Cloud-Assisted Privacy-Preserving Classification for IoT Applications

Lei Yang
Amazon LLC, Seattle, WA, USA
Email: ynglei@amazon.com

Fengjun Li
The University of Kansas, Lawrence, KS, USA
Email: fli@eecs.ku.edu

***Abstract*—The explosive proliferation of Internet of Things (IoT) devices is generating an incomprehensible amount of data. Machine learning plays an imperative role in aggregating this data and extracting valuable information for improving operational and decision-making processes. In particular, emerging machine intelligence platforms that host pre-trained machine learning models are opening up new opportunities for IoT industries. While those platforms facilitate customers to analyze IoT data and deliver faster and accurate insights, end users and machine learning service providers (MLSPs) have raised concerns regarding security and privacy of IoT data as well as the pre-trained machine learning models for certain applications such as healthcare, smart energy, etc. In this paper, we propose a cloud-assisted, privacy-preserving machine learning classification scheme over encrypted data for IoT devices. Our scheme is based on a three-party model coupled with a two-stage decryption Paillier-based cryptosystem, which allows a cloud server to interact with MLSPs on behalf of the resource-constrained IoT devices in a privacy-preserving manner, and shift load of computation-intensive classification operations from them. The detailed security analysis and the extensive simulations with different key lengths and number of features and classes demonstrate that our scheme can effectively reduce the overhead for IoT devices in machine learning classification applications.**

I. Introduction

The Internet of things (IoT) industry is considered a most promising industry in the future. As forecast by Cisco and Ericsson, more than 20 billion of IoT devices will be connected to the Internet by 2021 [1]. The IoT platform connects a huge number of sensors to the data network to collect realtime data that was previously unavailable or at a scale that was previously unreachable. More importantly, it integrates the ubiquitous sensing capability with advanced computing and data analysis capabilities of backend applications to provide automated extraction of insights by making sense of plethora of data generated by these sensors. This has led to a pervasive deployment of intelligence into our daily life, ranging from healthcare (e.g., remote patient monitoring, wearable fitness tracking) and security (e.g., community monitoring) to home automation, smart communities and smart cities (e.g., smart traffic control, distributed pollution monitoring).

In the IoT platform, the frontend IoT devices are usually resource-constrained, which have very limited storage and computing power to support complicated computations. Therefore, the intelligence is often provided by machine-learning applications running on powerful backend servers. With the large volume of data generated by IoT devices, these servers map feature vectors to categorical or real-valued outputs to train predictive models, which output classification or predication results for future sensing data to the clients. For example, wearable devices with accelerometer and gyroscope, depth cameras, etc. are deployed in the home of old adults for fall detection [2], [3]. With real-time sensor data, the predictive model running on the server analyzes the vertical state of objects and notify the caregiver when a fall is detected. More automatic medical assessment and risk profiling services can be provided from analyzing the physical measurements collected by the clients' wearable fitness tracking devices. The success of machine learning (ML) on IoT platforms has led to an explosion of demands. In fact, providing classification and predication services that are customized to different application domains is becoming an emerging business paradigm, known as "*machine-learning-as-a-service*" (MLaaS). Major cloud service providers such as Amazon, Google, Microsoft, and BigML are offering cloud-based MLaaS, which trains predictive models and charges future usage of the model at a pay-per-query or subscription-based cost.

However, IoT applications atop MLaaS also raise several privacy concerns, which may hinder further adoption of this emerging business paradigm. Privacy involved in the described application scenario is two-fold: *privacy of the subject* (i.e., IoT data) and *privacy of the service provider* (i.e., ML model). It is commonly recognized that in many applications data collected by IoT devices, such as electrocardiogram or environmental temperature, are sensitive or proprietary data of its owner. For privacy protection purposes, the client may not want to disclose her sensitive data to MLSP during the use of predictive models. This not only reflects the client's concern about probable inappropriate use of their data, but more importantly if MLSP can protect the data properly considering the numerous data breaches reported in recent years [4]. On the other hand, it is worth noting that ML models trained based on sensitive or proprietary data also need to be protected, as the predictive models (e.g., type, structure and parameters) are core intellectual property items of machine learning service providers (MLSPs). Moreover, revealing models may leak information about the underlying training data [5], [6]. Without proper protection, these models are under the risk of model extraction attacks, which query a prediction API to reverse-

978-1-5386-4586-4/18/$31.00 ©2018 IEEE

engineer model characteristics and extract an equivalent or near-equivalent model of the original one [7].

Therefore, in IoT applications that handle sensitive or proprietary data, it is important to keep both the data and the ML models private. Ideally, we can achieve this goal by requiring the ML model to process both input (i.e., data from the IoT device) and output (i.e., predication results) in the encrypted form, so that in the end of the classification, the client knows only the classification result but nothing else about the model, while the MLSP learns nothing about the client's input nor the classification result. This is known as the *privacy-preserving classification* problem [8], [9], [10].

In this work, we focus on the supervised classification schemes, which typically consist of two phases. In the training phase, the classification algorithm learns a model w from the data, and in the classification phase, the input data, denoted as a feature vector x, is fed into the trained classifier C to obtain the classification result as $C(x, w)$. It is often assumed that the model (i.e., the classifier C) has been trained through privacy-preserving training approaches (please refer to Section VII for details). Therefore, existing approaches [8], [9], [10] mostly focus on the second phase, and naturally adopt a two-party client-server setting to realize privacy-preserving classification as shown in Fig. 1.

However, these solutions cannot be directly extended to the IoT MLaaS applications. Some schemes adopt fully homomorphic encryption [9] for secure multiparty computing, which is considered several magnitudes slower than partially homomorphic schemes. Other two-party schemes such as [10] require the client to participant in the laborious process, which is computationally costly for resource-constrained IoT devices. The most promising scheme, as proposed by Bost et al. [8], also introduces computation and communication overhead that is impractical for IoT devices. For example, as shown in Fig. 1, MLSP first sends k encrypted weight vectors to the client, who computes k inner products between weight vector and her own input vector in the encrypted form and returns the greatest inner product without knowing its exact value. To avoid revealing the relative order and the classification result to MLSP, the client needs to blind the values and compare each pair with the assistance from MLSP (as shown in *Step* 3). This causes multiple rounds of interactions between the two parties, involving heavy cryptographic computation that is linear to the data size (i.e., number of attributes of the data).

