Cloud based Privacy Preserving Data Mining with Decision Tree

Echo P. Zhang, Yi-Jun He and Lucas C. K. Hui
Department of Computer Science, The University of Hong Kong, Hong Kong, Hong Kong

Keywords: Data Mining, Privacy Preserving Data Mining, Cloud Computing, Privacy in Cloud Computing.

Abstract: Privacy Preserving Data Mining (PPDM) aims at performing data mining among multiple parties, and at the meantime, no single party suffers the threat of releasing private data to any others. Nowadays, cloud service becomes more and more popular. However, how to deal with privacy issues of cloud service is still developing. This paper is one of the first researches in cloud server based PPDM. We propose a novel protocol that the cloud server performs data mining in encrypted databases, and our solution can guarantee the privacy of each client. This scheme can protect client from malicious users. With aid of a hardware box, the scheme can also protect clients from untrusted cloud server. Another novel feature of this solution is that it works even when the database from different parties are overlapping.

1 INTRODUCTION

1.1 Motivation

The privacy preserving data mining (PPDM) problem has draw a lot of attentions in recent years (Verykios et al., 2004). Multiple participants intend to collaboratively mine the data from their combined database. Also, no single participant's private data will be released to any other party. In this paper, we focus on solving one PPDM problem with decision tree technique. Note that although many previous works (Goethals et al., 2004), (Kantarcioglu and Clifton, 2004), (Kantarcioglu and Kardes, 2009), (Kantarcioglu et al., 2009), (Jha et al., 2005), (Zhan et al., 2005) did the research on PPDM problem through various directions, they did not give a concrete definition of PPDM. Here, we start with defining the PPDM problem with decision tree as follows (for convenience, we call it *PPDM_DT_PC*):

Definition 1: *PPDM_DT_PC*.

- There are n parties ($n \geq 2$). Call them $C_1, \ldots, C_n$.
- C_i has a private two-dimensional database $DB_i = \{d_{i,j,k}\}$, where $d_{i,j,k}$ means the cell data at the position of row j, column k in DB_i.
- A subset of $\{C_1, \ldots, C_n\}$, denoted by C_{sub}, cooperate to construct a decision tree based on their corresponding databases.
- After the decision tree is built up, C_i only gets the result (i.e. the decision tree DT) without knowing any data of DB_j for any $j \neq i$.

There have been many researches working on PPDM problem with decision tree, and they gave out many feasible solutions to different distributions of databases, such as horizontally distributed database (Lindell and Pinkas, 2000) and vertically (Wang et al., 2004), (Fung et al., 2005), (Du and Zhan, 2002), (Vaidya and Clifton, 2005), (Fang et al., 2010) distributed database. Since all previous researches proposed solutions based on the private databases stored in participants' personal computers, we collectively call the above definition as PC-based PPDM problem (*PPDM_DT_PC*). Up to our best knowledge, all *PPDM_DT_PC* solutions suffer the same two shortness: 1) Low efficiency. Most of *PPDM_DT_PC* solutions requires a large amount of communication messages, in order to achieve the privacy protection of each participant; 2) Limitation to non-overlapping database. There is no existing PC-based PPDM solution able to deal with overlapping database.

Besides the above problems, all *PPDM_DT_PC* solutions cannot be applied to the modern Cloud computing environment. As cloud computing is becoming more and more popular nowadays, therefore, we want to design a system of PPDM utilizing cloud computing (We call it *PPDM_DT_Cloud*). There is an additional advantage of using cloud environment to solve PPDM. The cloud service not only can handle overlapping databases from different participants, but also provide an efficient solution due to communication messages among different parties can be reduced.

P. Zhang E., He Y. and C. K. Hui L..
Cloud based Privacy Preserving Data Mining with Decision Tree.
DOI: 10.5220/0003996200050014
In *Proceedings of the International Conference on Data Technologies and Applications* (DATA-2012), pages 5-14
ISBN: 978-989-8565-18-1

Planting this problem into cloud platform is not easy. We have to notice another significant aspect of cloud computing – privacy issue. Due to the infrastructure of the cloud service, the privacy of cloud computing has been becoming a hot topic attracting many researchers attention (Pearson, 2009), (Singh et al., 2010). Cloud service provides clients a convenient environment to utilize services supplied by cloud server instead of PC. Nowadays, cloud server has been adopted to more and more applications. It is not avoidable that clients will enter personal data when utilizing the service provided by cloud server. Therefore, it is very important to consider the privacy service when design the cloud server.

Here, we would like to give out a version of definition of PPDM on decision tree in cloud environment. We call it *PPDM_DT_Cloud*. This definition in cloud scenario involves an extra party: the cloud server - *CS*. *CS* is usually a group of machines. Databases are stored in encrypted format in the cloud server to protect the privacy of clients. The definition is as follows:

Definition 2: *PPDM_DT_Cloud*

- There is a cloud server - *CS* and n parties $C_1,\ldots,C_n$ where $(n \geq 2)$.
- C_i has a private two-dimensional database $DB_i = \{d_{i,j,k}\}$, where $d_{i,j,k}$ means the cell data on the row j and column k in DB_i. The database is stored in the cloud service provided by *CS* in the encrypted format.
- A subset of $\{C_1,\ldots,C_n\}$, denoted by C_{sub}, decide to construct a decision tree based on their corresponding databases.
- After the decision tree is built up, *CS* will inform each party in C_{sub} about the decision tree. Doing so, C_i does not know DB_j for any $j \neq i$.

