Cluster Comput
DOI 10.1007/s10586-017-1019-9

Outsourced privacy-preserving C4.5 decision tree algorithm over horizontally and vertically partitioned dataset among multiple parties

Ye Li[1] · Zoe L. Jiang[2] · Lin Yao[3] · Xuan Wang[1] · S. M. Yiu[4] · Zhengan Huang[5]

Received: 18 March 2017 / Revised: 6 June 2017 / Accepted: 24 June 2017

Abstract Many companies want to share data for data-mining tasks. However, privacy and security concerns have become a bottleneck in the data-sharing field. The secure multiparty computation (SMC)-based privacy-preserving data mining has emerged as a solution to this problem. However, there is heavy computation cost at user side in traditional SMC solutions. This study introduces an outsourcing method to reduce the computation cost of the user side. We also preserve the privacy of the shared databy proposing an outsourced privacy-preserving C4.5 algorithm over horizontally and vertically partitioned data for multiple parties based on the outsourced privacy preserving weighted average protocol (OPPWAP) and outsourced secure set intersection protocol (OSSIP). Consequently, we have found that our method can achieve a result same the original C4.5 decision tree algorithm while preserving data privacy. Furthermore, we also implement the proposed protocols and the algorithms. It shows that a sublinear relationship exists between the computational cost of the user side and the number of participating parties.

Keywords Secure multiparty computation · Outsourced computation · C4.5 decision tree · Privacy preserving data mining · PPWAP · SSIP

✉ Ye Li
liye@cs.hitsz.edu.cn

✉ Zoe L. Jiang
zoeljiang@gmail.com

Lin Yao
yaolin@insun.hit.edu.cn

Xuan Wang
wangxuan@cs.hitsz.edu.cn

S. M. Yiu
smyiu@cs.hku.hk

Zhengan Huang
hza5288@163.com

1 Harbin Institute of Technology Shenzhen Graduate School, Shenzhen 518055, China

2 Harbin Institute of Technology Shenzhen Graduate School and Guangdong Provincial Key Laboratory of High Performance Computing, Shenzhen 518055, China

3 Harbin Institute of Technology School of Software, Harbin 150001, China

4 The University of Hong Kong, Hong Kong, SAR, China

5 Guangzhou University, Guangzhou 510006, China

1 Introduction

With the increasing development of the internet, a huge amount of data nowadays is generated every day. A data owner would like to share data with a third party, such as the Cloud Computing Center, for parallel deep data-mining tasks because of computational limitation. However, privacy and security concerns restrict data sharing [17]. The secure multi-party computation (SMC)-based privacy-preserving data mining algorithm has emerged as a solution to this problem [1]. Nevertheless, traditional SMC solutions are too inefficient and infeasible to enable truly large-scaled analytics to handle big data [32]. Therefore, computation and communication costs for ensuring accuracy and improving traditional SMC methods must be considered.

Cloud computing, which is the long-standing vision of computing as a utility, enables convenient and on-demand network access to a centralized pool of configurable computing resources. One of the most attractive benefits of this new computing environment is that the resource-constrained devices can outsource their computation workloads to cloud servers [39] and enjoy unlimited computation resources in a

Published online: 07 July 2017

ID	Age	Edu.	Salary	Loan	Overdue
1	31-40	H. S.	40-60 K	20 K	Y
2	21-30	U.	60-80 K	20 K	N
3	51-60	H. S.	60-80 K	30 K	Y
4	41-50	U.	>100 K	40 K	Y
5	51-60	H. S.	40-60 K	20 K	N
6	21-30	H. S.	20-40 K	20 K	Y
7	31-40	U.	60-80 K	0 K	N
8	41-50	H. S.	60-80 K	0 K	N
9	21-30	U.	40-60 K	30 K	Y
10	41-50	U.	>100 K	20 K	N

Party 1 Party 2

Fig. 1 Horizontally partitioned data

ID	Age	Edu.	Salary	Loan	Overdue
1	31-40	H. S.	40-60 K	20 K	Y
2	21-30	U.	60-80 K	20 K	N
3	51-60	H. S.	60-80 K	30 K	Y
4	41-50	U.	>100 K	40 K	Y
5	51-60	H. S.	40-60 K	20 K	N
6	21-30	H. S.	20-40 K	20 K	Y
7	31-40	U.	60-80 K	0 K	N
8	41-50	H. S.	60-80 K	0 K	N
9	21-30	U.	40-60 K	30 K	Y
10	41-50	U.	>100 K	20 K	N

Party 1 Party 2

Fig. 2 Vertically partitioned data

pay-per-use manner [16]. Cloud-based outsourcing enables data owners to employ third parties to process the data [34], secure integration, and verify Internet of things [18,28]. Doing some work based on encrypted data, such as identity-based encryption [12] and search work over encrypted outsourced data [6,42–44], is also very valuable.

Data owners can reduce the computation costs of a cryptographic scheme by outsourcing most of the computations to cloud servers. However, this solution may cause some leakage of the user's privacy with multiple parties. Moreover, the existence of two common types of distributed datasets (i.e., the horizontally distributed data and the vertically distributed data) worsens the user's privacy leakage problems.