To provide practical privacy-preserving classification services on the IoT platforms, we need to significantly reduce both computation and communication overheads at the resource-constrained IoT device. To tackle this problem, we propose a novel light-weight *cloud-assisted privacy-preserving classification* scheme, which employs an additional cloud server to outsource a part of computationally expensive operations from the client (i.e., IoT devices) to this assisting server. Moreover, we develop a privacy-preserving communication protocol to exchange intermediate results among the three parties, which avoids unnecessary communications between the client and the other two parties and thus reduces the participation of the client in the protocol. To the best of our knowledge, this is the first light-weight privacy-preserving classification scheme for IoT applications.

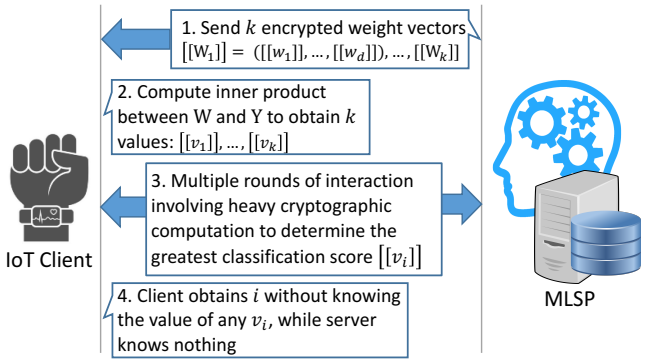

Fig. 1: Two-party privacy-preserving classification scheme (Bost et al. [8]).

The rest of the paper is organized as follows. We present the system mode, threat model and design goals in Section II and introduce the preliminaries in Section III. We present our three-party privacy-preserving classification scheme in Section IV, followed by security analysis in Section V and performance evaluation in Section VI. Finally, we discuss the related work in Section VII and conclude this work in Section VIII.

II. Background and The Problem

A. Privacy-Preserving Classification

In this work, we focus on a popular classification algorithm, called *hyperplane decision-based classifiers*, which is a parametric and discriminative classifier widely used in practice through various instances such as support vector machines (SVM), logistic regression and least squares.

Consider user input as a d-dimension vector $\mathbf{X} = \{x_1, ..., x_d\}$, where $x_i \in \mathbb{R}$ and $i \in \{1, ..., d\}$. Generally speaking, the hyperplane decision-based classifier consists of k weight vectors, corresponding to k distinct classes: $\hat{\mathbf{W}} = \{\mathbf{W}_1, \ldots, \mathbf{W}_k\}$, where $\mathbf{W}_i \in \mathbb{R}^d$ is a vector of d dimensions. To determine which class the input $\mathbf{X}$ belongs to, the classifier evaluates the index k^* such that:

$$k^* = \underset{1 \le i \le k}{\operatorname{argmax}} \langle \mathbf{W}_i, \mathbf{X} \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product operation.For example, in binary classification, i.e., $k = 2$, an input $\mathbf{X}$ is classified in class c_1 if $\langle \mathbf{W}, \mathbf{X} \rangle \ge 0$, or otherwise in class c_2. To extend to cases with $k > 2$ classes, it is common to adopt the "one-versus-all" approach to train k different binary classifiers, each discriminating one class from all the others. Therefore, the index i that leads to the greatest classification score can finally determine the classification result out of k classes.

Privacy-preserving classification requires evaluating the classifier while protecting the privacy of user data and the MLSP

classification model. From the above equation, we see that a hyperplane-based classifier involves two types of operations, i.e., inner product, comparison and argmax. Therefore, it demands new secure protocols for:

1) *privacy-preserving inner product*, which computes the inner product of $\mathbf{W}_i$ and $\mathbf{X}$ without disclosing the feature vector nor the weight vector to any party other than its owner.
2) *privacy-preserving integer comparison*, which compares two integers in the encrypted form.
3) *privacy-preserving* $argmax$, which determines the index $i(1 \leq i \leq [k])$ that leads to the greatest inner product without disclosing the value of i to the ML server and the assisting server.

B. Threat Model

Similar as previous work on privacy-preserving machine-learning-as-a-service [8], [9], [10], we adopt the "honest-but-curious" threat model for all involved parties. In an IoT MLaaS application, the user (i.e., IoT devices) is assumed to be curious about the machine learning model, which is regarded as the core intelligence property held by MLSP. Meanwhile, MLSP is interested in the private input from the user and the corresponding classification result. We introduce a new assisting cloud server in our scheme, which can only access the intermediate results. However, it is also interested in learning the input from the user, the output from the model, as well as the machine learning model. According to the "honest-but-curious" assumption, each of the three parties will faithfully follow the protocol, while trying its best to infer the private information of others. We also assume that MLSP and assisting cloud server do not collude. This can be easily achieved if the assisting cloud server is separately selected by the user. Finally, it is worth noting that either server may output incorrect results to the user accidentally or maliciously. While this problem can be solved by employing some existing verifiable computation techniques, it is out of the scope of this paper.

C. Overview of Our Scheme

In this work, we aim to design a light-weight and mutually privacy-preserving approach in the paradigm of "machine-learning-as-a-service" for IoT applications. In terms of security, our approach should minimize the privacy leak without requiring mutual trust between user and ML servers. In addition, due to the limited computation and storage capacity of IoT devices, it is expected that the overhead on the user side should be as small as possible. Therefore, our design goals are two-fold:

1) *Security*. After classification, the user should only know the classification result but nothing else about the machine learning model. On the other hand, MLSP should not know anything about the model's input and output. Moreover, ACS should know nothing at all.
2) *Lightweight operations on IoT device*. The computation and communication overheads incurred by the classification operations on the user side should be minimized to an amount affordable to resource-constrained IoT devices.

