Journal of Zhejiang University SCIENCE A
ISSN 1673-565X (Print); ISSN 1862-1775 (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

Distributed anonymous data perturbation method for privacy-preserving data mining*

Feng LI†, Jin MA, Jian-hua LI

(*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200030, China*)

†E-mail: atlas.lee@gmail.com

Received Apr. 8, 2008; Revision accepted Aug. 10, 2008; Crosschecked Apr. 29, 2009

Abstract: Privacy is a critical requirement in distributed data mining. Cryptography-based secure multiparty computation is a main approach for privacy preserving. However, it shows poor performance in large scale distributed systems. Meanwhile, data perturbation techniques are comparatively efficient but are mainly used in centralized privacy-preserving data mining (PPDM). In this paper, we propose a light-weight anonymous data perturbation method for efficient privacy preserving in distributed data mining. We first define the privacy constraints for data perturbation based PPDM in a semi-honest distributed environment. Two protocols are proposed to address these constraints and protect data statistics and the randomization process against collusion attacks: the adaptive privacy-preserving summary protocol and the anonymous exchange protocol. Finally, a distributed data perturbation framework based on these protocols is proposed to realize distributed PPDM. Experiment results show that our approach achieves a high security level and is very efficient in a large scale distributed environment.

Key words: Privacy-preserving data mining (PPDM), Distributed data mining, Data perturbation
doi:10.1631/jzus.A0820320 **Document code:** A **CLC number:** TP391.7

INTRODUCTION

With the rapid expansion of large-scale information systems, privacy is becoming more and more important in data analysis and computation. Specifically, privacy is a critical requirement when data mining is applied to financial, medical or government organizations (Cranor *et al.*, 2002; Ashley *et al.*, 2003; CSA, 2004; Evfimievski *et al*., 2004). As a result, privacy-preserving data mining (PPDM) (Agrawal and Srikant, 2000), which intends to protect the sensitive information in the process of data mining, has received much attention in recent years. Two main privacy-preserving approaches, namely data perturbation and secure multiparty computation (SMC), are proposed in the literature.

Data perturbation methods, inherited from secure database techniques, have been used to solve privacy-preserving problems in centralized environments (Agrawal and Srikant, 2000; Rizvi and Haritsa, 2002; Evfimievski *et al*., 2004). Agrawal and Srikant (2000) proposed two types of perturbation techniques, namely value-class membership and value distortion. Value-class membership is used to partition one attribute's values into disjoint classes, whilst value distortion, often used for association rule mining (Agrawal and Aggarwal, 2001; Chawla *et al*., 2005), adds random data with certain distribution to protect privacy, leaving statistical information to miners. Furthermore, more perturbation methods have been proposed in recent years. Data swapping (Fienberg and McIntyre, 2004) aims to protect the sensitive data in certain attributes by swapping a subset of attributes among the records. *k*-anonymity (Sweeney, 2002) focuses on substituting data of sensitive record following the rule that such a record is indistinct with at

* Project supported by the National Natural Science Foundation of China (Nos. 60772098 and 60672068), and the New Century Excellent Talents in University of China (No. NCET-06-0393)

least $k-1$ other records in the released set. In general, data perturbation methods are mainly used in centralized data mining.

SMC is used to solve the privacy-preserving problem in distributed data mining (DDM). Secure two-party computation was first investigated by Yao (1986), and later extended to multi-party computation in (Goldreich *et al*., 1987; Chaum *et al*., 1988). The basic rationale behind the SMC-based methods lies in: a designated cryptographic function is presented as a combinatorial circuit; involved parties run a short protocol with the function to exchange and compute encrypted data for data mining jobs. For most SMC-based methods, cost of communication is higher than $O(n^6)$ and the number of computation rounds is higher than $O(n)$ (Chaum *et al*., 1988; Beaver, 1991; Cramer *et al*., 2001) because of the cryptographic operations, where n refers to the number of nodes. Therefore, SMC is less practical in large-scale distributed applications due to its complexity.

In this paper, we propose a distributed anonymous perturbation method to realize efficient PPDM in a distributed environment. Key privacy constraints of consistency, privacy and integrity, and robustness in the distributed process are defined. With homographic techniques, a light-weight protocol is proposed to provide trustworthy anonymous perturbation in a distributed environment. Experiment and analysis show that the computation and communication cost of our method is low compared with SMC related methods, while preserving the privacy of original data. On the other hand, our work demonstrates that the random perturbation method works in both centralized and distributed environments.

The rest of the paper is organized as follows. In Section 2 we discuss the basic elements in distributed data perturbation. Privacy constraints are formally defined. Two privacy-preserving protocols are proposed to support distributed data perturbation. In Section 3 we propose the framework of distributed anonymous data perturbation (DADP). In Section 4 we analyze the security and privacy issues of DADP methods under collusion and malicious attacks with experiments, together with performance evaluation. Section 5 concludes the paper.

DISTRIBUTED DATA PERTURBATION ELEMENTS

In this section, we define the privacy constraints of distributed data perturbation, and propose two protocols to address these constraints. Since our method is based on traditional data perturbation techniques, we first briefly introduce the basic idea of the random value perturbation.

Random value perturbation

The data perturbation method covers the original dataset with a random data series following certain distribution (Agrawal and Srikant, 2000). In this study we focus on normalized distribution with zero mean and standard deviation σ.

Consider n original data $X_1, X_2, \ldots, X_n$, where X_i (i=1, 2, …, n) are variables following the same independent and identical distribution (i.i.d.). The distribution function of X_i is denoted as F_X, n random variables $Y_1, Y_2, \ldots, Y_n$ are generated to hide the real values of X_i by perturbation. Similarly, Y_i are i.i.d. variables. The distribution function of Y_i, F_Y, is a normalized distribution with zero mean and standard deviation σ.

