Mathematical Biosciences and Engineering

http://www.aimspress.com/journal/MBE

MBE, 17(4): 3478–3497.
DOI: 10.3934/mbe.2020196
Received: 15 February 2020
Accepted: 16 April 2020
Published: 6 May 2020

Research article

Privacy preserving anomaly detection based on local density estimation

Chunkai Zhang*, Ao Yin, Wei Zuo and Yingyang Chen

Department of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China

* **Correspondence:** Email: ckzhang812@gmail.com.

Abstract: Anomaly detection has been widely researched in financial, biomedical and other areas. However, most existing algorithms have high time complexity. Another important problem is how to efficiently detect anomalies while protecting data privacy. In this paper, we propose a fast anomaly detection algorithm based on local density estimation (LDEM). The key insight of LDEM is a fast local density estimator, which estimates the local density of instances by the average density of all features. The local density of each feature can be estimated by the defined mapping function. Furthermore, we propose an efficient scheme named PPLDEM based on the proposed scheme and homomorphic encryption to detect anomaly instances in the case of multi-party participation. Compared with existing schemes with privacy preserving, our scheme needs less communication cost and less calculation cost. From security analysis, our scheme will not leak privacy information of participants. And experiments results show that our proposed scheme PPLDEM can detect anomaly instances effectively and efficiently, for example, the recognition of activities in clinical environments for healthy older people aged 66 to 86 years old using the wearable sensors.

Keywords: anomaly detection; local density; privacy protection

1. Introduction

As an important branch of data mining, anomaly detection has a very wide range of application scenarios, such as, intrusion detection in network traffic, fraud detection for credit cards, disease detection in human health, video surveillance and so on. Anomaly detection is to find the instances that have different data characteristics from the most instances [1]. There are many anomaly detection algorithms that measure the different data characteristics of anomaly instances from different perspectives. For example, in distance-based algorithms [2–4], anomaly instances are the instances that are distant with most of the instances. In cluster-based anomaly detection algorithms [5–8], anomaly instances are the instances that do not lie in any large clusters. In angle-based anomaly

detection algorithms [9, 10], the variance of angles to pairs of instances remains rather small for anomaly instances. In density-based anomaly detection algorithms [11–15], anomaly instances are the instances with lower local density.

All of above anomaly detection algorithms are only suitable to the case of data sets stored in single-party participant, since these algorithms does not take into consideration data privacy protection. Therefore, as for the case of multi-party participants, it is important to design privacy preserving anomaly detection algorithms with the growing awareness of data privacy. There are some existing privacy preserving anomaly detection algorithms [8, 16–27], but most of them need to calculate the pair-wise distances on the ciphertext data, which not only needs many addition and multiplication operations on ciphertexts, but also need multiple communications between data owners, even in the case of two parties [18].

In order to avoid aforementioned disadvantages, we first propose a novel anomaly detection algorithms with linear time complexity, and then propose a privacy preserving anomaly detection scheme. And the main contributions of this paper are shown as below.

- We focus on the research of density-based anomaly detection algorithms. Most of existing density-based algorithms need calculate the pair-wise distances to determine the near neighbors of instances before obtaining the local density of instances, but this process is time-consuming. To void the $O(N^2)$ time complexity, we propose a fast anomaly detection algorithm based on local density estimation(LDEM). Different with the existing algorithms to calculate directly the local density of instances, LDEM estimate the density of instances by the average density of all features. Compared with the existing algorithms to obtain neighbors by calculating euclidean distance of instances, LDEM obtains the neighbors of each feature by the defined mapping function. And the time complexity of our algorithm only needs $O(N)$.
- What's more, we adopt our algorithm to the case of the data distributed in multiple parties. We propose an efficient anomaly detection scheme with privacy protection based on our proposed anomaly detection algorithm LDEM and homomorphic encryption scheme BCP [28–32]. Our scheme only needs outsource the sketch tables and each data owner only need constant communication times, which can reduce most communication cost. Furthermore, our scheme only needs linear addition operations on ciphertexts. From security analysis, it can easily prove that our scheme does not leak out any privacy information. And experiments results show that our algorithm can detect anomaly instances correctly with multi-party participation without leaking out any privacy information, for example, the recognition of activities in clinical environments for healthy older people aged 66 to 86 years old using the wearable sensors.

This paper is organized as follows. In section 2, we analysis the background used in our work. In section 3, we introduce the proposed local density estimation in detail. In section 4, we present the system model and introduce the proposed anomaly detection with privacy preserving in detail, and analyze the security of our scheme. In section 5, we perform some empirical experiments to illustrate the effectiveness of our algorithm. Lastly, our work is concluded in section 6.

2. Background

In this section, we will introduce the homomorphic encryption scheme BCP and two security protocols used in this paper.

2.1. Homomorphic encryption

The homomorphic encryption scheme used in our work is BCP cryptosystem, which is variant of the ElGamal cryptosystem [28] proposed by Bresson, Catalano and Pointcheval [33]. BCP has the property of additive homomorphic (see Eq (2.1)), and it can be competent at the computations on ciphers encrypted by different keys.

$$Enc_{pk}(x + y) = Enc_{pk}(x) * Enc_{pk}(u) \tag{2.1}$$

And BCP cryptosystem has two independent decryption mechanisms. In the first decryption mechanism, a given ciphertext can be decrypted by the corresponding private key. In the second decryption mechanism, any given ciphertext can be decrypted by the master key. Now we give a brief review of BCP, and the detail of BCP can be seen in [28, 33].

Setup (κ): Given a security parameter κ, choose a safe-prime RSA-modulus $N = pq(i.e., p = 2p^{'}+1$ and $q = 2q^{'} + 1$ for distinct primes $p^{'}$ and $q^{'}$, respectively of bitlength κ. Pick a random element $g \in \mathbb{Z}^*_{N^2}$ of order $pp^{'}qq^{'}$ such that $g^{p^{'}q^{'}} \bmod N^2 = 1 + kN$ for $k \in [1, N - 1]$. The plaintext space is $\mathbb{Z}_N$. We can get the public parameters and master secret as:

$$public\ parameters\ PP = (N, k, g)$$
$$master\ secret\ MK = (p^{'}, q^{'})$$

KeyGen (PP): Pick a random $a \in \mathbb{Z}_{N^2}$ and compute $h = g^a \bmod N^2$. So we can obtain the public key and secret key as:

$$\begin{aligned} public\ key\ pk &= h \\ secret\ key\ sk &= a \end{aligned} \tag{2.2}$$

Enc$_{(pp,sk)}$(m): Given a plaintext $m \in \mathbb{Z}_N$, pick a random $r \in \mathbb{Z}_{N^2}$. Then the calculation formula of ciphertext (A, B) as

$$\begin{aligned} A &= g^r\ mod\ N^2 \\ B &= h^r(1 + mN)\ mod\ N^2. \end{aligned} \tag{2.3}$$

