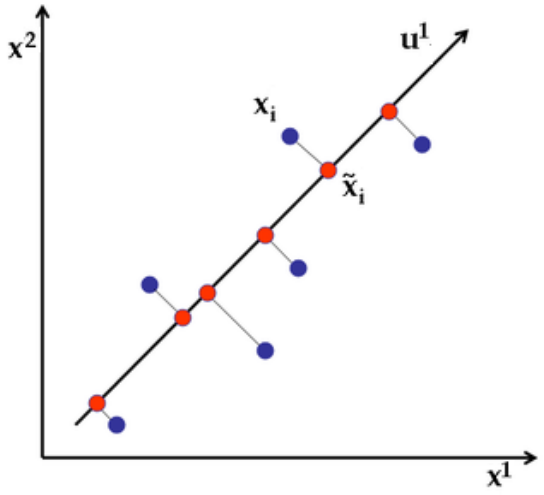


Fig. 2: Data Projection

B. Cryptographic Background

An important building block of our protocols is additive homomorphic encryption. There are multiple semantically-secure additive homomorphic encryption schemes, and without loss of generality, Paillier [17] will be used in this work as an example of such encryption schemes. Let function $\mathcal{E}_{pk}[\cdot]$ be an encryption operation indexed by a public key pk, and let $\mathcal{D}_{sk}[\cdot]$ be a decryption operation indexed by a secret key sk. The following rule holds for additive homomorphic encryption:

$$\mathcal{E}_{pk}[a+b] = \mathcal{E}_{pk}[a] \otimes \mathcal{E}_{pk}[b]$$

where $\otimes$ denotes the modulo multiplication operator in the encrypted domain. In addition, scalar multiplication can be achieved by: $\mathcal{E}_{pk}[a]^b = \mathcal{E}_{pk}[a \cdot b]$. Such encryption schemes only accept integers as plain text while machine learning data is expected to have real values. Hence, the data (feature values) should be discretized to obtain integer values [18].

We also utilize garbled circuits and oblivious transfer in our protocols. The main idea is having one party (the garbler) create an encrypted circuit that computes a function f, while the second party (the evaluator) executes this circuit on garbled input, and obtains the function output without learning any intermediate value. The garbled circuit is basically a collection of garbled gates, where each wire of this encrypted circuit would have two random cryptographic keys associated with it (for one and zero). The garbled input for the evaluator party is obtained from the garbling party by using oblivious transfer that does not allow the garbler to learn anything about the evaluator's input. The garbler would only provide the mapping from garbled output keys to bits to enable the evaluator to obtain said output.

IV. Problem Statement

A. Overview

We consider the case of horizontally-partitioned data across multiple data owners, and a single data user (fig. 3). Suppose there are L data owners. Each data owner ℓ holds a set of data vectors $x_i^\ell \in \mathbb{R}^M$ where M is the number of features (dimensions), and $i = 1, ..., N^\ell$ (N^ℓ is the number of data vectors held by data owner ℓ). Hence, each data owner ℓ would have a data matrix $X^\ell \in \mathbb{R}^{N^\ell \times M}$.

The data user would like to compute the PCA projection matrix from the data distributed across the data owners. The projection matrix would then be used by the data owners to reduce the dimensions of their data. Such reduced dimensions data could later be used an input to a certain privacy-preserving ML algorithm that performs classification, clustering or regression.

Traditionally, PCA was only used on the joint dataset in a centralized location. Computing the projection matrix $\mathbf{U}$ required the data owners to reveal all their data before applying PCA. Hence, it is necessary to modify the computation of the PCA projection matrix to make it distributed and privacy-preserving. This will be presented in section V.

B. Threat Model

The main privacy requirement of our protocols is enabling data owners to preserve the privacy of their data. We consider the adversaries to be the computation parties: the Crypto Service Provider (CSP) and the data user. Neither of these parties should have access to any of the data owner's input data or any intermediate values. The data user should only learn the output which is the PCA projection matrix and the Eigen values.

The CSP's main role is facilitating the privacy-preserving computation of the scatter matrices (section V). The CSP is assumed not to collude with the data user (similar to the privacy service provider in [11] and the CSP in [10]). The data user and the CSP can be different corporations that would not collude, at least in the interest of maintaining their reputation and their customer base.