In the horizontally distributed dataset, two or more parties hold different objects for the same set of attributes, whereas in the vertically distributed dataset, two or more parties hold different sets of attributes for the same set of objects.

Assume that n ($n \geq 2$) parties exist, and each party P_i has a database S_i, ($1 \leq i \leq n$). We annotate the entire database herein as S and $S = S_1 \cup \cdots \cup S_n$.

(1) The parties for the horizontally distributed dataset shared the set of general attributes $A = A_1 \cup A_2 \cup ... \cup A_n$ and the class attribute $C = c_1, ..., c_m$, but the number of records in the databases $|S_i|$ and the true record values $v_i \in A$ are unknown to the other parties. Fig. 1 shows an example.

(2) The parties for the vertically distributed dataset shared the number of records in the databases $|S_i|$ and the same order of the records, but hold different sets of attributes A_i for the same set of objects. The class attribute C is also known only by one party. Fig. 2 presents an example.

Aside from the algorithm output, the parties want to run the data-mining algorithm on the horizontally and vertically partitioned data for multiple parties without revealing their secrets to each other.

The proposed approach is based on the C4.5 [23] decision tree algorithm, developed by Quinlan et al. It is an extension of the ID3 algorithm [24] used to generate decision trees. C4.5 made a number of improvements to ID3 [23]. Some of these are: (1) handling both continuous and discrete attributes—in order to handle continuous attributes, C4.5 creates a threshold and then splits the list into those whose attribute value is above the threshold and those that are less than or equal to it. (2) Handling training data with missing attribute values. (3) Handling attributes with differing costs. (4) Pruning trees after creation—C4.5 goes back through the tree once it's been created and attempts to remove branches that do not help by replacing them with leaf nodes.

1.1 Our contributions

In this study, we mainly design two outsourced privacy-preserving C4.5 decision tree algorithm, which enables privacy-preserving data mining on both the horizontally and vertically partitioned data for multiple parties. In details, the OPPWAP and the OSSIP are proposed as the building blocks. When the outsourced privacy-preserving C4.5 (OPPC4.5) algorithm over horizontally partitioned data is proposed based on the OPPWAP, and the OPPC4.5 over vertically partitioned dataset is proposed based on the OSSIP.

The basic idea of this study is described as follows:

(1) For the horizontally partitioned dataset, we first reduce the key function of the distributed C4.5 decision tree algorithm to a weighted average problem (WAP), which only requires a secure calculation of $(x + y)/(a + b)$. We then propose the outsourced privacy preserving

weighted average protocol (OPPWAP) based on the BCP cryptosystem [2] to securely calculate the related values. After which, we build the outsourced privacy-preserving C4.5 algorithm over horizontally partitioned dataset based on the OPPWAP.

(2) For the vertically partitioned dataset, we transform all items in the vertically partitioned dataset to binary vectors. We then reduce the key function of the C4.5 decision tree algorithm over vertically partitioned dataset to the secure set intersection problem (SSIP), which only requires a secure calculation of the set of intersection operations. We also propose the outsourced secure set intersection protocol (OSSIP) based on the BCP cryptosystem [2] to securely calculate the related values. After which, we build the outsourced privacy-preserving C4.5 algorithm over vertically partitioned dataset based on the OSSIP.

As a result, all the parties and the outsourcing server jointly run the OPPC4.5 algorithms to finally get the mining result. At the result, either any party or server has no knowledge on the other party's privacy data. Also, server has no knowledge of the mining result.

2 Related work

Many techniques have recently been proposed to address the SMC-based privacy-preserving data mining problem, which can be classified into the two following categories according to the existence of cloud or not:

(1) *No cloud exist methods* The computation in the no cloud exist methods is distributed among the parties. In 2002, Lindell and Pinkas [40] proposed the first cryptography-based approach to build a decision tree over horizontally partitioned data among two parties. After which, similar works were performed to address the privacy-preserving decision tree construction problem. Zhan et al. [14], Emekci et al. [5], Samet and Miri [26], and Xiao et al. [19] discussed the ID3 decision tree on the horizontally distributed database, whereas Vaidya and Clifton [29] addressed the same problem on the vertically distributed database. Xiao et al. [38] discussed the C4.5 decision tree on the horizontally distributed database, whereas Shen et al. [27] and Gangrade et al. [7] addressed the C4.5 algorithm on the vertically distributed database. Veluru et al. [33] discussed privacy preserving text analytics.

(2) *Cloud exist methods* In the cloud exist methods, each party can outsource their computation workloads to cloud servers and enjoy unlimited computation resources in a pay-per-use manner [8]. However, the security and the privacy of outsourced data must be ensured. In this case, the cloud servers can only be assumed to be semi-trusted, while the computation tasks often contain some sensitive information that should not be exposed to the cloud servers. In 2005, Hohenberger et al. [10] provided a formal security definition for securely outsourcing computations and presented a scheme securely outsourcing cryptographic computations. In 2011, Kamara et al. [15] designed a general outsourcing multi-party computation protocol for a server-aided two-party. Meanwhile, in 2013, Peter et al. [22] proposed an efficient outsourcing multi-party computation construction under multiple keys, which can be applied in our case. In 2014, Liu et al. [3] applied a new encryption scheme for outsourcing privacy-preserving k-means data mining. Most of the computations in this scheme were finished on the cloud and reduced the computation work of the data owner, but its scheme was one partys data mining. Xiaoyan et al. [37] extended that scheme to the two-party case.