Dec$_{(pp,sk)}$(A,B): Given a ciphertext (A, B) and secret key $sk = a$. Then the plaintext m can be decrypted as

$$m = \frac{B/(A^a) - 1\ mod\ N^2}{N} \tag{2.4}$$

mDec$_{(pp,pk,sk)}$(A,B): Given a ciphertext (A,B), a user's public key $pk = h$ and the master secret MK. Let $sk = a$ denote the user's secret key corresponding to $pk = h$. First compute a mod N as

$$a\ mod\ N = \frac{h^{p^{'}q^{'}} - 1 mod\ N^2}{N} \cdot k^{-1}\ mod\ N \tag{2.5}$$

where k^{-1} denotes the inverse of k modulo N. Then compute r mod N as

$$r\ mod\ N = \frac{A^{p^{'}q^{'}} - 1 mod N^2}{N} \cdot k^{-1} mod N \tag{2.6}$$

Let δ denotes the inverse of pq modulo N and set $\gamma := ar$ modulo N. The algorithm outputs the plaintext m as

$$m = \frac{(B/(g^{\gamma}))^{p^{'}q^{'}} - modN^2}{N} \cdot k^{-1} modN \tag{2.7}$$

2.2. Security protocol

There are two security protocols used in our scheme. One is *ProbKey* protocol that can transform the ciphertexts encrypted by different pk_i into the ciphertexts encrypted by the master public key pk. The other is the reverse operation of the previous protocol. This security protocol is called $TransDec$, which can transform the ciphertexts encrypted by pk into the ciphertexts encrypted by pk_i. Both of these two security protocols are used between the server C and the server S. In these two security protocols, the public key pk_i belongs to a participant, and the master public key pk is owned in server S.

ProdKey: Given a message x encrypted by pk_i, $[x]_{pk_i}$. The steps of transforming this ciphertext into the ciphertext encrypted by pk as below.

1) Server C picks a random number $r \in \mathbb{Z}_N$, and encrypts it to get the cipher $[r]_{pk_i}$. So we can get $[x + r]_{pk_i} = [x]_{pk_i} * [r]_{pk_i}$, and send $[x + r]_{pk_i}$ to server S.
2) Server S decrypt cipher $[x + r]_{pk_i}$ by the master key, and encrypt the plain text by pk. So we can get $[x + r]_{pk}$, and send $[x + r]_{pk}$ and pk to server C.
3) Server C encrypt the $-r$ by pk, so it can get $[-r]_{pk}$. Then, it can get the raw plaintext encrypted by pk, as $[x]_{pk} = [x + r - r]_{pk} = [x + r]_{pk} * [-r]_{pk}$.

TransDec: Given a cipher $[x]_{pk}$, this protocol can transform it back to the cipher $[x]_{pk_i}$.

1) Server C picks a random number $r \in \mathbb{Z}_N$, and encrypts it to get the cipher $[r]_{pk}$. So we can get $[x + r]_{pk} = [x]_{pk} * [r]_{pk}$, and send $[x + r]_{pk}$ and pk_i to server S.
2) Server S decrypt cipher $[x + r]_{pk_i}$ by the master key, and encrypt the plain text by pk_i. So we can get $[x + r]_{pk_i}$, and send $[x + r]_{pk_i}$ and to server C.
3) Server C encrypt the $-r$ by pk_i, so it can get $[-r]_{pk_i}$. Then, it can get the raw plaintext encrypted by pk_i, as $[x]_{pk_I} = [x + r - r]_{pk_i} = [x + r]_{pk_i} * [-r]_{pk_i}$.

3. The proposed anomaly detection algorithm: LDEM

In this section, we propose an anomaly detection algorithm (LDEM) based on local density estimation with linear time and liner space complexity. Before introducing our method in detail, we will present the symbolics used in LDEM (see Table 1).

We can notice that if an instance is abnormal, some features of this instance may be different with these features of other normal instances. So based on the independence assumption of Naive Bayes, we can estimate the local density of each feature, and then determine the density of this instance. Then, we can judge whether this instance is abnormal based on the estimated local density.

Local Density Estimation: The key insight of estimating the density of instances is to estimate the density of each feature of instances. In our method, we define some mapping functions that can map similar values into the same key. So we only need count the instance number with the same key on

Table 1. The description of symbolic.

Symbolic	Description
X	A data set.
$X_{,j}$	A vector of j_{th} feature in X.
X_i	A vector of ith instance in X.
X_{ij}	The value of j_{th} feature in X_i.
N	The length of data set X.
d	The number of features in X.
M	The number of components.

each feature, and this number denotes the local density of the corresponding feature. Now, we will show the process steps in detail.

1) First, initialize d mapping functions. Randomly select global parameter w from the range $(1/ln(N), 1 - 1/ln(N))$. Generate a vector, $r = \{r_1, r_2, ..., r_d\}$, with length d, in which each r_j is selected uniformly at the range $(0, w)$. So, we can get the d mapping functions as

$$f(X_{ij}) = \lfloor \frac{X_{ij} + r_j}{w} \rfloor \tag{3.1}$$

Function 3.1 can be used to map similar values in feature $X_{,}j$ to the same key.

2) Normalize each X_j in data set X as Eq (3.2), in which u_j is the mean value of j_{th} feature in X and std_j is the standard deviation of the data of j_{th} feature in X.

$$X_{ij} = \frac{X_{ij} - u_j}{std_j} \tag{3.2}$$

3) Then, we can take the Eq (3.1) to map the value of each feature $X_{,j}$ in data set X, and count the times of each output value of Eq (3.1). So we will get d sketch table as Eq (3.3) for data set X, and the the form of each sketch table can be seen as below.

$$sketchtable_j = \{(k_{1j}, t_{1j}), (k_{2j}, t_{2j}), ..., (k_{qj}, t_{qj})\} \quad j \in [1, ..., d] \tag{3.3}$$

In this equation, each k_i denotes an output value of mapping function, and t_i is the corresponding times. The $|sketchtable| = q$ is the number of function output values in $X_{,j}$. (Note: the length of sketch table in different features may be not same.)

4) After we have built up these sketch tables, we can do estimate the local density of each instance X_i, as Eq (3.4). In which, $sketchtable_j[f(X_{ij})]$ denotes the value of t_{qj} with $k_{qj} = f(X_{ij})$. If no k_{qj} is equal to $f(X_{ij})$, the value of $sketchtable_j[f(X_{ij})]$ will be set to zero. Then, we will get the local density of instance X_i by calculating the average value of $sketchtable_j[f(X_{ij})], j = \{1, \ldots, d\}$. Obviously, this process only need scan data set X once, so our algorithm only needs $O(N)$ time complexity.

$$density(X_i) = \frac{1}{d} \sum_{j=1}^{d} sketchtable_j[f_j(X_{ij})] \tag{3.4}$$

Same as other density-based algorithms, the smaller value $density(X_i)$ is, the more likely abnormal X_i is.

