ATLANTIS PRESS International Journal of Computational Intelligence Systems, Vol. 11 (2018) 33–44

CPP-ELM: Cryptographically Privacy-Preserving Extreme Learning Machine for Cloud Systems

Ferhat Özgür Çatak [1] *, Ahmet Fatih Mustacoglu [1]

[1] TÜBİTAK - BİLGEM,
Baris Mh.,
Kocaeli, Turkey
E-mail: {ozgur.catak,afatih.mustacoglu}@tubitak.gov.tr

Received 9 June 2017

Accepted 18 September 2017

Abstract

The training techniques of the distributed machine learning approach replace the traditional methods with a cloud computing infrastructure and provide flexible computing services to clients. Moreover, machine learning-based classification methods are used in many diverse applications such as medical predictions, speech/face recognition, and financial applications. Most of the application areas require security and confidentiality for both the data and the classifier model. In order to prevent the risk of confidential data disclosure while outsourcing the data analysis, we propose a privacy-preserving protocol approach for the extreme learning machine algorithm and give private classification protocols. The proposed protocols compute the hidden layer output matrix **H** in an encrypted form by using a distributed multi-party computation (or cloud computing model) approach. This paper shows how to build a privacy-preserving classification model from encrypted data.

Keywords: Extreme learning machine, Privacy-preserving machine learning, Homomorphic encryption

1. Introduction

Classification algorithms are used for predicting the label of new instances by a classifier model that is built using correctly identified input observations. In order to achieve high performance results for accurate classification, the classification algorithm needs a large number of valid input instances.

Today, many organizations such as financial and medical institutions produce large-scale data sets. Machine learning-based data analysis tools are used by these organizations. The main problems with these data sets are privacy and security concerns. All of these tools want full authority to access private and confidential data to build a supervised or unsupervised model.

Even though various anonymization techniques are applied to confidential data sets to protect privacy, the confidential data itself can be still retrieved by an adversary via various method [15]. Imagine a scenario in which two or more institutions want to build a classification model [19] by combining each other's confidential dataset. In such a case, it is crucial to find a new privacy-based training method for classification algorithms that can run together on multiple confidential data sets without retrieving the private data.

In machine learning, a classification algorithm basically consists of two sequential steps: model building and labeling of new instances. The model

* TÜBİTAK - BİLGEM Cyber Security Institute, Kocaeli, Turkey

building phase of a classification algorithm creates a classifier hypothesis, h, from the correctly labeled input data set $\mathscr{D}$. In the phase of labeling new instances, the classifier hypothesis h predicts the label of unseen input instances $\mathscr{X}$. The traditional classifier algorithms require direct access to data, which poses security and privacy risks as mentioned above.

Extreme learning machines (ELMs) were recently introduced as a probabilistic neural network learning algorithm, proposed by Huang et al. [3] based on generalized single-hidden layer feed-forward networks (SLFNs). The major properties of the ELM method are the high generalization property of multi-class labeling of unseen input instances, small model building time in comparison to the traditional gradient-based learning methods, and being parameter-free with randomly generated hidden layer nodes.

In this research, we propose a privacy-preserving ELM classification model building method based on a Paillier cryptosystem [22] that constructs the global neural network classification model from encrypted and partitioned data sets from different sites. The labeled input data set is vertically partitioned and distributed among the different sites, and then the intermediate computation results of each party are aggregated by an independent trusted party (or at the client-side of the computation) to privately build a classification model that is used for predicting the correct label.

The contributions of our paper are as follows:

- The Paillier cryptosystem encryption-based ELM training model is proposed for learning and thus private classification model training is achieved.
- The computation of the hidden layer output matrix of the probabilistic neural network model is distributed to independent sites, thus minimizing the data communication.

Our paper is organized as follows: in Section 2, we briefly introduce some of the related works for privacy-preserving training. In Section 3, we describe the ELM, homomorphic encryption, the Paillier cryptosystem, and the arbitrary data partitioning technique. In Section 4, we describe the proposed secure multi-party training and model building method. Section 5 evaluates our proposed training and classification model. Section 6 concludes the paper.

2. Related Works

In this section, we review the existing works that have been developed for different security and privacy-preserving machine learning methods. We highlight detailed differences between our proposed learning model and the existing works. Encryption methods can be employed to address the security and privacy concerns in machine learning [16,17].

Recently, numerous significant privacy-preserving machine learning models have been proposed. Secretans et al. [26] proposed a new probabilistic neural network (PNN) training model. In their proposed method, the PNN is built with an approximation to the optimal classifier. The theoretically optimal classifier is known as the Bayesian optimal classifier. Their proposed method needs at least three sites to be involved in the computation of the secure matrix summation. The secure matrix is used for adding the partial class conditional probability vectors.

Aggarwal et al. [1] proposed a new condensation-based training method. In their paper, it was shown that the anonymized data closely matched the characteristics of the original data. Their proposed approach produces disclosable data that satisfy privacy concerns by providing utility for data analysis. Each instance in the data is randomized while a few statistical properties of these instances are kept.

Graepel et al. [8] implemented two binary classification algorithms for homomorphically encrypted data: linear means and Fisher's linear discriminant. They make scaling adjustments that preserve the results, but leave the fundamental methodology unchanged.

Samet et al. [25] proposed new privacy-preserving algorithms for both back-propagation training and ELM classification between several sites. Their proposed algorithms are applied to the perceptron learning algorithm and presented for only single-layer models.

Bost et al. [4] developed a two-party computa-

tion framework and used a mix of different partially and fully homomorphic encryption schemes, which allowed them to use machine learning techniques based on hyperplane decisions, naive Bayes, and binary decision trees. The fundamental methodologies are still unchanged, but substantial communication between two "honest, but curious" parties is required.