Perturbed data is composed as

$$\omega_1, \omega_2, \ldots, \omega_n, \text{ where } \omega_i=X_i+Y_i,\ i=1, 2, \ldots, n. \quad (1)$$

The problem turns out to consist of reconstructing the distribution function F_X from perturbed dataset $\omega_1, \omega_2, \ldots, \omega_n$ and the known distribution function F_Y of random data Y_i. Bayes' rule is adopted to resolve the problem. The posterior cumulative distribution function of X_1 is written as

$$F_{X_1}'(a)=\frac{\int_{-\infty}^{a} f_Y(\omega_1-z)\,f_X(z)\mathrm{d}z}{\int_{-\infty}^{+\infty} f_Y(\omega_1-z)\,f_X(z)\mathrm{d}z}, \quad (2)$$

where $f_Y()$ is the probability density function of Y_i and $f_X()$ is the probability density function of X_i.

The average of the cumulative distribution function of X_i is

$$F_X'(a)=\frac{1}{n}\sum_{i=1}^{n}\frac{\int_{-\infty}^{a} f_Y(\omega_i-z)\,f_X(z)\mathrm{d}z}{\int_{-\infty}^{+\infty} f_Y(\omega_i-z)\,f_X(z)\mathrm{d}z}. \quad (3)$$

Differentiate Eq.(3) to obtain the density function:

$$f_X{}'(a)=\frac{1}{n}\sum_{i=1}^{n}\frac{f_Y(\omega_i-a)f_X(a)}{\int_{-\infty}^{+\infty}f_Y(\omega_i-z)f_X(z)\mathrm{d}z}. \quad (4)$$

$f_X^0()$ is set to a uniform distribution in initialization because of the unknown distribution of $f_X()$; $f_X^1()$ is calculated using Eq.(4). Following the rule, the iteration process continues until the difference between $f_X^{j+1}()$ and $f_X^j()$ is less than a preset threshold. The function approximation is close to the original data distribution F_X when the sample dataset is sufficient.

Privacy constraints in distributed data perturbation

Data perturbation methods are proven to be efficient in centralized environments (Agrawal and Srikant, 2000; Agrawal and Aggarwal, 2001), and are applied in many fields (Liew *et al*., 1985; Bertino *et al*., 2005; Zhang *et al*., 2005). However, these PPDM technologies cannot be directly applied in the distributed environments because more privacy issues are concerned, as shown in Fig.1. It is generally the responsibility of the data holder to protect privacy. In centralized PPDM, the data center (DC) holds the original dataset. DC deals with the privacy problems to make sure that the output data contain less private information. While in the distributed environment, each distributed participant (DP) owns part of the original dataset. DP must take efforts to keep its source data private from both other DPs and DC. Hence the data released by DPs need to be preprocessed before DC retrieves them. The function of DC remains to be data collection. In a word, the data holder must process the data appropriately when they are going beyond the privacy range (gray area in Fig.1).

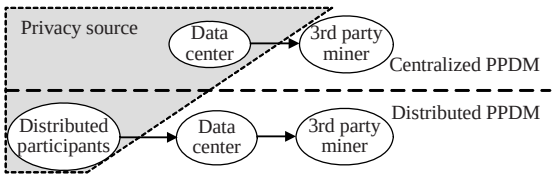

Fig.1 Privacy issues in centralized and distributed environment

Let a 4-tuple set {*DP*, *D*, *MCP*, *DC*} in *n*-participant environment, where DP_i is the distributed participant that holds partial source data, $DP_i \in DP$, i=1, 2, …, n; D_i is the dataset DP_i holds, $D_i \subset D$; MCP_i is the collusion or malicious participants, $MCP_i \in MCP$, i=1, 2, …, n, $MCP \in DP$; *DC* is the center collecting data from all DPs.

Two key privacy issues are concerned:

(1) $privacy(DP_i, DC)=D_i$. Function $privacy(A, B)=T$ represents that the content of target resource T that A holds is not allowed to be exposed to B. Therefore, D_i, the dataset DP_i holds, must not be exposed to *DC*.

(2) $privacy(DP_i, DP_j)=D_i$ (i, j=1, 2, …, n; $i \neq j$), DP_i should not expose the dataset D_i to other DPs including MCPs.

Based on the above discussion, we define the constraints for distributed data perturbation as follows:

(1) Consistency

It is necessary to generate the random dataset with a uniform distribution for perturbation. However, distributed nodes cannot exchange information about random data due to the privacy issue (2). How to distribute a uniform distribution of a random dataset with proper intensity to all DPs is a key issue.

(2) Privacy and integrity

Measures must be taken to make sure that the perturbation process is correctly and privately applied in distributed nodes, so that DC or the third party cannot have the exact data information about each node due to the privacy issue (1).

(3) Robustness

Collusion and malicious attacks must be considered in the distributed environment. Privacy and security risks occur when DP exchanges or shares information with colluding or malicious participants unconsciously. Detailed threats are discussed in Section 4.

Distributed privacy-preserving data perturbation protocols

1. Adaptive privacy-preserving summary protocol

The adaptive privacy-preserving summary protocol in this subsection is proposed to address constraint (1): consistency. Two entities can generate random data: DC and DPs. Generating a random

dataset by DPs would lead to unpredictable distribution. One solution is to generate the random dataset by DC, and deliver it to DPs separately. It is beyond doubt that DC can generate a random dataset with uniform distribution. However, how to generate the random data with proper intensity according to the original data remains a problem. It is clear that a too high intensity of random data will lead to inaccuracy of reconstruction (as described in the subsection 'Random value pertubation'), and DC has no privilege to summarize the intensity of original data due to the privacy issue (2). Therefore, the key issue is how to achieve the intensity of original data from DPs without violating privacy rules. Generally, variance is used for quantifying the intensity of the dataset.

In this subsection, we propose an adaptive privacy-preserving summary protocol to obtain the variance of original data from DPs privately. Variance is defined as

$$\sigma_{\text{origin}}^2 = \frac{1}{n}\left(\left(\sum_{i=1}^{n} X_i\right)^2 - \sum_{i=1}^{n} X_i^2\right). \tag{5}$$

Therefore, the task of achieving variance is equal to obtaining the sum of X_i and the sum of X_i squares. Some secure sum solutions have been proposed in recent years. Ashrafi *et al*.(2003) proposed an obfuscation/de-obfuscation procedure protocol (ODP) for distributed privacy-preserving association rule mining based on apriori algorithm:

Protocol 1 ODP

Initialization: DP_i holds datum X_i (*DPs* are randomly numbered).