Ensemble: Since the mapping function for each feature is generated randomly, the keys mapped by one mapping function in each feature may be biased. In order to ensure to obtain unbiased local density estimation for each feature, we will randomly generate M different mapping functions for getting M different sketch tables of each feature. Therefore, we will get M components, in which each component is composed of d sketch tables. Each sketch table summaries the information of each feature. Then, at the feature density estimation stage, each feature will be estimated M times. Hence, the final estimated local density of any instance X_i is the average of M estimation results, as Eq (3.5). After considering the ensemble, the time complexity of our algorithm will become $O(MN)$. But $O(M)$ is a constant, so the final time complexity is also linear.

$$\begin{aligned} Density(X_i) &= \frac{1}{M}\sum_{m=1}^{M} density_m(X_i) \\ &= \frac{1}{dM}\sum_{m=1}^{M}\sum_{j=1}^{d} sketchtable_{mj}[f_{mj}(X_{ij})] \end{aligned} \tag{3.5}$$

Rationality Analysis: First, we analyze the rationality of mapping function. Like the literature [34], assume there is a value v, and the output value of $f(v)$ is equal to p. Then, we can obtain v must be in the range of $[p*w-r,(p+1)*w-r)$. Assume there is another value y. If $f(y)$ is also equal to p, y must be in this range $[p*w-r,(p+1)*w-r)$. So we can get the inequality $|v-y|\leq w$. Hence, we can easily get the number of near neighbors of any value by this mapping value, without any distance calculation. But it is worth noting that the variable w of mapping function decides the range of values that can be mapped to the same key. If w has a larger value, some distant values may be mapped to the same key. If w is a very smaller value, the condition for mapping to the same keyword is too harsh. In our method, we get this variable randomly from the range of $(1/ln(N), 1-1/ln(N))$. It means that the range of mapping to the same keyword is less than one. What's more, we adapt the ensemble criteria by many mapping functions. Therefore, for each feature, we will get the unbiased local density estimation.

Then, we will show the interpretability of Eqs (3.4) and (3.5). It is clear that the estimated local density of each instance is the average value of local densities of all features from Eq (3.4). Feature space can be seen as a hyper-cube, and any one feature can be seen as an axis in this hyper-cube. $sketchtable_j[f_j(X_{ij})]$ can obtain the density of X_i on the j_{th} axis direction. Only by taking all features or all axes into account can we get the more accurate local density estimation, which is approximate to determining the location of a point in space by knowing all axes of this point. But we can not determine which feature plays a more important role on the local density of instance. So we can assume that each feature has the same influence on the local density estimation of any instance, according to the maximal entropy model [35]. Hence, the local density of an instance is the simple average value of local densities of all features, as Eq (3.4). Furthermore, according to the ensemble analysis in [36], it can be proved that combining M different components can ensure that the bias-variance trade-off is optimized, and our proposed algorithm LDEM can be called a variance reduction algorithm.

4. The proposed privacy preserving anomaly detection scheme: PPLDEM

4.1. System model

In our scheme, system model is composed of data owners and a cloud. This system model can be seen Figure 1.

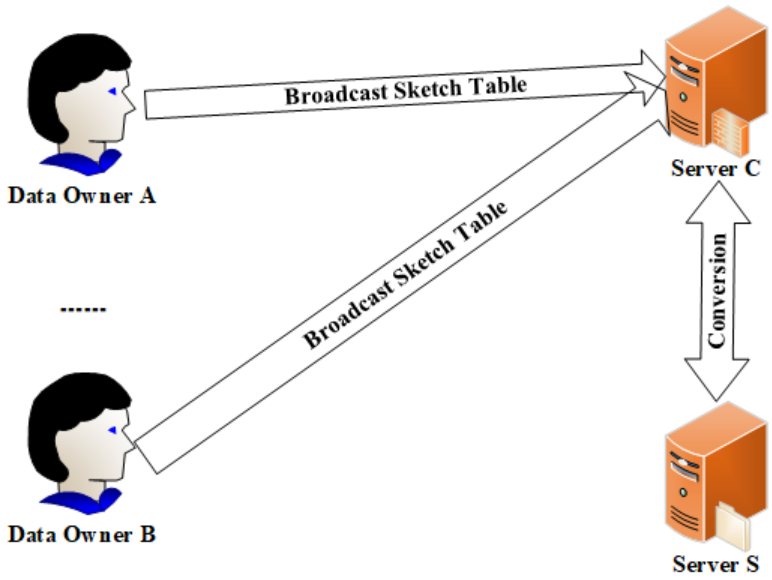


Figure 1. System model of our scheme (Data owner A has pk_a, sk_a. Data owner B has pk_b, sk_b and server S has pk and sk)

- The cloud consists of two servers. One is called as server S, which is responsible for initializing system parameters, including public parameters of BCP and parameters of our anomaly detection algorithm LDEM. Since this server has the master key, it is also in charge of conversion ciphers encrypted one public key into ciphers encrypted by other public key. This server only communicates with server C. The other server is called as server C, which is responsible for integrate the sketch tables received from all data owners. In the cloud, the server S is a trusted server and the server C is an untrusted server.
- Data owners can be also called the participant parties. Our scheme can apply to the case with multiple participants(two or more). Different with existing schemes, our scheme does not need data owners to outsource the original data set. Data owners only need sent their sketch tables to server C, and then server C will return integrated sketch tables encrypted by pk to data owners. These sketch tables contain all information of data owners contained in server C. Except requesting parameters, data owners can do any anomaly detection tasks by only communicating with server C.

4.2. Anomaly detection with privacy preserving

It is noticed easily that the key of detecting anomaly data is sketch table in LDEM. So it is very important to design a privacy preserving scheme to protect the information of sketch tables of each

participant, when there are many participants to do anomaly detection mining together. Protecting the information of sketch tables from being leaked means hiding the real keys and times in sketch tables. In our method, the crucial technologies of hiding the information are random disturbance and homomorphic encryption.

- **Random Disturbance** Random Disturbance means adding fictitious items in sketch tables, so attackers cannot guess the real items. Each fictitious item is a tuple $(key, 0)$, in which *key* can be any integer in the digital space that does not appear in the original sketch table, and 0 is used to mark fictitious *key*. For example, Table 2 is the original sketch table, and Table 3 shows the sketch table after adding some fictitious items in Table 2 and these fictitious items are marked in color red.

