Research Article

Privacy-Preserving KNN Classification Algorithm for Smart Grid

Zhuhuan Song, Yanli Ren, and Gang He

1School of Communication and Information Engineering, Shanghai University, Shanghai 200444, China
2Henan Key Laboratory of Network Cryptography Technology, Zhengzhou, Henan, China

Correspondence should be addressed to Yanli Ren; renyanli@shu.edu.cn

Received 30 March 2022; Accepted 26 April 2022; Published 11 May 2022

Copyright © 2022 Zhuhuan Song et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the development of Internet of Things (IOT), outsourcing data and tasks to a cloud server has become a popular and economical way for small devices with restricted ability. The $k$-nearest neighbors (KNNs) classification algorithm have been commonly applied to medical image classification, abnormal detection, defective product identification, and so on. The previous privacy-preserving KNN algorithms are based on two cloud servers, which have high computational and communication costs. In this paper, we design a privacy-preserving KNN classification (PPKC) algorithm with single cloud server for smart grid. Specifically, each smart meter and control center encrypted their data with Paillier cryptosystem and the cloud server does some calculation on the encrypted data. We prove that PPKC can protect the privacy of both smart meters and control centers, and the classification results are also private for the server. Besides, both smart meters and control centers can stay offline after uploading their data. The experiment results demonstrate that the proposed PPKC algorithm is more efficient than the previous algorithms and it can obtain almost the same accuracy as the original KNN, which means the PPKC algorithm is more applicable for small devices in IOT.

1. Introduction

In recent years, Internet of Things (IOT) has been widely used in medical devices, industrial detection, intelligent transportation, and so on. At the same time, more and more applications of machine learning have been applied to IOT, which can greatly facilitate our work and life [1].

As we know, a large amount of data can be generated in IOT. If massive data is given to small terminals for storage or calculation, it may undoubtedly increase the burden of these devices and reduce their efficiency [2]. In the worst case, this may even lead to calculation errors and the loss of stored data. Fortunately, the cloud server can be used to store and calculate data. Generally speaking, the cloud server has huge computing resources and storage capacity [3]. For these devices or users with insufficient computing power, they only need to spend a small amount of money and then enjoy the benefits of cloud platform including powerful computing ability and massive storage resources. For example, in the medical industry, using cloud platform can help doctors judge the specific situation of the disease quickly and then save the lives of patients [4].

Notice that training models of machine learning or deep learning on cloud platform can also bring great convenience to those users with limited computing power [5]. However, the users cannot directly outsource private data to the cloud server since it is always untrusted. Usually, the devices or users encrypt sensitive information before sending them to the cloud server. Unfortunately, calculating on the encrypted data may not get the correct answer. In addition, for small devices, many encryption algorithms may also consume a lot of computing resources and spend a lot of time. Therefore, it is urgent to propose an efficient privacy-preserving algorithm for the outsourced data.

At present, there are many machine learning algorithms, such as decision tree [6], $k$-means [7], support vector machine (SVM) [8], and so on. By using these algorithms, we can efficiently carry out the classification task and get the results. Note that $k$-nearest neighbor (KNN) [9] classification algorithm is one of the most commonly used machine
learning algorithms, which has been widely applied to IOT, including smart grid and smart home. Typically, KNN classification algorithm can obtain high accuracy in medical image classification, abnormal detection, defective product identification, and so on. For example, a lot of electrical consumption data will be generated in smart grid where the control centers collect the electricity consumption in a specific area and adjust the power generation to maintain the balance of supply and power demand. However, KNN classification may have a high computational complexity for big data, and the resource-constrained terminals may spend much time to complete the algorithm. Outsourcing these tasks to the cloud server is a solution to this problem. In the past few years, many privacy-preserving KNN classification algorithms have been presented. However, they either have security risks or are based on several noncolluding cloud servers [10, 11], which may be very difficult to be realized. Therefore, we aim to propose an efficient and secure privacy-preserving KNN classification algorithm for the outsourced data with only one cloud server.

1.1. Related Work. With the rapid development of machine learning, people pay more attention to the security risks. Rahulamathavan et al. [12] designed a secure SVM classification algorithm, which uses Paillier homomorphic encryption to protect the data. However, this scheme cannot protect the output result from the cloud server. Yuan and Tian [13] put forward a secure K-means clustering algorithm by using MapReduce, which encrypts the data based on the learning with error (LWE) problem, and the cloud server iterates on the encrypted data. However, the client should interact with cloud server several times, which appends the computational costs of the client. Liu et al. [14] proposed a secure decision tree training and evaluation algorithm, which realized data privacy protection by using public key encryption with distributed trapdoors and additive secret sharing. This protocol protects the secret of users well, but it can only be used in ID3 algorithm and is not applicable to other decision tree algorithms.

Many privacy-preserving KNN classification protocols have been extensively studied in the past years. Samanthula et al. [10] presented the first privacy-preserving KNN classification protocol, which used the Pailler cryptosystem and two cloud servers to build some secure blocks, including secure Euclidean distance, secure multiplication, secure minimum value, and secure frequency to hide the private data. However, the computation complexity is very high, and it is infeasible for limited users on large datasets. Rong et al. [15] presented a cooperative KNN protocol with ElGamal cryptosystem [16]. However, the protocol has high computational complexity and uses two cloud servers. For multiple data owners, Li et al. [17] proposed a secure KNN classification protocol. However, this protocol cannot protect the outputs from the servers. Cheng et al. [18] put forward a secure KNN query protocol with multiple keys. Unfortunately, the protocol is based on multiple servers and cannot realize the privacy of data. Yang et al. [19] designed an efficient KNN scheme by using vector homomorphic encryption (VHE) [20], which achieved high efficiency. However, it is proved that VHE has some secure risks in [21]. After that, Yang et al. [22] designed a secure VHE and applied it to KNN classification, but the query data cannot be protected from the server. Liu et al. [11] presented an efficient KNN classification protocol, which greatly improves the computational efficiency by using additive secret sharing. However, this protocol uses two noncolluding servers and needs multiple interactions between servers.