Start: From DP_1 to DP_{n-1} do

DP_i generates R_i, X_i and R_i are in the same number field.

DP_i submits X_i+R_i to DP_{i+1}.

DP_i has the value $\sum_{j=1}^{n-1}(X_j + R_j)$, and submits $B_n = \sum_{j=1}^{n}(X_j + R_j)$ to DP_1.

From DP_1 to DP_{n-1} do

DP_i submits $B_i' = B_i - R_i = \sum_{j=1}^{n} X_j + \sum_{j=i+1}^{n} R_j$ to DP_{i+1}.

DP_n obtains $\sum_{j=1}^{n} X_j + R_n$ and retrieves $\sum_{j=1}^{n} X_j$ by removing R_n.

Fukasawa *et al*.(2004) proposed another secure sum protocol (SSP) in distributed privacy-preserving association rule mining:

Protocol 2 SSP

Initialization: DP_i holds datum X_i (*DPs* are randomly numbered).

Start: From DP_1 to DP_{n-1} do

DP_i generates R_i, X_i and R_i are in the same number field.

DP_i submits X_i+R_i to DP_{i+1}.

DP_i has the value $\sum_{j=1}^{n-1}(X_j + R_j)$, and submits $B_n = \sum_{i=1}^{n}(X_j + R_j)$ to DP_1.

(Re-number *DPs* randomly)

From DP_1 to DP_{n-1} do

DP_i submits $B_i' = B_i - R_i = \sum_{j=1}^{n} X_j + \sum_{j=i+1}^{n} R_j$ to DP_{i+1}.

DP_n obtains $\sum_{j=1}^{n} X_j + R_n$ and retrieves $\sum_{i=1}^{n} X_j$ by removing R_n.

Both solutions assume that the collusion density is low. When the density of collusion grows, privacy of the data is threatened. For instance, when DP_{i-1} and DP_{i+1} are colluded in ODP, the values of r_i and x_i+r_i can be figured out, and therefore datum x_i that DP_i holds is exposed. To overcome the weakness of these two protocols, we propose a new distributed secure sum protocol (DSSP):

Protocol 3 DSSP

Initialization: DP_i holds datum X_i, and shares parameter m, $1<m\leq n$.

Start: From DP_1 to DP_n do

DP_i divides X_i into m shares: $(X_{i,1}, X_{i,2}, \ldots, X_{i,m} \mid X_i = \sum_{j=1}^{m} X_{i,j})$.

DP_i keeps $X_{i,1}$ and sends $\{X_{i,2}, X_{i,3}, \ldots, X_{i,m}\}$ to $m-1$ *DPs* randomly.

Each DP_i obtains $m-1$ shares from different *DPs*.

From DP_1 to DP_n do

DP_i sums up all m shares and submits the result to *DC*.

DC obtains $\sum_{i=1}^{n}\sum_{j=1}^{m} X_{i,j} = \sum_{i=1}^{n} X_i$.

$\sum_{i=1}^{n} X_i^2$ is retrieved through the same method. The variance of the original dataset is obtained through Eq.(5) without privacy leakage. Privacy and security features of these three protocols are analyzed in detail in Section 4.

2. Anonymous exchange protocol

As we can see from the above discussion, the main process of distributed data perturbation is that:

(1) DC generates random data and delivers to DPs; (2) DPs add random data to the original data they hold, and submit the result back to DC; (3) DC reconstructs the distribution finally.

Now we look into a new exchange protocol to address the second privacy constraint: privacy and integrity:

Define tracing status: Suppose *A* holds datum *T* and submits it to *B*, it is said that *A* has the relation map as *T*-*B* and *A* traces *B* by *T*.

Tracing status is considered as a privacy attack in distributed data perturbation. For instance, *DC* delivers r_i to DP_i. As a result, *DC* traces DP_i by r_i. If DP_i submits x_i+r_i to *DC*, *DC* can obtain the exact value of x_i, which violates constraint (2). In this subsection, we present a *k*-anonymous exchange protocol to reduce tracing status.

(1) 1-turn anonymous exchange

Assuming that DP_s holds object OBJ_s initially, DP_s randomly chooses another target DP_t, sends the OBJ_s to DP_t. DP_t holds two objects as a result. According to the principle that each *DP* holds one object, DP_t finds another *DP* and sends its original object OBJ_t. After that, DP_t will deny the invitation if a new exchange is requested from other *DP*. DP_s will finally obtain an object from a *DP*. Each *DP* holds a different object after 1-turn exchange.

(2) *k*-turn anonymous exchange

For *k*-turn exchange, each *DP* has to find *k* different participants to exchange the objects, $k<n$. In other words, each *DP* repeats 1-turn anonymous exchange protocol for *k* times. The exchange process of each *DP* is defined as being asynchronous, one *DP* continues to the next turn without waiting for all participants to finish a complete round. After *k*-turn anonymous exchange process, DP_s holds OBJ_u, where OBJ_u has no direct correlation to OBJ_s. Tracing status is eliminated after *k*-turn anonymous exchange. However, when collusion exists, tracing status is related to value *k*. Further discussion can be found in Section 4.

DISTRIBUTED ANONYMOUS DATA PERTURBATION FRAMEWORK

Based on the elements in Section 2, this section presents a framework for distributed anonymous data perturbation that addresses all the previously mentioned three privacy constraints. We name the framework DADP. Three types of units are included in the framework (Fig.2): *DC*, to manage the dataset and the process; *DP* (DP_1, DP_2, …, DP_n), to submit their data privately, supposing *n* nodes in the environment; third computation party (TCP), to provide trustworthy and private calculation.