We assume all participants to be honest-but-curious, i.e., we consider the semi-honest adversarial model. This means that all parties would correctly follow the protocol specification, but try to use the protocol transcripts to extract new information. Hence, both the data user and the CSP are considered semi-honest, non-colluding but otherwise untrusted servers. We also discuss extensions that could account for the possibility of collusion between the data user and a subset of the data owners in order to glean private information pertaining to a single data owner (section V-B).

V. Privacy-Preserving PCA

In this section, we describe the privacy preserving computation of the scatter matrix $\overline{\mathbf{S}}$ and the PCA projection matrix. We assume that there are L data owners that are willing to cooperate with a certain data user to compute the scatter matrices. We also assume the existence of a crypto service

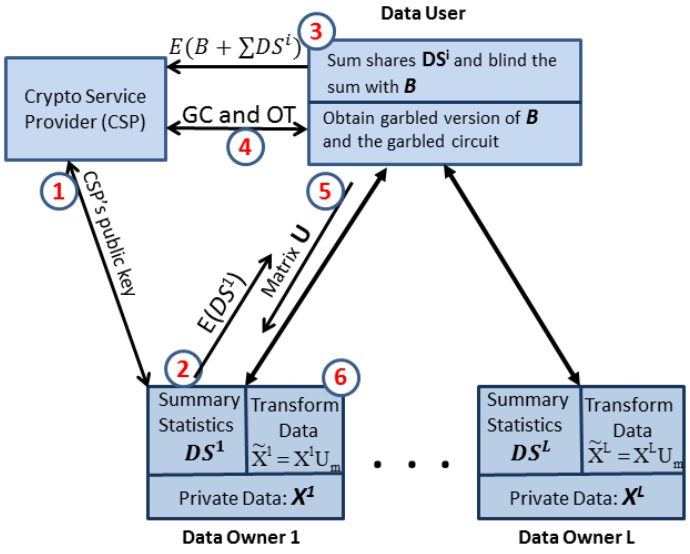


Fig. 3: System architecture and protocols

provider (CSP) who is trusted not to collude with the data user. Figure 3 shows the overall system architecture and the interaction between different parties.

This section covers steps 1-4 in fig. 3, while steps 5 and 6 show the usage of the PCA projection matrix by the data owner to transform its private data to lower dimensional data. This could be followed by using a privacy-preserving ML algorithm on the transformed data (such as [9]). We first present our protocols for the semi-honest model (section V-A), and follow that by an analysis of these protocol in section V-B.

A. The Semi-honest Model

In this section, we consider the case when all parties follow the protocol correctly, but might look at the information passed between entities. In the following, all communication between the data owners, the data user and the CSP is assumed to be carried on secure channels using well-known methods like SSL/TLS, digital certificates, and signatures. Hence, we only concentrate on our protocols for brevity.

As mentioned in section III-B, the individual shares have to be discretized before applying homomorphic encryption. Since such discretization involves scaling by an integer, such as $2^b - 1$, the resulting scatter matrix would take larger values than if it was directly computed from the original data. Therefore, the data user should divide all the scatter matrix elements by $(2^b - 1)^2$ after the protocol concludes.

We first describe the necessary equations to compute the scatter matrix in a distributed way, and we follow that by presenting the protocol that performs the PCA computation in a privacy-preserving way.

The total scatter matrix can be computed in an iterative fashion. Suppose there are L data owners, and denote P^ℓ as the set of training samples held by data owner ℓ. Each data owner ℓ can locally compute:

$$\mathbf{R}^\ell = \sum_{i \in P^\ell} \mathbf{x}_i \mathbf{x}_i^T \quad , \quad \mathbf{v}^\ell = \sum_{i \in P^\ell} \mathbf{x}_i \quad \text{and} \quad N^\ell = \left|P^\ell\right| \tag{3}$$