In summary, there is no privacy-preserving KNN algorithm that can simultaneously realize the privacy of data, query, and classification results based on only one cloud server.

1.2. Our Contributions. In this paper, we propose a privacy-preserving KNN classification (PPKC) algorithm with only one cloud server, which is much more secure and efficient compared with the previous algorithms [10, 11]. Specifically, the contributions of this paper are summarized as follows.

(1) Based on the Paillier cryptosystem, we design a privacy-preserving KNN classification (PPKC) algorithm, which only uses one cloud server. In [10, 11], the outsourcing algorithms of KNN are all based on two noncolluding servers, which needs multiple interactions between servers and is almost impossible in practical applications.

(2) The proposed PPKC can not only realize the privacy of both control centers and smart meters but also protect the final outputs from the servers. After the smart meters upload encrypted data to the cloud server, they only need to keep offline. Similarly, the control centers upload encrypted query data and then keep offline until the cloud server return the encrypted results. Therefore, the proposed algorithm can not only protect the sensitive information but also reduce the computational and communication costs of smart meters and control centers as much as possible.

(3) Some experiments are evaluated to analyze the performance of PPKC on four relevant datasets. The PPKC algorithm is the most efficient for the smart meters compare with the previous work. Besides, the accuracy of proposed PPKC is nearly same as the original KNN, which means PPKC can help control centers find the abnormal power consumption of smart meters without leaking privacy.

The rest of our paper is organized as follows. We briefly introduce some preliminaries in Section 2. Section 3 formalizes the concrete construction of PPKC algorithm. We give the security proofs, computational complexity, and efficiency comparisons in Section 4. Section 5 gives the performance evaluations of efficiency and accuracy. Finally, we conclude the paper in Section 6.

2. Preliminaries

In this section, we mainly introduce the original KNN algorithm, Paillier cryptosystem, and the system model, which are the basis of the PPKC algorithm.
2.1. K-Nearest Neighbors (KNNs) Classification Algorithm. KNN algorithm is a nonparametric classification or regression algorithm in the field of pattern recognition. In this subsection, we briefly introduce the KNN classification algorithm. Assume that there are a few training data which have several attributes and a label. Next, there is a group of testing data which only has some attributes without labels. The aim of KNN classification algorithm is to get the label of the testing data. The specific process is as follows.

1. Prepare training data and testing data. There are $m$ groups of training data and each group has $n$ attributes $\{\sigma_{11}, \sigma_{12}, \ldots, \sigma_{in}\}$ and a label $t_i$, where $i \in [1, m]$. Suppose there is a group of testing data $\{q_1, q_2, \ldots, q_n\}$.
2. Compute the Euclidean distance $d_i$ between each training data and the testing data.
   $$d_i = (q_1 - \sigma_{i1})^2 + \cdots + (q_n - \sigma_{in})^2, i \in [1, m].$$
3. All the distance $d_i$ and its corresponding label $t_i$ are used to generate an ordered collection $C = [(d_1, t_1), (d_2, t_2), \ldots, (d_m, t_m)]$.
4. Sort the ordered collection $C$ of distance $d_i$ and labels $t_i$ according to the distance $d_i$.
5. Select the first $K$ labels from the sorted collection $C$.
6. Count the frequency of the $K$ labels and set the label with the highest frequency as the final result of the testing data.

2.2. Paillier Cryptosystem. Paillier cryptosystem [23] is a public key cryptosystem, which realizes additive homomorphism and is mainly used in signal processing and data processing. Its security relies on the problem of determining $n$-order residue classes. We briefly give the introduction of Paillier cryptosystem as follows.

1. Key generation: input a security parameter $\kappa$ and two big primes $p$ and $q$ with bit length $\kappa$. Calculate $N = p \cdot q$ and $\lambda = \text{lcm}(p-1,q-1)$, where $\text{lcm}(\cdot)$ means the least common multiple. Define $L(x) = (x-1)/N$. Next, choose a random generator $g \in Z^*_N$ and calculate $\mu = (L(g^4 \mod N^2))^{-1} \mod N$. Finally, the public key $pk$ is $(N, g)$ and the private key is $(\mu, \lambda)$.
2. Encryption: input a plaintext $m \in Z_N$, the ciphertext $c$ can be computed with public key $pk$ as follows:
   $$c = E_{pk}(m) = g^m \cdot r^N \mod N^2,$$
   where $r$ is the random number in $Z^*_N$.
3. Decryption: for a ciphertext $c \in Z^*_N$, the plaintext $m$ can be computed with the private key $sk$ as follows:
   $$m = D_{sk}(c) = L\left(c^\lambda \mod N^2\right) \cdot \mu \mod N.$$

In addition, the Paillier cryptosystem has the following properties:

1. Homomorphic addition:

\[E_{pk}(a + b) = E_{pk}(a) + E_{pk}(b),\]

\[E_{pk}(a \cdot b) = E_{pk}(a) \cdot E_{pk}(b),\]

where $a$ and $b$ are any two plaintexts and $pk$ is the public key.

2.2.1. Control Centers. After collecting some abnormal data in this city, the control center encrypts and sends them to the cloud server. OX_ he cloud server uses KNN classification algorithm to obtain any sensitive data about plaintext from the ciphertext.

In the proposed PPKC algorithm, smart meters and control centers first use Paillier cryptosystem to encrypt sensitive data and then transmit the ciphertext to the cloud server. Notice that all data in the Paillier cryptosystem belong to $Z_N$. We omit mod $n$ in the following content for simplicity.

2.3. System Model. The proposed PPKC can be used in the smart grid and the participants include smart meters, control centers, and the cloud server. Figure 1 demonstrates the system model of PPKC algorithm.