Oliveria et al. [20] developed techniques for distorting private numerical attributes to protect confidentiality for clustering analysis methods. Their approach transforms data by using several different types of geometric transformations such as translation, rotation, and scaling while keeping the Euclidean distance.

Guang et al. [9] developed a new training method for a privacy-preserving back-propagation algorithm using horizontally partitioned data sets for a multi-site learning case. Their approach uses homomorphic encryption-based secure summation in its protocols.

Yu et al. [33] developed a new privacy-preserving training method for a support vector machine (SVM) classification algorithm. Their protocols build the global SVM classifier hypothesis from the distributed data at multiple sites, without retrieving or pooling the data of each site with others. Homomorphic encryption-based secure multi-party integer sums are applied in their protocols to compute the Gram matrix of the SVM algorithm formulation in a distributed manner from the encrypted partitioned data held by several different sites.

Yang et al. [31] proposed a privacy-preserving schema using homomorphic encryption in image processing. A secret key homomorphic encryption was constructed for encrypting image. Then the encrypted image can be processed homomorphically.

Tso et al. [29] proposed a new practical approach to prevent data disclosure from inside attack. Their approach is based on FairplayMP framework which enables programmers who are not experts in the theory of secure computation to implement such protocols. The method supports privacy-preserving machine learning for medical data stored in parties.

3. Preliminaries

In this section, we briefly introduce preliminary information about the ELM, homomorphic encryption, the Paillier cryptosystem, secure multi-party computation, and arbitrarily partitioned data.

3.1. Extreme Learning Machine

As mentioned before, a probabilistic neural network-based ELM classification algorithm was proposed for single-layer feed-forward neural networks [3,12,14,6]. Later, the ELM algorithm was expanded to a new model of generalized single-layer feed-forward networks. In this model, the hidden layer of the ELM may not be a neuron like in previous papers [10,11]. The first advantage of the ELM classification algorithm is its training time performance. The training phase of the ELM algorithm can be a hundred times faster than the standard SVM algorithm or the classical neural network algorithm. The reason for this rapid model building is that its hidden node biases vectors and input weight vectors are arbitrarily generated, and then the output layer weights of the model can be calculated by using a least-squares method [28,13]. As a result, the most appreciable property of the ELM training phase is based on the creation of probabilistic hidden layer parameters for performance issues.

Let us give a set of input instances $\mathscr{D} = \{(\mathbf{x}_i, y_i) \mid i = 1,...,n\}, \mathbf{x}_i \in \mathbb{R}^p, y_i \in \{1,2,...,K\}\}$ from some unknown distribution $\mathscr{X}$; they are sampled independently and identically distributed (i.i.d.). The main objective of a neural network model is to find a hypothesis $h : \mathscr{X} \to \mathscr{Y}$ where $\mathscr{X}$ is the input instances and $\mathscr{Y}$ is the set of all possible outcomes. The outcome of a single-layer feed-forward neural network with N hidden nodes in an n-dimensional vector space can be shown as:

$$f_N(\mathbf{x}) = \sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}), \mathbf{x} \in \mathbb{R}^n, \mathbf{a}_i \in \mathbb{R}^n \quad (1)$$

where $\mathbf{a}_i$ and b_i are the input weights and bias values of hidden nodes, respectively, and β_i is the weight that connects the ith hidden node to the output node.

The output function of the ELM algorithm for generalized SLFNs can be identified by:

$$f_N(\mathbf{x}) = \sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x}) = \beta \times h(\mathbf{x}) \tag{2}$$

For binary classification problems, the decision function of the ELM model becomes:

$$f_N(\mathbf{x}) = sign\left(\sum_{i=1}^{N} \beta_i G(\mathbf{a}_i, b_i, \mathbf{x})\right) = sign\left(\beta \times h(\mathbf{x})\right) \tag{3}$$

We can rewrite Eq. 2 in another form, as shown in Eq. 4:

$$\mathbf{H}\beta = \mathbf{T} \tag{4}$$

where $\mathbf{H}$ and $\mathbf{T}$ are the hidden layer matrix and the output matrix, respectively.

$$\beta = \mathbf{H}^{\dagger}\mathbf{T} \tag{5}$$

where $\mathbf{H}^{\dagger}$ is the *Moore–Penrose generalized inverse matrix* of the hidden layer matrix $\mathbf{H}$. The hidden layer matrix can be described as:

$$H(\widetilde{a}, \widetilde{b}, \widetilde{x}) = \begin{bmatrix} G(a_1, b_1, x_1) & \cdots & G(a_L, b_L, x_1) \\ \vdots & \ddots & \vdots \\ G(a_1, b_1, x_N) & \cdots & G(a_L, b_L, x_N) \end{bmatrix}_{N \times L} \tag{6}$$

where $\widetilde{a} = a_1, ..., a_L$, $\widetilde{b} = b_1, ..., b_L$, $\widetilde{x} = x_1, ..., x_N$. Then the output matrix of the ELM algorithm can be shown as:

$$T = \begin{bmatrix} t_1 \dots t_N \end{bmatrix}^T \tag{7}$$

The hidden nodes of the ELM can be arbitrarily created and they can be independent from the input data set.