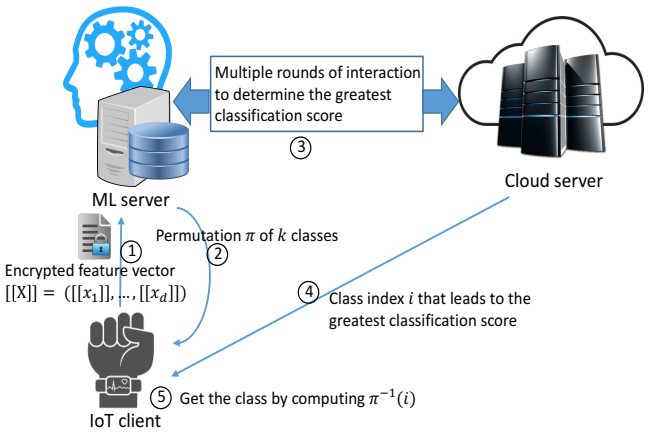

Fig. 2: The proposed three-party privacy-preserving classification framework.

To tackle this problem, we propose a three-party privacy-preserving classification framework for IoT applications. As shown in Fig. 2, the proposed framework consists of three entities: an IoT device (i.e., the client), a machine learning service provider (MLSP), and an assisting cloud server (ACS).

The IoT device, e.g., a smart watch or a smart gateway, is generating sensitive data about the client. As shown in *Step* 1, a feature vector of d dimensions, $\mathbf{X} = (x_1, \cdots, x_d) \in \mathbb{R}^d$, can be extracted from the data generated by the IoT device. $[\![\cdot]\!]$ denotes the feature vector is encrypted by the device so that its value is hidden from the server.

MLSP holds the machine learning model to provide the classification service to IoT devices. Taking $\mathbf{X}$ (in the encrypted form) as input, MLSP computes k inner products against k weight vectors for k classes, and meanwhile, sends a random permutation π of k classes to the IoT client. Different from the two-party schemes, we introduce an assisting cloud server (ACS) to mitigate the overhead in the IoT device while achieving the desired security goals. We design novel three-party privacy-preserving classification framework, and extend the idea of the two-stage Paillier cryptosystem [11] and the DGK secure comparison algorithm [12] to design new privacy-preserving protocols for inner product, integer comparison and argmax operations for the proposed framework. Our scheme can outsource most of the computation-costly device-end operations in previous two-party classification from the IoT device (*Step* 3 in Fig. 1) to the assisting cloud server (*Step* 3 in Fig. 2). This also reduces the interactions between the IoT device and MLSP for pairwise multiparty secure comparison, and thus reduces the communication overhead at the IoT devices.

In the end, ACS learns the index i (in the encrypted form) that leads to the greatest inner product, and returns it to the IoT device (*Step* 4), which can recover the classification result by computing the permutation over i (*Step* 5).

III. Preliminaries

A. Two-Stage Paillier-based Cryptosystem

Paillier cryptosystem is a probabilistic asymmetric algorithm proposed by Pascal Paillier [13] for public key cryptography. A notable feature of the Paillier cryptosystem is the homomorphic properties along with its non-deterministic encryption. More specifically, if the encryption function is additively homomorphic, to obtain the summation of two plaintexts, we can first product the corresponding two ciphertexts and decrypt the result to get $m_1 + m_2 = Dec(Enc(m_1) \cdot Enc(m_2))$.

Ateniese et al. proposed a proxy re-encryption scheme [11], which involves three parties, a *sender*, a *receiver* and a *proxy*. The sender encrypts the message with the receiver's public key. Once the proxy decrypts the ciphertext partially in the first stage and sends the intermediate result to the receiver, the receiver can recover the plaintext from the transformed ciphertext in a separate, second stage. In particular, the scheme consists of the following five functions (KeyGen, Enc, DirectDec, Transform, FinalDec):

- KeyGen: This algorithm generates the public/private keys for the receiver, where s is the private key, g^s is the public key, s_1 and s_2 are the private shares of s such that $s = s_1 + s_2$.
- Enc: This algorithm encrypts the plaintext with the input of the receiver's public key g^s.
- DirectDec: This algorithm decrypts the ciphertext with the private key s.
- Transform. This algorithm partially decrypts the ciphertext with the private key share s_1.
- FinalDec: This algorithm decrypts the transformed ciphertext to the plaintext with the private key share s_2.

The above scheme can be considered as a variant of the Paillier cryptosystem built from two-trapdoor Paillier [14], [15], with two promising properties: (1) it inherits the additively homomorphic property, and (2) the ciphertext decryption can be done in a two-stage procedure. The second property is essentially useful to our design, because it allows the private key owner to delegate the decryption capability to a third party in a control manner. In this way, the private key owner can control the decryption procedure so that without tight cooperation, any third party with one stage decryption key cannot recover the ciphertext in full.

B. Privacy-preserving integer comparison

The second building block of our model is the private-preserving integer comparison protocol. It enables two parties to jointly compare the integers they possess without revealing value of integers to each other. More specifically, we adopt the DGK comparison protocol proposed by Damgård, Geisler and Krøigaard [12]. Suppose that party $\mathcal{A}$ holds x and party $\mathcal{B}$ holds y, where x and y are two integers of l bits. The high-level idea of the protocol is to denote the integer by the binary representation such that $x = x_1 x_2 \cdots x_l$ and $y = y_1 y_2 \cdots y_l$, respectively. In this way, $x < y$ holds if and only if there exists an index $i \in [1, l]$ such that $x_i < y_i$ and for all $j < i$, $x_j = y_j$. In another word, if there exists $i \in [1, l]$, such that $z_i = x_i - y_i + 1 + \sum_{j<i}(x_j \oplus y_j) = 0$, then $x < y$.

Based on this idea, the protocol runs as follows (informal): First, $\mathcal{A}$ encrypts each bit of x using an additively homomorphic encryption scheme with $\mathcal{A}$'s public key and sends the bitwise encryptions to $\mathcal{B}$. $\mathcal{B}$ homomorphically computes encryptions of $z_1 r_1, \ldots, z_l r_l$, where $\{r_i, 1 \leq i \leq l\}$ are randomly chosen from $\mathbb{Z}_n$. Note that the encryption of z_i can be computed over the ciphertexts sent from $\mathcal{A}$, because $x_i \oplus 0 = x_i$ and $x_i \oplus 1 = 1 - x_i$, and y_i is known to $\mathcal{B}$. $\mathcal{B}$ sends the ciphertexts back to $\mathcal{A}$ in a random order, so that $\mathcal{A}$ can decrypt all ciphertexts to see whether there exists a single ciphertext decrypted to 0 and $l - 1$ ciphertexts decrypted to non-zero random numbers.