Assume that there are some smart meters in a city, which can separately collect the power consumption in a region of the city and send these collected data to the cloud server. At the same time, some control centers can collect and detect abnormal data in this city. After collecting some abnormal data, the control center encrypts and sends them to the cloud server. The cloud server uses KNN classification algorithm to calculate and obtain the label of abnormal data and send it back. The control center can then timely find out which region has abnormal power consumption according to the label and solve the problem in time. The detailed introduction of each participant is listed as follows.

2.3.1. Smart Meters. The smart meters are composed of various sensors which can collect the energy consumption information of electrical equipment. However, these sensors have limited storage and cannot store massive data. In the
PPKC model, the smart meters send the collected data to the cloud server. Suppose there are \( m \) smart meters \( x_i \), where \( i \in [1, m] \). Each smart meter \( x_i \) has a dataset \( D_i = \{x_{i1}, x_{i2}, \ldots, x_{in}, t_i\} \), where \( x_{i1}, x_{i2}, \ldots, x_{in} \) and \( t_i \) are separately the attributes and the label of the dataset. At the same time, each smart meter \( x_i \) needs to generate a tuple of random numbers \( \{a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)}\} \), which is used to encrypt their data. For saving storage, the smart meter uploads the ciphertext \( D'_i \) to the cloud server.

2.3.2. Control Centers. The control centers in the smart grid collect abnormal data to realize power adjustment, abnormal remind, or process optimization. In the PPKC model, the control centers submit some query data to obtain the labels of abnormal data. Suppose that there are \( l \) control centers \( Q_j \), where \( j \in [1, l] \). Each control center \( Q_j \) has a query dataset \( q_j = \{Q_{j1}, Q_{j2}, \ldots, Q_{jn}\} \) and sends encrypted query data \( q'_j \) to the cloud server. Each control center \( Q_j \) broadcasts the public key to smart meters.

2.3.3. Cloud Server. The cloud server has powerful computational resources and storage capacity to implement KNN classification algorithm. In the proposed PPKC algorithm, it will do some calculation on the encrypted data from smart meters and control centers and return the encrypted results to the control centers. The cloud server broadcasts the public key to smart meters and control centers. We suppose that the cloud server is honest but curious, which denotes that the cloud server correctly carries out every step in the algorithm, but may try to obtain the private data of the smart meters and control center.

2.4. Security Model. IND-CPA Security. In a public key cryptosystem, the indistinguishability of ciphertext against chosen plaintext attack (IND-CPA) [24] is executed by the following game between an adversary \( A \) and a challenger \( C \).

1. Setup. The challenger \( C \) generates public key \( PK \) and private key \( SK \) based on security parameter and publishes \( PK \) to the adversary \( A \) and keeps \( SK \) secret.

2. Queries. The adversary \( A \) can execute encryption queries to the challenger. The adversary \( A \) sends a plaintext \( m \) to the challenger. The challenger \( C \) executes Encrypt algorithm and generates the corresponding ciphertext \( c \) and sends it to \( A \).

3. Challenge. The adversary \( A \) randomly selects two plaintexts \( m_0, m_1 \) with the same length and returns them to \( C \). These two plaintexts \( m_0, m_1 \) are not queried in the Queries phase. The challenger randomly selects \( b \in \{0, 1\} \) and transmits the ciphertext \( c_b = \text{Encrypt}(PK, m_b) \) to \( A \).

4. Guess. \( A \) sends a bit \( b' \in \{0, 1\} \) as its guess. If \( b' = b \) with a probability which is larger than \( 1/2 \), the adversary wins the game and the advantage is defined as \( |\Pr[b' = b] - 1/2| \).

An encryption system is \((t, \varepsilon)\)-IND-CPA secure if all \( t \)-time IND-CPA adversaries have advantage of at most \( \varepsilon \) in winning the game.

3. The Proposed PPKC Algorithm

We introduce the privacy-preserving KNN classification algorithm in this section. Compared with [10, 11], the proposed PPKC algorithm is constructed based on single cloud server and realizes input privacy, query privacy, and output privacy simultaneously.

As shown in Section 2, there are three types of participants in the proposed PPKC algorithm, including smart meters (SM), control centers (CC), and cloud server (CS). All the data of smart meters and control centers belong to \( \mathbb{Z}_N \). The public keys and private keys of cloud server and control centers are denoted as \( PK, SK_{qj} \) and \( SK, SK_{qj} \). The proposed PPKC, the cloud server and control centers encrypt their data by using Paillier cryptosystem.

3.1. The Concrete Construction. The proposed PPKC algorithm includes uploading encrypted data, uploading query data, executing KNN on the ciphertext, and decrypting the ciphertext.

3.1.1. Uploading Encrypted Data. Assume that there are \( m \) smart meters. For a smart meter \( x_i \), it firstly chooses a tuple of random numbers \( \{a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)}\} \) and calculates \( x_{i1} - a_{i1}, x_{i2} - a_{i2}, \ldots, x_{in} - a_{in}, t_i - a_{i(n+1)} \). Next, it uses \( PK \) to encrypt the first \( n \) data and uses \( PK_{qj} \) to encrypt the last data and all the random numbers. Finally, it transmits these secure data to the cloud server. After this step, smart meters can stay offline.

3.1.2. Uploading Query Data. Assume the query data is \( \{Q_{j1}, Q_{j2}, \ldots, Q_{jm}\} \). The control center \( Q_j \) downloads \( \{EPK_{qj}(a_{j1}), EPK_{qj}(a_{j2}), \ldots, EPK_{qj}(a_{jn})\} \), \( i \in [1, m] \), from the cloud server. It then decrypts and obtains random numbers \( \{a_{j1}, a_{j2}, \ldots, a_{jn}\} \), \( i \in [1, m] \). Next, the control center \( Q_j \) calculates each random number minus all the query data and encrypts them with \( PK \). Finally, the control center \( Q_j \) sends the ciphertext of \( \{a_{j1} - Q_{j1}, a_{j2} - Q_{j2}, \ldots, a_{jn} - Q_{jm}\} \), \( i \in [1, m] \), to the cloud server. After that, the control center can stay offline.