Table 2. Original Sketch Table.

key	**-2**	**-1**	**1**	**3**	**4**
times	23	43	2	2	2

Table 3. Sketch Table After Random Disturbance.

key	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**
times	0	0	23	43	0	2	0	2	2

The purpose of adding fictitious items is to ensure that nobody can guess whether this key is a real key. Because the keys after adding fictitious items are still plain texts. To better hiding the real keys, we propose an adding fictitious items method. Since each feature data has been normalized by the Eq (3.2), the normalized data is distributed in two sides of *zero* and the size in two sides are equal approximately. So we can add some fictitious keys to ensure that the keys sent to server C also have this characteristic. For example, in Table 3 we add the fictitious items (-4,0), (-3,0), (0,0) and (2,0) to ensure the keys of sketch tables on both sides of 0 are symmetrical. In order to better privacy protection, we advise that each data owner can set the keys in each sketch table be the all digits in the range of -1000 to 1000, since the value of Eq (3.1) is almost impossible to beyond this range.

- **Homomorphic Encryption** After adding some fictitious items in sketch table, we need to do other operations to achieve the aim that the real keys and the fictitious keys are indistinguishable. So we need select a encryption system to encrypt the *times* that can distinguish the real keys and the fictitious keys. In order to ensure the addition operations on ciphertexts encrypted by different public key, we select a semi-homomorphic encryption system, BCP [28, 31–33], which supports homomorphism addition. Assume there is a participant party A who has a sketch table (Table 2), the public key of party A is pk_a. Then, after random disturbance and homomorphic encryption operations, this sketch table can be transformed into the sketch table as Table 4.

Table 4. Sketch Table After Random Disturbance and Homomorphic Encryption.

key	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**
times	$Enc_a(0)^*$	$Enc_a(0)$	$Enc_a(23)$	$Enc_a(43)$	$Enc_a(0)$	$Enc_a(2)$	$Enc_a(0)$	$Enc_a(2)$	$Enc_a(2)$

Then, we will introduce our scheme in detail, based on these two crucial technologies and the proposed anomaly detection algorithm LDEM. In our scheme, an anomaly detection task will be divided into two steps. One is the preprocessing step, and the other is the step of calculating anomaly scores of data.

Preprocessing: First, server S will initialize the public parameters of homomorphic encryption scheme BCP and generate the master key of BCP. Then, Data owners will initialize the public parameters of encryption system and the mapping functions of LDEM. For encryption system, data owners will request *public parameters* from server S, which are used to generate *public key* and *secret key*. For LDEM, data owners will request the number of component M and dM mapping functions from server S. Then, each data owner will do the follow operations.

1) Transform the data in their own database into sketch tables by the mapping functions requested from server C, as the description in Section 3.
2) Add fictitious items in each sketch table. Then, encrypt the times of each key in their sketch table by their public key pk_i, and they can send their sketch tables to server C.

After server C received sketch tables sent by data owners, it will transform these sketch table encrypted by different pk_i using the $ProdKey$ security protocol, and then merge these sketch tables.

Algorithm 1: PPLEDM:Anomaly Detection With Privacy Preserving.

Input: Data set X

Output: The local density $Density$.

// Request encrypted $SketchTables$ from server C;
$\hat{X} \leftarrow mapping(X)$; // Preprocessing data set X;
$Density \leftarrow \emptyset$;
for $\hat{x}$ *in* $\hat{X}$ **do**
    $density_x = 0$;
    for $j = 1$ *to* d **do**
        $density_x = density_x + Table[j][\hat{x}]$;
    end
    $Density \leftarrow [Density, density_x]$;
end
$Density \leftarrow TransDec(Density)$;
return $Density$;

Detection Stage: Assume there is a data owner (participant) A, and his public key is pk_a and his secret key is sk_a. If he wants to do anomaly detection tasks, he will firstly request the merged sketch tables encrypted by the public key pk of server C. Then, he will transform the data in each feature by the corresponding mapping functions. The data in each feature will be represented by the output values (*keys*) of mapping functions, and he can query the local density of each feature in the corresponding sketch tables. After getting the local densities of all features of each instance, the final local density of each feature will be obtained. But now, the final local densities are encrypted by pk. To get the plain density values, data owner A needs to send these encrypted densities and the public key pk_a to server

*$Enc(.)$ is the encryption function of BCP. $Enc_a(.)$ means that data is encrypted by the public key of participant party A.

C, and server C will transform these encrypted densities into the ciphertexts encrypted by pk_a using the $TransDes$ security protocol. And then, data owner A can get the plain densities through decrypting these ciphertexts by their own secret key sk_a. Lastly, data owner A can determine which instances are more likely to be anomaly instances, according to the local density ranking.The all process of this stage in data owner A can be seen in Algorithm 1.

5. Results

In this section, we first demonstrate the utility of our proposed detection algorithm LDEM with all original databases under single-party participation. Then, we will analyze the performance of our algorithm by encrypted data with multiple parties participation.

For comparability, we implemented all experiments on our workstation with 2.5 GHz, 64 bits operation system, 4 cores CPU and 16 GB RAM, and the algorithms codes are built in Python 2.7.

5.1. Evaluation metrics and experimental setup

Metics: In our experiment, we adopt Receiver Operating Characteristic (ROC) curve as the evaluation criterion for the proposed algorithm and other compared algorithms. The ROC curve is created by plotting the true positive rate against the false positive rate at various threshold settings, so it can evaluate anomaly detection algorithms in terms of false positive rate (FPR) and the true positive rate (TPR). The anomaly detection algorithms with larger area under ROC curve may have better detection accuracy, otherwise, the anomaly detection algorithms are less effective.

Experimental Setup: There are two experiments which are design to illustrate the effectiveness of our anomaly detection algorithm LDEM and and the significance of our anomaly detection algorithm with privacy preserving.

Table 5. Data sets information

Data set	Instances	Attribute	Anomaly Ratio
Breast	569	30	37.3%
Ann_thyroid	7200	6	7.4%
Waveform	1727	21	4.6%
Ecoli	336	7	2.7%
Arrhythmia	452	272	14.6%
Pima	768	8	34.9%
Satellite	6435	36	31.6%
Shuttle	14500	9	6.0%
Epileptic	11500	178	20.0%

- In the first experiment, all data sets are selected from UCI Repository [37], and these data sets are summarized in Table 5. Since many of these data sets contain more than two class labels, it needs some preprocessing operations to obtain the data sets which are suitable for anomaly detection. We preprocess these data sets according to some of the commonly used principles in the literatures [34]. We compare our proposed anomaly detection algorithm LDEM with other state-of-art algorithms. These compared algorithms contain LOF [11], FastABOD [9], iForest [38, 39]

and RDOS [14]. iForest detects anomaly instances based on the average path of instances in isolation forest. FastABOD determines anomaly instances based on the angle among instances. RDOS determines anomaly instances according to the relative kernel density distribution. For the parameter of all algorithms, the number of near neighbors of RDOS, FastABOD and LOF will be set to 10, and the tree number of iForest will be set to 25, and the height limit of iForest will be set to 6, and the sample size of iForest will be set to 256. The components of LDEM will be set to 10.

