Privacy-Preserving Multi-Class Support Vector Machine for Outsourcing the Data Classification in Cloud

Yogachandran Rahulamathavan, *Member, IEEE*, Raphael C.-W. Phan, Suresh Veluru, Kanapathippillai Cumanan, *Member, IEEE*, and Muttukrishnan Rajarajan, *Senior Member, IEEE*

Abstract—Emerging cloud computing infrastructure replaces traditional outsourcing techniques and provides flexible services to clients at different locations via Internet. This leads to the requirement for data classification to be performed by potentially untrusted servers in the cloud. Within this context, classifier built by the server can be utilized by clients in order to classify their own data samples over the cloud. In this paper, we study a privacy-preserving (PP) data classification technique where the server is unable to learn any knowledge about clients' input data samples while the server side classifier is also kept secret from the clients during the classification process. More specifically, to the best of our knowledge, we propose the first known client-server data classification protocol using support vector machine. The proposed protocol performs PP classification for both two-class and multi-class problems. The protocol exploits properties of Pailler homomorphic encryption and secure two-party computation. At the core of our protocol lies an efficient, novel protocol for securely obtaining the sign of Pailler encrypted numbers.

Index Terms—Privacy, data classification, cloud computing, support vector machine.

1 Introduction

In machine learning and artificial intelligence, data classification is a problem of identifying the category of unknown data sample by a classifier which is built using a training set of known data samples. Building a good classifier requires a large number of valid training samples; hence, it is not possible for individuals or small organizations to build their own classifier. Only viable solution to this problem is to outsource the data classification to third-party. Outsourcing the data classification mitigates the requirement of not only a large number of valid training data samples but also high computational and storage resources to clients (i.e. individuals or small organizations).

Recent advances in cloud computing replaces the traditional outsourcing techniques and provides various services to the clients over the Internet in a flexible manner (i.e. on-demand, pay-per use) [1]. This leads to a new paradigm of service where a server in a cloud could offer data classification to clients. In particular, the server can automatically process and classify the clients' data samples remotely based on privately owned training data samples.

However, releasing the data samples owned by the clients to the cloud raises privacy concerns since the data processed in the cloud are often outsourced to untrusted-third-party-servers (servers) [2], [3]. Furthermore, the server may not wish to disclose any details or parameters of its training data set even if it offers classification service to the client. Hence, in this paper we propose a method which preserves the privacy of both the client data samples and the server's training data while maximizing the benefits of the emerging cloud computing technology. The importance of cloud computing in the data classification can be categorized as follows:

1 the cloud is responsible for maintaining and updating training data set for classification
2 the cloud provides data classification as a service to any clients via Internet while preserving the privacy of client's data
3 the cloud helps to offload substantial amount of computation of clients

A widely used classification tool in many classification scenarios is support vector machine (SVM) due to its strong mathematical foundations and high reliability in many practical applications. The SVM classification involves two phases: training phase and testing phase. In the training phase, the server trains a SVM using labeled data (i.e. training data samples belong to different classes) in order to obtain the classification parameters. In the testing phase, any unlabeled data sample provided by a client can be

• *Yogachandran Rahulamathavan, Suresh Veluru and Muttukrishnan Rajarajan are with the School of Engineering and Mathematical Science, City University London, London, U.K. (e-mail: Yogachandran.Rahulamathavan.1@city.ac.uk; Suresh.Veluru.1@city.ac.uk; R.Muttukrishnan@city.ac.uk) Raphael C.-W Phan is with Faculty of Engineering, Multimedia University, 63100 Cyberjaya, Malaysia. (e-mail: raphael@mmu.edu.my). Kanapathippillai Cumanan is with the School of Electrical and Electronic Engineering, Newcastle University, Newcastle upon tyne, NE1 7RU, U.K. (e-mail: kanapathippillai.cumanan@ncl.ac.uk).*

classified and labeled to the matched class by a server using the classification parameters. Depending on the number of classes that the data samples are to be from, the SVM can be divided into two problems: two-class problem and multi-class problem.

In addition to the availability of a highly reliable SVM classification tool and easy access to cloud computing services, it is necessary to include security and privacy measures to ensure the privacy of the data provided by the client and the classification parameters made available by the server are preserved. Hence, it is crucial to develop a privacy-preserving (PP) protocol for SVM whereby a client could seek a server to run the SVM classifier to classify an unlabeled data sample; while maintaining the following privacy guarantees for both the client and server:

- hide the client's input data sample and the classification result from server
- hide the server side classification parameters from the client

In practical scenarios, the client might be a general practitioner (GP) who runs a private medical clinic, whereas classification service could be provided by a national hospital in order to detect a disease from patients' symptoms. This service could be provided by a hospital via Internet to any client.

In this paper, we present to the best of our knowledge, the first known client-server PP SVM protocol for both two-class and multi-class problems. More specifically, a client sends the input data sample in an encrypted format to the server. Then the server exploits the homomorphic encryption properties to perform the operations directly on the encrypted data sample. If there are any operations that cannot be handled by the homomorphic properties, then there will be a limited amount of interaction between the client and server based on two-party secure computation protocol. We assume that both the client and the server will execute the protocol correctly in order to maintain their reputation, hence they will behave in a semi-honest manner, i.e. they are honest but curious, so privacy is a real issue.

The remainder of this paper is organized as follows: In Section 2, we describe conventional SVM, i.e. the steps involved in training the SVM and classification in the plain-domain (PD). In Section 3, we first briefly describe one of the building blocks i.e. homomorphic encryption, and show how SVM classification can be extended to work in the encrypted-domain (ED) for the two-class problem setting. In particular, the core idea of finding the sign of an encrypted number is described in Section 3. In Section 4, we extend the protocol for two-class problem to the multi-class problem. We analyze the performance of these ED techniques in Section 5. We review related works in Section 6. Conclusions are discussed in Section 7.

Notation. We use boldface lower case letters to denote vectors; $(.)'$ denotes the transpose operator; $\|.\|_2$ the Euclidean norm; $\lfloor . \rceil$ the nearest integer approximation; $[\![m]\!]$ the encryption of message m; and $sign(m)$ denotes sign of the number m. The modular reduction operator is denoted by mod.

2 Support Vector Machine

SVM has been widely used in machine learning for data classification [10], [11], [20]. It has high generalization ability which provides high reliability in real-world applications such as image processing [16], computer vision [12], text mining [14], natural language processing [17], bioinformatics [18] and many more. SVM has been invented to classify a two-class problem, however, it has been extended later on to a multi-class problem. In a two-class problem, the goal of SVM is to separate both classes by a function, which is obtained from the training data samples. The multi-class problem can be solved by decomposing the multi-class problem into multiple two-class subproblems [19]. We formulate the classification functions of both two-class and multi-class problems in the following subsections. These classification functions are crucial to derive the PP SVM proposed in Sections 3 and 4.

2.1 Two-class Problem

We start with a training set of points $\tilde{\mathbf{x}}_i \in \mathbb{R}^n$, $i = 1, \dots, N$ where each point $\tilde{\mathbf{x}}_i$ belongs to one of the two classes denoted by a label $y_i \in \{-1, +1\}$, $i = 1, \dots, N$. Using these training data samples we can train a SVM to classify an unlabeled test sample. Initially, the training data need to be normalized to keep the numeric values of training samples on the same scale. This will prevent the samples with a large original scale from biasing the solution. Let us denote the normalized training data samples as $\mathbf{x}_i \in \mathbb{R}^n$, $i = 1, \dots, N$ where,

$$\mathbf{x}_i = \frac{\tilde{\mathbf{x}}_i - \bar{\mathbf{x}}}{\sigma^2}, \forall i, \tag{1}$$

where $\bar{\mathbf{x}}$ and σ denote the mean and standard deviation of the training data samples. Depending on the separability of the training data, the two-class problem is further divided into two: linear classification problem and non-linear classification problem.

2.1.1 Linear Classification Problem

The goal of linear classification problem is to obtain two parallel hyperplanes as shown in Fig. 1, $\mathbf{w}'\mathbf{x} + b = -1$ and $\mathbf{w}'\mathbf{x} + b = +1$, where $\mathbf{w}$ and b are the classification parameters obtained during the training process. Both hyperplanes separate the training data of the two classes such that the distance between those hyperplanes is maximized.

After the training stage, we can classify an unlabeled test sample, $\tilde{\mathbf{t}} \in \mathbb{R}^n$. Before the classification,

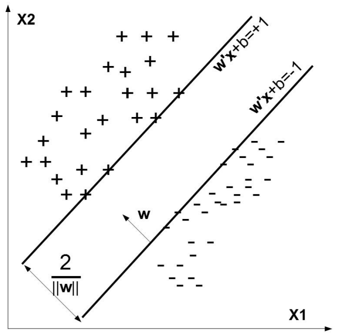

Fig. 1. Training data samples for two different classes are denoted by $+$ and $-$ signs.

the test sample is required to be normalized similar to (1) as

$$\mathbf{t} = \frac{\tilde{\mathbf{t}} - \bar{\mathbf{x}}}{\sigma^2}. \tag{2}$$

Now the normalized test sample, $\mathbf{t}$, can be substituted into the following classification function

$$f(\mathbf{t}) = sign(\mathbf{w}'\mathbf{t} + b) = sign\left(\sum_{s \in S} \alpha_s y_s \mathbf{x}_s'\mathbf{t} + b\right), \tag{3}$$

where $f(\mathbf{t}) \in \{-1, +1\}$, α_i, $i = 1, \ldots, N$ are Lagrangian variables and $\mathbf{x}_s$, $s = 1, \ldots, |S|$ are support vectors. If $f(\mathbf{t}) = +1$ then the test sample $\tilde{\mathbf{t}}$ belongs to $+ve$ class else it belongs to $-ve$ class. Please note that a decision function $d(\mathbf{t})$ can be extracted from (3) as

$$d(\mathbf{t}) = \mathbf{w}'\mathbf{t} + b = \sum_{s \in S} \alpha_s y_s \mathbf{x}_s'\mathbf{t} + b, \tag{4}$$

where $\mathbf{w}'\mathbf{x} + b = 0$ denotes the decision-hyperplane which lies between the two hyperplanes (i.e. $\mathbf{w}'\mathbf{x}+b = -1$ and $\mathbf{w}'\mathbf{x} + b = +1$).