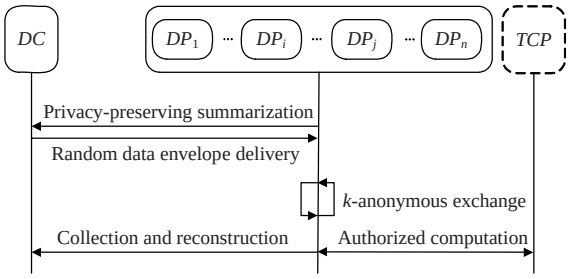

Fig.2 Framework of the distributed anonymous data perturbation method

The assumptions are: DPs are to be fully distributed for simplicity (each DP holds one datum only). DC and TCP have their public certificate and corresponding private key based on the Paillier public key mechanism (Paillier, 1999). DPs hold the public key of DC initially. The authentication method is applied so that unidentified users are not allowed to participate in the process.

The homomorphic encryption algorithm (Paillier, 1999) is used in the framework. It allows basic calculations of encrypted data, and is widely used in electronic voting, etc. The main feature of Paillier's algorithm is that, $E(a_1)E(a_2)=E(a_1+a_2)$, where $E()$ is the encryption function and a_1, a_2 are the data to be encrypted. For a dataset a_1, a_2, …, a_n, the equation is

$$E(a_1+a_2+\ldots+a_n)=E(a_1)E(a_2)\ldots E(a_n). \quad (6)$$

Paillier's public key based mechanism is applied in DADP to encrypt and sign the original and random data to keep data in privacy and support trusted computation.

The distributed data perturbation process can be summarized into the following four steps.

Step 1: Generating the perturbing data.

To generate an appropriate perturbing data series, *DC* requires the variance of original dataset σ_{origin} to determine the variance of perturbing dataset σ_{perturb}. In

general, the signal noise rate (SNR) is preset to measure the strength of perturbation: $SNR=20\log(\sigma_{origin}/\sigma_{perturb})$. DC uses DSSP to achieve the variance $\sigma_{perturb}$ from DPs securely.

DC generates the random dataset $Y=\{Y_1, Y_2, \ldots, Y_n\}$ with the distribution of zero mean and $\sigma_{perturb}$. The value of $\sigma_{perturb}$ depends upon σ_{origin} and preset SNR. The length of the random dataset is based on the number of nodes (n) in the environment.

Step 2: Delivery and exchange.

DC encrypts each random datum Y_i (i=1, 2, …, n) with its public key, and packages the encrypted datum and DC's signature into an envelope ($RDENV_i$) with its signature:

$$RDENV_i=\{E_{DC}(Y_i), Sig_{DC}(E_{DC}(Y_i))\},$$

where $E_{DC}()$ is the encryption function using DC's public key, and $Sig_{DC}()$ is DC's signature function. DC delivers envelopes ($RDENV_1$, $RDENV_2$, …, $RDENV_n$) to DPs, respectively. DP_i has $\{X_i, RDENV_i\}$ as a pair.

Taking RDENV as OBJ, DPs execute the k-anonymous exchange protocol (Fig.3). Three exchange modes are presented: push mode, cooperate mode, and fetch mode. In push mode, each DP randomly selects a target DP (TDP) to send its RDENV; in cooperate mode, each DP randomly finds a TDP to form a pair, and two DPs in pairs exchange the RDENV with each other; in fetch mode, each DP randomly sends a request to a TDP, and retrieves its RDENV. The difference among the modes will be discussed later, in terms of efficiency and privacy.

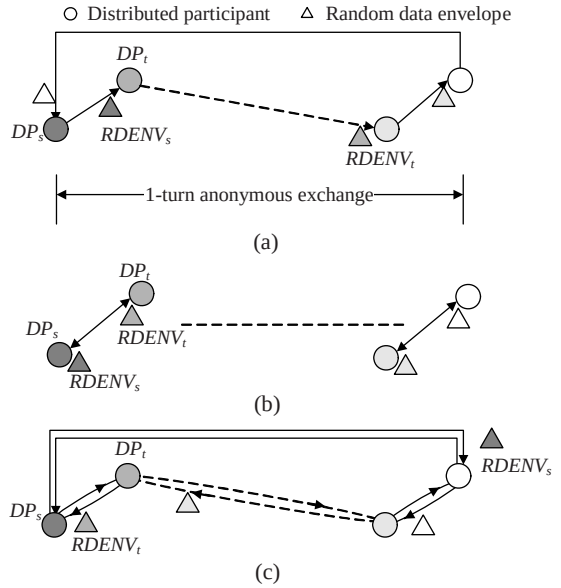

Fig.3 Random data envelope in anonymous exchange
(a) Push mode; (b) Cooperate mode; (c) Fetch mode

Each DP holds a new RDENV that differs from the one received from DC after Step 2.

Step 3: Authorized computation.

TCP helps to calculate the perturbed value in a secure way. The main operation in Step 3 is to add the random data to the original ones by TCP. Steps for DP_i to get trustworthy perturbation results through TCP are given as follows:

(1) DP_i encrypts its value X_i with DC's public key.

(2) DP_i updates the $RDENV_i$ and submits it to TCP, formatted as

$$RDENV_i'=\{E_{DC}(X_i), E_{DC}(Y_u), Sig_{DC}(E_{DC}(Y_u))\}.$$

(3) TCP verifies the signature $Sig_{DC}(E_{DC}(Y_u))$ to assure the random data is correctly generated and delivered by DC.

(4) TCP calculates $E_{DC}(\omega_i)$ through $E_{DC}(\omega_i)=E_{DC}(X_i)E_{DC}(Y_u)=E_{DC}(X_i+Y_u)$.

(5) TCP signs on the result and returns $TCPENV_i'=\{E_{DC}(\omega_i), Sig_{TCP}(E_{DC}(\omega_i))\}$ to DP_i.

Every DP gets the perturbed value with TCP's signature after Step 3.

Step 4: Collection and reconstruction.