It can be shown that the total scatter matrix (eq. 1) is given by summing the partial contributions from each party:

$$\begin{aligned} \overline{\mathbf{S}} &= \sum_{i=1}^{N} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T = \sum_{i=1}^{N} \mathbf{x}_i \mathbf{x}_i^T - N\mu\mu^T \\ &= \sum_{\ell=1}^{L} \mathbf{R}^\ell - \frac{1}{N}\mathbf{v}\mathbf{v}^T = \mathbf{R} - \frac{1}{N}\mathbf{v}\mathbf{v}^T \end{aligned} \tag{4}$$

where

$$\mathbf{R} = \sum_{\ell=1}^{L} \mathbf{R}^\ell \quad , \quad \mathbf{v} = \sum_{\ell=1}^{L} \mathbf{v}^\ell \quad \text{and} \quad N = \sum_{\ell=1}^{L} N^\ell \tag{5}$$

The data owners cannot send the local shares ($[\mathbf{R}^\ell, \mathbf{v}^\ell, N^\ell]$ for $\overline{\mathbf{S}}$) to the data user in cleartext since they include statistical summaries of their data. Alternatively, they can encrypt the local shares using an additive homomorphic encryption scheme (such as Paillier's cryptosystem [17]) where the public key is provided by the CSP. After receiving these encrypted shares, the data user can aggregate them to compute the encrypted intermediate values (namely $\mathbf{R}$, $\mathbf{v}$ and N), send them to the CSP for decryption (after blinding the values), and use these aggregate values to compute the scatter matrix and the PCA projection matrix by using garbled circuits and oblivious transfer. Blinding refers to adding random numbers to these encrypted values to prevent the CSP from learning anything about the data even in its aggregated form (*"Blinding"* the values using an equivalent of one-time pad).

For computing $\overline{\mathbf{S}}$, the protocol is as follows (fig. 3):

1) **Setup**: The CSP sends its public key pk for Paillier's cryptosystem to the data owners and the data user based on their request. This step could also include official registration of the data owners with the CSP.
2) **The data owners**: Each data owner ℓ would compute its own share $\mathbf{DS}^\ell = \{\mathbf{R}^\ell, \mathbf{v}^\ell, N^\ell\}$ using eq. 3. After discretizing all the values to obtain integer values, the data owners would then encrypt $\mathbf{R}^\ell$, $\mathbf{v}^\ell$ and N^ℓ using the CSP's public key to obtain $\mathcal{E}_{pk}[\mathbf{DS}^\ell] = \{\mathcal{E}_{pk}[\mathbf{R}^\ell], \mathcal{E}_{pk}[\mathbf{v}^\ell], \mathcal{E}_{pk}[N^\ell]\}$. Finally, each data owner ℓ sends $\mathcal{E}_{pk}[\mathbf{DS}^\ell]$ to the data user.
3) **The data user**: The data user receives $\mathcal{E}_{pk}[\mathbf{DS}^\ell] = \{\mathcal{E}_{pk}[\mathbf{R}^\ell], \mathcal{E}_{pk}[\mathbf{v}^\ell], \mathcal{E}_{pk}[N^\ell]\}$ from each data owner ℓ, and proceeds to compute the encryption of $\mathbf{R}$, $\mathbf{v}$ and N given by eq. 5. More explicitly, the data user is capable of computing $\mathcal{E}_{pk}[\mathbf{R}]$, $\mathcal{E}_{pk}[\mathbf{v}]$ and $\mathcal{E}_{pk}[N]$ from the encrypted data owner shares as follows:

$$\begin{aligned} \mathcal{E}_{pk}[\mathbf{R}] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\mathbf{R}^\ell] \\ \mathcal{E}_{pk}[\mathbf{v}] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[\mathbf{v}^\ell] \\ \mathcal{E}_{pk}[N] &= \otimes_{\ell=1}^{L} \mathcal{E}_{pk}[N^\ell] \end{aligned} \tag{6}$$

   The data user adds some random integers to these aggregated values to mask them from the CSP, thus obtaining the blinded shares $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\mathbf{v}']$ and $\mathcal{E}_{pk}[N']$ that can be sent to the CSP for decryption.
4) **The CSP and Data User**: Eigen Decomposition using Garbled Circuits