2.1.2 Non-linear Classification Problem

In the previous section, we have discussed the classification problem where the training data samples were linearly separable. However, it has been proven in the literature that the similar approach can be used for non-linear classification problems using kernel tricks [21], [22]. Hence, the non-linear classification algorithm is formally similar to linear classification algorithms except that the dot product in (3) (i.e. $\mathbf{x}_s'\mathbf{t}$) is replaced by a various non-linear kernel functions. These kernel functions map the data samples into a higher dimensional feature space; hence the non-linear classification problem transformed into linear classification problem (see Fig. 2). In this work, we only consider a polynomial kernel. Hence, the dot product between $\mathbf{x}_s$ and $\mathbf{t}$ in (3) can be replaced as

$$\mathbf{x}_s'\mathbf{t} \Rightarrow K(\mathbf{x}_s, \mathbf{t})^p = (\mathbf{x}_s'\mathbf{t} + 1)^p, \tag{5}$$

where p denotes the degree of the polynomial. Hence, the classification function in (3) can be modified as

$$f(\mathbf{t}) = sign\underbrace{\left(\sum_{s \in S} \alpha_s y_s (\mathbf{x}_s'\mathbf{t} + 1)^p + b\right)}_{\text{decision function } d(\mathbf{t})}. \tag{6}$$

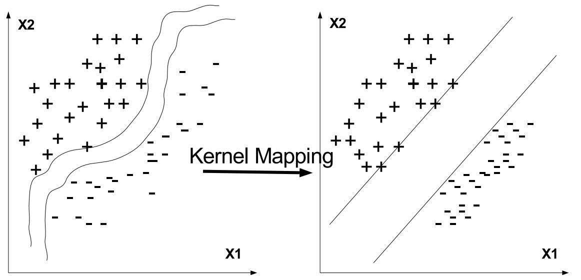

Fig. 2. Non-linear classification problem converted into linear classification problem after the kernel mapping.

2.2 Multi-class Problem

In order to solve a multi-class problem using SVM, it has to be decoupled into multiple two-class subproblems. There are two classical approaches to decouple the multi-class problem: one-versus-all (1VA) and one-versus-one (1V1) [13], [15].

2.2.1 One-versus-all Approach.

For a given N_c number of classes, we form N_c number of subproblems. Hence, we train N_c number of SVMs. For j^{th} subproblem, we train a SVM using j^{th} class training data samples as $+ve$ class and all the remaining samples as $-ve$ class. In the testing phase, for a given normalized test sample, $\mathbf{t}$, the matching class, M_c, can be obtained by modifying (6) as

$$M_c = \underset{j=1,\ldots,N_c}{argmax}\left(\sum_{s \in S_j} \alpha_s y_s (\mathbf{x}_s'\mathbf{t} + 1)^p + b_j\right), \tag{7}$$

where subscript j denotes the variables associated with the j^{th} subproblem. Hence, a subproblem with the largest margin value for the decision function (i.e. $\sum_{s \in S_j} \alpha_s y_s (\mathbf{x}_s'\mathbf{t} + 1)^p + b_j$) will be chosen.

2.2.2 One-versus-one Approach.

For this approach, we train $\frac{N_c(N_c-1)}{2}$ number of subproblems, where SVM for each subproblem trained using data only from two classes. If we consider training samples from classes i (i.e. $+ve$ class) and j (i.e. $-ve$ class) for a given subproblem, then the classification function in (6) can be modified as

$$f_{i,j}(\mathbf{t}) = sign\left(\sum_{s \in S_{i,j}} \alpha_s y_s (\mathbf{x}_s'\mathbf{t} + 1)^p + b_{i,j}\right), \tag{8}$$

where subscript i, j denotes the variables associated with the $(i, j)^{th}$ subproblem. The matching class for the given test sample $\mathbf{t}$ can be obtained using majority voting approach as

$$M_c = \underset{i=1,\ldots,N_c}{argmax} \left(\sum_{j=1, j\neq i}^{N} f_{i,j}(\mathbf{t}) \right), \quad (9)$$

where $f_{j,i}(\mathbf{t}) = -f_{i,j}(\mathbf{t})$, $\forall i, j$.

3 Classification in the Encrypted-Domain

We consider a client-server model where the server maintains a data set with training samples and obtains the classification parameters required for the classification functions (i.e. (6), (7) and (9)). In this section, we show how to preserve the privacy of the test sample, $\mathbf{t}$, and the classification result from the server and the SVM parameters from the client when the server executes the classification function to classify a test sample. Depending on the nature of the problem server executes one of the classification functions from (6), (7) and (9). First, let us explain the required building blocks in the next section.

3.1 Homomorphic Encryption

For concreteness and without loss of generality, our descriptions are based on the Paillier cryptosystem [9] although any other homomorphic encryption schemes could be used. The Paillier cryptosystem is a public-key encryption scheme. It's provable semantic security is based on the decisional composite residuosity problem. Hence, it is mathematically intractable to decide whether an integer z is an n-residue modulo n^2 for some composite n, i.e. whether there exists some $y \in \mathbb{Z}^*_{n^2}$ such that $z = y^n \mod n^2$. Let $n = pq$ where p and q are two large prime numbers. A message $m \in \mathbb{Z}_n$ can be encrypted using the Paillier cryptosystem as $[\![m]\!] = g^m r^n \bmod n^2$ where $g \in \mathbb{Z}^*_{n^2}$ and $r \in \mathbb{Z}^*_n$. The Paillier cryptosystem is said to be an additively homomorphic cryptosystem because for some given encryptions, $[\![m_1]\!]$ and $[\![m_2]\!]$, the encryption $[\![m_1+m_2]\!]$ of the sum $m_1 + m_2$ in the PD and the encryption $[\![m_1.\alpha]\!]$ of the product of m_1 with a constant α in the PD can respectively be computed efficiently in the ED as

$$[\![m_1 + m_2]\!] = [\![m_1]\!][\![m_2]\!],\ [\![m_1.\alpha]\!] = [\![m_1]\!]^{\alpha}. \quad (10)$$

In the setting considered by this paper, the client distributes its public-key to the server while keeping its private-key secret. The server performs encryptions under this public-key and exploits the homomorphic properties of the Paillier cryptosystem to perform the required linear operations in the ED. However, only the client can decrypt any encrypted messages using its corresponding private-key.

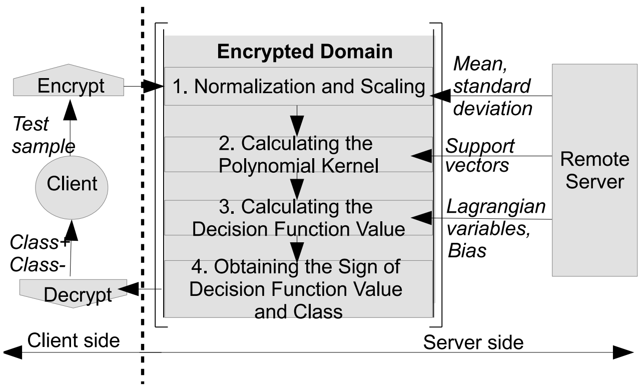
Encrypted Domain
1. Normalization and Scaling
2. Calculating the Polynomial Kernel
3. Calculating the Decision Function Value
4. Obtaining the Sign of Decision Function Value and Class
Encrypt
Test sample
Client
Class+
Class-
Decrypt
Mean, standard deviation
Support vectors
Lagrangian variables, Bias
Remote Server
Client side
Server side

Fig. 3. The block diagram of proposed PP SVM.

3.2 Encrypting Negative Integer

To represent negative integers, we exploit cyclic property in modulo arithmetic. We represent -1 by $n-1$ since $n - 1 \equiv -1$ in $mod\ n$. When the message space is $m \in \mathbb{Z}_n$, we represent Paillier encryption of $-m$ using homomorphic property as follows: $[\![-m]\!]$=$[\![-1 \times m]\!]$=$[\![m]\!]^{-1}$=$[\![m]\!]^{n-1}$. In the SVM classification problem considered in the work, Paillier security parameter $n >> \ classification\ variables$, hence, errors due to overflow can be avoided.

3.3 Two-class Problem in the Encrypted-domain

First we present our technique for SVM classification in the ED that solves the two-class problem. In particular, we show how server can execute (6) when the test sample is in the ED. Let us assume that each class is represented by an associated string $class_i$, $i \in \{-, +\}$. Fig. 3 depicts the overview of the proposed PP SVM: client supplies a test sample in an encrypted format to the server. The server has the normalization parameters (i.e. mean and standard deviation) and the SVM parameters (i.e. support vectors, Lagrangian variables and bias) in the PD. Hence, the server executes the four steps shown in Fig. 3, in the ED, in order to classify the encrypted test sample.