DPs submit the perturbed values to DC. DC verifies the correctness of the perturbation process through the signature. DC decrypts all $E_{DC}(\omega_i)$ to retrieve the perturbed dataset after verification:

$$D_{DC}(E_{DC}(\omega_i))=\omega_i=X_i+Y_u.$$

DC finally reconstructs the distribution function $f(\tilde{X})$ from $\{\omega_1, \omega_2, \ldots, \omega_n\}$ by the method in the subsection "Random value pertubation" and transfers the dataset to the miner for data mining.

SECURITY AND PERFORMANCE ANALYSIS

This section focuses on analyzing and testifying the performance of the distributed data perturbation scheme in terms of resolving the privacy constraints presented in Section 2. Specifically, we first analyze the robustness of DADP under two types of threat: collusion attacks and malicious attacks.

Robustness under Threat A: collusion attacks

Definition 1 (Threat A) Collusion participants (CPs) exist in DPs in a semi-honest environment. CPs obey the protocol but share the information they obtain during the process with other CPs to try to obtain extra information from other participants. Two steps in DADP that contain exchange operations are mainly considered in connection with collusion attacks.

1. DADP Step 1

Step 1 of DADP mainly involves the DSSP protocol. Here we compare the security feature of ODP, SSP and DSSP protocols. ODP and SSP use randomized DP queries to retain data in privacy. However, such methods are not safe when a collusion attack occurs, as previously described. In ODP, the collusion of two DPs (DP_{i-1} and DP_{i+1}) would lead to one datum leakage. In SSP, due to the different query order, the collusion of four DPs (DP_{i-1}, DP_{i+1}, DP'_{i-1} and DP'_{i+1}) is able to retrieve DP_i's datum.

To give a quantification of the privacy under a collusion attack, we assume that all DPs generate the randomized query together without a third-party. Each DP finds a random successor till the query completes.

Define three key parameters to evaluate the probability of a successful collusion attack on one DP: n, the number of all DPs; c, the number of CPs, $1<c<n$; m, the number of divided shares, $1<m\leq n$.

In ODP, a collusion attack is successful when a DP selects a CP as its successor; therefore the probability of a successful collusion attack is

$$p_{\mathrm{ODP}}=\frac{c-1}{n-1}. \tag{7}$$

p_{ODP} is proportional to c, the number of colluders. Furthermore, when $c>2$, CPs may use their relationship to the positions in query to attack more DPs, as shown in Fig.4. In this example, CP_2 is the output listener for the attack on HDP_1 (honest DP). Meanwhile, it is also the input listener for the attack on HDP_2. Therefore, three CPs successfully attack two HDPs. In general, c CPs may attack $c-1$ HDPs at worst.

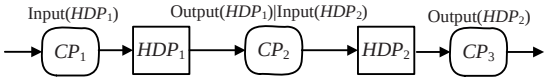

Fig.4 Collusion attack on ODP

In SSP, the probability of a successful attack is lowered due to the extra randomization operation on DPs:

$$p_{\mathrm{SSP}}=\frac{c-1}{n-2}\cdot\frac{c-3}{n-4}. \tag{8}$$

Similarly, CPs are able to attack more HDPs in a group attack, as shown in Fig.5. Four CPs can attack two HDPs. In general, c CPs may attack $\lfloor c/2\rfloor$ HDPs at worst.

In conclusion, methods based on a randomized query are vulnerable to collusion attacks, especially when the colluder density is high. The threat may be resolved if a third-party participates to generate the random query. However, the privacy issue is then transferred to the security and trust of the third-party.

In DSSP, all DPs find the successors initiatively. Therefore CP is not able to select the target, which lowers the collusion probability. As DP_i keeps one of the m shares $X_{i,1}$, the value of X_i will not be cracked even if the $m-1$ shares are sent to CPs. However, the situation changes if DC participates in the collusion. When all $m-1$ data holders who send their shares to DP_i are colluding, $X_{i,1}$ can be figured out through $X_{i,1}=sum_i-\sum_{q\in CP}mshare_q$, where sum_i is the sum of m different shares that DP_i sends to DC, and $mshare_q$ is

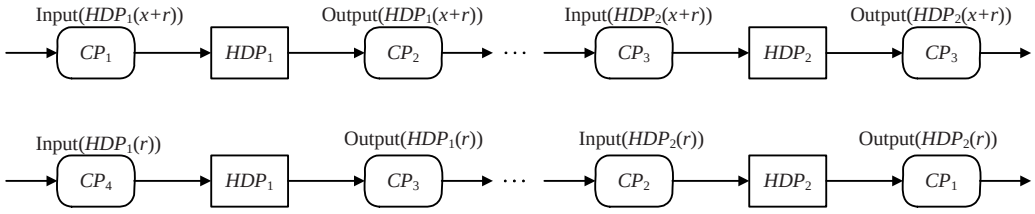

Fig.5 Collusion attack on SSP

the $m-1$ shares that DP_i gets from CPs. Therefore, $X_i = X_{i,1} + \sum_{j=2}^{m} X_{i,j}$ leaks out.

Using c, m and n to describe p_{DSSP0}, the probability of one or more successful collusion attacks on one DP:

$$p_{\mathrm{DSSP0}} = \begin{cases} (C_c^{m-1} / C_n^{m-1})^2, & m \le c+1, \\ 0, & m > c+1. \end{cases} \quad (9)$$

It is clear that the collusion attack would not succeed when $m>c+1$. In the extreme case, let $m=n$. It means that no information leaks out unless DC and all other $n-1$ participants are colluding. On the other hand, the minimum value of m can be computed through Eq.(9) to lower the communication cost, when the approximation number of CPs is given.