3.1.3. Executing KNN on the Ciphertext. After receiving the encrypted data from the control center \( Q_j \) and smart meters, the cloud server computes the dot products based on \( \{EPK(x_{j1} - a_{j1}), \ldots, EPK(x_{jn} - a_{jn})\} \) and \( \{EPK(a_{j1} - Q_{j1}), \ldots, EPK(a_{jn} - Q_{jm})\} \) and obtains \( \{EPK(x_{j1} - Q_{j1}), \ldots, EPK(x_{jn} - Q_{jm})\} \). Then, the cloud server decrypts and gets \( \{x_{j1} - Q_{j1}, x_{j2} - Q_{j2}, \ldots, x_{jn} - Q_{jm}\} \), \( i \in [1, m] \). Next, the cloud server computes \( d_i \), which is the Euclidean
distance between the data of smart meters and the query
data, and each $d_i$ has a corresponding encrypted label $E_{PK_{q_j}}(t_i)$. At this time, the cloud server has $m$ groups of
\{ $d_i, E_{PK_{q_j}}(t_i)$ \}. After that, the cloud server sorts and selects
the smallest $K$ numbers in $\{ d_1, d_2, \ldots, d_m \}$ and their corresponding encrypted labels. Finally, these $K$ encrypted
labels are transmitted to the control center $Q_j$.

3.1.4. Decrypting the Ciphertext. The control center $Q_j$
decrypts these $K$ encrypted labels and calculates the fre-
cquency of each label. Finally, $Q_j$ sets the label with the
highest frequency as the result.

Please see Algorithm 1 for the details of the proposed
PPKC algorithm.

3.2. Correctness. The correctness of PPKC relies on the
properties of Paillier cryptosystem, including homomorphic
addition and scalar multiplicative homomorphism.

After the smart meters upload encrypted data and the
control center uploads query data, the cloud server executes
KNN on the ciphertext. The cloud server calculates the dots
products based on $\{ E_{PK} (x_{i1} - a_{i1}), \ldots, E_{PK} (x_{in} - a_{in}) \}$ and $\{ E_{PK} (a_{i1} - Q_{j1}), \ldots, E_{PK} (a_{in} - Q_{jn}) \}$. According to the
homomorphic addition of Paillier cryptosystem, we can easily get the following equation:

$$
E_{PK}(x_{i1} - a_{i1}) \times E_{PK}(a_{i1} - Q_{j1})
\times E_{PK}(x_{i2} - a_{i2} + a_{i1} - Q_{j1})
\times \cdots
\times E_{PK}(x_{in} - a_{in} + n \times Q_{jn})
= E_{PK}(x_{i1} - Q_{j1}).
$$

(6)

Besides, the cloud server needs to calculate the en-
cryption labels. According to the homomorphic addition of Paillier cryptosystem, we can get the following equation:

$$
E_{PK_{q_j}}(t_i - a_{i(n+1)}) \times E_{PK_{q_j}}(a_{i(n+1)})
\times E_{PK_{q_j}}(t_i - a_{i(n+1)} + a_{i(n+1)})
\times \cdots
\times E_{PK_{q_j}}(t_i - a_{i(n+1)} + a_{i(n+1)})
= E_{PK_{q_j}}(t_i).
$$

(7)

Finally, the cloud server returns $K$ encrypted labels to the
control center $Q_j$. If the cloud server follows every step of
the proposed PPKC, the control center finally gets the correct
label.

Therefore, we can prove that the result obtained by the
control center is correct through the above analysis.

4. Analysis of the PPKC Algorithm

In this section, we analyze the data privacy of smart meters
and control centers and prove that the data given back by the
cloud server is secure. We assume that the cloud server is
semi-honest.

Therefore, we should prove that the private information of
both smart meters and control centers is all private for the
cloud server. Moreover, the outputs should also be well
protected from the server.

4.1. Security Analysis

Theorem 1. In the PPKC algorithm, the privacy of the smart
meter is ensured for the server based on the Paillier cryptosystem.

Proof. The security proof is executed between an ad-
versary $A$ and a challenger $C$. At the beginning of the game,
the challenger is given a ciphertext of Paillier cryptosystem
$E_{PK}(x_{b})$ to decide the corresponding plaintext is $x_{a}$ or $x_{b}$.
Suppose the adversary distinguish the ciphertext of smart
meter with an overwhelming probability; the challenger can
distinguish the ciphertext of Paillier cryptosystem with an
overwhelming probability.

(1) Setup. The public keys of smart meter, cloud server,
and control centers are published to the adversary $A$ and
the challenger $C$.

(2) Queries. Adversary $A$ first executes encryption
queries to the challenger. The adversary $A$ sends $\{ x_{i1}, x_{i2}, \ldots, x_{in}, t_i \}$ to the challenger. The challenger
$C$ executes the PPKC algorithm, selects some ran-
dom numbers $\{ a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)} \}$ and returns
the corresponding ciphertexts:

$$
E_{PK_{q_j}}(x_{i1} - a_{i1}), E_{PK_{q_j}}(x_{i2} - a_{i2}), \ldots, E_{PK_{q_j}}(x_{in} - a_{in})
\times E_{PK_{q_j}}(t_i - a_{i(n+1)}), E_{PK_{q_j}}(a_{i1}), \ldots, E_{PK_{q_j}}(a_{i(n+1)})
$$

(8)

where $PK, PK_{q_j}$ are the public keys of cloud server
and control center $Q_j$, respectively.