Initially, the client encrypts each element of the test sample $\tilde{\mathbf{t}}$ individually using the public-key and sends $[\![\tilde{\mathbf{t}}]\!]$ to the server. In general, the training samples, test sample and the variables involved in SVM classification are continuous data. Since the Paillier cryptosystem only supports integers, all the variables involved in the decision function (6) need to be quantized to the nearest integer value before encryption. However, if we quantize those variables without any preprocessing then this will deteriorate the classification accuracy. To address this issue, we scale the decision function in (6) using a positive number γ^{2p+1} before quantization. Hence, (6) can be modified into (11) (as shown at the top of the next page). Note that (6) and (11) provide the same results as long as $\gamma > 0$. It is obvious that if the scaling factor γ is sufficiently

$$f(\mathbf{t}) = sign\left\{\gamma^{2p+1}\left[\sum_{s\in S}\alpha_s y_s(\mathbf{x}_s'\mathbf{t}+1)^p + b\right]\right\} = sign\left\{\overbrace{\sum_{s\in S}\gamma(\alpha_s y_s)\underbrace{[(\gamma\mathbf{x}_s)'(\gamma\mathbf{t})+\gamma^2 1]^p}_{\text{polynomial kernel } K_s^p} + \gamma^{2p+1}b}^{\text{decision function } d(\mathbf{t})}\right\}. \quad (11)$$

$$\sum_{s\in S}\gamma(\alpha_s y_s)[(\gamma\mathbf{x}_s)'(\gamma\mathbf{t})+\gamma^2 1]^p + \gamma^{2p+1}b \approx \sum_{s\in S}\lfloor\gamma\alpha_s y_s\rceil\left(\lfloor\gamma\mathbf{x}_s\rceil'\lfloor\gamma\mathbf{t}\rceil + \lfloor\gamma^2 1\rceil\right)^p + \lfloor\gamma^{2p+1}b\rceil. \quad (12)$$

large (i.e. $\gamma > 10^6$) then the solution obtained before and after the quantization of variables will be nearly equal as shown in (12). This result is crucial in order to obtain higher accuracy when executing (11) in the ED. Executing (11) in the ED involves four different steps (see Fig. 3). In the following subsections, we propose methods to compute those four steps in the ED when the test sample, $\tilde{\mathbf{t}}$, is in encrypted format.

3.3.1 Step 1–Normalization and Scaling

As a first step, the test sample $\tilde{\mathbf{t}} = [\tilde{t}_1, \ldots, \tilde{t}_n]'$ has to be normalized as in (2). Let us define a mean vector $\bar{\mathbf{x}} = [\bar{x}_1, \ldots, \bar{x}_n]'$ and a normalized test sample as $\mathbf{t} = [t_1, \ldots, t_n]'$. Hence, the scaled and normalized test sample $\gamma\mathbf{t}$ can be written as

$$\begin{aligned}\gamma\mathbf{t} &= \gamma\left(\frac{\tilde{\mathbf{t}}-\bar{\mathbf{x}}}{\sigma^2}\right) = \frac{\gamma}{\sigma^2}\tilde{\mathbf{t}} - \frac{\gamma\bar{\mathbf{x}}}{\sigma^2},\\ \Rightarrow \gamma t_i &= \frac{\gamma}{\sigma^2}\tilde{t}_i - \frac{\gamma\bar{x}_i}{\sigma^2}, \quad \forall i. \end{aligned} \quad (13)$$

However, these operations have to be performed in the ED by a server who receives only the encrypted test sample, $[\![\tilde{\mathbf{t}}]\!] = \left[[\![\tilde{t}_1]\!], \ldots, [\![\tilde{t}_n]\!]\right]'$, from the client. Since the server knows the vector $\bar{\mathbf{x}}$, and scalars γ and σ in the PD, he can easily compute the values $-\frac{\gamma\bar{x}_i}{\sigma^2} = (-1).\frac{\gamma\bar{x}_i}{\sigma^2}$, $\forall i$ and encrypt each of its components by exploiting homomorphic properties as $[\![(-1).\frac{\gamma\bar{x}_i}{\sigma^2}]\!] = [\![\frac{\gamma\bar{x}_i}{\sigma^2}]\!]^{(-1)}$, $\forall i$. Similarly, encryption of $\frac{\gamma}{\sigma^2}\tilde{t}_i$ can be obtained as $[\![\frac{\gamma}{\sigma^2}\tilde{t}_i]\!] = [\![\tilde{t}_i]\!]^{\frac{\gamma}{\sigma^2}}$, $\forall i$. Hence, the scaled and normalized value of the test sample in (13) can be obtained in the ED as follows:

$$\begin{aligned}[\![\gamma t_i]\!] &= [\![\frac{\gamma}{\sigma^2}\tilde{t}_i - \frac{\gamma\bar{x}_i}{\sigma^2}]\!] = [\![\frac{\gamma}{\sigma^2}\tilde{t}_i]\!].[\![-\frac{\gamma\bar{x}_i}{\sigma^2}]\!], \quad \forall i,\\ &= [\![\tilde{t}_i]\!]^{\frac{\gamma}{\sigma^2}}.[\![\frac{\gamma\bar{x}_i}{\sigma^2}]\!]^{(-1)}, \quad \forall i. \end{aligned} \quad (14)$$

Note that every computation in (14) is performed by the server without interacting the client. Now the server obtains normalized and scaled test sample in ED as $[\![\gamma\mathbf{t}]\!] = [[\![\gamma t_1]\!], \ldots, [\![\gamma t_n]\!]]'$.

3.3.2 Step 2–Computing the Polynomial Kernel

The encrypted test sample $[\![\gamma\mathbf{t}]\!]$ is used to compute the polynomial kernel $K_s^p = [(\gamma\mathbf{x}_s)'(\gamma\mathbf{t}) + \gamma^2 1]^p$, $\forall s$ in (11) in the ED. Let us define the support vector $\mathbf{x}_s = [x_{s,1}, \ldots, x_{s,n}]'$, $\forall s$. Consider a case when $p = 1$, for which we have,

$$\begin{aligned}K_s &= [(\gamma\mathbf{x}_s)'(\gamma\mathbf{t}) + \gamma^2 1], \quad \forall s,\\ &= [(\gamma x_{s,1}).(\gamma t_1) + \ldots + (\gamma x_{s,n}).(\gamma t_n) + \gamma^2 1], \quad \forall s. \end{aligned} \quad (15)$$

Thus, the server can compute (15) in the ED as

$$\begin{aligned}[\![K_s]\!] &= [\![(\gamma x_{s,1}).(\gamma t_1) + \ldots + (\gamma x_{s,n}).(\gamma t_n) + \gamma^2 1]\!],\\ &= [\![(\gamma x_{s,1}).(\gamma t_1)]\!].\ldots.[\![(\gamma x_{s,n}).(\gamma t_n)]\!].[\![\gamma^2 1]\!],\\ &= [\![\gamma t_1]\!]^{\gamma x_{s,1}}.\ldots.[\![\gamma t_n]\!]^{\gamma x_{s,n}}.[\![\gamma^2 1]\!], \end{aligned} \quad (16)$$

$\forall s$. This can be computed as follows: since the server knows the scalars, $x_{s,i}, \forall s, i$, in the PD, he can perform the required multiplications by exploiting (10). For example, in order to multiply the first component, the server has to compute $[\![\gamma t_1]\!]^{\gamma x_{s,1}}$. To obtain the sum of all these products he has to multiply the encryptions of each component with other components. For the case $p = 1$, the server computes the kernel value without any interaction with the client. If $p > 1$ (i.e. K_s^p) then, the kernel could be computed via a secure two-party computation technique between the server and client. If the server sends $[\![K_s]\!]$ to the client then the client can decrypt it using her private-key, then raise the decrypted value by degree p and send the encrypted $[\![K_s^p]\!]$ back to the server. However, this will leak the server side SVM parameters to the client. Instead, the server blinds the encrypted $[\![K_s]\!]$ with some uniformly random element r_s from the PD as $[\![\tilde{K_s}]\!] = [\![K_s.r_s]\!] = [\![K_s]\!]^{r_s}$. Note that the server generates a fresh random element r_s for each $[\![K_s]\!]$. Then server sends the blinded $[\![\tilde{K_s}]\!]$ to the client. The client decrypts $[\![\tilde{K_s}]\!]$ using her private-key, computes $\tilde{K_s^p}$ and sends $[\![\tilde{K_s^p}]\!]$ back to the server. Now the server can remove the random mask and computes $[\![K_s^p]\!]$ as

$$[\![K_s^p]\!] = [\![\tilde{K_s^p}]\!]^{r_s^{-p}}, \quad (17)$$

which provides the required results due to the following homomorphic relation:

$$[\![\tilde{K_s^p}]\!]^{r_s^{-p}} = [\![\tilde{K_s^p}.r_s^{-p}]\!] = [\![K_s^p.r_s^p.r_s^{-p}]\!] = [\![K_s^p]\!].$$

Note that due to the blinding, the client does not learn the values of K_s even though she had access to $\tilde{K_s}$ in the PD (privacy analysis is given in Subsection 3.2.5).

3.3.3 Step 3–Calculating the Decision Function

The server can now execute the third step in Fig. 3 i.e. compute the value of the decision function in (11) in the ED as follows:

$$\begin{aligned}[\![d(\mathbf{t})]\!] &= [\![\sum_{s\in S}\gamma(\alpha_s y_s)[(\gamma\mathbf{x}_s)'(\gamma\mathbf{t})+\gamma^2 1]^p+\gamma^{2p+1}b]\!],\\ &= [\![\sum_{s\in S}\gamma(\alpha_s y_s)K_s^p+\gamma^{2p+1}b]\!],\\ &= [\![\gamma^{2p+1}b]\!].\prod_{s\in S}[\![K_s^p]\!]^{\gamma(\alpha_s y_s)}. \quad (18)\end{aligned}$$

This can be done as follows. Since the server knows the values of γ and $(\alpha_s y_s)$ in the PD, he can perform the required multiplications using (10). As an example, in order to compute $\gamma(\alpha_s y_s)K_s^p$ in the ED the server computes $[\![K_s^p]\!]^{\gamma(\alpha_s y_s)}$ for a given s. To obtain the sum of all these products together with $\gamma^{2p+1}b$, he multiplies the encryptions with each other.