3.2. Homomorphic Encryption

Homomorphic encryption schemes enable operations on plaintexts to be performed on their respective ciphertexts without revealing the plaintexts when data are divided between two or more parties as it facilitates computations with ciphertexts. One can say that a public-key encryption scheme is additively homomorphic if, given two encrypted messages such as $[\![a]\!]$ and $[\![b]\!]$, there exists a public-key summation operation $\oplus$ such that $[\![a]\!] \oplus [\![b]\!]$ is an encryption of the plaintext of $a+b$. The formal definition is that an encryption scheme is additively homomorphic if for any private key, public key (sk, pk), the plaintext space $\mathscr{P} = \mathbb{Z}_N$ for $a, b \in \mathbb{Z}_N$.

$$\begin{aligned} Enc_{pk}(a + b \bmod N) &= Enc_{pk}(x) \times Enc_{pk}(y) \\ Enc_{pk}(x \cdot y \bmod N) &= Enc_{pk}(x)^y \end{aligned} \tag{8}$$

3.2.1. Paillier's Encryption Scheme

The Paillier cryptosystem [22] is a probabilistic and asymmetric encryption algorithm. The Paillier cryptosystem is based on the problem of deciding whether a given number is an nth residue modulo n^2 where n is the product of two large primes [21]. The problem of computing the nth residue classes is believed to be computationally hard.

Let us give a set of possible plaintext messages M and a set of secret and public key pairs $K = pk \times sk$, where pk is the public key and sk is the secret key of the cryptosytem. Then the Paillier homomorphic encryption cryptosystem satisfies the following property of any two plaintext messages m_1 and m_2 and a constant value a:

$$Dec_{sk}\left(Enc_{pk}(m_1) \times Enc_{pk}(m_1)\right) = m_1 + m_2 \tag{9}$$

$$Dec_{sk}\left(Enc_{pk}(m_1)^a\right) = a \times m_1 \tag{10}$$

One of the main properties of the Paillier cryptosystem is that it is based on a probabilistic encryption algorithm. Large-scale data sets are sparse matrices in which most of the elements are zero. In order to prevent the guessing of elements of the input data set, the Paillier cryptosystem has probabilistic encryption that does not encrypt two equal plaintexts with the same encryption key into the same ciphertext.

3.3. Arbitrarily Partitioned Data

In this work, arbitrarily partitioned data exist between multplei sites (K), where $K > 2$ is considered. In the arbitrarily partitioned data scheme, there is not any specific order of how the data set is split between the multiple sites. More specifically, let us give a data set $X = \{\mathbf{x}_1, \cdots \mathbf{x}_n\}$, consisting of n instances, and each instance in X contains m numeric features $\mathbf{x}_i = \{x_i^1 \cdots x_i^m\}$. X_i^j is the subset of the data set owned by party P_j, and then we have $X_i^1 \cup X_i^2 \cdots \cup X_i^K = X_i$ and $X_i^1 \cap X_i^2 \cdots \cap X_i^K = \emptyset$. Then, in each instance (X_i), site P_k has a number of features t_i^k, where $\sum_{p=1}^{K} t_i^k = m$ and each site's feature size does not have to be equal. If a site has the same features in each instance, then the arbitrary partition becomes a vertical partition.

4. Cryptographically Privacy-Preserving ELM (CPP-ELM)

In this section, we briefly introduce our assumptions in the security model, our notations, and the sequence diagram of the overall method. In Section 4.6, we explain our proposed privacy-preserving training of the ELM algorithm in detail.

4.1. Security Model

In this paper, the aim is to enable multiple parties (or servers) to cooperatively build the ELM classification model without retrieving their own confidential input data set. Our main assumption is that the input data set is divided between two or more parties that are willing to train an ELM classifier if nothing beyond the expected end results is revealed [2]. Our other assumption is that all parties follow the protocol; this is called a semi-honest security model [18,32].

4.2. Notations

The input instance $\mathbf{x}$ of data set $\mathscr{X}$ is shown in the form of $\mathbf{x} = \{x_1, \cdots x_n\}$. In this work, two or more parties that hold vertically partitioned data are considered.

- Boldface lowercase letters denote the vectors (e.g., $\mathbf{x}$).
- $[\![m]\!]$ denotes the ciphertext of a message m.
- $sign(m)$ denotes the sign of the plaintext number m.
- $\oplus$ denotes encrypted addition and $\otimes$ denotes encrypted multiplication with the public-key operation.

4.3. Floating Point Numbers

The Paillier encryption scheme operates only on integer numbers. Thus, the proposed protocols manipulate only integers. However, the ELM classification algorithm is typically applied to continuous data. Nonetheless, in the case of an input data set with real numbers in the protocol, we need to map floating point input data vectors into the discrete domain with a conversion function (i.e. scaling).

Let $ConvertToInteger : \mathbb{R}^m \rightarrow \mathbb{Z}^m$ be the corresponding function that multiplies its floating point number argument by an exponent $(K : 2^K)$ and then rounds them to the nearest integer value and thus supports finite precision. Eq. 11 shows the conversion function:

$$\widehat{\mathbf{x}} \leftarrow ConvertInteger(\mathbf{x}) \ where \ \mathbf{x} \in \mathbb{R}^m, \widehat{\mathbf{x}} \in \mathbb{Z}^m \tag{11}$$