However, the above solution cannot be achieved directly, because it is not so easy for *CS* to build up the decision tree when all DB_i are in different encrypted formats. More specifically, each DB_i is encrypted by the unique key of the owner C_i. This paper will propose a novel scheme to solve this problem. Even if the cloud server is untrusted (in cases of using a public cloud), we introduce an easy-to-deploy hardware to protect the system from the untrusted server. If the server is trusted (in cases of using a private cloud), a software implementation is sufficient to achieve the needed security and privacy.

1.2 Potential Applications

Cloud computing is a proper platform for multiple users to cooperate on some applications. For example, a bank performs a collaborated data mining using its customer database and the database of another bank, to judge whether it should approve a debt for some bank client according to his personal information and previous credit situation (like Table 1 & 2). In the meantime, the bank cannot see the database details of another bank.

Table 1: Practical example (Bank A).

Cus.	Age	Edu.	Salary	Loan	Overdue
C1	50	-	-	12k	No
C2	30	High Sch.	-	50k	-
C3	-	Univer-city	30k	-	-
C4	40	-	40k	100k	Yes
C5	-	-	10k	-	-

Table 2: Practical example (Bank B).

Cus.	Age	Edu.	Salaty	Loan	OD
C1	50	H.S.	20k	-	-
C2	-	-	20k	20k	Yes
C3	25	-	-	50k	No
C4	-	Uni.	-	-	-
C5	60	H.S.	-	10k	No

Another application is in the area of digital forensics. Law enforcing authorities may need to look into information on many different database to mine criminal behaviours, but at the same time should preserve privacy.

The cloud server platform provides a feasible and efficient way to solve these problems. However, the private information of bank clients and suspected criminals are usually confidential. As a result, we must propose a novel solution for cloud-based privacy preserving data mining, which works on encrypted confidential information. We will elaborate this novel scheme in the following sections.

1.3 Our Contributions

1. We propose the first solution for PPDM on decision tree utilizing the cloud server. An additional advantage of planting the PPDM into the cloud server platform is much more efficient than PC-based PPDM.
2. *PPDM_DT_Cloud* can also handle the overlapping databases case which has not been resolved before.
3. We modify the traditional ElGamal cryptographic algorithm in our scheme, which can protect par-

ticipants' privacy from the threat of any malicious parties.

4. Also, we design an easy-to-deploy hardware Black Box which is an auxiliary device to unify the data from various sources, to protect the threat of an untrusted cloud server.

The rest of the paper is organized as follows: In the next section, we introduce the related work in the field of privacy preserving data mining. In Section 3, we explain some preliminary knowledge which is required for understanding our scheme. Section 4 analyse the security requirements of *PPDM_DT_Cloud*. In Section 5, we depict our novel algorithm to solve the *PPDM_DT_Cloud* using Paillier cryptography and ElGamal algorithm. Sections 6 presents a brief security analysis. Section 7 concludes our paper.

2 RELATED WORK

Privacy preserving data mining (PPDM) can be traced back to Yao's millionaire problem (Yao, 1986). PPDM helps participators in collaborating their private databases to reach an accountable result without exploring their own secret data. PPDM can be classified by different criteria.

Since our work is about building decision tree, here we focus on previous research works in this topic.

(Yang et al., 2005) developed a privacy-preserving frequency mining algorithm based on the homomorphic cryptography property, which can be utilized in naive Bayes classifier, ID-3 trees and association rule mining. However, this algorithm was restricted by the efficiency, e.g. it only could deal with a small number of attributes.

(Lindell and Pinkas, 2000) discussed the ID-3 decision tree on the horizontally distributed database, while (Wang et al., 2004); (Du and Zhan, 2002); (Vaidya and Clifton, 2005) and (Fang et al., 2010) discussed algorithms for vertically distributed database. In particular, (Wang et al., 2004) and (Du and Zhan, 2002) only considered the situation that both parties know the class attributes. Jaideep et al. conquered this limitation in (Vaidya and Clifton, 2005) by building an ID-3 decision tree. (Du and Zhan, 2002) could not deal with the situation if one party lies about its input.

Since the rapid development of cloud computing, the privacy issue just begins to draw attentions of cloud service providers and users. It is unavoidable that applications involving the private data must be used in cloud service. As a result, designing a secure cloud environment which protects privacy becomes more significant. Detailed reasons of why privacy is so important for cloud computing can be found in (Pearson, 2009).

In 2010, Singh et al. gave out a cryptography based PPDM solution for cloud computing (Singh et al., 2010), which mainly addressed the clustering problem with k-NN classifier. They extended the Jaccard measure to test the equality of two encrypted items. In this protocol, it needed a semi-honest third party who can access the decrypted data and this assumption allowed the threat to the privacy of users. Up to our best knowledge there are no previous work in cloud computing environment that builds decision tree.

3 PRELIMINARY KNOWLEDGE

3.1 ID-3 Algorithm

Decision tree is one important traditional model for data mining. ID-3 (Quinlan, 1986) is one data mining algorithm for building the decision tree.

There are two important issues in such algorithms based on possibility statistic: one is the best split point and the other one is stopping criteria (Bhatnagar et al., 2010).