- In the second experiment, we adopt a sensor data to verify the effectiveness of PPLDEM. This sensor data is sequential motion data from 14 healthy older people aged 66 to 86 years old using a batteryless, wearable sensor on top of their clothing for the recognition of activities in clinical environments [40]. This data contain four classes. In this experiment, we chose class 1 as the abnormal class, and the class 3 as the normal class. Therefore, the selected data sets are summarized in Tables 6 and 7. All these data sets are selected from the data sets allocated in Room1, of which the setting uses 4 RFID reader antennas around the room. Table 6 describes the data sets collected from female, and Table 7 describes the data sets from male.

Table 6. Data sets of female in Room1

Data set	Instances	Attribute	Anomaly Ratio
d1p13F	168	8	13.09%
d1p14F	124	8	7.25%
d1p18F	219	8	0.456%
d1p49F	2555	8	35.6%
d1p50F	4299	8	28.2%
d1p53F	4363	8	27.98%

Table 7. Data sets of male in Room1

Data set	Instances	Attribute	Anomaly Ratio
d1p01M	334	8	34.13%
d1p05M	448	8	29.01%
d1p06M	614	8	11.88%
d1p40M	807	8	40.64%
d1p41M	806	8	33.49%
d1p43M	2332	8	17.32%

5.2. Performance efficiency of LDEM

Accuracy Analysis: For fairness, we build up our algorithm and reproduce all compared algorithms in Python Language. All algorithms are executed many times to obtain stable results on all data sets in Table 5, and the experiment results are shown in Figure 2. Figure 2(a) plots the ROC curve of all compared algorithms on *Breast* data set. Figure 2(b) plots the ROC curve of all compared algorithms on *Ann_thyroid* data set. Figure 2(c) plots the ROC curve of all compared algorithms on *Waveform* data set. Figure 2(d) plots the ROC curve of all compared algorithms on *Ecoli* data set. Figure 2(e) plots

the ROC curve of all compared algorithms on *Arrhythmia* data set. Figure 2(f) plots the ROC curve of all compared algorithms on *Pima* data set. Figure 2(g) plots the ROC curve of all compared algorithms on *Satellite* data set. Figure 2(h) plots the ROC curve of all compared algorithms on *Shuttle* data set. Figure 2(i) plots the ROC curve of all compared algorithms on *Epileptic* data set.

In Figure 2(a–c), the ROC curves of other compared algorithms are completely below the ROC curves of LDEM, so the detection accuracy of our algorithm LDEM outperforms than the compared algorithms on these three data sets. In Figure 2(d–f,h,i,), the ROC curves of LDEM are approximate to the best ROC curves of other compared algorithms. In Figure 2(g), although the ROC curve of LDEM cannot completely cover the ROC curves of the compared algorithms, the area under ROC curve of LDEM is great than others. Therefore, this experiment illustrates that our proposed algorithm is effective.

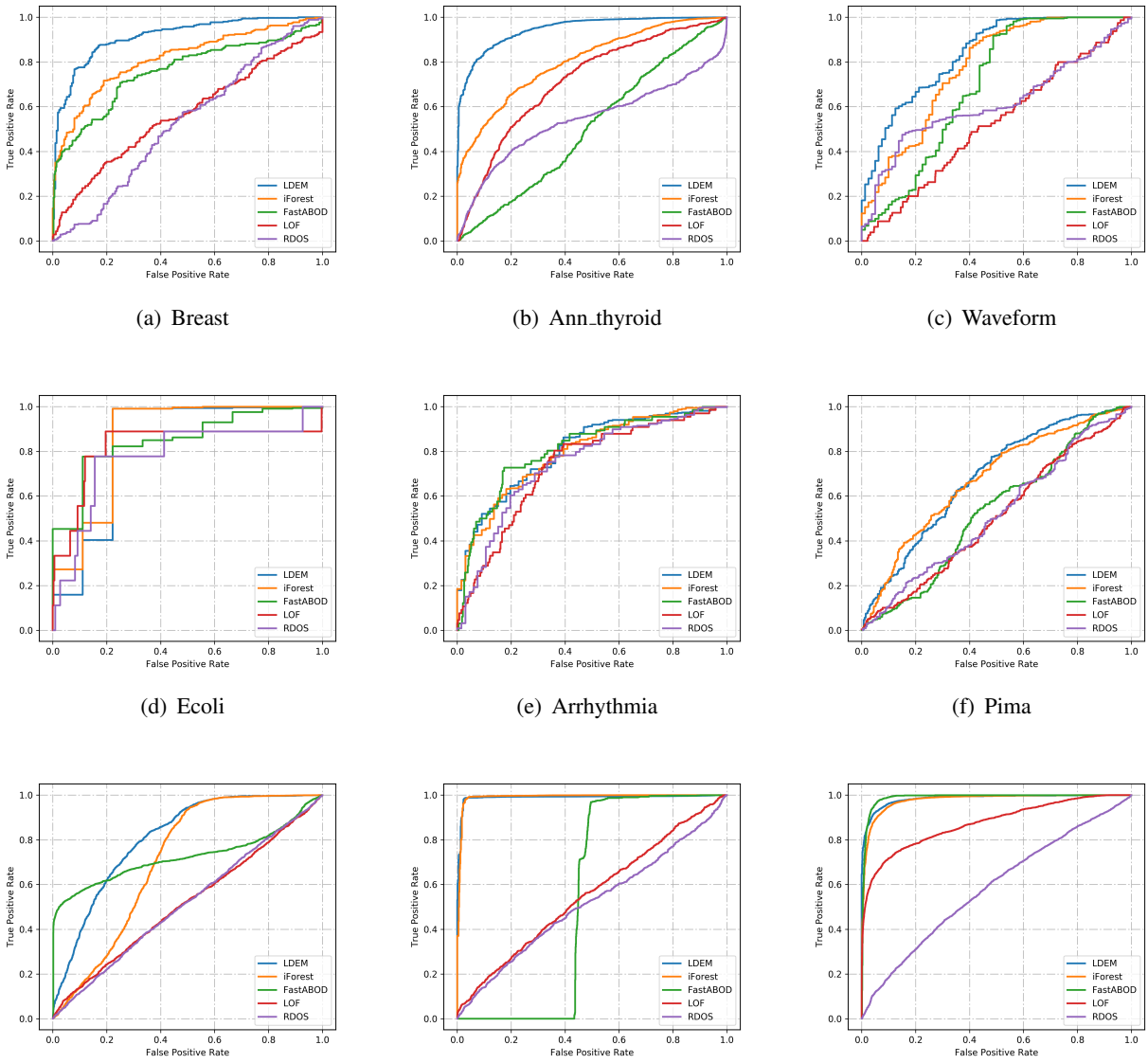
Running Time Analysis: To illustrate that our algorithm not only gets good detection results, but also needs less running time. We do an experiment to compare the running time of all aforementioned algorithms on the selected four data sets with different size from Table 5. The results of this experiment are shown in Figure 3. The running time of LOF is more than 20 seconds on *Ann_thyroid* and *Satellite* data sets, and the running time of LOF is more than 20 seconds on *Ann_thyroid*, *Waveform* and *Satellite* data sets. Analysis the process of these four algorithms in Figure 3, we can know that LOF, RDOS and FastABOD need to calculate the pair-wise distances, which needs $O(N^2)$ time complexity, while LDEM only needs to build up M sketch tables, which only needs linear time complexity. Therefore, we can get the results in Figure 3. It can be easily seen that the running time of our algorithm is less than LOF, FastABOD and RDOS.