4.4. Vertically Partitioned Data

The input data set that is used for finding an ELM-based classifier consists of m rows in n-dimensional vector space shown with $\mathscr{D} \in \mathbb{R}^{m \times n}$. Input matrix $\mathscr{D}$ is arbitrarily and vertically partitioned into k different sites of $P_0, P_1, \cdots, P_k$ and each attribute of rows is owned by a private site as depicted in Fig. 1. As denoted in Eq. 6, in the second phase of the ELM training, hidden layer output matrix $\mathbf{H}$ is computed using randomly generated hidden node parameters $\mathbf{w}$, $\mathbf{b}$, and β. The ELM training phase then computes the output weight vector, β, by multiplying $\mathbf{H}$ and $\mathbf{T}$. Each element of matrix $\mathbf{H}$ is computed with activation function g. The global G matrix is such that $G(\mathbf{w}_i, \mathbf{x}_i, b_i) = g(\mathbf{x}_i \cdot \mathbf{w}_i + b_i)$ for sigmoid or $G(\mathbf{w}_i, \mathbf{x}_i, b_i) = g(b_i - ||\mathbf{x}_i - \mathbf{w}_i||)$ for radial-based functions. The $(i, j)^{th}$ element of $\mathbf{H}$ is:

$$G(\mathbf{w}_i, \mathbf{x}_i, b_i) = sign(\mathbf{x}_i \cdot \mathbf{w}_i + b_i) \tag{12}$$

where $\mathbf{x}_i$ is the i^{th} row of the input data set $\mathscr{D}$, $\mathbf{w}_i$ is the hidden node input weight vector of the i^{th} row, and $\mathbf{x}_i, \mathbf{w}_i \in \mathbb{R}^n$.

$$\mathcal{D} = \left(\begin{array}{ccc|c|ccc} \multicolumn{3}{c|}{Party_1} & \cdots & \multicolumn{3}{c}{Party_k} \\ \hline \mathbf{x}_{1,1} & \cdots & \mathbf{x}_{1,t-1} & \cdots & \mathbf{x}_{1,t} & \cdots & \mathbf{x}_{1,n} \\ \mathbf{x}_{2,1} & \cdots & \mathbf{x}_{2,t-1} & \cdots & \mathbf{x}_{2,t} & \cdots & \mathbf{x}_{2,n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathbf{x}_{m,1} & \cdots & \mathbf{x}_{m,t-1} & \cdots & \mathbf{x}_{m,t} & \cdots & \mathbf{x}_{m,n} \end{array} \right)$$

Fig. 1. Vertically Partitioned Data Set $\mathscr{D}$.

Let $\mathbf{x}_i^1, \cdots \mathbf{x}_i^k$ be vertically partitioned vectors of an input instance $\mathbf{x}_i$, let $\mathbf{w}_j^1, \cdots \mathbf{w}_j^k$ be vertically partitioned vectors of the j^{th} hidden node input weight $\mathbf{w}_j$, and let $b_j^0, \cdots, b_j^k$ be the j^{th} node input bias over k different parties. Then the output of the j^{th} input node with the i^{th} instance of an input data set using k different parties is:

$$\begin{aligned} &sign(\mathbf{x}_i \cdot \mathbf{w}_j + b_j) = \\ &sign\left(\left(\mathbf{x}_i^0 \cdot \mathbf{w}_j^0 + b_j^0\right) + \cdots + \left(\mathbf{x}_j^k \cdot \mathbf{w}_j^k + b_j^k\right)\right) \end{aligned} \tag{13}$$

From Eq. 13, the computation of the hidden layer output matrix $\mathbf{H}$ can be distributed into k different parties by using the secure sum of matrices, such that:

$$\mathbf{H} = sign\left(\mathbf{T}_1 + \cdots + \mathbf{T}_k\right) \tag{14}$$

where:

$$\mathbf{T}_i = \begin{bmatrix} \left(\mathbf{x}_1^i \cdot \mathbf{w}_1^i + b_1^i\right) & \cdots & \left(\mathbf{x}_1^i \cdot \mathbf{w}_L^i + b_L^i\right) \\ \vdots & \ddots & \vdots \\ \left(\mathbf{x}_N^i \cdot \mathbf{w}_1^i + b_1^i\right) & \cdots & \left(\mathbf{x}_L^i \cdot \mathbf{w}_N^i + b_N^i\right) \end{bmatrix}_{N \times L} \tag{15}$$

4.5. *Sequence Diagram of Overall Method*

In this section, we present the conceptual level of the CPP-ELM. We define the converting of the flow of messages into the flow of methods. In the sequence diagram, we use the following notation for the actions executed by an actor when receiving a signal:

- **M**: Method invocation.
- :User, :ClientApp, :CloudApp: Actors

The CPP-ELM is composed of four steps, as shown in Fig. 2 - 3:

- Client initialization
- Information sharing
- Hidden layer output matrix computation
- Classifier model building

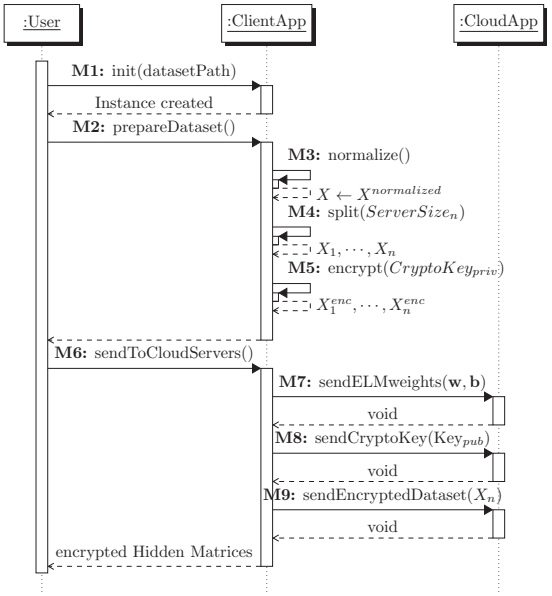


Fig. 2. Sequence diagram of overall method: Client computation.