For the former one, the ID-3 algorithm chooses the attribute with the maximum *Gain* value to split. To calculate the *Gain*, firstly the information entropy ($E(S)$) of the dataset S is calculated as follows:

$E(S) = -\sum_{i=1}^{n} f_S(i)\log_2 f_S(i)$,

where n is the number of different values of attributes in S; $f_S(i)$ is the frequency of the value i in the dataset S.

After that, the *Gain* ($G(S,A)$) is calculated as follows:

$G(S,A) = E(S) - \sum_{i=1}^{m} f_S(A_i)E(S_{A_i})$,

where $G(S,A)$ denotes the gain of the dataset S after a split over the attribute A and A_i is ith possible value of A. m is the number of different values of the attribute A in S. $f_S(A_i)$ is the frequency of the A_i in S. S_{A_i} is a subset of S containing all items where the value of A is A_i.

For the latter one, there are three main criteria for stopping growing the tree: a) The tree has already reached the maximum depth; b) there has already been minimum number of data points for one branch; c) all of the points share the same label.

3.2 Homomorphic Encryption and Paillier Cryptography

In short, homomorphic encryption can satisfy the demand that one kind of algebraic calculation on cipher text will be reflected as another kind of algebraic calculation on plain text (Fontaine and Galand, 2007). This concept was firstly proposed by R.L. Rivest et al. in (Rivest et al., 1978), and in 2009 Craig Gentry (Gentry, 2009) published the first fully homomorphic encryption scheme using lattice-based cryptography.

The homomorphic encryption property is usually presented as:

$$E(a) \bigoplus E(b) = E(a \bigotimes b). \quad (1)$$

In our paper, we choose Paillier cryptography (Paillier, 1999) which is a probabilistic asymmetric algorithm for public key cryptography.

The biggest advantage of Paillier cryptography is that we can get different cipher text from the same plain text, with different random numbers used in the encryption process. This advantage can help our scheme to protect the client from eavesdropping.

4 REQUIREMENTS OF PPDM_DT_Cloud

Before we describe the details of *PPDM_DT_Cloud* scheme, we firstly need to investigate requirements for solving this problem.

In *PPDM_DT_Cloud*, we concern that only authorized user can have access to the data. Even if unauthorized user access to the data, he will get the wrong one.

Here, we try to design a scheme to protect active attacks from the cloud server, as well as a malicious client. Now we list the structural components in *PPDM_DT_Cloud*:

- **Central Cloud Server (CCS):** There is one CCS which acts as the interface between cloud service users and other cloud services.
- **Distributed Storage Server (DSS):** There are many DSS which provide the storage space.
- **Distributed Computation Server (DCS):** There are many DCS which carry out the computing task.
- **Cloud Service Users (Client):** the users who communicate with CCS to indirectly get services from different DSS and DCS.

Note that CCS, DCS and DSS together is the Cloud Server (CS) in the definition *PPDM_DT_Cloud*. Also, DSS and DCS are conceptual entities. In practice, they can be the same machine.

We have to design a system which can satisfy the following criteria:

1. **Client Security:** In private cloud, the cloud server can be trusted but the user may suffer the attack from other malicious users.
2. **Strong Client Security:** In a public cloud environment, there is no guarantee that the cloud server or other users of cloud service are honest. We have to assume the cloud server is untrusted. Therefore, all of data which is stored in the cloud server has to be encrypted, and only the data owner can decrypt them. Note that it is s stronger security requirement than the first criteria (Client Security).
3. **Service Availability:** The system has to make sure that the owner of the data can access / search / proceed any function on his own data at any time.
4. **Correctness:** The system also need to guarantee the correctness of the data which is stored in the cloud server.
5. **PPDM Feasibility:** This requirement involves two mainly aspects: 1) satisfy the general requirement on PPDM, which means that any participant will have no idea about the other parties' private data; 2) the data mining result is correct (i.e. gives out the same output as *PPDM_DT_PC*).

It is not easy to design a scheme to satisfy all above five criteria simultaneous. For the first and second requirements, the data is encrypted by its owner and the owner keeps the secret key. No one else can access to this secret key.

As for the fifth requirement, the data mining is implemented on the combined database from multiple clients, which means that databases from different owners are encrypted by separate secret keys. However, the necessary but not sufficient condition for data mining on the encrypted database is that the same message has to share the same transformed format in the database.

For example, in Table 1, Customer C1's age is 50. This cell is also in Bank-2's database. In the cloud server, these two cells are kept as $E_{Bank-1}(50)$ and $E_{Bank-2}(50)$ which are encrypted by Bank-1 and Bank-2's key separately. However, it is impossible for the cloud server to process data mining on encrypted data with two different keys. Therefore, before data mining, the cloud server has to transform databases

from different clients, into a unified format. We assume that a transformation function $f(\cdot)$ can achieve this purpose. After going through $f_{key}(\cdot)$ we always get $f_{key}(E_{Bank-1}(50)) = f_{key}(E_{Bank-2}(50))$. Otherwise, the system cannot satisfy the third requirement.

Note that the second requirement is in fact a stronger version of the first requirement. Actually, in our designed scheme, if all steps are implemented in software, requirements 1, 3, 4, 5 are satisfied but requirement 2 is not (i.e. there is an attack involving a malicious untrusted cloud server). In order for our scheme to satisfy requirement 2 as well, we need the help of a **Black Box** (BB), which is a tamper-resistant hardware token.