IV. Three-party Privacy-preserving classification

The proposed three-party privacy-preserving classification framework is shown in Fig. 2. To outsource the secure comparison operations from the IoT device to the assisting server, we need new privacy-preserving protocols for the *inner product*, *integer comparison*, and *argmax* operations over the encrypted data. In this section, we present our design for the three operations.

A. Privacy-Preserving Inner Product

In hyperplane-based classification, the first step is to compute the inner product of the feature vector extracted from user data and the weight vector of the ML model. Both vectors are encrypted by user and server respectively for privacy protection. To compute the inner product of the two encrypted vectors, additive homomorphic encryption schemes (e.g., Paillier cryptosystem) are commonly adopted, which can be computed in two directions, denoted as *user-initiated* and MLSP-*initiated* approaches.

The two-party privacy-preserving classification scheme proposed by Bost et al. took the MLSP-initiated approach, in which MLSP generates the public/private keys and sends the encrypted weight vectors of the machine learning model to the user. In this approach, the user is responsible to conduct the secure inner product operation over the encrypted data. Since the aggregation operation for inner product is less costly than the encryption operation, the MLSP-*initiated* scheme has a smaller computation overhead at the user end. However, the MLSP-*initiated* approach will consume additional storage of the encrypted user data at either the IoT device or an additional storage cloud. Based on this consideration, we follow the user-initiated approach, in which user generates a pair of public/private key and sends the encrypted input to MLSP, which in turn performs the private inner product.

In Fig. 3, we present an instantiation that implements the user-initiated strategy by using the aforementioned two-stage Pallier cryptosystem. More specifically, the client performs the KeyGen and Enc algorithms (*Step* 1 of Fig. 2):

User Input : $\mathbf{X} = (x_1, \cdots, x_d) \in \mathbb{Z}^d$, secret key SK_A and public key PK_A
MLSP **Input** : $\mathbf{W} = (w_1, \cdots, w_d) \in \mathbb{Z}^d$, public key PK_A
MLSP **Output:** $[\![\langle \mathbf{W}, \mathbf{X} \rangle]\!]$

1) The user encrypts $x_1, \cdots, x_d$ and sends the encryptions $[\![x_1]\!], \cdots, [\![x_d]\!]$ to MLSP.
2) MLSP performs the private inner product by computing $[\![v]\!] = \prod_i [\![x_i]\!]^{w_i} \mod N^2$.
// $v = \sum w_i x_i$
3) MLSP outputs $[\![v]\!]$, which is used as the input to the $argmax$ protocol.

Fig. 3: Privacy-preserving inner product

- KeyGen: Given two large safe primes p and q, such that p and q are primes of the form $p = 2p' + 1$, $q = 2q' + 1$, where p' a nd q' are also primes, the algorithm computes $n = pq$, selects $s \in [1, n^2/2]$ uniformly at random and sets the public key as $(n, g, h = g^s)$ where g of order $\lambda(n) = 2p'q'$ (As remarked in [14], such that g can be easily generated by selecting a random value $a \in \mathbb{Z}^*_{n^2}$ and setting $g = -a^{2n}$.), and the private key as s. In order to enable the two-stage decryption, the private key s can be divided into two shares s_1 and s_2 such that $s = s_1 + s_2$. The client holds the private key s, and distributes one private key share s_1 to MLSP and another share s_2 to ACS securely.
- Enc: To encrypt a message $m \in \mathbb{Z}_n$, it selects r uniformly at random from $\mathbb{Z}_{n^2}$ and computes the ciphertext $C = (c_1, c_2)$ where

$$c_1 = g^r \bmod n^2, c_2 = h^r(1 + mn) \bmod n^2.$$

For the simplicity of exposition, we denote the ciphertext of value y as $[\![y]\!]$, which actually consists of two parts: c_1 and c_2. As we can see, MLSP holds its model (i.e., a set of weight vectors), and performs the inner product computation together with the user's input (i.e., an encrypted feature vector), which is referred to as *Step* 2 of Fig. 2. Here, we leverage the homomorphic property of the two-stage Paillier, which states that: **(1)** the product of two ciphertexts will decrypt to the sum of their corresponding plaintexts, and **(2)** an encrypted plaintext raised to a constant k will decrypt to the product of the plaintext and the constant, namely, $\mathsf{DirectDec}(\mathsf{Enc}(m_1) * \mathsf{Enc}(m_2)) = m_1 + m_2$ and $\mathsf{DirectDec}(\mathsf{Enc}(m)^k) = km$.

B. Privacy-Preserving Integer Comparison

After MLSP computes the inner products (i.e., encrypted inner product values) by taking as input its owned model and the encrypted feature vector from the user, it needs to figure out which ciphertext corresponds to the maximum value. Therefore, we propose a private comparison protocol to preserve the data privacy while allowing MLSP to determine the maximum value. The protocol leverages the decryption technique discussed in Section III-A to exchange data securely between MLSP and the assisting cloud ACS.

MLSP **Input:** $[\![v_1]\!]$, $[\![v_2]\!]$, partial secret key s_1, and public QR key PK_{QR}
ACS **Input** : partial secret key s_2, and private QR key SK_{QR}