a) The CSP would use its private key to decrypt the blinded shares $\mathcal{E}_{pk}[\mathbf{R}']$, $\mathcal{E}_{pk}[\mathbf{v}']$ and $\mathcal{E}_{pk}[N']$ received from the data user. Without knowing the random values added by the data user, the CSP can not learn the aggregated values.
b) The CSP would then proceed to construct a garbled circuit to perform Eigen Decomposition on the scatter matrix computed from the aggregated shares. The input to this garbled circuit is the garbled version of: (i) the blinded aggregate shares which were decrypted by the CSP (step 4a), and (ii) the blinding values which are generated and held by the data user (step 3). Since the CSP constructs the garbled circuit, it can obtain the garbled version of its input by itself. However, the data user needs to interact with the CSP using oblivious transfer to obtain the garbled version of its input: the blinding values. Using oblivious transfer guarantees that the CSP would not learn the blinding values held by the data user.
   The garbled circuit constructed by the CSP takes the two garbled inputs and does the following: (1) computes the scatter matrix from the shares $\mathbf{R}'$, $\mathbf{v}'$ and N' after subtracting the data user blinding values added in the previous steps, and (2) follows that by performing Eigen decomposition on the scatter matrix to obtain the PCA projection matrix.
c) The data user would receive the garbled circuit as its evaluator. This garbled circuit already has the CSP's garbled input which is the decrypted and blinded aggregate shares, and obtains the garbled version of the blinding values using oblivious transfer.
d) Finally, the data user executes the garbled circuit and obtains the projection matrix and Eigen values as output.

The details of implementing the Eigen Decomposition using garbled circuits will be deferred to section VI.

B. Analysis

We assume that all the parties participating in our protocols are semi-honest. This includes the data owners, the data user and the CSP. In the semi-honest adversarial model, corrupted parties would still correctly follow the protocol specification, however, the adversary obtains the internal state of the corrupted parties including the exchanged messages, and attempts to use this to extrapolate new information. We further assume that the data user and the CSP do not collude, which can be ensured by having two different organizations control each of these two parties.

In section IV-B, we have set our privacy goal to prevent revealing the data owners input, or any intermediate values, to the CSP, the data user or any other data owner. It should be noted that each of the data owner shares were encrypted using a semantically secure homomorphic encryption. This is Paillier's cryptosystem [17]. We shall expand on the security of our protocols in the following discussion by considering the view of each party:

1) Data Owners: None of the data owners would have any interaction with other data owners in our protocols. Each data owner would receive the CSP's public key, and send their encrypted shares to the data user (step 2 in sections V-A). It is clear that none of the data owners can learn extra information about other data owners, or any other party.

2) Data user: The data user receives the individual shares from each data owner (step 3 in sections V-A). However, these shares are provided by the data owner in encrypted form using the CSP's public key. Hence, the data user cannot learn the summary statistics of any individual data owner from these encrypted shares. As the data user does not know the CSP's private key, the received encrypted shares would be indistinguishable from random.

After the data user aggregates all individual shares, these encrypted shares are sent to the CSP (after blinding them), and the data user does not receive their decryption. The data user only receives the Eigen Decomposition garbled circuit with the blinded aggregate shares garbled (to act as the CSP's input to the garbled circuit). Hence, the data user does not learn anything about the individual shares or their aggregates. After executing the garbled circuit, the data user only learns the intended protocol output which is the PCA projection matrix and the associated Eigen values.

Finally, in order to ensure that a malicious data user does not use a single data owner input to compute the projection matrix in an attempt to reveal some private information about that data owner. Certain measures from the literature could be utilized to enable the CSP to ensure that the aggregates received were based on input from all the intended data owners. Such method was outlined by [10] where zero knowledge proofs could be used by the data user to prove to the CSP that the ciphertexts it received were the product of all data owners' ciphertexts (section IV.G in [10]).

3) CSP: The CSP does not receive any data from the data owners. The CSP receives values from the data user to decrypt (step 4 in sections V-A). These values are aggregates and not individual shares. Moreover, the data user blinds these aggregate values by adding random numbers to these aggregates to mask their values from the CSP (which is equivalent to the one-time pad). The data user also needs to obtain the garbled version of its blinding values from the CSP, but the CSP does not learn the blinding values as oblivious transfer is used for this interaction. Overall, the CSP does not learn any useful information from its interaction with the data user.

VI. Eigen Decomposition Garbled Circuit

A. Eigen Decomposition

Eigen Decomposition could be performed in a variety of ways. Some methods such as the QR algorithm find all the Eigen vectors/values at once. However, PCA is used to reduce the dimensions; hence, not all the Eigen vectors are needed. For this reason, we will rely on methods that only find a subset of the Eigenvalues/vectors to avoid the extra computation associated with finding unneeded Eigenvectors.