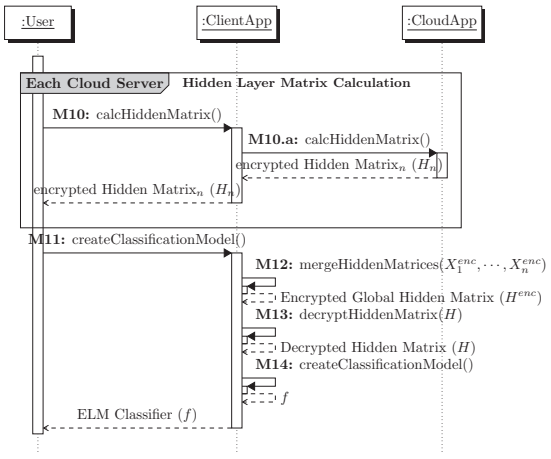


Fig. 3. Sequence diagram of overall method: server computation.

4.6. *Privacy-Preserving ELM Algorithm*

In this section, a cloud-based privacy-preserving multi-party CPP-ELM learning algorithm over arbitrarily partitioned data is presented.

4.6.1. *Client Initialization*

In the client initialization phase of the proposed method, first the client normalizes the input data set $\mathscr{D} = \{(\mathbf{x}, y)|\mathbf{x} \in \mathbb{R}^n, y \in \mathbb{R}\}$. Let $\mathbf{x}_{min}$ and $\mathbf{x}_{max}$ be an input data set of minimum and maximum values of each corresponding feature. Hence, the normalized data set can be written as:

$$\mathscr{D} = \frac{\mathscr{D} - \mathbf{x}_{min}}{\mathbf{x}_{max} - \mathbf{x}_{min}} \tag{16}$$

Next, the client splits the whole data set $\mathscr{D}$ based on the server size of the cloud systems and then it encrypts each element of the input data set $\mathscr{D}$ individually by using its public-key through the Paillier cryptosystem as shown in Eq. 17:

$$[\![\mathscr{D}_i]\!] = E_{pk}(\mathscr{D}_i) \ \forall i \in k \tag{17}$$

Algorithm 1 shows the overall process in the initialization phase.

Algorithm 1: Client Initialization.

Data: data set: $\mathscr{D} \in \mathbb{R}^{m \times n}$
server size: k
Crypto key length: l
Hidden layer size: nh
Result: Encrypted sub-data sets for each party P_s
begin
  $\mathscr{D} \longleftarrow normalize(\mathscr{D})$;
  // *Generate public and private keys* ;
  $Key_{pub}, Key_{priv} \longleftarrow KeyGen(l)$;
  // *Create weight vectors* ;
  $\mathbf{w} \longleftarrow random(nh, m)$ where $\mathbf{w} \in \mathbb{R}^{nh \times m}$;
  $\mathbf{b} \longleftarrow random(1, m)$ where $\mathbf{b} \in \mathbb{R}^m$;
  for $i \in k$ **do**
    create sub-data sets $\mathscr{D}^i$ with random feature index for party P_i ;
    $[\![\mathscr{D}]\!]^i \longleftarrow encrypt(\mathscr{D}^i, Key_{priv})$;

4.6.2. *Information Sharing*

In the second phase of the CPP-ELM, first the client sends the encrypted sub-data set $[\![\mathscr{D}_i]\!]$ to each party P^i.

Second, the client generates public and secret keys, as described in Algorithm 1: the client generates and assigns random weights $\mathbf{w} \in \mathbb{R}^n$ and $\mathbf{b} \in \mathbb{R}^n$, and then sends the weight vectors' corresponding items to each P_k. Algorithm 2 shows the overall process.

Algorithm 2: Information Sharing $\mathscr{D}^i_{enc}$.

Data: encrypted sub-data sets: $[\![\mathscr{D}]\!]^i \in \mathbb{R}^{m \times n}$
public key: Key_{pub}
weight vectors: $\mathbf{w} \in \mathbb{R}^{nh \times m}$ and $\mathbf{b} \in \mathbb{R}^m$
Result: information sharing is a success
begin
  for $i \in k$ **do**
    $res \longleftarrow sendELMWeights(\mathbf{w}, \mathbf{b})$;
    if $res == TRUE$ **then**
      $res \longleftarrow sendCryptoKey(Key_{pub})$;
      if $res == TRUE$ **then**
        $res \longleftarrow$ $sendEncryptedDataSet([\![\mathscr{D}]\!]^i)$;

4.6.3. *Hidden Layer Output Matrix Computation*

The encrypted sub-data sets $[\![\mathscr{D}_i]\!]$ of each server and weight vectors $\mathbf{w}, \mathbf{b}$ are used for computing the encrypted hidden layer output matrix for each data chunk $[\![H_i]\!], \forall i \in k$. The computed $[\![H_i]\!]$ is sent to the client. Considering a case where a party $k = 1$, we have:

$$\begin{aligned}[\![H_1]\!] =& [([\![\mathscr{D}_{1,s}]\!] \cdot \mathbf{w}) + b_s], \\ & where \ \mathscr{D} \in \mathbb{R}^{m \times n}, s = 1, \cdots, k\end{aligned} \tag{18}$$

The party (server) knows the plaintext weight vectors $\mathbf{w}, \mathbf{b}$. It can perform the required multiplication and additions with its own encrypted sub-data set $[\![\mathscr{D}_i]\!]$ by performing dot multiplication of each row with $\mathbf{w}$ and addition of b.

Algorithm 3 shows the calculation of the hidden layer output matrix for each server in the encrypted domain.

Algorithm 3: Hidden Layer Output Matrix Calculation.