1) MLSP: $[\![\delta]\!] \leftarrow [\![v_2]\!] \cdot [\![2^l]\!] \cdot [\![v_1]\!]^{-1} \mod n^2$
// $\delta \leftarrow v_2 + 2^l - v_1$
2) MLSP chooses a random value r to blind δ
3) MLSP: $[\![\delta']\!] \leftarrow [\![\delta]\!] \cdot [\![r]\!] \mod N^2$
// $\delta' \leftarrow \delta + r$
4) MLSP runs the algorithm Transform to partially decrypt $[\![\delta']\!]$ to $[\![\delta']\!]'$ with s_1, and sends $[\![\delta']\!]'$ to ACS, where the Transform algorithm is defined as letting the transformed ciphertext $[\![\delta']\!]' = (c_1', c_2')$: $c_1' = c_1$ and $c_2' = c_2 / c_1^{s_1}$.
5) ACS runs the algorithm FinalDec to decrypt $[\![\delta']\!]'$ with s_2: $\delta' = L(c_2'/c_1'^{s_2} \bmod n^2)$.
6) MLSP: $c \leftarrow r \mod 2^l$
7) ACS: $d \leftarrow \delta' \mod 2^l$.
8) MLSP and ACS privately compute the encrypted bit $[t]$ such that $t = 1$ if $d < c$, and $t = 0$ otherwise.
9) ACS encrypts z_l using QR encryption and sends $[z_l]$ to MLSP.
10) MLSPencrypts r_l to $[r_l]$.
11) Server computes $[t'] \leftarrow [\delta'_{l+1}] \cdot [r_{l+1}] \cdot [t]$ and sends $[t']$ to ACS // $t' \leftarrow \delta'_{l+1} \oplus r_{l+1} \oplus t \equiv (t' = \delta \div 2^l)$
12) ACS decrypts to get t'

Fig. 4: Comparison protocol of two encrypted integers

MLSP needs to compare two encrypted integers $[\![v_1]\!]$ and $[\![v_2]\!]$ of l bits without revealing the real values. In particular, MLSP holds a private key share s_1 and ACS holds the other private key share s_2 (s_1 and s_2 can be used to recover the user's private key s). Based on the secure comparison scheme proposed by Veugen in [16], we implement a new comparison scheme, as shown in Fig. 4, which uses two homomorphic cryptosystems, the Paillier cryptosystem and the Quadratic Residues (QR) cryptosystem [17].

Let $[\cdot]$ denote the ciphertext of the QR cryptosystem, and z_{l+1} denote the $(l+1)$-th bit of integer z. The basic idea of the comparison protocol is to use the significant bit of the value $\delta = v_1 + 2^l - v_2$, determining whether $v_1 > v_2$. Note that if the significant bit δ_{l+1} is 1, then $v_1 \geq v_2$, and $v_1 < v_2$ otherwise. Given that δ is a $(l+1)$-bit number (at most), its significant bit can be obtained through $\delta \div 2^l$ where $\div$ is the integer division, and $\delta = 2^l(\delta \div 2^l) + (\delta \bmod 2^l)$ where $0 \leq (\delta \mod 2^l) < 2^l$. Moreover, in order to hide the difference between v_1 and v_2, let $\delta' = \delta + r$ where r is a $l+1$-bit random value. Therefore,

$$\begin{aligned}\delta' &= 2^l(\delta' \div 2^l) + (\delta' \bmod 2^l) \\ &= 2^l(\delta \div 2^l + r \div 2^l) + (\delta \bmod 2^l + r \bmod 2^l).\end{aligned}$$

and $\delta' \div 2^l = \delta \div 2^l + r \div 2^l + t$ where $t = 0$ if $(\delta \bmod$

$2^l) + (r \bmod 2^l) < 2^l$, and $t = 1$ otherwise.

On the other hand, $t = 0$ means that $(\delta' \bmod 2^l)$ is equal to $(\delta \bmod 2^l) + (r \bmod 2^l)$. Therefore, if $(\delta' \bmod 2^l) \geq (r \bmod 2^l)$, then $t = 0$. Hence, the significant bit δ_l can be determined as follows:

$$\delta \div 2^l = (\delta' \div 2^l) - (r \div 2^l) - t \bmod 2 = z_{l+1} \oplus r_{l+1} \oplus t.$$

Based on the above idea (together with the two-stage decryption cryptosystem), we present the detailed comparison protocol as shown in Fig. 4. This comparison protocol corresponds to one interaction between MLSP and ACS in *Step* 4 of Fig. 2. Note that in *Step* 8 of Fig. 4, MLSP and ACS need to privately compare two integers. This is known as millionaires problem [18], which can be achieved by any garbled circuit scheme. Nevertheless, we implement it by applying a cryptography-based scheme, known as DGK [12], for the sake of compatibility to our cryptography-based framework. Unlike existing work [12], [8], [10], in which the comparison protocol was implemented with the assistance of the client holding the secret key through multiple-round interaction, our protocol does not require the user to be involved in the comparison interactions and thus reduces the heavy computational overhead. This also provides an opportunity to outsource computationally costly operations to the assisting cloud ACS and leverage the two-stage decryption cryptosystem.

C. Privacy-preserving $argmax$

Given the comparison protocol for two encrypted integers as above, we extend it to a complete protocol implementing $argmax$. An intuitive way is that ACS sends the comparison result back to MLSP so that the comparison process can be repeated until finding the greatest one, incurring linear complexity (i.e., $\mathcal{O}(k)$). This approach, however, violates the security goals as mentioned in Section II-C, because each comparison reveals the greater one to MLSP, which therefore eventually leaks the classification result. Another alternative is to let MLSP and ACS compute all two-element combinations of the k values $[\![v]\!]$. In this way, MLSP learns nothing but ACS will learn which one is the greatest. Since ACS is semi-honest, it should not be allowed to see the final classification result. Moreover, this approach incur $\mathcal{O}(k^2)$ complexity.

To overcome the drawbacks of the above two approaches, we propose a three-party protocol that applies a random permutation to hide the classification result from MLSP and ACS, as depicted in Fig. 5. It incurs only a linear complexity $\mathcal{O}(n)$.

Here we explain the basic idea ot the protocol (referred to *Step* 3, 5, 6 of Fig 2): MLSP applies a random permutation π over k values $[\![v]\!]$ such that the i-th value becomes the $\pi(i)$-th value, and then sends the permutation map to the client; then MLSP and ACS iteratively perform the comparison protocol (Fig. 4) until the index n that leads to the greatest $[\![v]\!]$ after $k-1$ iterations. More specifically, at each iteration, MLSP compares the current maximal value $[\![max]\!]$ to the next value with the help from ACS. Once the maximum of the two compared values is determined, to prevent MLSP from linking

MLSP **Input:** k encrypted integers $[\![v_1]\!], \cdots, [\![v_k]\!]$, partial secret key s_1, and public QR key PK_{QR}
ACS **Input :** partial secret key s_2, and private QR key SK_{QR}
User Output: classification result i