One of the most notable algorithms for Eigen decomposition is the power iteration method. This method finds the dominant Eigen value (largest value) with its associated Eigen vector. A matrix deflation method could be used afterwards to remove the effect of the already found dominant Eigen value while leaving the remaining Eigen values unchanged. By repeatedly applying the power iteration method and matrix deflation, we can find the required number of Eigen vectors.

The power iteration method starts with a non-zero vector x_0, and attempts to reach a good approximation of the dominant Eigenvector in a number of iterations J. In each iteration j, a new approximation is computed as $x_j = Ax_{j-1}$ where A is the matrix for which we want to find the Eigen values/vectors. The algorithm stops when the difference between x_j and x_{j-1} is negligible. After that, the dominant Eigenvalue can be obtained using $\lambda = \frac{Ax_J \cdot x_J}{x_J \cdot x_J}$ (Rayleigh quotient). It can be seen that if J is known, this iterative computation is equivalent to $x_J = A^J x_0$; however, computing A^J would most likely lead to an overflow. Hence, scaling x_j in each iteration is important. This is demonstrated in algorithm 1.

As input, algorithm 1 takes the scatter matrix $\overline{\mathbf{S}}$, the number of required Eigenvalues/vectors m, and the number of iterations for the power method J. The inputs m and J are provided by the data user. Algorithm 1 outputs the Eigenvalues and vectors. The algorithm has two loops: the outer loop runs m times which is the number of required Eigenvectors (principal components), while the inner loop performs the power method in J iterations. For each Eigenvector, the power method is run followed by the deflation method. In line 6, the infinity norm of the vector x is computed, and used in step 7 to scale the vector x down (to prevent overflow). The deflation method is shown in step 10 and is preceeded by normalizing the Eigenvector x in step 9.

Algorithm 1 Eigen Decomposition on Scatter matrix $\overline{\mathbf{S}}$

Input: the scatter matrix $\overline{\mathbf{S}}$ of dimensions M x M
Input: the number of required Eigen values/vectors m
Input: the number of iterations J
Output: PCA projection matrix (Eigenvectors) U
Output: Eigenvalues Λ

1: Set x_0 to be a unit vector
2: **for** `i = 1 to` m **do**
3: $x = x_0$ ▷ initialize the Eigenvector
4: **for** `j = 1 to` J **do**
5: $x = \overline{\mathbf{S}}x$
6: $\lambda = ||x||_\infty$
7: $x = \frac{x}{\lambda}$
8: **end for**
9: $x = \frac{x}{||x||_2}$ ▷ Normalize vector x
10: $\overline{\mathbf{S}} = \overline{\mathbf{S}} - xx^T\overline{\mathbf{S}}xx^T$ ▷ Deflation
11: Set the i^{th} Eigenvalue $\Lambda_i = \lambda$
12: Set the i^{th} Eigenvector $U_i = x$
13: **end for**

As can be seen from algorithm 1, setting the number of required principal components (PCs) m to a high number would increase the computation time. It would be possible for m to be set to a low number initially, and based on the Eigenvalues obtained as output of the protocol, the data user can choose to generate additional principal components easily. Another garbled circuit can be created that would be similar to the one described in algorithm 1 with the exception of having U as an additional input. The algorithm would be modified to include m number of deflations (line 10) using the supplied Eigenvectors in U, then proceeds to normal execution to obtain the additional principal components.

The initial Eigenvalues can be used to determine if enough number of principal components were obtained either using (1) the Kaiser method: if any Eigenvalues had values less than one, then there is no need to generate more PCs, or (2) the scree test: where the Eigenvalues can be plotted, and if the curve is still steep after having m Eigenvalues, more PCs could be generated.

B. Implementation

We implemented our garbled circuit using Obliv-C [19]. Obliv-C includes an implementation of Yao's garbled circuit protocol and oblivious transfer, and it further incorporates recent optimizations. Obliv-C provides an extensible secure computation programming tool with the ability to be integrated with standard C code. It basically provides garbled integer and binary data types similar to the standard C data types, and the basic arithmetic and logic operations associated with them.