Sensitivity Analysis: In our algorithm, there is only one parameters, the number of components M, which can affect the detection accuracy. In order to verify whether the proposed algorithm LDEM is sensitive to the parameter M, we will record the AUC value of these ten data sets in this experiment, when the range of this number is from 1 to 50. AUC is the area under ROC curve, which can intuitively reflect the classification ability of ROC curve expression. The results of this experiment are shown in Figure 4. From this figure, we can notice that LDEM has almost the same AUC value on most of selected data sets, when the parameter M takes different value in the range of 1 to 50. The AUC value of over half of data sets has almost not any fluctuation. Only a few data sets have small fluctuations in their AUC values. As a result, this experiment can illustrate that our algorithm is not very sensitive to the parameter M.

5.3. Performance efficiency of PPLDEM

Accuracy Analysis: We will present the improvement of our algorithm under multiple participants based on the data sets in Tables 6 and 7. First, we can calculate the AUC value of each data set based on the sketch tables created by their own data set. This process can be finished on each data owners. Also, we can calculate the AUC value of each data set based on all sketch tables created from all data sets. Since each data set in these two tables is created by corresponding person, it is necessary to use our privacy preserving scheme to protect the privacy information of each data owners.

The results based on the sketch tables obtained from the data set of each data owners are recorded in the column *AUC_BY_SELF* of Tables 8 and 9, and the results based on the sketch tables obtained from the data sets of all participants are recorded in the column *AUC_BY_ALL* of Tables 8 and 9. Compare the results of these two columns, it can be seen that the column *AUC_BY_ALL* has a better AUC value



(a) Breast (b) Ann_thyroid (c) Waveform

(d) Ecoli (e) Arrhythmia (f) Pima

(g) Satellite (h) Shuttle (i) Epileptic

Figure 2. The ROC curve of all anomaly detection algorithms on the selected data sets from UCI.

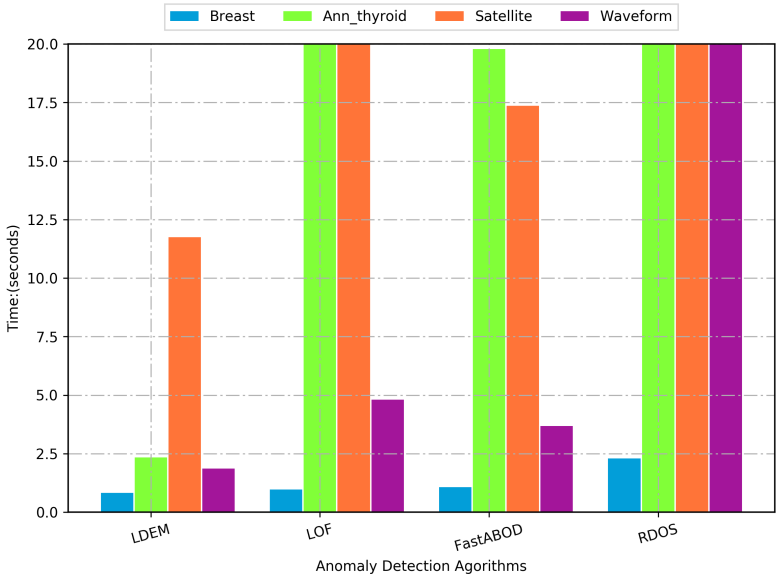


Figure 3. Running time of these algorithms on four of the data sets.

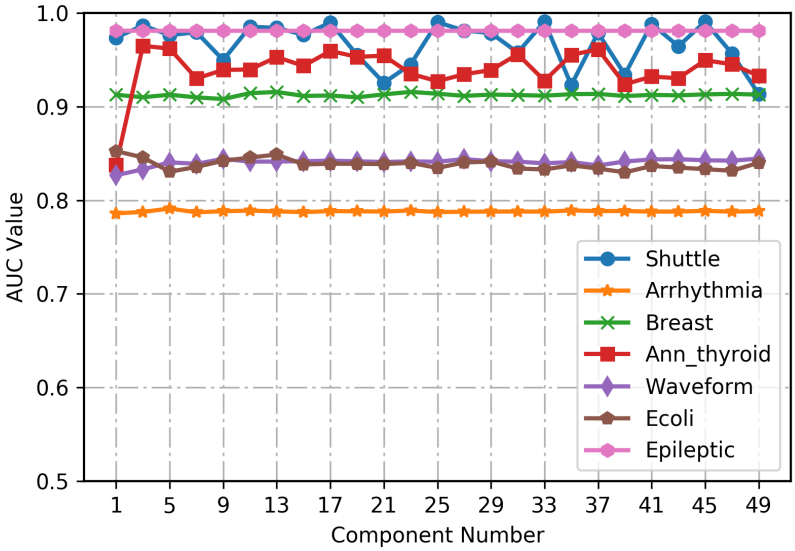


Figure 4. Sensitive analysis of the parameter M.

on most of all data sets. Hence, from this experiment, it can prove that our algorithm is effective under multiple participants.

Table 8. AUC value on different sketch tables from female data set.

Data set	AUC_BY_SELF	AUC_BY_ALL
d1p13F	0.968	**0.979**
d1p14F	0.993	**1.00**
d1p18F	0.972	**0.972**
d1p49F	0.770	**0.892**
d1p50F	0.766	**0.818**
d1p53F	0.919	**0.975**

Table 9. AUC value on different sketch tables from male data set.

Data set	AUC_BY_SELF	AUC_BY_ALL
d1p01M	0.961	**0.993**
d1p05M	0.954	**0.959**
d1p06M	0.997	**0.999**
d1p40M	0.831	**0.997**
d1p41M	0.908	**0.995**
d1p43M	0.991	**0.991**

Security Analysis: We use the BCP cryptosystem to encrypt private data and our anomaly detection scheme with privacy preserving is based on semi-honest model. In semi-honest model, all participants will comply with the security protocols, but they may collect the received information (inputs, outputs, calculated results) to look for some privacy information [31]. In our scheme, we assume that all participants, including server C, server S and data owners, will do anomaly detection tasks on the basis of the proposed protocols.