1) MLSP: chooses a random permutation π over $\{1, \cdots, k\}$, re-indexes the k values, and sends the permutation map to the user.
2) MLSP and ACS perform the following iteration to get index n that leads to the greatest $[\![v]\!]$:
   MLSP: Let $[\![max]\!] = [\![v_{\pi(1)}]\!]$
   ACS: Let $[\![max]\!] = [\![v_{\pi(1)}]\!]$ and $n = 1$
   for $i = 2$ **to** k **do**
      Server↔ACS: interact to compare $[\![max]\!]$ and $[\![v_{\pi(i)}]\!]$ using the comparison protocol in Figure 4
      if $[\![max]\!] < [\![v_{\pi(i)}]\!]$ **then**
         $[\![max]\!] = [\![v_{\pi(i)}]\!]$
         $n = i$
      end
      ACS: randomizes $[\![max]\!]$ as $[\![max]\!] = [\![max]\!] \cdot [\![0]\!]$ and sends $[\![max]\!]$ back to MLSP //max = max + 0
   end
3) ACS: sends n to client
4) User: gets the classification result $\pi^{-1}(n)$

Fig. 5: $argmax$ protocol

the maximum to the value compared, ACS randomizes it by adding an encrypted $[\![0]\!]$ to $[\![max]\!]$. Since Paillier encryption allows adding randomness to the ciphertext, the same $[\![max]\!]$ blinded with different encryptions of 0 looks like different random numbers to MLSP. Hence, MLSP does not know which value is greater in the comparison. When comparisons finish, ACS sends the index n that leads to the greatest value to the user, who in turn gets the classification result by reversely mapping s to the class.

Eventually, MLSP learns nothing about the order of the k values. Although ACS learns that the n-th value is the greatest, it cannot correlate this value to a specific class due to permutation. Therefore, the classification result is hidden from both MLSP and ACS as we expect. Note that in the $argmax$ protocol, the user only receives a mapping (generated by the permutation π) and performs a lightweight lookup, which incurs very small communication and computation overheads to the user.

V. Security Analysis

In this section, we analyze the security of the proposed scheme in terms of input data privacy, machine learning model privacy and classification result privacy. Note that in our model,

the three parties (i.e., the user, ACS and MLSP) cannot collude with each other.

Input data privacy. When computing the inner product protocol as shown in Fig. 3, the feature vector is encrypted by the user's public key. Note that either MLSP or ACS only possesses a share of the secret key, they cannot decrypt the encrypted data as we assume that both MLSP and ACS cannot collude together. In addition, in the comparison protocol as shown in Fig. 4, ACS blinds the inner product with a random value (i.e., r in *Step* 3), so that MLSP cannot know the inner product without knowing that random value. Furthermore, with the privacy-preserving integer comparison protocol (i.e., DGK protocol in our proposed scheme), ACS only knows which inner product value is the greatest but not knowing its exact value (only learning the index of the classification result). That is, either ACS or MLSP cannot learn either the input data or the inner product values.

Machine learning model privacy. Note that in the inner product and comparison protocols, because the comparison protocol (Fig. 4) is implemented in a privacy-preserving way, either the user or ACS cannot know the inner product values with the given feature vectors. That is, the user and ACS has no chance to learn the machine learning model.

Classification result privacy. MLSP permutes the indexes of the classes so that the permutation mapping is only shared between the user and MLSP. Moreover, ACS only knows the permuted index for the classification result (from the comparison protocol), but MLSP has no knowledge about the classification result. Only the user can get the classification result by using the permuted index and the permuted mapping. Therefore, only the user can know the classification result while the other two parties are blind to that.

VI. Performance Evaluation

We have implemented the proposed cloud-assisted machine learning service scheme in real-world clouds, i.e., Amazon EC2, and compared the performance of our scheme with existing works [8], [9] in terms of asymptotic complexity and the experimental performance. More specifically, we compare our work to Bost *et al.*'s scheme [8], which adopts lightweight cryptographic primitives to protect the user's input data and MLSP's model mutually in a two-party setting. Note that we did not compare our proposed scheme with that in [9], which uses fully homomorphic encryption (FHE) to allow the server to compute some medical predication functions over patient's encrypted input. The reason is two-fold: FHE introduces significant computation overhead and is still considered impractical. On the other hand, [9] assumes that MLSP's model is known to the public and only protects users' input, therefore it provides a weaker privacy guarantee when compared to our protocol.

When conducting the performance comparison, we focus on the computational overhead at the user side, due to the fact that our goal is to minimize the cost on the recourse-constrained IoT devices and it is commonly assumed that MLSP and ACS have unlimited computational capability.

Scheme	Inner Product	Argmax
Ours	$d\times$Enc	Dec
Bost's	$k \times d\times$(Exp + Add)	$k\times$(5Add + Exp + 2Enc + DGK)

TABLE I: Comparison of asymptotic complexity at the user side between our proposed scheme and Bost's [8].

Scheme	Number of messages
Ours	$(d+1)\times$ ciphertext + Permutation-Map
Bost's	$(d + k \times (3 + 2l))\times$ ciphertext

TABLE II: Comparison of communication overhead on the client of two schemes in terms of the number of messages.

A. Asymptotic Complexity Analysis

Table I shows the asymptotic complexity comparison of the two schemes at the user side, where k denotes the number of classification classes, d denotes the dimension of the feature vector, Exp denotes the exponentiation operation, DGK denotes the overhead on the user side when executing the DGK protocol used in argmax protocol, Enc, Dec, Add denote encryption, decryption and adding two ciphertexts, respectively. We can see that when comparing with that of [8], our proposed scheme greatly offloads the computational overhead on the user side.

In addition, we compare the communication overhead for the user. In our scheme, the communication overhead only comprises d ciphertexts as the user's input, a permutation map and a ciphertext containing the classification result. All communication involved in the argmax protocol has been offloaded to ACS, therefore, the communication cost is greatly reduced for the user. In contrast, user in [8] have to be engaged in both inner product protocol and argmax protocol, which results in a significant communication overhead.

A detailed communication overhead comparison on the user side is shown in Table II, in terms of the number of messages. We choose this metrics because in our scheme, the size of permutation map (which is sent to the user) is much less significant than that of ciphertexts to be transferred, which dominates the communication overhead complexity. From Table II, we can see that the communication overhead on the user side in our scheme is much less than that of the Bost's scheme.