In order to implement algorithm 1, we implemented fixed-point arithmetic functions, and a linear algebra library. We used fixed-point arithmetic for its speed in comparison to floating point arithmetic. We utilized 32-bits integer and used the format Q15.16. Functions for multiplication and division were created to work with Q15.16 fixed-point numbers.

Because of the need to normalize the Eigenvectors (step 9 in algorithm 1), we needed to compute the square root using an iterative algorithm: $y_{i+1} = 0.5(y_i + \frac{x}{y_i})$ where x is the value for which the square root is desired, and y_i and y_{i+1} are the previous and current estimations. It can be seen that each iteration includes a division. Furthermore, to normalize a certain vector, each of its values has to be divided by the resulting square root. Knowing that multiplication is generally more efficient than division, we should find the inverse square root (instead of the square root); thus, performing multiplication by the vector rather than division. Moreover, the iterations required to compute the inverse square root do not include divisions: $y_{i+1} = y_i(1.5 - 0.5xy_i^2)$. Hence, avoiding divisions for all of the vector normalization steps.

In addition, we implemented a basic linear algebra library that performs matrix multiplication and addition, dot products, scaling or matrices and vectors, computing norms, and some other minor operations.

VII. Experiments

In this section, we evaluate our PCA computation protocol in terms of efficiency and accuracy (performance of ML

TABLE I: Distributed PCA Efficiency

Dataset	Features	Classes	Avg. Data Owner time	Avg. Data User Coll / Add time	CSP Decryption time	EigenDecomposition using Garbled Circuits
Diabetes	8	2	0.63 sec	10 ms	0.67 sec	28.3 sec (8)
Breast Cancer	10	2	0.93 sec	11 ms	1 sec	49.6 sec (8)
Australian	14	2	1.7 sec	12 ms	1.8 sec	119.1 sec (8)
German	24	2	5 sec	17 ms	5 sec	16.3 min (15)
Ionosphere	34	2	9.8 sec	24 ms	9.9 sec	43.2 min (15)
SensIT Acoustic	50	3	22.5 sec	40 ms	22.7 sec	126.7 min (15)

algorithms). We implemented our protocols using Java and Obliv-C [19]. For homomorphic encryption using Paillier's cryptosystem, we used the Java library: THEP [20]. For some of our experiments, we also utilized python and libraries like Scikit-Learn and NumPy.

The datasets used were from the UCI machine learning repository [21]. While our protocols work for any type of machine learning algorithm, we chose classification as the number of datasets available for classification, which was 255, far outnumbered those available for clustering or regression (around 55 each). Since the efficiency of our protocols depend on the data dimensions (number of features), we chose datasets with varying numbers of features: 8-50. Both the number of features and classes for each dataset can be seen in table I.

All the experiment were performed on a commodity computer with i5-6600K CPU @ 3.5GHz and 8GB of RAM. In all experiments, we use SVM, and perform cross validation and grid search to find the SVM parameters C and γ. The number of data owners was set to 10 in all experiments. The initial stage of aggregating encrypted shares was implemented using Java, while the second stage where these encrypted shares were utilized to compute the projection matrix (using a garbled circuit) was implemented using Obliv-C.

A. Efficiency

Table I show the timing data for performing distributed PCA on different datasets when using Paillier's key length of 1024 bits. In this table, the *"Avg. data owner time"* refers to the total time it took the data owner to compute the individual shares and encrypt them. The *"Avg. data user Coll/Add time"* represents the time needed to *collect* each individual share from each data owner, and *add* it to the current sum of these shares (in the encrypted domain). We also show the time it took the CSP to decrypt the blinded aggregated values received from the data user. Finally, we show the time needed to run the Eigen Decomposition using garbled circuits in order to compute the PCA projection matrix. The number between the brackets refers to the number of principal components generated using the garbled circuit. Naturally, for a given dataset, reducing this number would decrease the computation time. As will be seen in the next section, even 15 PCs for the SensIT Acoustic dataset is enough to achieve adequate accuracy. Finally, it can be noticed that increasing the dimension of the data would increase the computation time for all stages of our protocols especially the Eigen Decomposition. However, even at 50 dimensions, the computation time was still reasonable especially that no specialized servers were used in our experiments

B. Accuracy

In this section, we perform experiments to test the accuracy of classification tasks after using our protocols, and to compare such results to those obtained using the python library Numpy.