The only part missing now is the computation of sign value of $d(\mathbf{t})$ and inform the matched class i.e. $class_+$ or $class_-$ to the client. Note that, this classification result cannot be revealed to the server. In the next section, we describe the required secure protocol to obtain the classification result.

3.3.4 Step 4–Obtaining the Sign of the Value of the Decision Function

This section studies how to obtain a sign of Paillier encrypted number. According to (11), $f(\mathbf{t}) = sign(d(\mathbf{t}))$. However, the server now has the value of $d(\mathbf{t})$ in the ED as in (18). Let us assume that $\mid d(\mathbf{t}) \mid< 10^l$, $l \in \mathbb{Z}$ in the PD. Note that since the training and test data samples are normalized, the value of l can be determined using the range of $\sum_{s\in S}\alpha_s y_s(\mathbf{x}_s'\mathbf{t}+1)^p+b$ and the scaling factor γ^{2p+1} in (11).

Now the server computes a new variable in the ED as

$$[\![z]\!] = [\![10^l + d(\mathbf{t})]\!] = [\![10^l]\!].[\![d(\mathbf{t})]\!]. \quad (19)$$

Since $|d(\mathbf{t})| < 10^l$, the most significant digit of z in PD is either 1 (i.e. if $d(\mathbf{t}) > 0$) or 0 (i.e. if $d(\mathbf{t}) < 0$). Let us denote the most significant digit of z as $\tilde{z} \in \{1, 0\}$. If $\tilde{z} = 1$ then test sample belongs to $class_+$ and if $\tilde{z} = 0$ then test sample belongs to $class_-$. Hence, the matched class M_c, can be obtained as

$$M_c = \tilde{z}.(class_+ - class_-) + class_-. \quad (20)$$

The most significant digit $\tilde{z}$ could be computed using the following linear operation:

$$\tilde{z} = 10^{-l}.\left[z - (z \mod 10^l)\right], \quad (21)$$

where subtraction sets the least significant digits of z into 0 while the multiplication shifts the most significant digit down. Since z in (19) is in the ED the server needs to obtain the $\tilde{z}$ in (21) in the ED. This can be done as follows:

$$\begin{aligned}[\![\tilde{z}]\!] &= [\![10^{-l}.\left[z-(z \mod 10^l)\right]]\!],\\ &= \left([\![z]\!].[\![z \mod 10^l]\!]^{-1}\right)^{10^{-l}}. \quad (22)\end{aligned}$$

However, z available at the server is encrypted; thus similar to the process leading to the server being able to compute (17), the server engages the client in a secure two-party computation protocol to compute $[\![z \mod 10^l]\!]$.

The server blinds the $[\![z]\!]$ using an uniformly random value r as

$$[\![z_r]\!] = [\![z+r]\!] = [\![z]\!].[\![r]\!],$$

and this is sent to the client who decrypts the blinded $[\![z_r]\!]$ and reduces z_r mod 10^l. The result i.e. z_r mod 10^l is then encrypted and sent back to the server who retrieves $[\![z \text{ mod } 10^l]\!]$ as

$$[\![z \text{ mod } 10^l]\!] = [\![z+r \text{ mod } 10^l]\!].[\![r \text{ mod } 10^l]\!]^{-1}[\![\lambda]\!]^{10^l},$$

where $\lambda \in \{0, 1\}$ used to avoid the underflow (i.e. $\lambda = 0$ if $z + r$ mod $10^l > r$ mod 10^l or $\lambda = 1$ if $z + r$ mod $10^l < r$ mod 10^l). The client knows $z + r$ mod 10^l in PD while server knows r mod 10^l in PD. Comparing two integers in private has been widely studied in literature [27], [28]. Now the server can compute $[\![\tilde{z}]\!]$ using (22). The matching class label of the test sample can be computed in the ED using $[\![\tilde{z}]\!]$ and (20) as follows:

$$\begin{aligned}[\![M_c]\!] &= [\![\tilde{z}.(class_+ - class_-) + class_-]\!],\\ &= [\![\tilde{z}]\!]^{(class_+ - class_-)}.[\![class_-]\!]. \quad (23)\end{aligned}$$

Now $[\![M_c]\!]$ can be returned to the client who decrypts it to find the matched label of the test sample.

Fig. 4 shows the flow diagram of the proposed method. Note that, since the classification process is in the ED, it is not feasible for the server to obtain the classification result in PD without prior knowledge of the test sample. The computational complexity during the interactions is always less than the computational complexity required to encrypt the test sample because the number of elements in the test sample is sufficiently larger than the number of support vectors (i.e. $s \ll n$). Hence, efficiency of the proposed algorithm only depends on the number of interaction between the client and server, where server interacts with a client only when the homomorphic properties are not sufficient to complete the task. Since the server masks the kernel value, $[\![K_s]\!]$, by multiplying by random noise, r_s, instead of addition, number of interaction required to compute decision function in (18) is only one for any number of the polynomial (i.e. $p > 1$).

3.3.5 Privacy Analysis of the Proposed Algorithm

In the proposed work, the client uses Paillier homomorphic encryption to encrypt the test sample. The

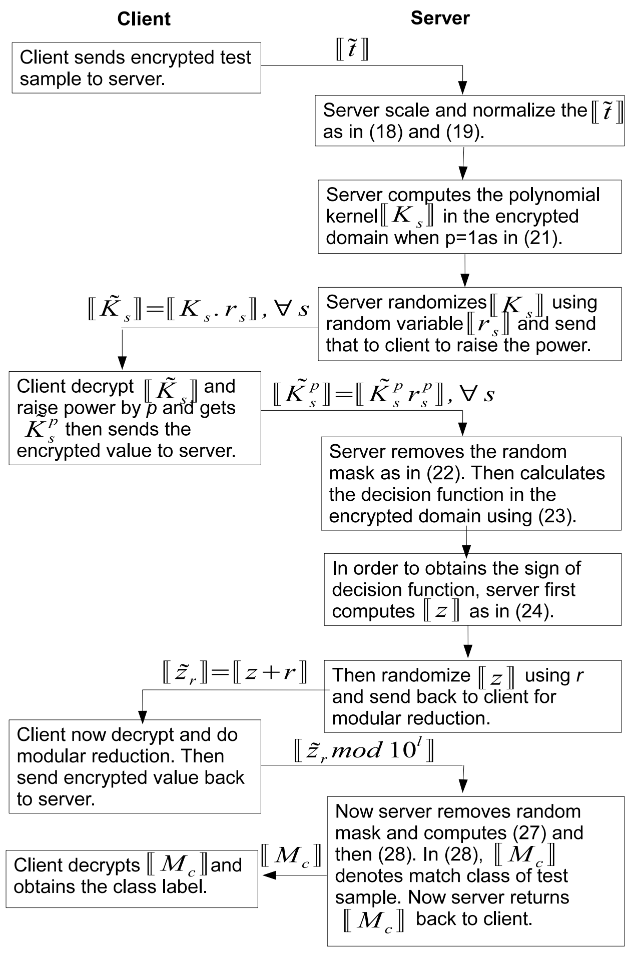
Client
Server
Client sends encrypted test sample to server.
$[\![\tilde{t}]\!]$
Server scale and normalize the $[\![\tilde{t}]\!]$ as in (18) and (19).
Server computes the polynomial kernel $[\![K_s]\!]$ in the encrypted domain when p=1as in (21).
$[\![\tilde{K}_s]\!]=[\![K_s.r_s]\!], \forall s$
Server randomizes $[\![K_s]\!]$ using random variable $[\![r_s]\!]$ and send that to client to raise the power.
Client decrypt $[\![\tilde{K}_s]\!]$ and raise power by p and gets K_s^p then sends the encrypted value to server.
$[\![\tilde{K}_s^p]\!]=[\![\tilde{K}_s^p r_s^p]\!], \forall s$
Server removes the random mask as in (22). Then calculates the decision function in the encrypted domain using (23).
In order to obtains the sign of decision function, server first computes $[\![z]\!]$ as in (24).
$[\![\tilde{z}_r]\!]=[\![z+r]\!]$
Then randomize $[\![z]\!]$ using r and send back to client for modular reduction.
Client now decrypt and do modular reduction. Then send encrypted value back to server.
$[\![\tilde{z}_r mod\ 10^l]\!]$
Now server removes random mask and computes (27) and then (28). In (28), $[\![M_c]\!]$ denotes match class of test sample. Now server returns $[\![M_c]\!]$ back to client.
$[\![M_c]\!]$
Client decrypts $[\![M_c]\!]$and obtains the class label.

Fig. 4. The flow diagram of the proposed algorithm.

server conducts the mathematical operations in the ED. Hence, the server cannot learn about the test sample, the intermediate results or the classification outcome. However, the homomorphic encryption is not sufficient to complete any non-linear operations such as exponentiation in the ED. Hence, the server and client securely interact with each other to perform any non-linear operations. For this interaction, we exploited a secure two-party computation technique and assumed that the parties are semi-honest.

In a semi-honest model, each party follows the rules of the protocol using its correct input. However, the parties are free to use the intermediate values obtained during the execution of the protocol in order to learn about other parties input [37]–[39]. However, a secure two-party protocol must satisfy the following privacy definition in a semi-honest model:

Privacy definition for the secure two-party computation: a secure two-party protocol should not reveal more information to a semi-honest party than the information that can be induced by looking at that party's input and output. The formal proof of the definition can be found in [36].