To obtain the variance, each DP needs to submit two values: X_i and X_i^2. Hence, the total probability of one or more successful attacks is

$$p_{\mathrm{DSSP}} = \begin{cases} 1-(1-p_{\mathrm{DSSP0}})^2 \\ \quad = 1-(1-(C_c^{m-1} / C_n^{m-1})^2)^2, & m \le c+1, \\ 0, & m > c+1. \end{cases} \quad (10)$$

Experiments were conducted in a JAVA environment using Windows 2003. Two hundred DPs (n=200) were simulated to compare the privacy performance among the protocols. In Fig.6, the percentage of compromised HDPs is calculated as $a_{\mathrm{H}}/(n-c)$, in which a_{H} is the number of attacked HDPs. DSSP 0/1/5 represent not colluding with DC ($m\in[1, n]$), colluding with DC and m=1%×n=2, colluding with DC and m=5%×n=10, respectively. Fig.6 shows that for ODP and SSP, the percentage of compromised HDPs grows rapidly when the number of CPs rises, especially when c>50%×n. But for DSSP, when DC is not colluding with CPs, a collusion attack is not successful even if $n-1$ CPs are colluding. Even when DC colludes, DSSP shows much higher security robustness than ODP and SSP. The number of compromised HDPs is still close to 0 when $c/n\le$50% and m=5.

In summary, the DSSP protocol with a proper value of m can achieve privacy in a semi-honest environment with collusion attacks. The value of m can be calculated using Eq.(10) when an estimation of c is given.

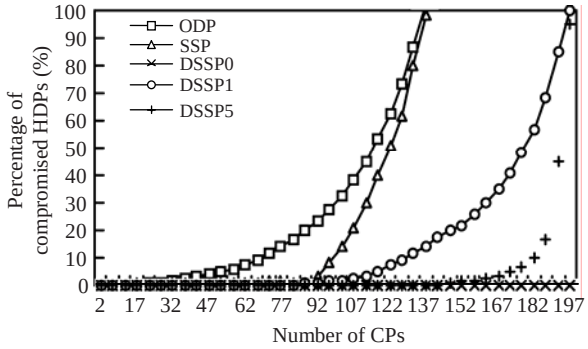

Fig.6 Comparison of different secure sum protocols under collusion attacks

2. DADP Step 2

Exact values in *RDENVs* are invisible to DPs due to Paillier's cryptographic method applied in the whole exchange process. But when DC, the private key holder, participates in the collusion, DPs are under threat. We discuss two main security tasks in DADP Step 2:

(1) Remove identical information to resolve the tracing problem.

Definition 2 (Tracing problem) The process where DC sends RDENVs to DPs is actually a process of mapping construction of DP-RDENV by DP. As a result, the exact random values that DPs hold is clear to DC. It is said that RDENV is distinguishable and DP is traced if DC can maintain the mapping relation.

In DADP, DC takes the role of a random data generator and deliverer. Hence DC keeps the full mapping relation of all RDENVs and DPs. DPs are initially traced by DC. If DPs add the random data directly to their original data, DC can retrieve the original data of DPs with a simple minus operation: $X_i=\omega_i-Y_i$. Using the k-turn anonymous exchanging process without DC's participation, DPs make RDENVs indistinguishable to DC. k=1 is sufficient to disturb the initial tracing status in most cases.

(2) Reduce the possibility of information disclosure on condition of CPs' collusion with DC. The tracing problem gets worse when DC colludes together with CPs. On condition that CPs share the exchanging information with DC, DC can still update the mapping DP-RDENV to keep tracing a certain DP. Note that there are two modes in multiple turns' exchange: (i) DP waits for all others to finish the

exchange before the next round starts; (ii) DP finds k different participants to exchange, without waiting for the completion of all peers in the current round.

In the synchronous mode, DP will be traced if only its exchange partner colludes with DC in the last turn (the kth turn), regardless of all former turns. As a result, k has no effect in resolving the tracing problem in such a mode.

While in the asynchronous mode, CPs must involve all k turns with one DP to keep its tracing status; otherwise, the target status is lost. The main reason is that CPs would not be able to count the exact turn that the target DP is in unless they get the status of its all turns. Hence, the probability for one DP to be tracked after the k-turn exchange process is

$$pca_{s2} = \begin{cases} C_c^k / C_{n-1}^k, & k \le c < n, \\ 0, & k > c. \end{cases} \tag{11}$$

The probability of one or more successful collusion attacks was simulated in a JAVA environment using Windows 2003 to illustrate the performance and robustness of DADP (Fig.7). Assume that there are 500 nodes in the distributed privacy-preserving data mining (DPPDM) environment and data are fully distributed (each DP holds one record). At least one DP is honest (HDP). We focus on one HDP threatened by collusion attacks.

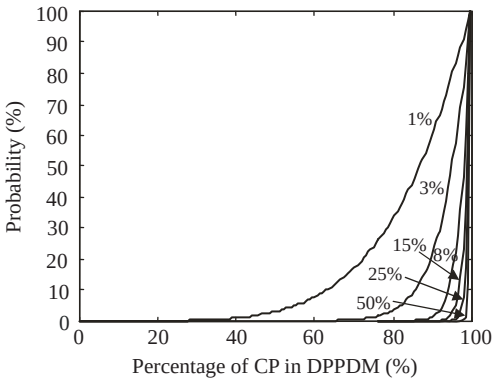

Fig.7 Probability of one or more successful collusion attacks on one DP with various parameters in DADP Step 2

Fig.7 shows the probability of one or more successful collusion attacks on one DP in DADP Step 2. The probability of a successful collusion attack reduces fast when k increases. Specifically, when c/n increases to 75 (15% of the number of total nodes), one DP can survive in the environment even when 90% of the participants are colluding. In other words, a lower value of k, compared to c, can protect the DP from the collusion attack. To a certain degree, the robustness benefits from the initiative target search mode (Fig.3).

In summary, comparatively higher values of key parameters m and k will keep the method away from collusion attacks described in Threat A. On the other hand, a balanced solution between security and performance can be achieved. Using Eqs.(10) and (11), the values of m and k can be computed when a maximum threshold of attack probability and the estimation number of colluders are given, so that the communication and computation cost can be minimized. Kargupta *et al*.(2007) proposed a light-weight method to let each DP estimate the percentage of colluders in the environment. The main idea is that each DP will receive a collusion invitation from CPs before the process. Therefore, an approximate percentage of CPs in the environment is obtained.

Robustness under threat B: malicious attacks

Definition 3 (Threat B) Malicious attacks in data exchange or malicious actions taken by some participants that do not follow the protocol.