In this paper, for convenience, we describe our solution *PPDM_DT_Cloud* with the hardware Black Box. In applications (such as in a private cloud) that requirement 1 is needed instead of requirement 2, the solution can be trivially modified by putting all BB implementation in software and some simple key structure modifications. Details will be described in the beginning of Section 5.

Black Box: Here, we need a tamper-resistant hardware Black Box to help the system defending the eavesdropping threat. Inside the BB, there is prestored key pair (a private key and a public key). The private key can only be utilized by functions inside the BB. In other words, the private key will never be leaked out. Note that as all BB contain the same key structure. In our system, we need one BB is a Client (we call it a Client-side BB) and the same kind of BB is also used in a DSS (we call it a DSS-side BB). Therefore only one kind of BB is needed to be manufactured, and this is a significant advantage in production. We will elaborate the detailed usage of this Black Box in Section 5

Algorithm 1: *PPDM_DT_Cloud* with ElGamal.

Key Generation:

1: CCS generates an efficient description of a multiplicative cyclic group G of order q with generator g, which are published to each client of this cloud service.
2: C_i secretly chooses a random s_i from $\{0,\ldots,q-1\}$, which is C_i's private ElGamal key. While $h_i = g^{s_i}$ is C_i's semi-public ElGamal key $PubEG_{C_i}$, which is kept from knowing by CCS.
3: Meanwhile, there is one pair of Paillier key prestored in Client-side and DSS-side Black Box.

Progress:

Phase 1: C_i calls Algorithm 2 to encrypt his private database DB_i and then submits the encrypted database to CCS.
Phase 2: CCS assigns DB_i to different DSS and each DSS keeps one part of DB_i.
Phase 3.1: Whenever there is a subset of clients C_{sub} agreed on PPDM, they negotiate on a Group Key *GPkey* and send a PPDM request to CCS.
Phase 3.2: CCS forwards the PPDM request to each DSS. Each DSS, who keeps any part of database belonging to the member of C_{sub}, calls Algorithm 3 to unify the database.
Phase 3.3: According to Algorithm 4, DSS combines the unified databases from different sources into one, and sends it back to CCS.
Phase 4.1: CCS further combines input from different DSS according to Algorithm 4.
Phase 4.2: CCS asks different DCS to build up the Decision Tree DT, and gets the result $E_{GPkey}(DT)$ from DCS.
Phase 5.1: CCS calls Algorithm 5 and sends $E_{GPkey}(DT)$ back to each member of C_{sub}.
Phase 5.2: All members in C_{sub} collectively decrypt $E_{GPkey}(DT)$ to get the final result and check the correctness of DT.

5 CLOUD-BASED PPDM WITH UNTRUSTED CLOUD SERVER

We roughly divide the whole process into five phases and the overall system is depicted in Algorithm 1:

We consider the public and private cloud environment individually. If cloud server is untrusted (i.e. the public cloud case), there is a threat that the cloud server can get the plain text of data from clients. To protect clients' privacy from the untrusted cloud server, as stated in Section 4, we need help of an external hardware (the BB) to defend this attack.

On the other hand, if the cloud server is trusted (i.e. the private cloud case), as mentioned in Section 4, strong client security is not needed. We can have a simple scheme without using the hardware BB. In this case, BB can be substituted by software. Also as BB is not needed, the private key of BB will be replaced by a private key shared by CCS and all DSS.

For the sake of ease discussion, in this paper we only present the solution with BB for PPDM involving an untrusted cloud server. In our scheme, we choose a modified ElGamal cryptography model combining with Paillier cryptography as the encryption cryptography.

In our scheme, we use Black Box (BB) to implement three algorithms, namely Algorithm 2 in Phase 1, Algorithm 3 in Phase 3, and Algorithm 5 in Phase 4. Any outside party can *only* call BB to execute these

three algorithms.

5.1 Phase 1: Initialization

Before the client submits his private database to CCS, he needs to encrypt the data since there is a threat of unauthorized access. For simplicity, we use the notation in Section 1. $d_{i,j,k}$ is the cell data in C_i's database, at the position of row j, column k.

As mentioned in Section 4, the necessary but not sufficient condition for *PPDM_DT_Cloud* is that the same message has to be encrypted to the same cipher text. Therefore, we choose the mechanism of Group Key in our scheme. However, not any cryptographic algorithm can be used with Group Key. Here, we choose the modified ElGamal and Paillier cryptography. The encryption and decryption of the modified ElGamal algorithm are denoted by $EG_{key}(\cdot)$ and $DG_{key}(\cdot)$. While those of the Paillier algorithm are denoted by $P_{key}(\cdot)$ and $Q_{key}(c)$. Besides the pre-stored Paillier key pair, there is also a pre-stored hash function $H(\cdot)$ inside BB.

The core part of Phase 1, Algorithm 2, is shown below.

Algorithm 2: The first function in BB.