Let us verify whether the proposed two-party computation satisfies the privacy definition. In our protocol, the server cannot learn any information since all results are protected by Paillier homomorphic encryption. Hence, adversary in our model is a client. The adversary follows the protocol correctly, but try to infer the classification parameters of the server. The proposed algorithm composed of four steps: 1. normalization and scaling, 2. computing the polynomial kernel, 3. calculating the decision function and 4. obtaining the sign of the value of the decision function. Interaction between the client and server happens only in the Step 2 (when $p > 1$) and the Step 4. We now formally analyses whether the client can infer the server side parameters from these steps. Let us begin with Step 2 followed by Step 4.

In Step 2, the server interacts with the client in order to exponentiate the encrypted value when polynomial parameter $p > 1$. This interaction is more important than step 4 as it is directly including the support vectors of the training data during the computation, e.g (15). Let us consider the worse case scenario of semi-honest model where the adversary (client) trying infer support vectors by feeding selected input instead of actual test sample value during the interaction. There are $|S|$ number of support vectors where each support vector contained n number of elements. Lets assume a simple threat model to see whether an adversary can find the first element from s^{th} support vector $\mathbf{x}_s$ i.e. $x_{s,1}$ $\forall s \in S$. To verify this, lets assume, the client could send the following test sample $\mathbf{t} = [1\ 0\ 0\ 0 \ldots 0]^T$ in the encrypted format as many times as he can (assume that it is normalized) to the server. The server computes K_s in the ED. Since only one element in the test sample is one and all other values are zeros, $K_s = \gamma x_{s,1} + \gamma$. However, the server blinds K_s by multiplying it by a random r_s, hence, the client only obtains $\gamma x_{s,1} r_s + \gamma r_s$ at the end. Lets assume that the client might know the scaling factor γ; hence only task left for the client is to extract $x_{s,1}$ from $x_{s,1}r_s + r_s$. Since the server generates a random $r_s \in \mathbb{Z}_n$ (i.e. where $\mathbb{Z}_n$ is a very large number) every time, it is infeasible for a client to distinguish the $x_{s,1}$ from $x_{s,1}r_s + r_s$.

In Step 4, the server interacts with the client for modulo reduction in order to obtain the sign of the decision function $d(\mathbf{t})$. Since, $d(\mathbf{t})$ is included in z in (19), revealing z may leak the decision function value to the client. Hence, the server adds random value r with z before sending it to the client. This randomization makes it infeasible for the client to extract any server side information. Hence, two-party protocols used in Step 2 and Step 4 satisfy the privacy definition.

4 Multi-class Problem in the Encrypted-domain

We considered two classical approaches to decouple the multi-class problem into multiple two-class subproblems in Section 2.2. In the following subsections, we propose methods for both the approaches in the ED.

4.1 1VA Approach in the Encrypted-domain

In 1VA approach, the server has decoupled the N_c-class problem into N_c-number of two-class subproblems and obtained the corresponding support vectors, Lagrangian variables and bias for each subproblem in the PD. Any unlabeled test sample can be labeled according to the predicted class using (7) in the PD. However, in the ED the client could supply a test sample $\tilde{\mathbf{t}}$ only in the ED. Hence, the server needs to obtain the value of the decision function for each subproblem in the ED. A subproblem with the largest decision function value will be chosen and the test sample labeled to the corresponding class using (7). Let us define the value of the decision function (scaled version) of the j^{th} subproblem using (7) and (11) as

$$d_j(\mathbf{t}) = \sum_{s\in S_j}(\gamma\alpha_s y_s)[(\gamma\mathbf{x}_s)'(\gamma\mathbf{t})+\gamma^2 1]^p + \gamma^{2p+1}b_j. \quad (24)$$

The server obtains the $d_j(\mathbf{t}),\ j=1,\ldots,N_c$ in the ED (i.e. $[\![d_j(\mathbf{t})]\!],\ j=1,\ldots,N_c$) using our methodology as in (18). Hence, (7) can be redefined as

$$[\![M_c]\!] \quad = \quad [\![\underset{j=1,\ldots,N_c}{argmax}\ (d_j(\mathbf{t}))]\!]. \quad (25)$$

The only part remaining is to find the largest value among $d_j(\mathbf{t}),\ \forall j$ and the corresponding class $class_j$ from the ED $[\![d_j(\mathbf{t})]\!]$.

4.1.1 Finding the Largest Value in the Encrypted-domain

At this stage, the server has computed N_c encrypted values of decision functions. Hence, the aim now is to determine the largest value and the corresponding class based on (25) in the ED. Securely computing the maximum of two Paillier encrypted numbers was studied in [29]. We adopted the fundamental building blocks used in [29]. Let us explain how to obtain the maximum of two encrypted values $[\![d_i(\mathbf{t})]\!]$ and $[\![d_j(\mathbf{t})]\!]$ where $|d_i(\mathbf{t})|, |d_j(\mathbf{t})| \ < \ 10^l$. To do this, the server computes a positive new variable $b_{i,j}$ as follows:

$$\begin{aligned}[\![b_{i,j}]\!] \quad &= \quad [\![10^{l+1} + d_i(\mathbf{t}) - d_j(\mathbf{t})]\!] \\ &= \quad [\![10^{l+1}]\!][\![d_i(\mathbf{t})]\!][\![d_j(\mathbf{t})]\!]^{-1}.\end{aligned}$$

Let us denote the most significant digit of $b_{i,j}$ as $\tilde{b}_{i,j}$, where $\tilde{b}_{i,j}$ can be obtained using the following linear operation:

$$\tilde{b}_{i,j} = 10^{-l-1}.\left[b_{i,j} - (b_{i,j} \bmod 10^{l+1})\right].$$

Note that $\tilde{b}_{i,j}$ can be 0 or 1 and $\tilde{b}_{i,j}=0$ implies $d_i(\mathbf{t}) < d_j(\mathbf{t})$. Now the server can obtain the largest value of two values in the ED as

$$[\![d_*(\mathbf{t})]\!] = [\![\tilde{b}_{i,j}.(d_i(\mathbf{t}) - d_j(\mathbf{t})) + d_j(\mathbf{t})]\!] \quad (26)$$

using an interactive two-party computation protocol with the client, and encryption of the matched class corresponding to the winning subproblem is given by

$$[\![class_*]\!] = [\![\tilde{b}_{i,j}.(class_i - class_j) + class_j]\!]. \quad (27)$$

We apply a straightforward recursive algorithm in Table 1 to obtain the largest value and the corresponding class out of N classes. This is returned to the client who decrypts it to find the matched class of the test sample.

TABLE 1
Algorithm 1: Finding the largest value and the corresponding class in the ED.

```
1. for $n = 1 : N-1$
2.   $[\![d_i(\mathbf{t})]\!] \leftarrow [\![d_n(\mathbf{t})]\!]$, $[\![d_j(\mathbf{t})]\!] \leftarrow [\![d_{n+1}(\mathbf{t})]\!]$
3.   $[\![class_i]\!] \leftarrow [\![class_n]\!]$, $[\![class_j]\!] \leftarrow [\![class_{n+1}]\!]$
4.   find maximum $[\![d_*(\mathbf{t})]\!]$ using (26)
5.   find corresponding class $[\![class_*]\!]$ using (27)
6.   $[\![d_{n+1}(\mathbf{t})]\!] \leftarrow [\![d_*(\mathbf{t})]\!]$ (overwrite)
7.   $[\![class_{n+1}]\!] \leftarrow [\![class_*]\!]$ (overwrite)
8. end
```

4.2 1V1 Approach in the Encrypted-domain

In the 1V1 approach, the server has $\frac{N_c(N_c-1)}{2}$ number of SVMs, where each SVM was trained using the samples only from two classes. Then the test sample will be classified using each SVM. According to the majority voting approach as in (9), a class, which has the highest number of $+ve$ predictions, will be chosen. For our context, (8) and (9) need to be computed in the ED. However, (8) can be computed in the ED using the approach described in Section 3.3. Hence, the server has $[\![f_{i,j}(\mathbf{t})]\!],\ i,j=1,\ldots,N_c,\ i\neq j$. Now server needs to sum up the votes for each class in the ED as in (9), in order to find the class which obtained the majority voting. For an i^{th} class, summation of voting in the ED can be performed by the server as

$$[\![\sum_{j=1,j\neq i}^{N} f_{i,j}(\mathbf{t})]\!] \quad = \quad \prod_{j=1,j\neq i}^{N}[\![f_{i,j}(\mathbf{t})]\!],\ i=1,\ldots,N_c,$$

where $[\![f_{j,i}(\mathbf{t})]\!] = [\![f_{i,j}(\mathbf{t})]\!]^{-1}$. What remains is to find the largest value out of N_c in the ED and the corresponding class (i.e. majority class). This can be done by the server using an approach similar to Algorithm 1 described in Table 1.

4.2.1 Computational Savings for the Client

We compares the required mathematical operations for traditional classification and two-party based classification. In the traditional approach, the testing phase of the SVM classification is done by only one party. However, in the two-party case (i.e. similar to the proposed method), the testing phase is done collaboratively between two-parties. The proposed algorithm performs PP classification in the ED where the test sample and the classification result are known only to one party and the training features are known to another party. To the best of our knowledge, there is no one-party classification algorithm (i.e. traditional approach) in the literature which performs PP classification in the ED. Hence, in this section, we show the saving in the two-party approach against the traditional approach.