B. Implementation and Performance Evaluation

We have implemented the Bost's scheme and our proposed scheme with Java where three cryptographic primitives are used, i.e., the original Paillier, the revised Paillier and the Goldwasser-Micali (QR) cryptosystems.

In our implementation, we set the bit length of the large prime numbers p and q used throughout three cryptosystems to a strong security level, i.e., 512, so n is 1024-bit length. The experiments are conducted on the Raspberry Pi 2, with 700MHz ARM A6 microprocessor and 512MB RAM, to simulate the resource constrained IoT devices such as smart phone acting as the gateway for wearable devices and smart meters. In Bost's scheme MLSP is running on an Amazon EC2 C3.2xlarge instance, while in our scheme MLSP is running on

the same EC2 instance and ACS is running on a Microsoft Azure F2 instance.

Fig. 6 compares the computation time spent on the client finishing a classification service with different parameter settings. Since the performance goal of this work is to mitigate the overhead on resource-constrained IoT device, we measure the time spent on all computational operations which are performed by the client. Theses computational operations include encryption of feature vector, multiple-round comparison of encrypted integers and decryption of classification result, if any of above operation is applicable in the compared schemes. Note that the time is not the entire time from initiating the service request to receiving the classification result, which excludes the time of network communication and processing time on MLSP and ACS.

For example, in the setting where there are 5 classes and 10 features, and the key length is 1024, we can see in Fig 6, the overhead on user in Bost's two-party scheme [8] is almost 90 times than our three-party scheme. Our asymptotic complexity analysis is confirmed by the real implementation, that is, the overhead on the client is much more lightweight than Bost's scheme, since our scheme outsources the costly operations to the cloud. It is worth noting that the overhead on the client in Bost's scheme is proportional to the number of classes, while the client's overhead in our scheme is independent from the number of classes but proportional to the number of features due to the encryption of features.

We also compare the time spent on MLSP and ACS. For the setting where there are 5 classes and 10 features, and the key length is 1024, the average time spent on MLSP is 2.5 seconds in Bost's scheme while in our scheme the time on MLSP is 4.8 seconds and the time on ACS is 2.9 seconds. We also measure the communication time between EC2 instance and Azure instance. The round-trip-time between them is around 60 milliseconds. We can see that our scheme cannot necessarily reduce the total processing time, but it offloads the computation cost from the IoT device to MLSP and ACS. Such an overhead reduction on IoT devices makes machine learning classification over encrypted data feasible for resource-constrained IoT devices.

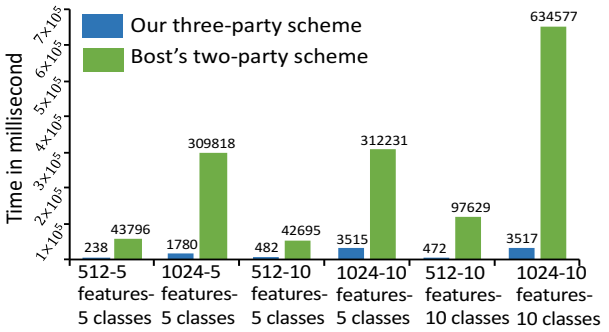

Fig. 6: Comparison of overhead on client between our scheme and Bost's [8] with varying key length, number of features and number of classes.

VII. Related Work

Related work to this paper can be classified into three categories.

Privacy-preserving training. Training the model is the first phase of a complete machine learning process. Most existing work falls into this category, which either use cryptographic techniques [19], [20] or privacy-preserving data mining techniques such as value distortion [21], randomized response [22] and partitioned data [23], [24]. These work span over variety of training algorithms including logistic regression [25], decision trees [23], [24], [22], clustering [20], Naive Bayes [26], etc.

Privacy-preserving classification. Little work has been done in the category of privacy-preserving classification, namely, apply the trained model to client input. Our paper falls in this category. Erkin *et al.* proposed a privacy-preserving face recognition scheme to hide biometrics from the server conducting the matching operation [27]. Bos *et al.* proposed a fully homomorphic encryption (FHE) based scheme to allow third party to perform predication over the encrypted medical record of a patient [9]. However, the performance problem of FHE makes their scheme less practical in reality. Besides, in their work, the server's model is assumed to be known by the public, so they provide no protection to the predictive model, and thus weaker security guarantee than our scheme.

Wu *et al.* proposed a protocol to privately evaluate decision trees [10] using DGK comparison protocol [12], oblivious transfer and tree permutation techniques. In their scheme, the client's input and the server's tree model are mutually hidden from the opposite party. In contrast to the previous schemes that focus on a specific learning algorithm, Bost *et al.* proposed a set of building blocks to construct more complex classifiers, and meanwhile they also protect the information of client and server simultaneously [8]. However, to make their scheme support more classification algorithms such as decision trees besides hyper-plane based classifiers focused in our paper, they view the decision tree as a polynomial and adopt FHE to compute the polynomial, which inevitably results in worse performance. More importantly, all the above schemes work in a two-party client-server model and did not pay special attention to the fact that the client may have very limited capacity of computation and communication, such as the IoT devices on which we focus.

Cloud-assisted IoT security. A broader topic related to our work is cloud-assisted IoT security. Since IoT sensors such as wearable devices, smart meters, in-home monitoring cameras usually collect and report sensitive data, and plenty of application use the data in various ways, to keep the data secure, cryptographic techniques are often an alternative option. However, resource-constrained IoT devices usually cannot afford costly cryptographic techniques and large data storage. Many schemes try to solve this problem by leveraging cloud which provides unlimited computation and storage capacity [28], [29], [30], [31]. Zhou *et al.* proposed a privacy-preserving

key management scheme for cloud-assisted wireless body area networks where the computationally-intensive key material updating is outsourced to the cloud in a privacy-preserving way [28]. [29], [31] both focus on cloud-assisted healthcare IoT, which mainly use the storage resources of the cloud. In [29], they proposed a scheme to add watermark into the collected data of a patient to avoid the privacy leakage on the cloud, while Yang *et al.* proposed a scheme that allows health service providers such as doctors to access and verify the encrypted medical records stored on the cloud by using a searchable encryption with forward privacy support [31]. In contrast, [30] utilizes the computation resources of the cloud to implement a data publishing scheme adopting attribute-based encryption.