Data:
Result: Encrypted sub-hidden layer output matrix $[\![H]\!]^s$ for each party P_s
begin
  for $i \in k$ **do**
    $[\![H]\!]^i \longleftarrow$ $calcHiddenLayerMatrix([\![\mathscr{D}]\!]^i, \mathbf{w}^i, \mathbf{b}^i)$;
    $sendOutputLayerToClient([\![H]\!]^i)$

4.6.4. *Classifier Model Building*

The client can now compute the last step given in Fig. 3, i.e. compute the value of the ELM-based classifier function in the private domain. First, the client decrypts each encrypted hidden layer output matrix $[\![H_i]\!]$ received from each party by using its own private key and then computes the global hidden layer output matrix $\mathbf{H}$. The smallest norm least-squares solution of the linear system [5] is:

$$\widehat{\beta} = \mathbf{H}^{+}\mathbf{T} \tag{19}$$

where $\mathbf{H}^{+}$ is the *Moore–Penrose generalized inverse* of matrix $\mathbf{H}$ [24].

Algorithm 4 shows the classifier model building in the decrypted domain.

Algorithm 4: Classification Model Building.

Data: Encrypted sub-hidden layer output matrix $[\![H]\!]^s$ for each party P_s
Result: final global classifier f
begin
  $H \longleftarrow \emptyset$;
  for $i \in k$ **do**
    $H \longleftarrow H +$ $decryptHiddenLayerMatrix([\![H]\!]^i, Key_{priv})$;
  find β then calculate final classifier f ;

5. Experiments

5.1. *Experimental Setup*

We have implemented our proposed protocols and the classifier training phase in Python by using the *scikit-learn* library for machine learning and the *PyPhe* library for the partially homomorphic encryption implementation.

To show the training phase time performance of the proposed protocols, we tested different public data sets with parties and encryption/decryption keys that have different lengths. For the experiments, we consider four different data sets from the UCI Machine Learning Repository called Australian, Breast Cancer, Ionosphere, and Sonar. Table 5.1 shows the details of the data sets.

Table 1. Details of Data Sets Used in Experiments.

Data Set	Sample	Class	Attributes
Ionosphere	351	2	34
Sonar	208	2	60
Breast Cancer	683	2	10
Australian	690	2	14

Each data set is arbitrarily vertically partitioned among each party ($K \in \{2,3,5,7,10\}$), and then the results in the encrypted-domain are compared with the results of the plain-domain. In plain-domain training, the conventional ELM training phase is performed for all data sets in a single pass. All experiments are repeated 5 times and the results are averaged. All data sets are publicly available in SVM-light format on the LIBSVM website.

The Ionosphere data set consists of a phased array of 16 high-frequency antennas with a total transmitted power on the order of 6.4 kilowatts. "Good" radar returns are those showing evidence of some type of structure in the ionosphere. "Bad" returns are those that do not; their signals pass through the ionosphere [27]. The data set contains 351 training instances.

The Sonar data set is used for the study of the classification of sonar signals using a neural network. The task is to train a network to discriminate between sonar signals bounced off a metal cylinder

and those bounced off a roughly cylindrical rock [7]. The data set contains 208 training instances.

The Breast Cancer data set contains 699 instances, with 458 benign (65.5%) and 241 (34.5%) malignant cases. Each instance is described by 9 attributes with an integer value [30].

The Australian data set concerns credit card applications. The data set contains 690 instances and each instance is described by 14 attributes [23].

5.2. Experimental Results

Table 2 shows the best performance of the conventional ELM method of each experimental data set.

Table 2. Accuracy Results in the Plain-Domain for Each Data Set.

Data Set	Accuracy
Ionosphere	0.8613
Sonar	0.7293
Breast Cancer	0.9592
Australian	0.8438

The proposed CPP-ELM training method has been implemented in Python 2.7 using the PyPhe library, version 1.0. Both the server and the clients are modeled as different process with the Python multiprocessing library. Each process sends the variables to each other via file exchange. The developed software is tested on a computer with 1.6 GHz Intel(R) i5 (R) processor and 4 GB of RAM running on a Windows 64-bit operating system. The system has been tested while varying the party size of each data set from 3 to 10, and with key lengths of 512 and 1024 bits. Performance results are shown in Table 3.

It is noted that the average training and model building time depends on the total number of instances used for classification. Fig. 4 shows the encryption time for each data set, which is affected by the number input instances and feature size. This process is computed at the client-side and can be pre-computed offline.

Table 3. Performance Results (in Seconds) for Each Data Set.

Data	Key	Number of Parties				
		2	3	5	7	10
Ion.	512	7477,43	8280,77	9743,51	11481,17	12542,17
	1024	39550,93	40020,45	45148,68	56751,82	82240,62
Sonar	512	8345,99	7824,33	8621,37	10192,59	11047,74
	1024	33775,32	30149,92	35418,41	38780,89	44357,66
Breast	512	7681,71	9245,26	10630,49	13672,9	18651,20
	1024	44035,76	48736,21	58554,61	72524,98	99986,21
Aust.	512	8709,58	10463,64	11629,93	15406,08	17817,07
	1024	39188,52	45621,76	52264,14	71074,53	79875,20