Let us denote the computational power required for mathematical operations such as multiplication, addition and sign function are as c_m, c_a, and c_s, respectively. Let us denote the size of the test sample as n, and the number of support vectors required for data classification as $|S|$. Firstly, we analysis the two-class problem followed by the multi-class problem (see Table 2). The second column in Table 2 denotes computational cost for the client to run the data classifier without cloud-server assistance (i.e. traditional approach). The third column denotes the computational cost for the client in two-party approach. The fourth column, which is calculated by subtracting computational cost in the third column from the second column, denotes the computational saving for the client.

Since, n, $|S| > 0$, the client saves a substantial amount of computational power in two-party approach. Let us now compute the computational savings for multi-class problem. Let us denote the number of classes in a multi-class problem as N_c. Hence, the multi-class problem can be decomposed into N_c-, and $\frac{N_c(N_c-1)}{2}-$numbers of two-class subproblems in 1VA approach and 1V1 approach, respectively. Hence, the computational savings for 1VA and 1V1 approaches are $N\{2(|S|+1)c_a + [|S|(n+2)+1]c_m\}$, and $\frac{N_c(N_c-1)}{2}\{2(|S|+1)c_a + [|S|(n+2)+1]c_m\}$, respectively.

5 Performance Analysis

In this section, we compare the accuracy of the proposed ED methods with the conventional PD methods. For the experiments, we consider three popular data sets from the UCI machine learning repository called the Wisconsin Breast Cancer (WBC), Puma Indian Diabetic (PID), and Iris data sets [23]. We also consider a popular facial expression data set called JAFFE [24]. The details of the data sets are given in Table 3. The WBC and PID data sets are medical data sets, and the Iris data set is about flowers while the JAFFE data set is a facial expression data set. The WBC data set contains 681 data samples where 444 samples are benign (non-cancerous) and 237 samples are malignant (cancerous). The each sample of WBC data is composed with nine different attributes. The PID data set contains 768 data samples for two different classes: benign (500 samples) and malignant (268 samples). The each sample in PID data set composed with eight different attributes. The Iris data set contains samples for three different classes: Setosa, Versicolour, and Virginica, where each class contains an equal number of data samples (i.e. 50 per class). The JAFFE data set contains facial images of ten Japanese females. Each subject in JAFFE database has six facial expressions [25], i.e. angry (AN), disgust (DI), fear (FE), happy (HA), sad (SA) and surprise (SU), and a neutral (NE) face. In total, there are 213 gray-scale facial expression images in this database, each with a pixel resolution of 256×256. For the purpose of computational efficiency, all the images were resized to 51×51 pixels.

TABLE 3
The details of the data sets.

Data set	No. Classes	No. Samples	No. Attributes
WBC	2	681	9
PID	2	768	8
Iris	3	150	4
JAFFE	7	213	51×51

For experiment purpose, we use the leave-one-out approach [26], that is, one data sample is removed from the data set and all the remaining samples are used for training the SVM. The removed sample will be used as a test sample. This procedure will be repeated for a different left out test sample each time until all the data samples in a data set are tested. In order to verify the proposed methods, we first conduct the experiments with the test sample in the PD. Later we repeat the experiments, however, the test sample now in the ED, in order to compare the performance.

5.1 Experiments in the Plain-Domain

In all experiments, we assume that the training data is not linearly separable and therefore, we use polynomial kernel method as in (6). First we consider the two-class case, which will then be followed by the multi-class case.

5.1.1 Two-class Case

We consider the WBC and PID data sets for two-class experiment. Tables 4 and 5 respectively depict the maximum classification accuracy of the WBC and PID data sets in the PD. In our settings, the maximum accuracy achieved using the WBC data set was 98.24% when $p = 2$. Total number of correctly classified benign and malignant samples were 436 out of 500 and 233 out of 237, respectively. Similarly, Table 5 shows that the maximum classification accuracy in PID data set was 86.98%. It is noted during the experiment that the average number of support vectors used for the WBC and PID data sets were 10 and 55, respectively.

TABLE 2
Total number of mathematical operations required for the client for the two-class classification in traditional and two-party approaches.

	Traditional Approach	Two-party Approach	Savings
Step 1	$n(c_a + c_m)$	0	$n(c_a + c_m)$
Step 2	$\|S\|(nc_m + c_a) + \|S\|(p-1)c_m$	$\|S\|(p-1)c_m$	$\|S\|(nc_m + c_a)$
Step 3	$2\|S\|c_m + (\|S\|+1)c_a$	0	$2\|S\|c_m + (\|S\|+1)c_a$
Step 4	c_s	c_s	0
Total	$(2\|S\| + n + 1)c_a + [\|S\|(n+2)+1]\, c_m$ $+\|S\|(p-1)c_m + c_s$	$\|S\|(p-1)c_m + c_s$	$(2\|S\| + n + 1)c_a$ $+ [\|S\|(n+2)+1]\, c_m$

TABLE 4
The classification accuracy achieved for the WBC data set when $p = 2$ in the PD.

WBC	Accuracy
Benign (444)	98.20% (436)
Malignant (237)	98.31% (233)
Overall Accuracy (681)	**98.24**% (669)

TABLE 5
The classification accuracy achieved for the PID data set when $p = 3$ in the PD.

PID	Accuracy
Benign (500)	90.20% (451)
Malignant (268)	80.97% (217)
Overall Accuracy (768)	**86.98**% (668)

5.1.2 *Multi-class case*

For the multi-class case, we considered the Iris and JAFFE data sets. Let us explain how we solve the multi-class problem using a simple example. Since the JAFFE data set has seven classes, the number of required SVMs to classify a test sample for 1VA and 1V1 approaches are 7 and 21, respectively. Tables 6 and 7 show some examples for 1VA and 1VA approaches, respectively. In Table 6, decisional function values of seven different SVM classifiers for four different test samples are displayed. Each test sample is categorized to a class using (7) if the highest decision function value belongs to that class. The highest decision function value of the test sample 1 in Table 6 is 0.5070 and it corresponds to the AN-vs-all SVM, which means the test sample 1 can be categorized to the AN class. Similarly, test samples 2, 3 and 4 are also classified to the AN class. Only one test sample has been considered in Table 7 as an example to demonstrate the 1V1 method. The 7-class problem is now decoupled into 21 number of subproblems and Table 7 shows the sign of decisional function values using (8) for all subproblems. The sign values in Table 7 form a skew-symmetric matrix because $f_{j,i}(\mathbf{t}) = -f_{i,j}(\mathbf{t})$. The last column of Table 7 shows the summation of the sign values in each row. Using the majority-voting rule in (9) the AN class obtained maximum votes (i.e. 4) and, therefore, the test sample considered for this example can be categorized to the AN class.

TABLE 6
Example: 1VA method.

	Test image			
	1	2	3	4
AN-vs-all	**+0.5070**	**+1.0931**	**+0.4706**	**+0.4325**
DI-vs-all	-1.2123	+0.7910	-0.5935	-0.8492
FE-vs-all	-1.4276	-2.3656	-1.3893	-2.0274
HA-vs-all	-1.2985	-2.3203	-2.1607	-0.6268
NE-vs-all	-2.0509	-2.8030	-2.1769	-1.7336
SA-vs-all	-0.9503	-2.5222	-2.6247	-1.3851
SU-vs-all	-1.9416	-1.8771	-1.8144	-1.1769

TABLE 7
Example: 1V1 method.

	AN	DI	FE	HA	NE	SA	SU	$\sum$
AN	0	1	1	1	1	-1	1	**4**
DI	-1	0	-1	1	1	1	1	2
FE	-1	1	0	1	1	1	-1	2
HA	-1	-1	-1	0	1	-1	1	-2
NE	-1	-1	-1	-1	0	-1	1	-4
SA	1	-1	-1	-1	-1	0	1	-2
SU	-1	-1	1	-1	-1	-1	0	-4

The classification results for both the Iris and JAFFE data sets are shown in Tables 8 and 9, respectively. The 1V1 approach provides higher accuracy (87.33%) for Iris data set than the 1VA approach (85.33%). However, in JAFFE data set, the 1VA approach outperforms the 1V1 approach by nearly 1%. It is noted from the experiment that the average numbers of support vectors (including all subproblems) used for Iris data set were 174 for 1VA approach while 123 for 1V1 approach. In JAFFE data set, 551 support vectors were used for 1VA approach while 679 support vectors were used for 1V1 approach. In the next section, we perform the experiments described above, but where all the test data samples are in the ED, to validate our proposed PP SVM methods.

TABLE 8
The classification accuracy achieved for the Iris data set in the PD.

Iris	1VA ($p = 3$)	1V1 ($p = 1$)
Setosa (50)	84.00% (42)	98.00% 49
Versicolour (50)	78.00% (39)	74.00% 37
Virginica (50)	94.00% (47)	90.00% 45
Total (150)	**85.33**% (128)	**87.33**% (131)

TABLE 9
The classification accuracy achieved for the JAFFE data set in the PD.

	1VA ($p = 1$)	1V1 ($p = 1$)
AN (30)	93.33% (28)	100.00% (30)
DI (29)	93.10% (27)	89.66% (26)
FE (32)	90.63% (29)	84.38% (27)
HA (31)	87.10% (27)	83.87% (26)
NE (30)	100.00% (30)	93.33% (28)
SA (31)	90.32% (28)	90.32% (28)
SU (30)	90.00% (27)	86.67% (26)
Total (213)	**92.02**% (196)	**89.67**% (192)

5.2 Experiments in the Encrypted-domain

Our proposed PP SVM methods have been implemented in C++ using GNU GMP library version 4.2.4. Both the server and client were modeled as different threads of a single program, which passes the variables to each other. The program is tested on a computer with a 3.40 GHz Intel(R) Xeon(R) processor and 8GB of RAM running on Windows 64−operating system. The size of the Paillier security parameter was 2048 bits long.