(3) Challenge. The adversary $A$ randomly picks two
messages $\{ x_{01}, x_{02}, \ldots, x_{0n}, t_0 \}, \{ x_{11}, x_{12}, \ldots, x_{1m}, t_1 \}$
and transmits them to $C$. Notice that these two
plaintexts are not queried in the Queries phase. The
challenger randomly selects $b \in \{ 0, 1 \}$ and returns the
ciphertext to adversary $A$:

$$
\{ E_{PK_{q_j}}(x_0), E_{PK_{q_j}}(x_{b1} - a_{b1}), \ldots, E_{PK_{q_j}}(x_{bn} - a_{bn}) \},
\{ E_{PK_{q_j}}(t_b - a_{b(n+1)}), E_{PK_{q_j}}(x_{b1} - x_b), E_{PK_{q_j}}(a_{b2}), E_{PK_{q_j}}(a_{b3}) \ldots, E_{PK_{q_j}}(a_{b(n+1)}) \}
$$

(9)
where $E_{PK_{qj}}(x_{b1} - x_{b}) = E_{PK_{qj}}(x_{b1})E_{PK_{qj}}(x_{b})^{-1}$, and \{a_{b2}, \ldots, a_{b(n+1)}\} are random numbers chosen by the challenger.

(4) Guess. A sends a bit $b' \in \{0, 1\}$ as its guess, and then the challenger returns $b' \in \{0, 1\}$. If the adversary wins the game with an advantage of $\epsilon$, then the challenger can distinguish the ciphertext of Paillier cryptosystem with an overwhelming probability.

Therefore, the privacy of the smart meter is assured for the server based on the Paillier cryptosystem.

**Theorem 2.** In the PPKC algorithm, the privacy of the control center is also ensured for the server based on the Paillier cryptosystem.

Proof. The same as Theorem 1, the challenger is firstly given a ciphertext of Paillier cryptosystem $E_{PK}(x_{b})$ to decide the corresponding plaintext is $x_{0}$ or $x_{1}$. Suppose the adversary distinguish the ciphertext of control center with an overwhelming probability, the challenger can distinguish the ciphertext of Paillier cryptosystem with an overwhelming probability.

(1) Setup. The same as Theorem 1.

(2) Queries. The adversary $A$ can execute encryption queries to the challenger. The adversary $A$ sends \{Q_{01}, Q_{02}, \ldots, Q_{m}\} to the challenger. The challenger $C$ executes the PPKC algorithm, selects some numbers $\{a_{11}, a_{12}, \ldots, a_{m}, a_{m1}, a_{m2}, \ldots, a_{mn}\}$, and returns the corresponding ciphertexts:

\[
\{E_{PK}(a_{11} - Q_{j1}), \ldots, E_{PK}(a_{m} - Q_{j})\} \tag{10}
\]

\[
\{E_{PK}(a_{m1} - Q_{j1}), \ldots, E_{PK}(a_{mn} - Q_{j})\}. \tag{11}
\]

where $PK_{qj}$ is the public key of the control center and $\{a_{1}, \ldots, a_{K}\}$ are random numbers chosen by the challenger.

(3) Challenge. The adversary $A$ randomly chooses two queries $\{Q_{01}, Q_{02}, \ldots, Q_{m}\}, \{Q_{11}, Q_{12}, \ldots, Q_{in}\}$ and sends them to $C$. Notice that these two tuples of labels are not queried in the Queries phase. The challenger selects $b \in \{0, 1\}$ and returns the ciphertext to $A$:

\[
\{E_{PK}(x_{b}), E_{PK}(a_{i1} - Q_{b1}), \ldots, E_{PK}(a_{in} - Q_{bn})\}
\]

\[
\{E_{PK}(a_{m1} - Q_{b1}), \ldots, E_{PK}(a_{mn} - Q_{bn})\},
\]

where $\{a_{12}, \ldots, a_{in}, a_{m1}, a_{m2}, \ldots, a_{mn}\}$ are random numbers chosen by the challenger.

(4) Guess. The same as Theorem 1.

Therefore, the result of the query is protected from the server based on the Paillier cryptosystem.

**Theorem 3.** In the PPKC algorithm, the result of query is protected from the server based on the Paillier cryptosystem.

Proof. As same as Theorem 1, the challenger is firstly given a ciphertext of Paillier cryptosystem $E_{PK_{qj}}(t_{b})$ to decide the corresponding plaintext is $t_{0}$ or $t_{1}$. Suppose the adversary distinguish the ciphertext of the query result with an overwhelming probability; the challenger can distinguish the ciphertext of Paillier cryptosystem with an overwhelming probability.

(1) Setup. The same as Theorem 1.

(2) Queries. Adversary $A$ can execute encryption queries to the challenger. The adversary $A$ sends the labels $\{t_{b1}, t_{b2}, \ldots, t_{bK}\}$ to the challenger. The challenger $C$ executes the PPKC algorithm and returns the encrypted labels:

\[
\{E_{PK_{qj}}(t_{1} - a_{1}), \ldots, E_{PK_{qj}}(t_{K} - a_{K}), E_{PK_{qj}}(a_{1}), E_{PK_{qj}}(a_{2}), \ldots, E_{PK_{qj}}(a_{K})\}. \tag{12}
\]

where $PK_{qj}$ is the public key of cloud server.

(3) Challenge. The adversary $A$ randomly chooses two numbers chosen by the challenger.

(4) Guess. The same as Theorem 1.

Therefore, the result of the query is protected from the server based on the Paillier cryptosystem.
Suppose that there are \( m \) smart meters and \( l \) control centers. Each smart meter \( x_i \) has \( \{x_{i1}, x_{i2}, \ldots, x_{in}, t_i\} \), \( i \in [1, m] \), where \( t_i \) is the label and the attributes are \( \{x_{i1}, x_{i2}, \ldots, x_{in}\} \). Each control center \( Q_j \) has a query data \( \{Q_{j1}, Q_{j2}, \ldots, Q_{jn}\} \), \( j \in [1, l] \). PK, SK are public key and private key of the cloud server while \( PK_{Q_j}, SK_{Q_j} \) are public key and private key of the control center \( Q_j \).