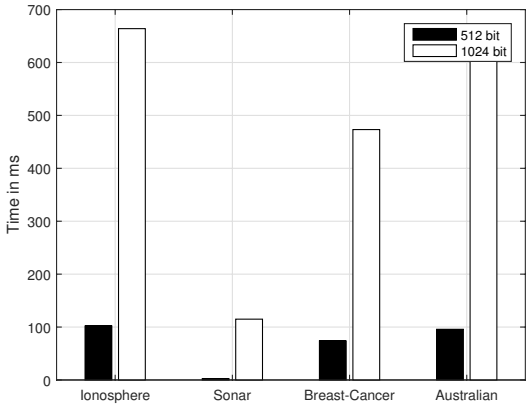
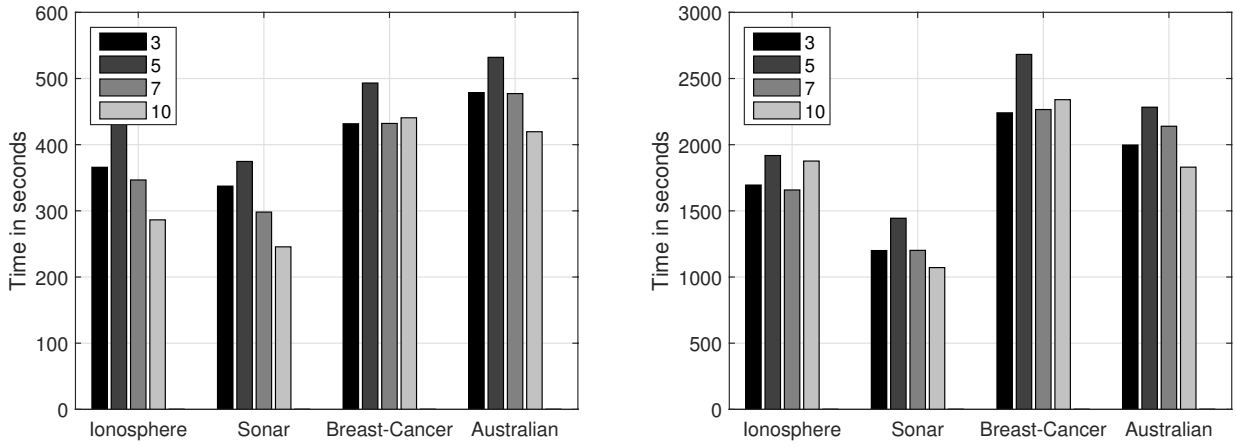


Fig. 4. Encryption Costs for Different Data Sets.

In the proposed method, CPP-ELM, there are three different factors (party size, complexity of the input data set, and key bit length) that affect the computation time according to the experimental results.

As show in Fig. 5, when the number of parties varies from 3 to 10, then the overall hidden layer output matrix **H** computation stays relatively stable in the proposed model, at about 5 minutes to 8.5 minutes for 512-bit key length encryption and 20 minutes to 43 minutes for 1024-bit key length encryption.

In cryptographic systems, all computations work with integers. Thus, all real numbers used in our algorithms are mapped into finite fields by using a scaling procedure. Before starting the learning phase, the client converts the input data set to integer representation by multiplying and rounding the results. In the last stage, in Algorithm 4, the client scales down the hidden layer output matrix **H**. This



(a) 512-Bit Key Length.

(b) 1024-Bit Key Length.

Figure 5: **H** Computation Time for Different Party Numbers and Data Sets.

scaling operation causes accuracy loss in the final classifier model. Table 4 shows the best performance of the CPP-ELM method for each experimental data set.

Table 4. Accuracy Results in the Encrypted-Domain on Each Data Set.

Data Set	512 Bits	1024 Bits
Ionosphere	0.7311	0.6598
Sonar	0.5976	0.5856
Breast Cancer	0.9433	0.9204
Australian	0.6877	0.6444

6. Conclusion

In this work, we proposed a privacy-preserving and practical multi-party ELM learning scheme over arbitrarily vertically partitioned data between two or more parties. We have also provided cryptographically secure protocols for computing the hidden layer output matrix and have shown how to aggregate encrypted intermediate matrices securely. In our proposed approach, the client encrypts its input data, creates arbitrarily and vertically partitioned data, and then uploads the encrypted messages to a cloud system. The cloud system can execute the most time-consuming operation of ELM training without knowing any confidential information. One interesting future work is to extend the privacy-preserving training to other existing classification algorithms.

We suggest that the method proposed here is appropriate for areas such as healthcare where data privacy is highly sensitive. Data privacy is protected by transferring the identification information to the remote cloud service provider in an encrypted manner. Using the high computational power provided by the cloud service provider, the classification model is built on the encrypted data set. In this way, the cloud service provider can create a classification model without reaching plaintext data.

References

1. Charu C. Aggarwal and Philip S. Yu. A condensation approach to privacy preserving data mining. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Bhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004*, volume 2992 of *Lecture Notes in Computer Science*, pages 183–199. Springer Berlin Heidelberg, 2004.
2. Ankur Bansal, Tingting Chen, and Sheng Zhong. Privacy preserving back-propagation neural network learning over arbitrarily partitioned data. *Neural Computing and Applications*, 20(1):143–150, 2011.