1: Input: $(d_{i,j,k}, PubEG_{C_i})$
2: Utilizing the pre-stored function $H(\cdot)$ to generate a random number: $r_{i,j,k} = H(d_{i,j,k})$ where $r_{i,j,k} \in \{0,\ldots,q-1\}$.
3: Encrypting this random number with public Paillier key as the first part of cipher text: $c_1 = P_{BB}(r_{i,j,k})$
4: Calculating the second part of cipher text with ElGamal cryptography: $c_2 = EG_{PubEG_{C_i}}(d_{i,j,k}) = d_{i,j,k} \cdot h_i^{r_{i,j,k}}$.
5: Output: $E(d_{i,j,k})$=(c_1, c_2)

There are two significant points needed to be emphasized:

1. The pre-defined function $H(\cdot)$ for generating the pseudo random number, must have the uniqueness property. This means that for any $m_1 = m_2$, we must have $H(m_1) = H(m_2)$. $H(\cdot)$ can be a collision resistant cryptographic hash function.

   The purpose of requiring the uniqueness of the pseudo random function is to make sure that CCS can successfully process data mining on the combined database from various clients.

2. The Paillier cryptography $P(\cdot)$ implemented by BB, has the property of a non-unique mapping, which means that for any $m_1 = m_2$, there can be $P_{key}(m_1) \neq P_{key}(m_2)$. This non-unique mapping can protect the private data from unauthorized access. Once the unauthorized party accesses the cipher-text of $E(d_{i,j,k}) = (P_{BB}(r_{i,j,k}), EG_{PubEG_{C_i}}(d(i,j,k)))$, he cannot deduce $d_{i,j,k}$ from $P_{BB}(r_{i,j,k})$ even if he also has the same cell in his database. The property of non-unique mapping can eliminate this threat.

5.2 Phase 2: Distributed Storage Mechanism

One service supplied by the CCS is availability, which means that the client can access to their data even though some DSS are down. In another word, the CCS has to keep the backup of clients' data in different DSS. CCS can save the whole database from one client into n ($n \geqslant 2$) separate DSS, or divide the database into several overlapping parts and assign them into various DSS. No matter which approach CCS chooses, any eavesdropper could not recognize the source according to the cipher-text which are sent back by DSS. After CCS gets the encrypted database from client, it will distribute the database to one or more DSS.

5.3 Phase 3: Unifying the Database

5.3.1 Phase 3.1: Agreement on PPDM

When a group of clients C_{sub} want to carry out a *PPDM_DT_Cloud* action, they will issue a PPDM command. Then each member C_i needs to send a command to CCS with $PubEG_{C_i}$. CCS will check the correctness of received commands, then broadcasts the identity of members in C_{sub} to DSS.

5.3.2 Phase 3.2: Unifying Individual Database Components

A DSS who stores some part of encrypted database belong to a member in C_{sub} will pass the database component to a DSS-side BB, and returns the output to CCS.

Before carrying out data mining, the preparation work is to unify various formats of encrypted data, with a Group Key negotiated by C_{sub}.

With public ElGamal keys of all M members in C_{sub}, DSS can construct one partial Group Keys for each group member. The partial Group Key for C_i is:

$$GPkey_i = \prod_{l=1 \wedge l \neq i}^{M} PubEG_{C_l}$$

$$= \prod_{l=1 \wedge l \neq i}^{M} g^{s_l} = g^{\sum_{l=1 \wedge l \neq i}^{M} s_l}. \quad (2)$$

In a DSS-side BB, Algorithm 3 is executed. Firstly, BB checks the correctness of cipher text of $d_{i,j,k}$. Then BB combines the public ElGamal key of C_i with partial Group Key $GPkey_i$ as the whole Group Key $GPkey$, to unify the data from different sources.

Algorithm 3: The second function in BB.

INPUT: $[P_{BB}(r_{i,j,k}), EG_{C_i}(d_{i,j,k})]$, $PubEG_{C_i}$, $GPkey_i$

1: Decrypt $r_{i,j,k}$ with BB's secret key.
2: Decrypt $d_{i,j,k} = \frac{EG_{C_i}(d_{i,j,k})}{PubEG_{C_i}^{r}}$.
3: If $r_{i,j,k} = H(d_{i,j,k})$ then go to next step. Otherwise, return an error message to DSS.
4: Encrypt the data $d_{i,j,k}$ with $GPkey$: $EG_{GPKey}(d_{i,j,k}) = EG_{C_i}(d_{i,j,k}) \cdot (GPkey_i)^r = d_{i,j,k} \cdot (g^{\sum_{l=1}^{M} s_l})^{r_{i,j,k}}$.

OUTPUT: $EG_{GPkey}(d_{i,j,k})$)

5.3.3 Phase 3.3: Combine Different Databases

DSS combines partial uniformed database of different C_{sub} members together. DSS also can deal with the special case of overlapped parts among different members' databases. As a matter of fact, this kind of situation is quite common. For example, one bank custom may choose more than one commercial banks to borrow money. Once one bank wants to decide whether it should continue to lend the money to this custom, the bank prefers to make the decision based on this custom's previous loan history of all other banks rather than this bank only.

However, as stated in Section 1, all previous researches did not give out any solution to this special situation, because the PC-based solution cannot securely solve this problem in an efficient way. The cloud computing technology provides a feasible platform to resolve this kind of problem. In our scheme, we utilize the homomorphic property of ElGamal algorithm to make the calculation on cipher text possible. The exact work which CCS needs to do is shown in Algorithm 4:

The rough idea of the combination work is as follows. DSS judges whether there are any records from different sources sharing the same primary key. If yes, DSS checks under the same attribute whether the value from different sources are more than one. Again, if yes, DSS carry out the calculation according to the pre-defined rules. If the rule asks for keeping the same value or multiplying different items, DSS can complete the computation on the cipher text directly.