The use of a cryptographic algorithm and signature method based on the Paillier public key mechanism has made data exchange and the authentication process of DADP robust to a malicious attack:

(1) Tampering attack: Cryptographic signature is applied as a method of authentication. DPs and TCP verify whether the random value is generated by DC through DC's signature in RDENV. Similarly, DC can verify whether the DPs' private values have been added to the right random data delivered from DC via TCP's signature. Therefore, the random data, the main object transfer red in the whole process, are protected against tampering and replacing attacks.

(2) Integrity attack: Malicious participants (MPs) cannot generate a real RDENV due to the signature. But they have chances to disturb the existing envelopes in the exchange process. For instance, one MP has accessed k' random envelopes after k' turns exchange, as ($RDENV_a$, $RDENV_b$, …, $RDENV_f$, $RDENV_{k'}$), $k'<k$. It should use the latest random envelope $RDENV_{k'}$ to exchange with its partner in the

next round, as defined in the protocol. If MP uses one random envelope $RDENV_v$ ($v \in \{a, b, \ldots, f\}$), it accessed in former turns rather than $RDENV_{k'}$, some envelope is abandoned and some is duplicated in global view. Integrity of the model breaks down, which leads the reconstruction to failure. One way to detect such attacks is as follows: due to the zero mean of the random dataset, the sum of the perturbed data is equal to sum of the original ones: $\sum_{i=1}^{n} \omega_i = \sum_{i=1}^{n} X_i + \sum_{j=1}^{n} Y_i = \sum_{i=1}^{n} X_i$. Therefore, the comparison result between $\sum_{i=1}^{n} X_i$ (retrieved in DADP Step 1) and $\sum_{i=1}^{n} \omega_i$ (retrieved in DADP Step 4) will tell DC whether an integrity attack occurs.

(3) Initiative collusion attack (ICA) based on tracing: In a semi-honest environment, the assumption is made that the target to exchange RDENV is randomly selected in DADP Step 2, as described in Section 3. However, malicious purposeful target selection would threaten a single DP's privacy. For instance, MPs choose HDPs purposefully rather than randomly to send the RDENVs. At least one target in HDPs is traced when $mp>k$, where mp is the number of MPs. The main reason for successful ICA is the mapping of DP-RDENV. Three exchange modes are introduced (Fig.3):

In mode (a), Push Mode, if MP initiatively pushes the traceable RDENV to a selected target HDP in one turn, the MP knows the exact RDENV that the HDP holds after that turn. Therefore the mapping of DP-RDENV remains maintainable when MPs collude, $mp>k$.

In mode (b), Cooperate Mode, suppose that n is even, DP_s finds DP_t to exchange RDENVs with each other as a pair. Therefore, both DP_s and DP_t can update the counterpart's DP-RDENV map. At least $k/2$ MPs are able to crack one HDP in this way: DP_s actually gets two items of mapping information about DP_t in one turn, i.e., the RDENV that DP_t holds in the previous turn (DP_t-$RDENV_t$) and the RDENV that DP_t holds in the current turn (DP_s-$RDENV_s$). Therefore, with the inference method, MPs can reckon up the whole mapping status based on time sequence with one-turn-interval target-selected ICA.

In mode (c), Fetch Mode, DP_s asks for a randomly selected target DP_t to retrieve its RDENV. DP_s, the exchange initiator, can only get the mapping of DP_t on the previous turn. Therefore, MPs cannot retrieve the final status of HDP, as no initiative method can be applied in the last turn. The probability of one HDP being cracked in mode (c) is $(mp-k)/(n-k)$. The probability increases when mp is up to n.

Table 1 shows a comparison on status of the three modes. It is observed that mode (c) is more secure in a high-density malicious environment with DC. From another point of view, a comparatively higher value of k is the most effective way to defend against ICA.

Table 1 Mapping status of different exchange modes

Mode	DP_s	DP_t	Probability of ICA when $mp>k$
(a)	Updates DP_t-$RDENV_t$	–	100%
(b)	Updates DP_t-$RDENV_t$	Updates DP_s-$RDENV_s$	100%
(c)	–	Updates DP_s-$RDENV_s$	$\frac{mp-k}{n-k}\times 100\%$

In conclusion, the analysis shows the robustness of DADP under two types of attacks. DADP resists both types of attack on the condition that DC is trustworthy. With appropriate parameters m and k, DADP shows high robustness in an environment of high-density collusion and malicious attackers. In addition, in a semi-honest environment (Definition 1), TCP may be omitted as all units follow the process. Therefore, DPs are trustworthy in performing the addition operation of random data and the original ones. The simplified model would cost less in communication and computation.

Other performance evaluation

In order to evaluate the efficiency of DADP, this subsection estimates the operational cost of the model. We focus on communication and computation costs in the evaluation.

1. Communication cost

The communication cost of each step in DADP is listed in Table 2.

It is clear that the additional communication cost is mainly caused by the anonymous process (AP) in DADP Steps 1 and 2 and is proportional to the parameters m and k. Therefore, the main costs L_1 and L_2 are in the range between $O(n)$ and $O(n^2)$. In general, a balanced solution between performance and security

Table 2 Communication cost of DADP

Step	Communication cost	
	With AP	Without AP
1	$L_1=2l_0n$	$L_1=2l_0mn$
2	$L_2=l_{env2}n$	$L_2=l_{env2}n(k+1)$
3	$L_3=(l_{env2}+l_{env3})n$	$L_3=(l_{env2}+l_{env3})n$
4	$L_4=2l_{env4}n$	$L_4=2l_{env4}n$

Note: AP: anonymous process; l_0, l_{env2}, l_{env3}, l_{env4} represent the lengths of random shares of original data, random data envelope delivered from DC, envelope updated by DP, and envelope signed by TCP, respectively

can be used to lower the cost under reasonable privacy constraints (described in the subsection "Robustness under threat A: collusion attacks"). It should be noted that the length of the envelope may have more effects on the cost when $l>>n$, but it is still far lower than the one in SMC methods.