We use the weighted F1 score as a measure for testing the accuracy of classifiers (it is basically an F1 score that considers the labels imbalance). The F1 score can be thought of as a weighted average of the precision and recall scores, and a classifier is at its best when its F1 score is 1, and worst at 0.

Figure 4 shows the results for distributed PCA. From fig. 4, it is clear that our protocols are correct, and their results are equivalent to the ones obtained using Numpy. It should be noted that the fluctuation in the weighted F1 score is mostly due to the SVM parameter selection, however, it can still be seen that the accuracy of both methods are almost the same.

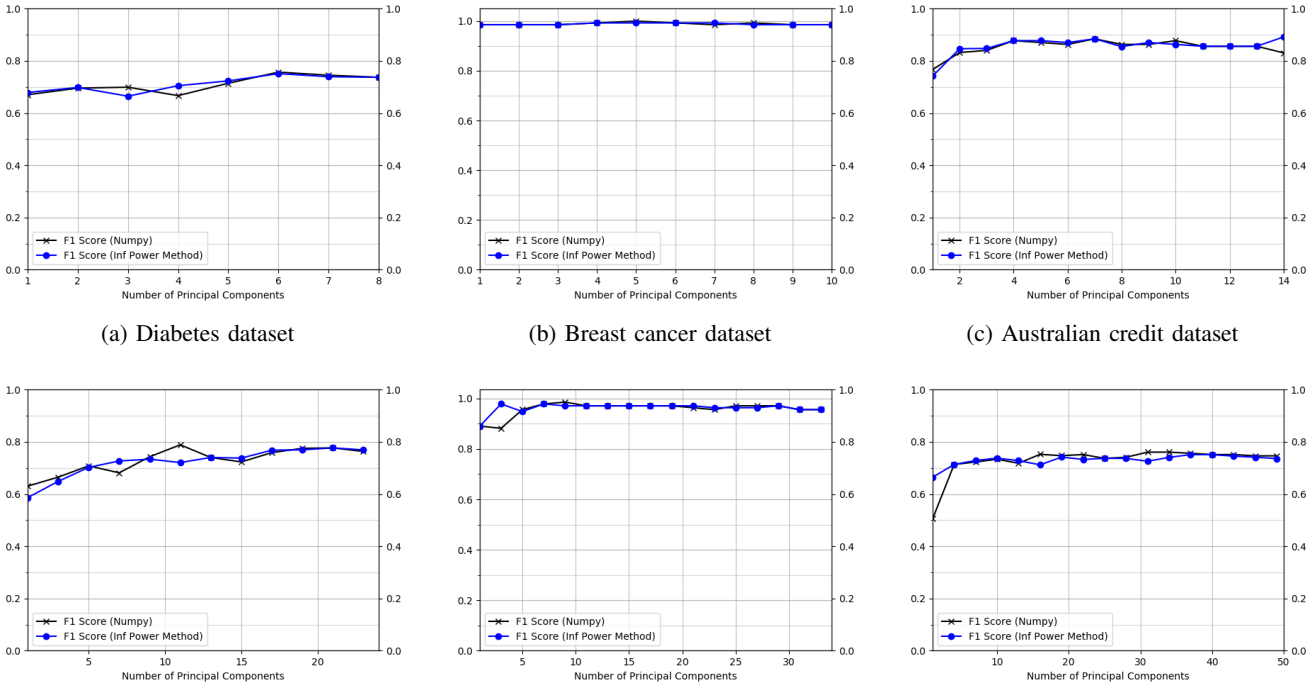
VIII. Conclusion

We introduced an efficient privacy-preserving protocol for computing PCA on horizontally-partitioned data (distributed across multiple data owners). This protocol is based on additive homomorphic encryption and garbled circuits, and it was implemented using Java and Obliv-C. We performed experiments which have shown that our protocol is efficient, and that it maintains the utility for data users. In our future work, we intend to extend our protocols to Discriminant Component Analysis (DCA) [22]. This would require designing protocols to compute the noise matrix using additive homomorphic encryption, and computing matrix inverse using garbled circuits. We also intend to extend our protocols to the Kernel version of PCA.

Acknowledgment

This material is based on research sponsored by the DARPA Brandeis Program under agreement number N66001-15-C-4068.[1]

[1]The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.