As we mentioned in Section 3.3.1, the scaling factor γ has an influence on the classification accuracy in the ED because the Paillier cryptosystem only encrypts integers. Table 10 shows the classification accuracy for all four data sets for various scaling factors in the ED. The crucial point is that the classification accuracy in the ED eventually becomes equal to the classification accuracy in the PD when γ is at a sufficient level, in this case at $S = 10^5$. We also observed the corresponding tables in the ED when $S = 10^5$, which are the same as in Tables 4, 5, 8 and 9. This shows that our proposed PP SVM enables classification to work, in fact, it achieves the same classification accuracy as in the PD, despite test samples having been encrypted, and, therefore, privacy of the test samples is guaranteed.

The scaling factor γ can be determined by the server. If we closely look at the classification function in equation (6), classification variables and test sample (i.e. $\alpha_s y_s, \mathbf{x}_s, b$ and $\mathbf{t}$) are normalized. In traditional SVM classification, i.e. in PD, we approximate these variables and test samples to particular decimal points based on the accuracy required before computing the decision function in equation (6) (i.e. keeping the variables beyond a certain number of decimal points will not make any significant difference in the output). The range of variables in the test sample should also lie in the same range as in the training sample. Hence, it is obvious for a server to obtain the number of decimal points required to provide a higher accuracy. In the example considered in this paper, five decimal points are sufficient, hence γ value above 10^5 will not make any difference in the classification results.

5.2.1 Communication Complexity

Since only the Paillier encrypted values are being communicated between the server and client, the communication cost of the proposed algorithms highly depends on the size of the Paillier security parameter n; in our implementation $n = 2048$, hence, the size of a Paillier ciphertext is 2048 bits. Hence, sending an encrypted test sample with N number of features consumes $2048N$ bits of bandwidth in the communication channel. In the proposed algorithm, the server interacts with the client for two times: for computing polynomial kernel and for computing the sign of an encrypted value. The server sends $|S|$ number of ciphertexts (i.e. equal to the total number of support vectors) during the first interaction (i.e. polynomial kernel computation) while only one ciphertext in the last interaction. Hence, the communication cost for our algorithm is upper-bounded by the first interaction which requires $2048|S|$ bits of bandwidth. Since the number of support vectors should be less than the size of the data set, the worst case bandwidth requirement for WBC, PID, Iris, and JAFFE data sets are 1.394MB, 1.573MB, 0.3MB, and 0.436MB, respectively.

5.2.2 Computational Complexity

We measure the computational complexity in terms of average runtime required for the proposed algorithms. Table 11 depicts the average computational time required for the proposed PP algorithms in order to complete the classification task in all four data sets. It is noted that the average computational time depends on the degree of polynomial, the number of support vectors, and the total number of two-class subproblems used for the classification. For an example, let us consider the Iris 1V1 problem ($p = 1$)

TABLE 11
The average computational times required for all four data sets in the ED classification.

Data set	Time (seconds)
WBC	7.71
PID	32.09
Iris (1VA)	98.31
Iris (1V1)	4.18
JAFFE (1VA)	120.20
JAFFE (1V1)	149.82

TABLE 10
Classification result of all four data sets in the ED for various scaling factor γ.

Scaling Factor	WBC	PID	Iris		JAFFE	
			One-vs-All	One-vs-One	One-vs-All	One-vs-One
10^2	34.80%	67.45%	62.67%	42.67%	56.00%	2.35%
10^3	93.54%	**86.98%**	73.33%	69.33%	51.17%	40.85%
10^4	**98.24%**	**86.98%**	**85.33%**	**87.33%**	86.38%	**89.67%**
10^5	**98.24%**	**86.98%**	**85.33%**	**87.33%**	**92.08%**	**89.67%**
10^6	**98.24%**	**86.98%**	**85.33%**	**87.33%**	**92.08%**	**89.67%**

and the Iris 1VA problem (i.e. $p = 3$). Since there is no interaction required between the server and client, in order to compute the polynomial kernel when $p = 1$, the average computational time required for the earlier problem is less than the later problem. It is also noted that the average computational time is increasing linearly with the number of support vectors used for classification problems when $p > 1$ (i.e. the WBC and PID data sets) or when $p = 1$ (i.e. JAFFE 1VA problem and JAFFE 1V1 problem).

6 Related work

There were several classification algorithms developed in pattern recognition and machine learning for different applications [35]. However, only a few of them have been redesigned for PP classification in literature ([4]–[7], [29]–[34] and references therein). In this section, we review some of the PP SVM literature. Majority of the work in the literature were developed for the distributed setting where different parties hold parts of the training data sets and securely train a common classifier without each party needing to disclose its own training data to other parties [4]–[7]. After the training, each party holds part of the classification parameters. In order to classify a new test sample, each party has to be involved equally to compute part of the kernel matrix and then all parties together, or the trusted third-party will classify the test sample. The works in [5]–[7] exploited the secure multi-party integer summation in order to compute the kernel matrix. Basically, each party generates a Gramm matrix using scalar products of training and test data samples. This Gramm matrix is later revealed to the trusted third party who will compute the kernel matrix and then classify the test sample. Revealing the Gramm matrix may leak the private data and, therefore, privacy cannot be entirely preserved.

The work in [4] proposed for the first time a strongly privacy-enhanced protocol for SVM using cryptographic primitives where the authors assumed that the training data is distributed. Hence, in order to preserve the privacy they developed a protocol to perform secure kernel sharing, prediction and training using secret sharing and homomorphic encryption techniques. At the end of the training, each party will hold a share of the secret. In the testing phase, all parties collaboratively perform the classification using their shared secrets. At the end of the protocol, each party will hold the share of the predicted class label. Since the work is based on secret sharing, all parties must be involved in every operation of calculating the kernel values and predicting the class. Hence, it is suitable only for the distributed scenario and not for the client-server model considered in this paper. In the client-server model, the client just sends the test sample in the ED and is minimally involved in interactions with the server during the classification process.

The recent work in [8] discusses the issue of releasing the trained SVM classifier without violating the privacy of classification parameters. While the Gaussian kernel was considered therein, however, Taylor series was exploited to approximate the infinite dimension of the Gaussian kernel into finite dimension for negligible performance loss. Since this works purely in the PD, it cannot be modified to the client-server scenario considered in this paper.

Note that all the early works in PP SVM classification are restricted only to the two-class problem. Hence, in this paper, we proposed for the first time a PP protocol not only for the two-class but that solves the multi-class problem.

7 Conclusions

In this paper, we have proposed PP classifier to outsource the data classification. In particular, we proposed the first known client-server PP SVM classification protocol for the two-class problem, and the first known PP protocol for multi-class SVM classification. At the core of the proposed method lies an efficient, novel, and secure protocol to obtain the sign of a Paillier encrypted value. In order to validate the proposed methods, we have experimented our method on four different data sets. The experiment results show that the classification accuracies of the proposed ED methods are same as those in the plain non-encrypted-domain, which proves reliability of the proposed methods. Moreover, the benefit in our ED methods is that test samples need not be revealed unnecessarily as they can remain in encrypted form at all times, even during the classification process.

ACKNOWLEDGEMENTS

The authors would like to thank the Associate Editor and the anonymous reviewers for their constructive comments that greatly improved the quality of this manuscript. This work was supported through the EPSRC Grants – "The Uncertainty of Identity: Linking Spatiotemporal Information Between Virtual and Real Worlds (EP/J005266/1)" and "TRUMP: A Trusted Mobile Platform for the Self-Management of Chronic Illness in Rural Areas (EP/J00068X/1)".

REFERENCES

[1] Sundareswaran, S., Squicciarini, A. C., Lin, D.: Ensuring distributed accountability for data sharing in the cloud. In: IEEE Trans. Dependable and Secure Computing, vol. 9, no. 4, pp. 555–567. (Jul.-Aug. 2012)

[2] Pearson, S., Charlesworth, A.: Accountability as a way forward for privacy protection in the cloud. In: Proc. First Int'l Conf. Cloud Computing. (2009)

[3] Pearson, S., Shen, Y., Mowbray, M.: A privacy manager for cloud computing. In: Proc. Int'l Conf. Cloud Computing (CloudCom), pp. 90–106. (2009)

[4] Lipmaa, H., Laur, S., Mielikainen, T.: Cryptographically private support vector machines. In: Proc. 12th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 618–624, Philadelphia, USA. (Aug. 2006)

[5] Yu, H., Jiang, X., Vaidya. J.: Privacy-preserving SVM using nonlinear kernels on horizontally partitioned data. In: Proc. ACM Symp. Applied Computing (SAC). (2006)

[6] Yu, H., Vaidya, J., Jiang, X.: Privacy-preserving SVM classification on vertically partitioned data. In: Proc. 10th Pacific-Asia Conf. Knowledge Discovery and Data Mining (PAKDD). (2006)

[7] Chen, K., Liu, L.: Privacy preserving data classification with rotation perturbation. In: ICDM, IEEE Computer Society, pp. 589–592. (2005)

[8] Lin, K-P., Chen, M-S.: On the design and analysis of the privacy-preserving SVM classifier. In: IEEE Trans. Knowledge and Data Engineering, vol. 23, no. 11, pp. 1704–1717. (Nov. 2011)

[9] Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238, Springer, Heidelberg. (1999)

[10] Cortes, C., Vapnik, V.N.: Support-vector networks. In: Machine Learning, vol. 20, no. 3, pp. 273–297. (1995)

[11] Vapnik, V. N.: An overview of statistical learning theory. In: IEEE Trans. Neural Netw. vol. 10, pp. 988–999. (1999)

[12] Qian, H., Mao, Y., Xiang, W., Wang, Z.: Recognition of human activities using SVM multi-class classifier. In: Pattern Recognition Letters, vol. 31, pp. 100–111. (2010)