Algorithm 4: Check flow.

Let PK be the primary key attribute and the set of value is $PK = \{PK_1, PK_2, PK_3, \ldots\}$
$|PK|$ is the size of PK
for $i = 1$ to $|PK|$ **do**
  if there are more than one records whose primary key $= PK_i$ **then**
    let A be the set of all attributes
    for $j = 1$ to $|A|$ **do**
      if the rule for A_j is keeping the same value of those records **then**
        keeping the same value
      else if the rule for A_j is multiplying different values of those records **then**
        multiplying those records
      end if
    end for
  end if
end for

After the combination, DSS only sends $EG_{GPkey}(d_{i,j,k})$ to CCS. The reason is to save the communication flow and for the purpose of security,

5.4 Phase 4: Cloud Decision Tree Building

In Phase 4.1, CCS will build the decision tree DT using the ID-3 algorithm. Before CCS carries out the decision tree building, it firstly combines the databases from different DSS. As for the overlapped parts, CCS also goes through Algorithm 4.

After that, Phase 4.2 is executed. With an encrypted database, CCS can easily follows the standard ID-3 algorithm to construct a decision tree DT from this encrypted database. In our design, CCS will not run the ID-3 algorithm on its own. Instead, CCS takes advantage of the cloud computing environment, to coordinate different DCS to run the ID-3 algorithm in a distributed manner. Doing so, the running time of ID-3 will be reduced.

For a large size database, the workload for statistic computation in ID-3 is heavy for a single CCS and thus is a suitable computational task to be distributed to different DCS. Many DCS can share workload for statistic computation in parallel. Meanwhile, CCS plays the role as a controller, which means that CCS assigns the computational work to DCS.

As mentioned in Section 3, the major computa-

tional part of ID-3 is the statistic computation of attribute frequency and calculating the *Gain* value.

Here we describe how the calculation of the frequency and the *Gain* value can be distributed to different DCS.

CCS firstly groups the database entries according to the value of classified attribute. After that CCS sends the grouped encrypted data of one column to one DCS. Then each DCS calculates the *Gain* value corresponding to that column, and sends this value to CCS. After collecting all *Gain* values corresponding to different columns from different DCS, CCS will select the maximum one as the current node. CCS and DCS repeat the above steps until reaching the stopping point of the ID-3 algorithm.

Let us take the Table 1 & 2 as example. After grouping the database according to the value of classified attribute *Overdue*, the order of records should be $C2, C4$ in the *Yes* Group and $C1, C3, C5$ in the *No* Group. Then taking the attribute of *Age* for instants again, CCS sends $E(30), E(40)$ in the *Yes* Group and $E(50), E(25), E(60)$ in the *No* Group to DCS. So do other attributes. Each DCS returns the *Gain* value back to CCS, and CCS picks up the attribute with the maximum *Gain* as the root node. In this example, this attribute is $Education$. CCS sends this column, $E(HighSchool(HS)), E(University(U))$ in the *Yes* Group and $E(HS), E(U), E(HS)$ in the *No* Group, to other DCS who has the attribute with non-maximum *Gain* value. Similarly, DCS calculate *Gain* value and CCS choose the maximum one as the second-level node. CCS and DCS repeat the above steps until reaching the stopping point.

5.5 Phase 5: Output

5.5.1 Phase 5.1

After CCS and DCS finish the collaboration work on decision tree building, CCS has a whole encrypted decision tree. The last step is to send back the result to each participant C_i. CCS calls Algorithm 5 to get $g^{r_{i,j,k}}$, and send it back with $EG_{GPkey}(d_{i,j,k})$ to each C_i. CCS uses Algorithm 5, the third function in BB, to decrypt $r_{i,j,k}$.

Algorithm 5: The third function inside the Black Box.

1: INPUT: $P_{BB}(r_{i,j,k})$
2: Decrypt $r_{i,j,k} = Q_{BB}(P_{BB}(r_{i,j,k}))$
3: OUTPUT: $g^{r_{i,j,k}}$

Each participant C_i calculates $(g^{r_{i,j,k}})^{s_i}$ and sends it to each other member in C_{sub}. The plain text of decision tree (all are $d_{i,j,k}$ values) can be decrypted: $d_{i,j,k} = DG_{GPkey}(EG_{GPkey}(d_{i,j,k})) = \frac{d_{i,j,k} \cdot g^{r_{i,j,k} \Sigma_{l=1}^{M} s_l}}{\Pi_{l=1}^{M} (g^{r_{i,j,k}})^{s_l}}$.

Then each C_i gets the whole decision tree which is built from the combined database.

5.5.2 Phase 5.2

C_{sub} also need to check the correctness of DT and we use the method stated in (Du and Zhan, 2002) for this checking process. For simplicity, here we give a simple walk-through of the checking as follows:

C_{sub} chooses one entry with the same primary key value in each C_i's database, $d_{i,j} = \{d_{i,j,k} | k = 1, 2, \ldots\}$. $d_{i,j}$ is one single row in C_i's database. Each C_i traverses through the DT based on the value of this entry to create a vector $V_i = v_1, \ldots, v_p$ where p is the number of leaf nodes in DT. If a node is split using C_is attribute, C_i traverses to all children of the node. If C_i reaches a leaf node q, he changes the qth entry of V_i to 1. At the end, vector V_i records all the leaf nodes that C_i can reach.