**Smart Meter \( x_i \), Uploading Encrypted Data:**

1. Generate a series of random numbers \( \{a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)}\} \) and calculate \( \{x_{i1} - a_{i1}, x_{i2} - a_{i2}, \ldots, x_{in} - a_{in}, t_i - a_{i(n+1)}\} \).
2. Use public key \( PK \) and \( PK_{Q_j} \) to encrypt \( x_{i1} - a_{i1}, x_{i2} - a_{i2}, \ldots, x_{in} - a_{in} \) and \( t_i - a_{i(n+1)} \), respectively.
3. Send these encrypted numbers, \( \{EPK_{Q_j}(x_{i1} - a_{i1}), EPK_{Q_j}(x_{i2} - a_{i2}), \ldots, EPK_{Q_j}(x_{in} - a_{in})\} \) and \( \{EPK_{Q_j}(t_i - a_{i(n+1)}), EPK_{Q_j}(a_{i1}), EPK_{Q_j}(a_{i2}), \ldots, EPK_{Q_j}(a_{in}), EPK_{Q_j}(a_{i(n+1)})\}, j \in [1, l] \), to the cloud server.

**Control Center \( Q_j \), Uploading Query Data:**

1. Download \( \{EPK_{Q_j}(a_{i1}), EPK_{Q_j}(a_{i2}), \ldots, EPK_{Q_j}(a_{in})\}, i \in [1, m] \), from the cloud server.
2. Decrypt by using the private key \( SK_{Q_j} \) and obtain these random numbers \( \{a_{i1}, a_{i2}, \ldots, a_{in}\} \), \( i \in [1, m] \).
3. Calculate \( \{a_{i1} - Q_{j1}, a_{i2} - Q_{j2}, \ldots, a_{in} - Q_{jn}\} \).
4. Encrypt the above random numbers by using public key \( PK \) and get \( \{EPK(a_{i1} - Q_{j1}), EPK(a_{i2} - Q_{j2}), \ldots, EPK(a_{in} - Q_{jn})\} \).
5. Send these encrypted random numbers to the cloud server.

**The Cloud Server Executing KNN on the Ciphertext:**

\[
\begin{align*}
&\{EPK(x_{i1} - a_{i1}) \cdot EPK(a_{i1} - Q_{j1}) = EPK(x_{i1} - Q_{j1}), \ldots, EPK(x_{in} - a_{in}) \cdot EPK(a_{in} - Q_{jn}) = EPK(x_{in} - Q_{jn})\}, i \in [1, m],
\end{align*}
\]
and then use \( SK \) to decrypt these encrypted numbers.

2. Calculate \( d_i = (x_{i1} - Q_{j1})^2 + (x_{i2} - Q_{j2})^2 + \cdots + (x_{in} - Q_{jn})^2 \), \( i \in [1, m] \), which is the Euclidean distance between the data of smart meters and the query data.
3. Calculate the encrypted label of each Euclidean distance \( d_i \). In detail, \( EPK_{Q_j}(t_i - a_{i(n+1)}), EPK_{Q_j}(a_{i(n+1)}) = EPK_{Q_j}(t_i) \). Thus, there are \( m \) groups Euclidean distance \( d_i \) and its corresponding encrypted label \( \{d_i, EPK_{Q_j}(t_i)\}, \{d_2, EPK_{Q_j}(t_2)\}, \ldots, \{d_m, EPK_{Q_j}(t_m)\}\).
4. Sort and select the smallest \( K \) numbers in \( \{d_1, d_2, \ldots, d_m\} \) and their corresponding encrypted labels.
5. Send these \( K \) encrypted labels \( \{EPK_{Q_j}(t_1), EPK_{Q_j}(t_2), \ldots, EPK_{Q_j}(t_k)\} \), \( a \in [1, K] \), to the control center \( Q_j \).

**Control Center \( Q_j \), Decryption the Ciphertext:**

1. Decrypt these \( K \) encrypted labels with \( SK_{Q_j} \) and get \( \{t_1, t_2, \ldots, t_K\} \).
2. Calculate the frequency of each label.
3. Set the label with the highest frequency as the final result.

**Algorithm 1:** PPKC.

### 4.2. Computational Complexity

In Table 1, we give the final computation overheads of each participant, where \( n, m, l \) separately mean the quantity of data records, smart meters, and control centers. As mentioned in Section 3, there are totally \( m \) smart meters and each smart meter has \( n + 1 \) data records including \( n \) data attributes and one label. At the same time, there exist \( l \) control centers and each control center has \( n \) data records. We ignore the additions and multiplications in the following analysis since these operations have low computational costs and modular exponentiation (ME) and modular multiplication (MM) cost far more than them. For the Paillier cryptosystem, one encryption needs 2 ME and 1 MM, while one decryption needs 2 ME and 3 MM.

In the proposed PPKC, each smart meter needs to use \( PK_{Q_j} \) to encrypt \( \{t_i - a_{i(n+1)}, a_{i1}, a_{i2}, \ldots, a_{in}, a_{i(n+1)}\} \) and use \( PK \) to encrypt \( \{x_{i1} - a_{i1}, x_{i2} - a_{i2}, \ldots, x_{in} - a_{in}\} \). Thus, each smart meter performs \( n + l(n + 2) \) encryptions and the total number of all smart meters is \( m(n + (n + 2)) \) for encryption.

After downloading the encrypted random numbers, the control center \( Q_j \) decrypts and then encrypts the query data by using random numbers. Thus, each control center \( Q_j \) performs \( mn \) decryptions and \( mn \) encryptions. In the proposed PPKC, the cloud server needs to compute the Euclidean distance. For an Euclidean distance, the cloud server needs to perform \( n \) MM and \( n \) decryptions. Thus, computing \( m \) Euclidean distances needs to perform \( mn \) MM and \( mn \) decryptions. In the Decryption operation, the control center needs to decrypt \( K \) encrypted labels and calculates the label with the highest frequency. We ignore the operations in this paper because \( K \) is a small number.