2. Computation cost

Computation cost is mainly caused by cryptographic computation, such as data encryption, decryption, signature and verification process. Computations are: (1) DC encrypts random data with signature. (2) DPs verify the random envelope before exchange in each turn and encrypt the source data. (3) TCP verifies the signature on the random envelope and signs the computation result. (4) DC certifies and decrypts perturbed data from peers.

To summarize, most of these computations have a linear relationship with ln, except (2) which is proportional to kln. When the data length of each *DP* increases, a hash method must be used before producing the signature, in order to lower the computation cost. As a result, the performance cost is at the same level as in SMC methods introduced in Section 1. Security and performance advantages will become greater with the growth of n.

CONCLUSION

Security, along with the collusion problem, becomes a critical issue with DPPDM. This paper presents an anonymous perturbing method DADP that adapts the perturbation-based PPDM technologies to a distributed environment. Incorporating homomorphic cryptography techniques, DADP is more efficient and scalable than SMC related methods. With flexible parameters, DADP provides a balanced solution between performance and security, according to the environmental requirement. DADP proves to be robust in high-density collusion attacks and linear growth of communication cost upon node number. This basic framework of DADP can also be applied to perturbation based distributed clustering and association rule mining.

An open question is whether other centralized PPDM methods based on perturbation can also be transformed to fit into the distributed environment using the method given in this paper. We believe that these issues are worth researching in the future.

References

Agrawal, D., Aggarwal, C.C., 2001. On the Design and Quantification of Privacy Preserving Data Mining Algorithms. Proc. 20th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems, p.247-255. [doi:10.1145/375551.375602]

Agrawal, R., Srikant, R., 2000. Privacy-preserving data mining. *ACM SIGMOD Record*, **29**(2):439-450. [doi:10.1145/335191.335438]

Ashley, P., Hada, S., Karjoth, G., 2003. The Enterprise Privacy Authorization Language (EPAL 1.1), IBM. Available from: http://www.zurich.ibm.com/security/enterprise-privacy/epal/

Ashrafi, M.Z., Taniar, D., Smith, K., 2003. Towards Privacy Preserving Distributed Association Rule Mining. Distributed Computing-IWDC, p.279-289. [doi:10.1007/b94926]

Beaver, D., 1991. Foundations of secure interactive computing. *CRYPTO*, **1991**:377-391. [doi:10.1007/3-540-46766-1]

Bertino, E., Fovino, I.N., Provenza, L.P., 2005. A framework for evaluating privacy preserving data mining algorithms. *Data Min. Knowl. Discov.*, **11**(2):121-154. [doi:10.1007/s10618-005-0006-6]

Chaum, D., Crepeau, C., Damgard, I., 1988. Multiparty Unconditionally Secure Protocols. Proc. 20th Annual ACM Symp. on Theory of Computing, p.11-19. [doi:10.1145/62212.62214]

Chawla, S., Dwork, C., McSherry, F., Smith, A., Wee, H., 2005. Toward Privacy in Public Databases. Theory of Cryptography Conf., p.363-385. [doi:10.1007/b106171]

Cramer, R., Damgard, I., Nielsen, J.B., 2001. Multiparty Computation from Threshold Homomorphic Encryption. Proc. EUROCRYPT, p.280-300. [doi:10.1007/3-540-44987-6]

Cranor, L., Langheinrich, M., Marchiori, M., Presler-Marshall, M., Reagle, J. (Eds.), 2002. The Platform for Privacy Preferences 1.0 (P3P1.0) Specification. W3C. Available from: http://www.w3.org/TR/P3P/

CSA (Canadian Standards Association), 2004. Privacy Code. Available from: http://www.csa.ca/standards/privacy/Default.asp?laguage=English

Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J., 2004.

Privacy preserving mining of association rules. *Inf. Syst.*, **29**(4):343-364. [doi:10.1016/j.is.2003.09.001]

Fienberg, S.E., McIntyre, J., 2004. Data swapping: variations on a theme by dalenius and reiss. *Priv. Statist. Datab.*, **3050**:14-29. [doi:10.1007/b97945]

Fukasawa, T., Wang, J., Takata, T., Miyazaki, M., 2004. An Effective Distributed Privacy-preserving Data Mining Algorithm. Intelligent Data Engineering and Automated Learning (IDEAL), p.320-325. [doi:10.1007/b99975]

Goldreich, O., Micali, S., Wigderson, A., 1987. How to Play Any Mental Game or a Completeness Theorem for Protocols with Honest Majority. 19th ACM Symp. on the Theory of Computing, p.218-229. [doi:10.1145/28395.28420]

Kargupta, H., Das, K., Liu, K., 2007. Multi-party, privacy-preserving distributed data mining using a game theoretic framework. *LNCS*, **4702**:523. [doi:10.1007/978-3-540-74976-9]

Liew, C.K., Choi, U.J., Liew, C.J., 1985. A data distortion by probability distribution. *ACM Trans. Datab. Syst.*, **10**(3):395-411. [doi:10.1145/3979.4017]

Paillier, P., 1999. Public-key cryptosystems based on composite degree residuosity classes. *Advances in Cryptology EUROCRYPT*, **99**:223-238. Available from: http://www.springerlink.com/content/kwjvf0k8fqyy2h3d/

Rizvi, S.J., Haritsa, J.R., 2002. Maintaining Data Privacy in Association Rule Mining. Proc. 28th Int. Conf. on Very Large Data Bases, **28**:682-693. [doi:10.1016/B978-155860869-6/50066-4]

Sweeney, L., 2002. Achieving *k*-anonymity privacy protection using generalization and suppression. *Int. J. Uncert., Fuzz. and Knowl.-based Syst.*, **10**(5):571-588. [doi:10.1142/S021848850200165X]

Yao, A.C., 1986. How to Generate and Exchange Secrets. Proc. 27th IEEE Symp. on Foundations of Computer Science, p.162-167.

Zhang, P., Tong, Y.H., Tang, S.W., Yang, D.Q., 2005. Privacy Preserving Naive Bayes Classification Advanced Data Mining and Applications. **3584**:744-752. [doi:10.1007/b11111]