(a) Diabetes dataset (b) Breast cancer dataset (c) Australian credit dataset

(d) German credit dataset (e) Ionosphere dataset (f) SensIT Acoustic dataset

Fig. 4: Distributed PCA Privacy and Utility

REFERENCES

[1] A. K. Jain, A. Ross, and S. Pankanti, "Biometrics: a tool for information security," *Information Forensics and Security, IEEE Transactions on*, vol. 1, no. 2, pp. 125–143, 2006.

[2] J. M. Chang, C.-C. Fang, K.-H. Ho, N. Kelly, P.-Y. Wu, Y. Ding, C. Chu, S. Gilbert, A. E. Kamal, and S.-Y. Kung, "Capturing cognitive fingerprints from keystroke dynamics," *IT Professional*, vol. 15, no. 4, pp. 24–28, 2013.

[3] G. V. Cormack, "Email spam filtering: A systematic review," *Foundations and Trends in Information Retrieval*, vol. 1, no. 4, pp. 335–455, 2007.

[4] L. Delamaire, H. Abdou, and J. Pointon, "Credit card fraud and detection techniques: a review," *Banks and Bank systems*, vol. 4, no. 2, pp. 57–68, 2009.

[5] M. A. Musen, B. Middleton, and R. A. Greenes, "Clinical decision-support systems," in *Biomedical informatics*. Springer, 2014, pp. 643–674.

[6] Y. Qu, G. Ostrouchov, N. Samatova, and A. Geist, "Principal component analysis for dimension reduction in massive distributed data sets," in *Proceedings of IEEE International Conference on Data Mining (ICDM)*, 2002.

[7] Z.-J. Bai, R. H. Chan, and F. T. Luk, "Principal component analysis for distributed data sets with updating," in *International Workshop on Advanced Parallel Processing Technologies*. Springer, 2005, pp. 471–483.

[8] M. A. Pathak and B. Raj, "Efficient protocols for principal eigenvector computation over private data." *Transactions on Data Privacy*, vol. 4, no. 3, pp. 129–146, 2011.

[9] J. Sedenka, S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure outsourced biometric authentication with performance evaluation on smartphones," *Information Forensics and Security, IEEE Transactions on*, vol. 10, no. 2, pp. 384–396, 2015.

[10] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE, 2013, pp. 334–348.

[11] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 3, pp. 1053–1066, 2012.

[12] C. F. Lee, J. C. Lee, and A. C. Lee, *Statistics for business and financial economics*. Springer, 2000, vol. 1.

[13] J.-P. Bouchaud and M. Potters, "Financial applications of random matrix theory: a short review," *arXiv preprint arXiv:0910.1205*, 2009.

[14] J. Zhan, L. Chang, and S. Matwin, "Privacy-preserving support vector machines learning," in *Proceedings of the 2005 International Conference on Electronic Business (ICEB05)*, 2005.

[15] S. Laur, H. Lipmaa, and T. Mielikäinen, "Cryptographically private support vector machines," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2006, pp. 618–624.

[16] H. Yu, X. Jiang, and J. Vaidya, "Privacy-preserving svm using nonlinear kernels on horizontally partitioned data," in *Proceedings of the 2006 ACM symposium on Applied computing*. ACM, 2006, pp. 603–610.

[17] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in cryptologyEUROCRYPT99*. Springer, 1999, pp. 223–238.

[18] S. Govindarajan, P. Gasti, and K. S. Balagani, "Secure privacy-preserving protocols for outsourcing continuous authentication of smartphone users with touch data," in *Biometrics: Theory, Applications and Systems (BTAS), 2013 IEEE Sixth International Conference on*. IEEE, 2013, pp. 1–8.

[19] S. Zahur and D. Evans, "Obliv-c: A language for extensible data-oblivious computation." *IACR Cryptology ePrint Archive*, vol. 2015, p. 1153, 2015.

[20] THEP, "The homomorphic encryption project," Java Library, 2016, accessed 3-April-2016. [Online]. Available: https://github.com/diegode/thep

[21] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[22] S.-Y. Kung, "Discriminant component analysis for privacy protection and visualization of big data," *Multimedia Tools and Applications*, pp. 1–36, 2015.