Other members of C_{sub} create the vector in the same way. It should be noted that $V = V_1 \wedge \ldots \wedge V_M$ has one and only one non-zero entry (here $\wedge$ is the logical-and operation) because there is one and only one leaf node that all C_{sub} can reach. Therefore, $V = V_1 \wedge \ldots \wedge V_M$ should also have one and only one non-zero entry. Otherwise, C_{sub} verify that the DT is not correct.

6 DISCUSSION OF SECURITY PROPERTY

The encryption scheme used in *PPDM_DT_Cloud* can ensure that confidentiality, integrity, and authenticity of all the records in the database. For simplicity, here we only show how our scheme can defend some interesting attacks. In particular, Attacks 2 and 3 are related to the Strong Client Security property.

6.1 Attack 1

Throughout the whole scheme, note that r has the unique relationship with m. If one malicious party C_l gets the $r_{i,j1,k1}$ belong to C_i, meanwhile, in C_l's database, he also contains the same value of $r_{l,j2,k2} = r_{i,j1,k1}$, then there is a very high possibility that $m_{l,j2,k2}$ is equal to $m_{i,j1,k1}$. This is an attack that we want to avoid. We choose Paillier cryptographic algorithm, which has the property that encrypting the same message will give a different cipher text, to encrypt r to avoid this attack.

6.2 Attack 2

In Algorithm 3, there are two main points which can defend the attack from untrusted cloud server. One is using the partial Group Key and the original cipher text to unify the data, the other one is checking the correctness of cipher text and random number. For the first point, if BB directly uses the Group Key as the input parameter, there is a threat that DSS may replace *GPkey* with its own key $PubEG_{DSS} = g^{s_{DSS}}$. The details of this attack is listed as follows:

1. DSS sends: $[P_{BB}(r_{i,j,k}), EG_{C_i}(d_{i,j,k})]$, $PubEG_{C_i}$, $PubEG_{DSS}$ to BB.
2. DSS calls Algorithm 3:
   - Decrypt $r_{i,j,k}$ with BB's secret key.
   - Decrypt $d_{i,j,k} = \frac{EG_{C_i}(d_{i,j,k})}{(C_i)^{r_{i,j,k}}}$.
   - Encrypt the data with the Group Key: $EG_{DSS}(d_{i,j,k}) = d_{i,j,k} \cdot (g^{s_{DSS}})^{r_{i,j,k}}$.
3. DSS gets $EG_{DSS}(d_{i,j,k})$ from BB.
4. DSS gets $g^{r_{i,j,k}}$ with the Algorithm 5 from BB.
5. Now, DSS can decrypt $d_{i,j,k}$ from $EG_{DSS}(d_{i,j,k}))$ with his private key s_{DSS}.

To avoid this attack from untrusted DSS, the Algorithm 3 could not take *GPkey* as the parameter directly. Since DSS wants to get $d_{i,j,k}$ which means that it cannot change $EG_{C_i}(d_{i,j,k}$, otherwise, it will get the wrongly decrypted $d_{i,j,k}$. $EG_{C_i}(d_{i,j,k})$ is calculated from g^{s_i}, meanwhile, g^{s_i} is one of component of *GPkey*. Therefore, we utilize this point to unify the data, $EG_{GPKey}(d_{i,j,k}) = EG_{C_i}(d_{i,j,k}) \cdot (GPkey_i)^r$. By doing this, DSS cannot attack the privacy of data through sending a wrong parameter to BB.

6.3 Attack 3

As mention in the above, the other one significant point of Algorithm 3 is checking the correctness of $EG_{C_i}(d_{i,j,k})$ and $r_{i,j,k}$. Without the checking, DSS can execute the following attack:

1. DSS sends $[P_{BB}(r_{i,j,k}), \frac{1}{EG_{C_i}(d_{i,j,k})}]$, $PubEG_{C_i}$.
2. DSS calls Algorithm 3:
   - Decrypt $r_{i,j,k}$ with BB's secret key.
   - Decrypt $r_{i,j,k}$ with BB's secret key to get $r_{i,j,k}$.
   - Encrypt the data with the Group Key: $EG_{PubEG_{C_i}}(\frac{1}{d_{i,j,k}}) = \frac{(g^{s_i})^{r_{i,j,k}}}{d_{i,j,k} \cdot (PubEG_{C_i})^{r_{i,j,k}}} = \frac{(g^{s_i})^{r_{i,j,k}}}{d_{i,j,k} \cdot (g^{s_i})^{r_{i,j,k}}}$.
3. DSS gets the output of $\frac{1}{d_{i,j,k}}$ from BB.

Hence, DSS gets the information of $\frac{1}{d_{i,j,k}}$. This is also an attack which we want to avoid. Therefore, before unifying $d_{i,j,k}$, BB has to check the correctness of parameters by comparing the value of $Q_{BB}(P_{BB}(r_{i,j,k}))$ with the hash value of $\frac{EG_{C_i}(d_{i,j,k})}{(PubEG_{C_i})^{r_{i,j,k}}}$. By doing this, we can prevent DSS to modify the cipher text or random number.