[13] Lei, H., Govindaraju, V.: Which is the best multiclass SVM method? An empirical study. In: Proc. 6th Int'l Conf. Multiple Classifier Systems, Springer-Verlag Berlin, Heidelberg, pp.156–164. (2005)

[14] Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. In: Journal of Machine Learning Research, Vol. 2, pp. 45–66. (2002)

[15] Hsu, C. -W., Lin, C. -J.: A comparison of methods for multiclass support vector machines. In: IEEE Trans. Neural Netw., vol. 13 no. 2, pp. 415–425. (2002)

[16] Kotsia, I., Pitas, I.: Facial expression recognition in image sequences using geometric deformation features and support vector machines. In: IEEE Trans. Image Process., vol. 16, no. 1, pp. 172–187. (2007)

[17] Bergsma, S., Lin, D., Schuurmans, D.: Improved natural language learning via variance-regularization support vector machines. In: Proc. 14th Conf. Computational Natural Language Learning, pp. 172–181. (2010)

[18] Ben-Hur, A., Ong, C., Sonnenburg, S., Scholkopf, B., Ratsch, G.: Support vector machines and kernels for computational biology. In: PLoS Computational Biology — www.ploscompbiol.org, vol. 4, no. 10, pp. 1–10. (2008)

[19] Platt, J., Cristianini, N., Shawe-Taylor, J.: Large margin DAGs for multiclass classification. In: Advances in Neural Information Processing System, vol. 12, pp. 547–553. (2000)

[20] Burges, C. J. C.: A tutorial on support vector machines for pattern recognition. In: Data Min. Knowl. Discov., vol. 2, pp. 121–167. (1998)

[21] Vapnik, V. N.: The nature of statistical learning theory. In: Springer-Verlag New York, Inc., New York, USA. (1995)

[22] Boser, B. E., Guyon, I., Vapnik, V. N.: A training algorithm for optimal margin vlassifiers. COLT, pp. 144-152. (1992)

[23] Mangasarian, O. L., Street, W. N., and Wolberg, W. H.: Breast cancer diagnosis and prognosis via linear programming. In Operations Research, vol. 43, pp. 570–577. (Jul.-Aug. 1995)

[24] Lyons, M.J., Budynek, J., Akamatsu, S.: Automatic classification of single facial images. In: IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 12, pp. 1357–1362. (Dec. 1999)

[25] P. Ekman, W. Friesen: Facial Action Coding System. Consulting Psychologist Press, Palo Alto, CA. (1978)

[26] Cawley, G.C., Talbot, N.L.C.: Efficient leave-one-out cross-validation of kernel fisher discriminant classifiers. In: Pattern Recognition, vol. 36, no. 11, pp. 2585–2592. (2003)

[27] Damgård, I., Geisler, M., Krigaard, M.: Efficient and secure comparison for online auctions. In: ACISP 2007. LNCS, vol. 4586, pp. 416–430. Springer, Heidelberg (2007)

[28] Garay, J. A., Schoenmakers, B., Villegas, J.: Practical and secure solutions for integer comparison. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 330–342. Springer, Heidelberg (2007)

[29] Erkin, Z., Franz, M., Guajardo, J., Katzenbeisser, S., Lagendijk, I., Toft. T.: Privacy-preserving face recognition. In privacy enhancing technologies (PET09). LNCS, vol. 5672 pp 235–253. Springer. (2009)

[30] Rahulamathavan, Y., Phan, R. C.-W., Chambers, J. A., Parish, D.: Facial expression recognition in the encrypted-domain based on local fisher discriminant analysis. In: IEEE Trans. Affective Computing, vol. 4, no. 1, pp. 83–92. (Jan.-Mar., 2013)

[31] Barni, M., Bianchi, T., Catalano, D., Raimondo, M. D., Labati, R. D., Failla, P., Fiore, D., Lazzeretti, R., Piuri, V., Scotti, F., Piva, A.: Privacy-preserving fingercode authentication. In Proc. 12th ACM workshop on Multimedia and security, New York, NY, USA, 231–240. (2010)

[32] Du, W., Zhan, Z.: Building decision tree classifier on private data. In Proc. IEEE Int'l Conf. Privacy, Security and Data Mining - vol. 14. Australian Computer Society, Inc., Darlinghurst, Australia, 1–8. (2002)

[33] Barni, M., Failla, P., Lazzeretti, R., Sadeghi, A-R., Schneider, T.: Privacy preserving ECG classification with branching programs and neural networks. In: IEEE Trans. Information Forensics and Security, vol. 6, no. 2, pp. 452–468. (Jun. 2011)

[34] Barni, M., Orlandi, C., Piva, C.: A privacy-preserving protocol for neural network-based computation. In Proc. ACM Multimedia and Security Workshop,26-27, Geneva, Switzerland. (Sept. 2006)

[35] Duda, R. O., Hart, P. E., and Stork, D. G.: Pattern classification and scene analysis 2nd ed. (1995)

[36] Goldreich, O.: Secure multiparty computation. (working draft), available: http://www.wisdom.weizmann.ac.il/ oded/pp.html. (Sep. 1998)

[37] Kantarcioglu, M., Clifton, C.: Privacy-preserving distributed mining of association rules on horizontally partitioned Data. In: IEEE Trans Knowledge and Data Engineering, vol. 16, no. 9, pp. 1026–1037. (2004)

[38] Lindell, Y., Pinkas, B.: Privacy preserving data mining. In Advances in CryptologyCRYPTO 2000, pp. 36–54. Springer Berlin Heidelberg, (2000).

[39] Vaidya, J., Clifton, C., Zhu, M.: Privacy-preserving data mining. In Advances in Information Security bookseries, Sushil Jajodia, Series Editor, Springer-Verlag, ISBN:0-387-25886-8. (Nov. 2005)

Yogachandran Rahulamathavan (S'08-M'12) received the B.Sc. degree (first-class honors) in electronic and telecommunication engineering from the University of Moratuwa, Sri Lanka, in 2008 and a Ph.D. degree in Signal Processing from Loughborough University, UK in 2011.

From April 2008 to September 2008, he was an Engineer at Sri Lanka Telecom, Sri Lanka and from November 2011 to March 2012, he was a Research Assistant with the Advanced Signal Processing Group, School of Electronic, Electrical and Systems Engineering, Loughborough University, UK. He is currently working as a Research Fellow with the Information Security Group, School of Engineering and Mathematical Sciences, City University London, UK.

Dr Rahulamathavn received a scholarship from Loughborough University to pursue his Ph.D. degree. His research interests include signal processing, machine learning and information security and privacy.

Dr. Suresh Veluru is a Research Fellow in the School of Engineering and Mathematical Sciences at City University London, United Kingdom. Prior to this, he worked as a research fellow in Computer Science at University of York, United Kingdom and University of New Brunswick, Canada. He received his PhD in Computer Science from Indian Institute of Technology Guwahati, India in 2009. His research interests include pattern recognition, data mining, natural language processing, artificial intelligence and privacy preserving data mining.

Raphael Phan is Professor at the Multimedia University, Malaysia, having previously worked in British, Swiss and Australian universities before assuming his current position. He has Erdős number 2, been invited to serve in over 90 technical program committees of international security conferences, and has published in excess of 50 journal papers and 100 conference papers. He researches diverse aspects of security and privacy, including cryptography, security protocols, secure signal processing, pattern recognition for surveillance, and forensics. Besides research, he spends his time educating young minds on the beauty of security research and analyzing real-life mechanisms from a security viewpoint.

Kanapathippillai Cumanan (M'10) received the B.Sc. degree with first class honors in electrical and electronic engineering from the University of Peradeniya, Peradeniya, Sri Lanka, in 2006 and the Ph.D. degree in signal processing from Loughborough University, Loughborough, UK.,in 2009. He is currently working as a research associate in Communications, Sensors and Signal processing group, School of Electrical and Electronic Engineering, Newcastle University, UK. Prior to this, he was a Research Associate with the Advanced Signal Processing Group, School of Electronic, Electrical and Systems Engineering, Loughborough University, UK and he was also an academic visitor at Department of Electrical and Computer Engineering, National University of Singapore. From January 2006 to August 2006, he was a Teaching Assistant with the Department of Electrical and Electronic Engineering, University of Peradeniya, Sri Lanka. His research interests include physical layer security, cognitive radio networks, relay networks, convex optimization techniques and resource allocation techniques.

Dr. Cumanan was the recipient of an overseas research students award from Cardiff University, Wales, UK., where he was a research student between September 2006 and July 2007.

Muttukrishnan Rajarajan (SM'05) received the BEng and PhD degrees from City University London in 1994 and 1999 respectively. He then worked at the same institute as a postdoctoral research fellow on a Ministry of Defence funded research project. He moved to Logica UK in 2000 to work as a Network Security consultant. He joined City University London back in 2002 where he is currently a full Professor and heads the Information Security Research Group. He has research expertise in the areas of privacy preserving techniques, mobile security and Cloud security. He acts as an advisor to the government of India research laboratories in the area of cyber security. He is a visiting scientist at the British Telecommunications (BT) security innovation laboratories, UK. Professor Rajarajan is a Senior Member of Institute of Electrical and Electronic Engineering and is a member of the academic advisory board of the institute of information security professionals (IISP). He has been involved in several recent policy debates in the area of cyber security. He has published more than 150 journal and conference papers and has recently published a book entitled Mobile Security and Privacy. He is in the advisory board of several start-up companies in the area of Cloud security and identity assurance. He serves on several journal editorial boards and international conferences programme committees. More details can be found at www.staff.city.ac.uk/∼raj