VIII. Conclusion

In this paper, we proposed a cloud-assisted, privacy-preserving machine learning classification scheme for resource-constrained IoT devices. By introducing an additional cloud server and employing a two-stage decryption Paillier-based cryptosystem, our scheme allows an IoT device to offload expensive classification computations to the cloud server in privacy-preserving manner, thereby ensuring data privacy for both IoT client and machine learning service provider. The extensive complexity analysis and performance evaluation demonstrate that the proposed scheme provides an efficient solution for conducting machine learning on IoT devices, where compared to the existing solutions in the literature.

Acknowledgment

This work was partially supported by the National Science Foundation Grant DGE-1565570.

References

[1] P. Cerwall, "Ericsson mobility report," 2015. [Online]. Available: http://www.ericsson.com/res/docs/2015/mobility-report/ericsson-mobility-report-nov-2015.pdf

[2] E. E. Stone and M. Skubic, "Fall detection in homes of older adults using the microsoft kinect," *IEEE journal of biomedical and health informatics*, vol. 19, no. 1, pp. 290–301, 2015.

[3] T. N. Gia, I. Tcarenko, V. K. Sarker, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Iot-based fall detection system with energy efficient sensor nodes," in *Nordic Circuits and Systems Conference (NORCAS), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[4] S. Kuranda, "The 10 biggest data breaches of 2015 (so far)," 2015. [Online]. Available: http://www.crn.com/slide-shows/security/300077563/the-10-biggest-data-breaches-of-2015-so-far.htm

[5] G. Ateniese, L. V. Mancini, A. Spognardi, A. Villani, D. Vitali, and G. Felici, "Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers," *International Journal of Security and Networks*, vol. 10, no. 3, pp. 137–150, 2015.

[6] M. Fredrikson, S. Jha, and T. Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 1322–1333.

[7] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction apis," in *USENIX Security*, 2016.

[8] R. Bost, R. A. Popa, S. Tu, and S. Goldwasser, "Machine learning classification over encrypted data." in *NDSS*, vol. 4324, 2015, p. 4325.

[9] J. W. Bos, K. Lauter, and M. Naehrig, "Private predictive analysis on encrypted medical data," *Journal of biomedical informatics*, vol. 50, pp. 234–243, 2014.

[10] D. J. Wu, T. Feng, M. Naehrig, and K. Lauter, "Privately evaluating decision trees and random forests," *Proceedings on Privacy Enhancing Technologies*, vol. 4, pp. 1–21, 2016.

[11] G. Ateniese, K. Fu, M. Green, and S. Hohenberger, "Improved proxy re-encryption schemes with applications to secure distributed storage," *ACM Transactions on Information and System Security (TISSEC)*, vol. 9, no. 1, pp. 1–30, 2006.

[12] I. Damgård, M. Geisler, and M. Krøigaard, "Efficient and secure comparison for on-line auctions," in *Australasian Conference on Information Security and Privacy*. Springer, 2007, pp. 416–430.

[13] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 1999, pp. 223–238.

[14] R. Cramer and V. Shoup, "Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption," in *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2002, pp. 45–64.

[15] E. Bresson, D. Catalano, and D. Pointcheval, "A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2003, pp. 37–54.

[16] T. Veugen, "Comparing encrypted data," *Multimedia Signal Processing Group, Delft University of Technology, The Netherlands, and TNO Information and Communication Technology, Delft, The Netherlands, Tech. Rep*, 2011.

[17] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of computer and system sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[18] A. C. Yao, "Protocols for secure computations," in *Foundations of Computer Science, 1982. SFCS'08. 23rd Annual Symposium on*. IEEE, 1982, pp. 160–164.

[19] Y. Lindell and B. Pinkas, "Privacy preserving data mining," in *Annual International Cryptology Conference*. Springer, 2000, pp. 36–54.

[20] J. Vaidya and C. Clifton, "Privacy-preserving k-means clustering over vertically partitioned data," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 206–215.

[21] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *ACM Sigmod Record*, vol. 29, no. 2. ACM, 2000, pp. 439–450.

[22] W. Du and Z. Zhan, "Using randomized response techniques for privacy-preserving data mining," in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2003, pp. 505–510.

[23] J. Vaidya and C. Clifton, "Privacy-preserving decision trees over vertically partitioned data," in *IFIP Annual Conference on Data and Applications Security and Privacy*. Springer, 2005, pp. 139–152.

[24] F. Emekçi, O. D. Sahin, D. Agrawal, and A. El Abbadi, "Privacy preserving decision tree learning over multiple parties," *Data & Knowledge Engineering*, vol. 63, no. 2, pp. 348–361, 2007.

[25] K. Chaudhuri and C. Monteleoni, "Privacy-preserving logistic regression," in *Advances in Neural Information Processing Systems*, 2009, pp. 289–296.

[26] R. Wright and Z. Yang, "Privacy-preserving bayesian network structure computation on distributed heterogeneous data," in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 713–718.

[27] Z. Erkin, M. Franz, J. Guajardo, S. Katzenbeisser, I. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2009, pp. 235–253.

[28] J. Zhou, Z. Cao, X. Dong, N. Xiong, and A. V. Vasilakos, "4s: A secure and privacy-preserving key management scheme for cloud-assisted wireless body area network in m-healthcare social networks," *Information Sciences*, vol. 314, pp. 255–276, 2015.

[29] M. S. Hossain and G. Muhammad, "Cloud-assisted industrial internet of things (iiot)–enabled framework for health monitoring," *Computer Networks*, vol. 101, pp. 192–202, 2016.

[30] L. Yang, A. Humayed, and F. Li, "A multi-cloud based privacy-preserving data publishing scheme for the internet of things," in *Proceedings of the 32nd Annual Conference on Computer Security Applications*. ACM, 2016, pp. 30–39.

[31] L. Yang, Q. Zheng, and X. Fan, "RSPP: A Reliable, Searchable and Privacy-Preserving e-Healthcare System for Cloud-Assisted Body Area Networks," in *INFOCOM, 2017 Proceedings IEEE*. IEEE, 2010.