3. Guang bin Huang, Qin yu Zhu, and Chee kheong Siew. Extreme learning machine: A new learning scheme of feedforward neural networks. In *IN PROC. INT. JOINT CONF. NEURAL NETW*, pages 985–990, 2006.
4. Raphael Bost, Raluca Ada Popa, Stephen Tu, and Shafi Goldwasser. Machine learning classification over encrypted data. *IACR Cryptology ePrint Archive*, 2014:331, 2014.
5. A. Castaño, F. Fernández-Navarro, and C. Hervás-Martínez. Pca-elm: A robust and pruned extreme learning machine approach based on principal component analysis. *Neural Processing Letters*, 37(3):377–392, 2012.
6. Yao-Wei Fu, Hui-Ling Chen, Su-Jie Chen, Li-Juan Li, Shan-Shan Huang, and Zhen-Nao Cai. *Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings, Part I*, chapter A Hybrid Extreme Learning Machine Approach for Early Diagnosis of Parkinson's Disease, pages 342–349. Springer International Publishing, Cham, 2014.
7. R. Paul Gorman and Terrence J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural networks*, 1(1):75–89, 1988.
8. Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In *Information Security and Cryptology–ICISC 2012*, pages 1–21. Springer, 2012.
9. Li Guang, Wang Ya-Dong, and Su Xiao-Hong. A privacy preserving neural network learning algorithm for horizontally partitioned databases. *Inform. Technol. J*, 9:1–10, 2009.
10. Guang-Bin Huang and Lei Chen. Convex incremental extreme learning machine. *Neurocomputing*, 70(1618):3056 – 3062, 2007. Neural Network Applications in Electrical Engineering Selected papers from the 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005) 3rd International Work-Conference on Artificial Neural Networks (IWANN 2005).
11. Guang-Bin Huang and Lei Chen. Enhanced random search based incremental extreme learning machine. *Neurocomputing*, 71(1618):3460 – 3468, 2008. Advances in Neural Information Processing (ICONIP 2006) / Brazilian Symposium on Neural Networks (SBRN 2006).
12. Guang-Bin Huang, Lei Chen, and Chee-Kheong Siew. Universal approximation using incremental constructive feedforward networks with random hidden nodes. *Neural Networks, IEEE Transactions on*, 17(4):879–892, July 2006.
13. Guang-Bin Huang, Ming-Bin Li, Lei Chen, and Chee-Kheong Siew. Incremental extreme learning machine with fully complex hidden nodes. *Neurocomputing*, 71(46):576 – 583, 2008. Neural Networks: Algorithms and Applications 4th International Symposium on Neural Networks 50 Years of Artificial Intelligence: a Neuronal Approach Campus Multidisciplinary in Perception and Intelligence.
14. Guang-Bin Huang, Qin-Yu Zhu, and Chee-Kheong Siew. Extreme learning machine: Theory and applications. *Neurocomputing*, 70(13):489 – 501, 2006. Neural Networks Selected Papers from the 7th Brazilian Symposium on Neural Networks (SBRN '04) 7th Brazilian Symposium on Neural Networks.
15. Zhanglong Ji, Zachary Lipton, and Charles Elkan. Differential privacy and machine learning: a survey and review. *arXiv preprint arXiv:1412.7584*, 2014.
16. Hiroaki Kikuchi, Kei Nagai, Wakaha Ogata, and Masakatsu Nishigaki. Privacy-preserving similarity evaluation and application to remote biometrics authentication. *Soft Computing*, 14(5):529–536, 2009.
17. Xuan Li, Jin Li, and Faliang Huang. A secure cloud storage system supporting privacy-preserving fuzzy deduplication. *Soft Computing*, 20(4):1437–1448, 2015.
18. Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Advances in CryptologyCRYPTO 2000*, pages 36–54. Springer, 2000.
19. Yehuda Lindell and Benny Pinkas. Secure multiparty computation for privacy-preserving data mining. *Journal of Privacy and Confidentiality*, 1(1):5, 2009.
20. Stanley RM Oliveira and Osmar R Zaiane. Privacy preserving clustering by data transformation. *Journal of Information and Data Management*, 1(1):37, 2010.
21. Claudio Orlandi, Alessandro Piva, and Mauro Barni. Oblivious neural network computing via homomorphic encryption. *EURASIP Journal on Information Security*, 2007:18, 2007.
22. Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Advances in cryptologyEUROCRYPT99*, pages 223–238. Springer, 1999.
23. J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3):221–234, 1987.
24. Calyampudi Radhakrishna Rao and Sujit Kumar Mitra. *Generalized inverse of matrices and its applications*, volume 7. Wiley New York, 1971.
25. Saeed Samet and Ali Miri. Privacy-preserving back-propagation and extreme learning machine algorithms. *Data Knowl. Eng.*, 79-80:40–61, September 2012.
26. Jimmy Secretan, Michael Georgiopoulos, and Jose Castro. A privacy preserving probabilistic neural network for horizontally partitioned databases. In *Neural Networks, 2007. IJCNN 2007.*, pages 1554–1559, Aug 2007.
27. Vincent G Sigillito, Simon P Wing, Larrie V Hutton,

and Kile B Baker. Classification of radar returns from the ionosphere using neural networks. *Johns Hopkins APL Technical Digest*, 10(3):262–266, 1989.

28. Jiexiong Tang, Chenwei Deng, Guang-Bin Huang, and Baojun Zhao. Compressed-domain ship detection on spaceborne optical image using deep neural network and extreme learning machine. *Geoscience and Remote Sensing, IEEE Transactions on*, 53(3):1174–1185, March 2015.
29. Raylin Tso, Abdulhameed Alelaiwi, Sk Md Mizanur Rahman, Mu-En Wu, and M. Shamim Hossain. Privacy-preserving data communication through secure multi-party computation in healthcare sensor cloud. *Journal of Signal Processing Systems*, 89(1):51–59, Oct 2017.
30. William H. Wolberg and Olvi L. Mangasarian. Multisurface method of pattern separation for medical diagnosis applied to breast cytology. *Proceedings of the national academy of sciences*, 87(23):9193–9196, 1990.
31. Pan Yang, Xiaolin Gui, Jian An, and Feng Tian. An efficient secret key homomorphic encryption used in image processing service. *Security and Communication Networks*, 2017.
32. Zhiqiang Yang, Sheng Zhong, and Rebecca N. Wright. Privacy-preserving classification of customer data without loss of accuracy. In *SDM*, pages 92–102. SIAM, 2005.
33. Hwanjo Yu, Xiaoqian Jiang, and Jaideep Vaidya. Privacy-preserving svm using nonlinear kernels on horizontally partitioned data. In *Proceedings of the 2006 ACM Symposium on Applied Computing*, SAC '06, pages 603–610, New York, NY, USA, 2006. ACM.