- **Security analysis under attacks on encrypted data.** BCP cryptosystem is semantically secure in the standard model, based on the decisional Diffie-Hellman assumption modulo a square composite number [33]. Semantic secure [41] is widely admitted to be the main security notion in secrecy of communication. For data encrypted by BCP cryptosystem, cryptotexts can not be decrypt without the private key or the master key. Therefore, privacy information will not be leaked by the encrypted data.
- **Security analysis under attacks from server C.** In our scheme, server C is responsible for two things, merging sketch tables received from different data owners and transforming the densities encrypted by pk into ciphertexts encrypted by pk_i. For the first thing, server C may do the keys attack on the basis of the received sketch tables. But in our scheme, the keys in these sketch tables are composed of real keys and fictitious keys, and these two type keys are marked by the times encrypted by the pk_i. Since any two times ciphertexts are indistinguishable based on the security of BCP encryption system [31], server C can not distinguish the true and false of any two keys, such as $(k_i, Enc(0))$ and $(k_j, Enc(1))$. It only can know the frequencies of any received key, but it can not infer the true times of these keys in there original sketch tables. For the other thing, all

operations in server C are based on ciphertexts. Any information will not be leaked in this thing, since cracking BCP is difficult NP problem. From the above analysis, it can be determined that our scheme will not leak any information of data owners in server C.

- **Security analysis under attacks from server S.** Server S is a trusted server, so it can do initialize public parameters of BCP, and generate the parameters of LDEM. What's more, it is responsible for transforming ciphertexts with server C together. In the process of transforming ciphertexts, it only receives encrypted digits, but it do not know the meaning of these digits, and these digits are the disturbed digits. Therefore, there is no more than one-half probability to guess the original true digit, and random perturbation can achieve indistinguishable.
- **Security analysis under attacks from different data owners.** The original data set of any data owner is secure, since there is no any communication about original data set among data owners in our scheme. For each data owner, they may do the key's times attack. For example, data owner A has finished an anomaly detection task with the help of sketch tables received from server C, and obtain the density $Density(x)$ of an instance x. He may try to guess the times in others' sketch tables by subtracting the $density(x)$ obtained from his own sketch tables. Assume the received sketch tables of data owner A are combined with Q data owners, and then he can only get the sum density of others, $Density(x) - density(x) = \sum_{q=1}^{Q-1} density_q(x)$. Analysis the value $Density(x) - density(x)$, he only can know whether there are other participants, and can not defer any other privacy information with more than one-half probability. Therefore, for any data owners, they only can know their own data set and the density value of their data set in our scheme.

6. Discussion and conclusions

In this paper, we first propose a fast local density estimation method LDEM, which can be used in anomaly detection. LDEM obtains the local density of instances by calculating the average local density of all features, and the local density of each feature can be estimated by the defined mapping function. Compare with the existing density-based algorithms, LDEM has no any distance calculation, and it only need $O(N)$ time complexity.

Then, we extend this algorithm to the case of multi-party participants, and we propose an efficient scheme PPLDEM to detect anomaly instances with considering privacy protection under multi-party participants. PPLDEM is finished with the aid of a cloud, and this cloud is composed of two servers, server S and server C. In our scheme, the detection stage of PPLDEM are executed in each participant own, based on the sketch tables requested from server C. So compare with the existing anomaly detection algorithms with privacy preserving, our scheme need less communication cost and less calculation, under the premise of ensuring safety and detection accuracy. What's more, PPLDEM can be used to the cases both horizontally distributed data and vertically distributed data. And experiments and theoretical analysis show that our proposed scheme PPLDEM can detect anomaly instances effectively and efficiently.

Acknowledgments

This study was supported by the Shenzhen Research Council (Grant No. GJHZ20180928155209705).

Conflict of Interests

All authors declare no conflicts of interest in this paper.

References

1. D. M. Hawkins, *Identification of Outliers*, Springer, (1980).
2. E. M. Knox, R. T. Ng, *Algorithms for mining distancebased outliers in large datasets*, Proceedings of the international conference on very large data bases, Citeseer, 1998, 392–403. Available from: https://dl.acm.org/doi/10.5555/645924.671334.
3. X. Wang, X. L. Wang, M. Wilkes, *A fast distance-based outlier detection technique*, Industrial Conference on Data Mining-Poster and Workshop, 2008, 25–44. Available from: https://www.researchgate.net/publication/26621806_A_Fast_Distance-Based_Algorithm_to_Detect_Outliers.
4. M. Sugiyama, K. Borgwardt, *Rapid distance-based outlier detection via sampling*, Advances in Neural Information Processing Systems, 2013, 467–475. Available from: http://papers.nips.cc/paper/5127-rapid-distance-based-outlier-detection-via-sampling.
5. Z. He, X. Xu, S. Deng, Discovering cluster-based local outliers, *Pattern Recognit. Lett.*, **24** (2003), 1641–1650.
6. Z. Chen, A. W. C. Fu, J. Tang, *On complementarity of cluster and outlier detection schemes*, International Conference on Data Warehousing and Knowledge Discovery, Springer, 2003, 234–243. Available from: https://link.springer.com/chapter/10.1007/978-3-540-45228-7_24.
7. C. Zhang, H. Liu, A. Yin, *Research of detection algorithm for time series abnormal subsequence*, International Conference of Pioneering Computer Scientists, Engineers and Educators, Springer, 2017, 12–26. Available from: https://link.springer.com/chapter/10.1007/978-981-10-6385-5_2.
8. C. Zhang, A. Yin, Y. Wu, Y. Chen, X. Wang, *Fast time series discords detection with privacy preserving*, 2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications (TrustCom), IEEE, 2018, 1129–1139. Available from: https://ieeexplore.ieee.org/abstract/document/8456026.
9. H. P. Kriegel, M. Schubert, A. Zimek, *Angle-based outlier detection in high-dimensional data*, Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining, 2008, 444–452. Available from: https://dl.acm.org/doi/abs/10.1145/1401890.1401946.
10. N. Pham, R. Pagh, *A near-linear time approximation algorithm for angle-based outlier detection in high-dimensional data*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2012, 877–885. Available from: https://dl.acm.org/doi/abs/10.1145/2339530.2339669.