7 CONCLUSIONS

In this paper, we designed *PPDM_DT_Cloud*, a Cloud based PPDM solution, which can defend against the malicious participant. With the aid of a Black Box, our scheme even can protect clients' privacy from the untrusted cloud server as well.

Our scheme allows any subset of clients to carry out PPDM action. We choose the mechanism of Group Key to unify the cipher text encrypted by various clients separately, and through the whole process, each client share the same rights that no one has privilege to other clients. To make the Group Key feasible, we utilize the modified ElGamal and Paillier cryptographic algorithm as our encryption algorithms. ElGamal is used to keep the uniqueness of cipher text which is the necessary condition for PPDM, while Paillier is used to defend the attack from malicious participant as well as the untrusted server. This solution works in a private cloud. In the public cloud environment, we need the help of the hardware Black Box (BB) to protect the user from untrusted cloud. BB keeps never leaks out the Paillier key pair. Besides this, we design three functions which are implemented inside BB. The first two carry out the encryption action. As for the third one, BB does not give out the decrypted data directly, it gives $g^{r_{i,j,k}}$ instead of $r_{i,j,k}$. This design can avoid the malicious participant or untrusted cloud server unauthorised decrypting the data.

This *PPDM_DT_Cloud* solution is also the first PPDM solution that can take care of overlapping databases. So this solution is powerful, efficient, and easy to deploy. We believe that it can be applied to many practical situations.

REFERENCES

Bhatnagar, V., Zaman, E., Rajpal, Y., and Bhardwaj, M. (2010). Vistree: Generic decision tree inducer and visualizer. In *DATABASES IN NETWORKED INFORMATION SYSTEMS*. Springer-Verlag.

Du, W. and Zhan, Z. (2002). Building decision tree classifier on private data. In *Proceedings of the IEEE ICDM Workshop on Privacy*.

Fang, W., Yang, B., and Song, D. (2010). Preserving private knowledge in decision tree learning. In *Journal of Computers*. ACADEMY PUBLISHER.

Fontaine, C. and Galand, F. (2007). A survey of homomorphic encryption for nonspecialists. In *EURASIP Journal on Information Security*. Hindawi Publishing Corp. New York, NY, United States.

Fung, B. C. M., Wang, K., and Yu, P. S. (2005). Top-down specialization for information and privacy preservation. In *ICDE 2005*.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *STOC '09 Proceedings of the 41st annual ACM symposium on Theory of computing*. ACM New York, NY, USA.

Goethals, B., Laur, S., Lipmaa, H., and Mielikainen, T. (2004). On private scalar product computation for privacy-preserving data mining. In *Information Security and Cryptology ICISC 2004*, volume 3506/2005. Springer-Verlag.

Jha, S., Kruger, L., and McDaniel, P. (2005). Privacy preserving clustering. In *COMPUTER SECURITY ESORICS 2005*. Springer-Verlag.

Kantarcioglu, M. and Clifton, C. (2004). Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *Knowledge and Data Engineering*.

Kantarcioglu, M. and Kardes, O. (2009). *Privacy-preserving data mining in the malicious model*, volume 2. Springer-Verlag.

Kantarcioglu, M., Nix, R., and Vaidya, J. (2009). An efficient approximate protocol for privacy-preserving association rule mining. In *ADVANCES IN KNOWLEDGE DISCOVERY AND DATA MINING*.

Lindell, Y. and Pinkas, B. (2000). Privacy preserving data mining. In *ADVANCES IN CRYPTOLOGY CRYPTO 2000*. Springer-Verlag.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Prof. of the EUROCRYPT'99*. Springer-Verlag.

Pearson, S. (2009). Taking account of privacy when designing cloud computing services. In *CLOUD '09. ICSE Workshop*.

Quinlan, J. R. (1986). *MACHINE LEARNING*. Springer-Verlag.

Rivest, R. L. L., Adleman, M., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. In *In Foundations of Secure Computation*. ACADEMY PUBLISHER.

Singh, M. D., Krishna, P. R., and Saxena, A. (2010). A cryptography based privacy preserving solution to mine cloud data. In *COMPUTE '10 Proceedings of the Third Annual ACM Bangalore Conference*. ACM New York, NY, USA.

Vaidya, J. and Clifton, C. (2005). Privacy-preserving decision trees over vertically partitioned data. In *DATA AND APPLICATIONS SECURITY XIX*. Springer-Verlag.

Verykios, V. S., Bertino, E., Fovino, I. N., Provenza, L. P., Saygin, Y., and Theodoridis, Y. (2004). State-of-the-art in privacy preserving data mining. In *ACM SIGMOD Record*, volume Vol. 33, No. 1. ACM New York, NY, USA.

Wang, K., Yu, P. S., and Chakraborty, S. (2004). Bottom-up generalization: a data mining solution to privacy protection. In *ICDM'04*.

Yang, Z., Zhong, S., and Wright, R. N. (2005). Privacy-preserving classification of customer data without loss of accuracy. In *Proceedings of the 5th SIAM International Conference on Data Mining*.

Yao, A. (1986). How to generate and exchange secrets. In *Proceedings of the IEEE 27th Annual Symposium on Foundations of Computer Science*.

Zhan, J., Matwin, S., and Chang, L. (2005). Privacy-preserving collaborative association rule mining. In *19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security University of Connecticut*. Springer-Verlag.