Therefore, all the smart meters need to perform \( 2m(n + l(n + 2)) \) ME and \( m(n + l(n + 2)) \) MM. Each control center needs to perform \( mn \) encryptions and \( mn \) decryptions. Thus, the control center needs to execute \( 4mn \) ME and \( 4mn \) MM. The cloud server needs to perform \( 2mn \) ME and \( 4mn \) MM.
4.3. Comparisons. Table 2 shows the comparison of the proposed PPKC and other privacy-preserving KNN algorithms, where $m$, $k$, $v$ separately mean the quantity of data owners, the parameter of KNN, and the length of a value. Samanthula et al. [10] put forward a privacy-preserving KNN classification protocol, which is the first one to achieve query privacy, data privacy, and output privacy. Unfortunately, this protocol has high computational cost. Liu et al. [11] designed a secure KNN classification protocol with additive secret sharing, which protects all the data of each participant. Moreover, this protocol is much more efficient than [10]. However, these two protocols are constructed based on two noncolluding servers, which is difficult to be realized in reality. For multiple data owners, Li et al. [17] proposed a secure KNN classification protocol. Unfortunately, this protocol cannot totally protect the data from the servers. Cheng et al. [18] designed a secure KNN query protocol by utilizing the distributed two trapdoors public-key cryptosystem [25]. However, this protocol uses two cloud servers and cannot achieve the privacy of data. Yang et al. [19] presented another secure KNN classification algorithm with vector homomorphic encryption which is proved insecure in [21]. Thus, this protocol cannot protect any private information of data owners and query users. Yang et al. [22] used a secure VHE designed a privacy-preserving KNN classification algorithm, but the query data cannot be well protected.

Therefore, the proposed PPKC is the first algorithm which can protect the private information of smart meters, control centers, and the query results based on single cloud server.

5. Performance Evaluation

To observe the efficiency and accuracy of the PPKC algorithm, we execute the following two experiments. In the first experiment, we compare the PPKC with the previous protocols to demonstrate the efficiency of PPKC. Next, we test the accuracy of original KNN with that of PPKC to show that the accuracy does not decrease in the privacy-preserving setting.

All the experiments are designed on a MacBook Pro laptop equipped with 4 cores rated at 1.4 GHZ and 16 GB of RAM by Java language.

5.1. Computational Overheads. In this subsection, we do some experiments to show the effectiveness of the PPKC. As shown in Table 2, the proposed PPKC and [10, 11] achieve data privacy, query privacy, and result privacy, but [17–19, 22] cannot realize the properties simultaneously; we thus only compare the PPKC with these two protocols. Our experiments are performed on the Car Evaluation dataset from University of California Irvine (UCI) dataset [26], which is also used in [10, 11]. Specifically, there are total 1728 records and each record has 6 data attributes in this dataset, which is chosen from 5 to 25. Moreover, we test the time of control centers, smart meters, and cloud server during executing PPKC.

Since the length of $N$ has an effect on the efficiency of PPKC, we separately select $N = 512$ and $N = 1024$ in the following experiments, which is the same as [10, 11]. Tables 3 and 4 mainly present the computation cost of SM, CC, and CS in the PPKC algorithm for different $N$. We define $t_{PPKC}$ as the total time of executing PPKC. $t_{SM}$, $t_{CC}$, and $t_{CS}$ separately denote the time of smart meters, control center, and cloud server during executing PPKC.

We specifically discuss the running time of smart meters, control center, and cloud server, respectively. As shown in

### Table 1: Computational complexity of PPKC.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Smart meters</th>
<th>Control center</th>
<th>Cloud server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular exponentation</td>
<td>$2m(n+l(n+2))$</td>
<td>4 $mn$</td>
<td>2 $mn$</td>
</tr>
<tr>
<td>Modular multiplication</td>
<td>$m(n+l(n+2))$</td>
<td>4 $mn$</td>
<td>4 $mn$</td>
</tr>
</tbody>
</table>

### Table 2: Comparison of privacy-preserving KNN algorithms.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Data privacy</th>
<th>Query privacy</th>
<th>Output privacy</th>
<th>Single cloud server</th>
<th>Computation cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Samanthula et al. [10]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>O (mkv)</td>
</tr>
<tr>
<td>Liu et al. [11]</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>O (mk)</td>
</tr>
<tr>
<td>Li et al. [17]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>O (mk)</td>
</tr>
<tr>
<td>Cheng et al. [18]</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>O (mk)</td>
</tr>
<tr>
<td>Yang et al. [19]</td>
<td>✗</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>O (m)</td>
</tr>
<tr>
<td>Yang et al. [22]</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
<td>✓</td>
<td>O (m)</td>
</tr>
<tr>
<td>The PPKC algorithm</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>O (m)</td>
</tr>
</tbody>
</table>