11. M. M. Breunig, H. P. Kriegel, R. T. Ng, J. Sander, *Lof: identifying density-based local outliers*, ACM sigmod record, 2000, 93–104. Available from: https://dl.acm.org/doi/abs/10.1145/342009.335388.
12. J. Gao, W. Hu, Z. M. Zhang, X. Zhang, O. Wu, *Rkof: Robust kernel-based local outlier detection*, Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2011, 270–283. Available from: https://link.springer.com/chapter/10.1007/978-3-642-20847-8_23.
13. L. Duan, L. Xu, G. Feng, J. Lee, B. Yan, A local-density based spatial clustering algorithm with noise, *Inf. Syst.*, **32** (2007), 978–986.
14. B. Tang, H. He, A local density-based approach for outlier detection, *Neurocomputing*, **241** (2017), 171–180.
15. C. Zhang, A. Yin, Y. Deng, P. Tian, X. Wang, L. Dong, *A novel anomaly detection algorithm based on trident tree*, International Conference on Cloud Computing, 2018, 295–306. Available from: https://link.springer.com/chapter/10.1007/978-3-319-94295-7_20.
16. M. Kantarcıoglu, C. Clifton, *Privately computing a distributed k-nn classifier*, PKDD2004: 8th European Conference on Principles and Practice of Knowledge Discovery in Databases, Pisa, Italy, 2004, 279–290. Available from: https://link.springer.com/chapter/10.1007/978-3-540-30116-5_27.
17. X. Lin, C. Clifton, M. Zhu, Privacy-preserving clustering with distributed em mixture modeling, *Knowl. Inf. Syst.*, **8** (2005), 68–81.
18. L. Li, L. Huang, W. Yang, X. Yao, A. Liu, Privacy-preserving lof outlier detection, *Knowl. Inf. Syst.*, **42** (2015), 579–597.
19. C. Zhang, Y. Zhou, J. Guo, G. Wang, X. Wang, Research on classification method of high-dimensional class-imbalanced data sets based on svm, *Int. J. Mach. Learn. Cybern.*, **10** (2019), 1765–1778.
20. L. T. Dung, H. T. Bao, *A distributed solution for privacy preserving outlier detection*, 2011 Third International Conference on Knowledge and Systems Engineering, 2011, 26–31. Available from: https://ieeexplore.ieee.org/abstract/document/6063441.
21. T. Li, Z. Huang, P. Li, Z. Liu, C. Jia, Outsourced privacy-preserving classification service over encrypted data, *J. Network Comput. Appl.*, **106** (2018), 100–110.
22. Z. Yu, C. Gao, Z. Jing, B. B. Gupta, Q. Cai, A practical public key encryption scheme based on learning parity with noise, *IEEE Access*, **6** (2018), 31918–31923.
23. T. Li, W. Chen, Y. Tang, H. Yan, A homomorphic network coding signature scheme for multiple sources and its application in iot, *Secur. Commun. Networks*, **2018** (2018), 9641273.
24. R. H. Jhaveri, N. M. Patel, Y. Zhong, A. K. Sangaiah, Sensitivity analysis of an attack-pattern discovery based trusted routing scheme for mobile ad-hoc networks in industrial Iot, *IEEE Access*, **6** (2018), 20085–20103.
25. C. Gao, S. Lv, Y. Wei, Z. Wang, Z. Liu, X. Cheng, M-sse: An effective searchable symmetric encryption with enhanced security for mobile devices, *IEEE Access*, **6** (2018), 38860–38869.

26. M. Xi, J. Wu, J. Li, G. Li, *Sema-icn: Toward semantic information-centric networking supporting smart anomalous access detection*, 2018 IEEE Global Communications Conference (GLOBECOM), 2018, 1–6. Available from: https://ieeexplore.ieee.org/abstract/document/8647325.
27. V. Sharma, R.KUMAR, W. Cheng, M. Atiquzzaman, K. Srinivasan, A. Y. Zomaya, Nhad: Neuro-fuzzy based horizontal anomaly detection in online social networks, *IEEE Trans. Knowl. Data Eng.*, **30** (2018), 2171–2184.
28. T. ElGamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Trans. Inf. Theory*, **31** (1985), 469–472.
29. R. Bendlin, I. Damgård, C. Orlandi, S. Zakarias, *Semi-homomorphic encryption and multiparty computation*, Annual International Conference on the Theory and Applications of Cryptographic Techniques Springer, 2011, 169–188. Available from: https://link.springer.com/chapter/10.1007/978-3-642-20465-4_11.
30. I. Damgård, V. Pastro, N. Smart, S. Zakarias, *Multiparty computation from somewhat homomorphic encryption*, Advances in Cryptology–CRYPTO 2012, Springer, 2012, 643–662. Available from: https://link.springer.com/chapter/10.1007/978-3-642-32009-5_38.
31. A. Peter, E. Tews, S. Katzenbeisser, Efficiently outsourcing multiparty computation under multiple keys, *IEEE Trans. Inf. Forensics Secur.*, **8** (2013), 2046–2058.
32. X. Liu, R. H. Deng, K. K. R. Choo, J. Weng, An efficient privacy-preserving outsourced calculation toolkit with multiple keys, *IEEE Trans. Inf. Forensics Secur.*, **11** (2016), 2401–2414.
33. E. Bresson, D. Catalano, D. Pointcheval, *A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications*, International Conference on the Theory and Application of Cryptology and Information Security, Advances in Cryptology-ASIACRYPT 2003, 37–54. Available from: https://link.springer.com/chapter/10.1007/978-3-540-40061-5_3.
34. S. Sathe, C. C. Aggarwal, *Subspace outlier detection in linear time with randomized hashing*, Data Mining (ICDM), 2016 IEEE 16th International Conference on, IEEE, 2016, 459–468. Available from: https://ieeexplore.ieee.org/abstract/document/7837870.
35. W. Harper, Statistics: Theory and methods, *Technometrics*, **33** (1991), 369–370.
36. C. C. Aggarwal, S. Sathe, Theoretical foundations and algorithms for outlier ensembles, *ACM SIGKDD Explor. Newsl.*, **17** (2015), 24–47.
37. Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, et al., *The ucr time series classification archive*, 2015, Available from: www.cs.ucr.edu/ eamonn/time_series_data/.
38. F. T. Liu, K. M. Ting, Z. H. Zhou, *Isolation forest*, Eighth IEEE International Conference on Data Mining, IEEE, 2008, 413–422. Available from: https://ieeexplore.ieee.org/abstract/document/4781136.
39. F. T. Liu, K. M. Ting, Z. H. Zhou, Isolation-based anomaly detection, *ACM Trans. Knowl. Discovery Data*, **6** (2012), 3.

40. R. L. S. Torres, R. Visvanathan, S. Hoskins, A. V. D. Hengel, D. C. Ranasinghe, Effectiveness of a batteryless and wireless wearable sensor system for identifying bed and chair exits in healthy older people, *Sensors*, **16** (2016), 546.

41. S. Goldwasser, S. Micali, Probabilistic encryption, *J. Comput. Syst. Sci.*, **28** (1984), 270–299,