### Table 3: Computation cost of PPKC when $N = 512$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$t_{SM}$ (sec)</th>
<th>$t_{CC}$ (sec)</th>
<th>$t_{CS}$ (sec)</th>
<th>$t_{PPKC}$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7.44</td>
<td>4.508</td>
<td>6.777</td>
<td>18.726</td>
</tr>
<tr>
<td>10</td>
<td>7.499</td>
<td>4.742</td>
<td>6.802</td>
<td>19.043</td>
</tr>
<tr>
<td>15</td>
<td>7.549</td>
<td>4.777</td>
<td>6.92</td>
<td>19.246</td>
</tr>
<tr>
<td>20</td>
<td>7.532</td>
<td>4.819</td>
<td>6.902</td>
<td>19.253</td>
</tr>
<tr>
<td>25</td>
<td>7.453</td>
<td>4.98</td>
<td>6.915</td>
<td>19.348</td>
</tr>
</tbody>
</table>
Tables 3 and 4, we make a comparison of $t_{SM}$, $t_{CC}$, and $t_{CS}$. When $N = 512$ and $k = 5$, $t_{SM}$, $t_{CC}$, and $t_{CS}$ is 7.44 s, 4.508 s, and 6.778 s, respectively. When $N = 512$ and $k = 25$, $t_{SM}$ is changed from 7.44 s to 7.453 s, $t_{CC}$ is changed from 4.508 s to 4.98 s, and $t_{CS}$ is changed from 6.778 s to 6.915 s. With the increase of $k$, $t_{SM}$, $t_{CC}$ and $t_{CS}$ are almost unchanged. Similarly, when $N = 1024$ and $k = 5$, $t_{SM}$, $t_{CC}$, and $t_{CS}$ are 40.918 s, 26.763 s, and 43.949 s, respectively. When $N = 1024$ and $k = 25$, $t_{SM}$, $t_{CC}$, and $t_{CS}$ are 41.72 s, 28.01 s, and 43.621 s, respectively. With the increase of $k$, $t_{SM}$, $t_{CC}$ and $t_{CS}$ do not obviously change. Thus, we can conclude that the increase of $k$ does not have a great impact on $t_{SM}$, $t_{CC}$, and $t_{CS}$.

Considering there are 1728 data records in this dataset, which means there are 1728 smart meters. Since the total time of smart meters executing PPKC is 7.453 s and 41.72 s, respectively, each smart meter only needs 4.31 ms and 24.14 ms to encrypted their data records when $N$ is varied from 512 to 1024, which is quite time-saving. As shown in Tables 3 and 4, each control center may spend more time than each smart meter since it needs to encrypt all the query data. It is acceptable for the control center because there is only one cloud server in PPKC algorithm and the control center need to do more computation to protect the query.

Next, we compare the efficiency of PPKC with those of Samanthula et al. [10] and Liu et al. [11] during executing the whole algorithm as shown in Figures 2 and 3. When $k$ is varied from 5 to 25 and $N = 512$, the total running time of PPKC is varied from 18.726 s to 19.348 s. Similarly, when $N = 1024$, the total running time of PPKC is varied from 111.63 s to 113.352 s. In [10, 11], when $N = 512$ and $k = 5$, their running time is 10.01 min and 0.59 min, respectively. Simultaneously, when $N = 1024$ and $k = 5$, their running time is 67.97 min and 3.091 min, respectively. Therefore, the proposed PPKC is much more efficient than the previous privacy-preserving KNN protocols.

### 5.2. Classification Accuracy
We test the classification accuracy of KNN and PPKC on some datasets from UCI dataset [26] in this subsection.

Assume that $m$, $n$, $t$, $q$ are separately the number of data records, the attributes, the training data records, and query data. The query data are randomly selected from the dataset. Thus, each query data has its own label, which can be judged whether the classification results of PPKC are true or false. Table 5 demonstrates some information of the datasets.

We totally use 4 different datasets. The first dataset is Car Evaluation dataset. As demonstrated in previous part, there are 1728 data records and 4 different labels, including unaccuracy, accuracy, good, and very good, while each data record has 6 attributes. Here, we set the first 1400 records as training data and the other 328 records as testing data.

The second dataset is Mammographic Mass dataset. There are a total of 830 data records with 5 attributes and 2 different labels including benign and malignant. In this dataset, the first 600 data records are training data and the other 230 data records are testing data.

The third dataset is Electrical Grid Stability dataset, which is widely used in smart grid. In this dataset, there are some attributes including power consumed, value for electricity producer, and so on. There are a total of 10000 data records with 11 attributes and 2 different labels including stable and unstable. In this dataset, the first 8000
data records are training data and the other 2000 data records are testing data.

The fourth dataset is Letter Recognition dataset. In this dataset, there are a total of 20000 data records which have 16 attributes. The labels are 26 capital letters in the English alphabet. The first 16000 records are training data and the other 4000 records are testing data.

For different datasets, we totally carry out 5 groups of experiments with different $k$. For each dataset, we set the value of $k$ as 1, 2, 3, 5, and 10.

As shown in Table 6, we give the accuracy of original KNN and PPKC with different $k$. In the first dataset, the accuracy of original KNN is varied from 89.975% to 65.853%, while the accuracy of PPKC is varied from 89.975% to 65.548%. In the other three datasets, it is also not much difference between the accuracy of original KNN and PPKC. No matter what $k$ is, the accuracy of KNN is almost the same as that of PPKC. Besides, the proposed PPKC can protect the data of smart meters and control centers, which can prevent cloud server from obtaining their private information. Moreover, the results of Electrical Grid Stability dataset shows that the proposed PPKC can be well applied to the smart grid. Thus, PPKC is secure and practical for smart grid.

### 6. Conclusion

In this paper, we propose a privacy-preserving KNN classification algorithm based on single cloud server for smart grid. The proposed PPKC algorithm can protect private data of smart meters, control centers, and the classification results based on Paillier cryptosystem. Compared with the previous protocols, PPKC only uses one cloud server, which is more practical in reality. The experiments results demonstrate that PPKC is much more efficient than the previous protocols and the classification accuracy of PPKC is almost the same as that of the original KNN, which means PPKC is efficient and feasible for smart grid. However, the proposed PPKC algorithm is only applicable to the KNN algorithm, and we will pay more attention to other privacy-preserving machine learning algorithms in the future.

### Data Availability

All data are included within the article.

### Conflicts of Interest

The authors declare that they have no conflicts of interest.

### Acknowledgments

The work described in this paper was supported by the Natural Science Foundation of Shanghai (20ZR1419700 and 22ZR1481000) and Henan Key Laboratory of Network Cryptography Technology (LNCT2021-A13).

### References