Table 1 presents the characteristic comparison among different schemes. We focus herein on the outsourced privacy-preserving C4.5 decision tree protocol with the cloud exist method. Unlike the existing work, we propose the protocol over both horizontally and vertically partitioned data and extend the protocol among multiple parties.

2.1 Organization

The rest of the paper is organized as follows: the system model and security model used in this study are presented in Sect. 3; the Secure Multi-party Computation, BCP homomorphic encryptions, our OPPWAP, and the OSSIP we used in our protocol are reviewed in Sect. 4; our privacy-preserving

Table 1 The characteristics comparison among different schemes

Paper	Algorithm	Data partition	Number of parties	With cloud?
[19,21,40]	ID3	Horizontally	2	No
[14]	ID3	Horizontally	2	No
[5,26]	ID3	Horizontally	n	No
[29]	ID3	Vertically	n	No
[38]	C4.5	Horizontally	2	No
[27]	C4.5	Vertically	2	No
[7]	C4.5	Vertically	n	No
[3]	K-means	Horizontally	1	Yes
[37]	K-means	Horizontally	2	Yes
Our scheme	C4.5	Horizontally & vertically	n	Yes

C4.5 protocol is discussed in Sect. 5; the experiments of our PPC4.5 protocol are presented in Sect. 6; and the conclusions are drawn in Sect. 7.

3 Models and design goals

In this section, we formalize the system and security models and identify the design goals.

3.1 System model

The system model (Fig. 3) comprises two or more data owners and a cloud. Each data owner has a private database and can request the cloud to mine the C4.5 decision tree from a joint database on their behalf. The semi-honest cloud is tasked with the mining of the C4.5 decision tree for data owners and sending of the mining result to relevant data owners. The system comprises servers C and S in the cloud and n different parties.

3.2 Security model

In this study, the cloud is considered to be semi-honest and is assumed to have some background knowledge of the attribute names and the number of their attribute values. For each of these datasets, the cloud may also know which data owners private database the item belongs to. Moreover, we do not assume the cloud to be colluding with any data owner. Hence, colluding and insider attacks are not considered in this study.

The data owners herein are also considered as semi-honest. Furthermore, we assume that each data owner has some knowledge of the other owners' private data-bases. With some knowledge of the other owners' private databases, these data owners believe that they can benefit from the collaborative mining. We assume that each data owner knows the attribute names and the size of any other data owner's private database.

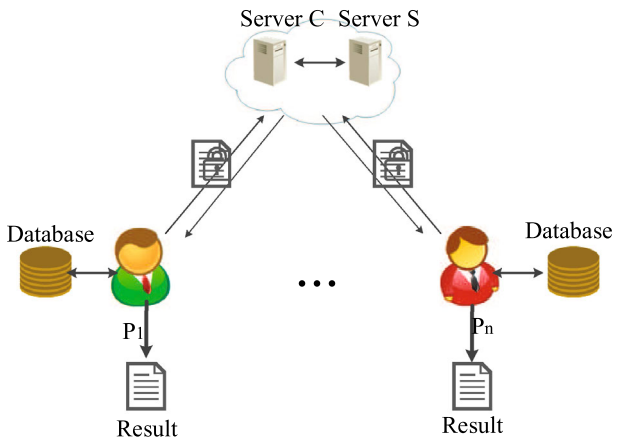


Fig. 3 System model under consideration

4 Building blocks

4.1 Secure multi-party computation

SMC [1] is mainly concerned with the problem of evaluating a function with the private input of two or more parties. Thus, after running the protocol, each party holds a share of the output with no additional information revealed. There are basically two main security models: semi-honest and malicious.

4.1.1 Semi-honest adversaries

In the semi-honest adversarial model, even corrupted parties correctly follow the protocol specification. However, the adversary obtains the internal state of all corrupted parties, including a transcript of all the messages received, and attempts to use this to learn information that should remain private. Semi-honest adversaries are also called honest-but-curious and passive adversaries.

4.1.2 Malicious adversaries

The corrupted parties in this adversarial model can arbitrarily deviate from the protocol specification according to the adversarys instruction. Providing security in the presence of malicious adversaries is generally preferred because it ensures that no adversarial attack can succeed. Malicious adversaries are also called active adversaries.

This study mainly focuses on solving the SMC problem of the semi-honest model and assumes that all parties and the cloud server are the semi-honest adversaries.

4.2 The BCP homomorphic cryptosystem

The BCP homomorphic cryptosystem [2] is an additive homomorphic cryptosystem introduced by Bresson, Catalano, and Pointcheval in 2003 and has the following property:

$$Enc_{pk}(m_1 + m_2) = Enc_{pk}(m_1) \odot Enc_{pk}(m_2) \quad (1)$$

where m_1 and m_2 are two messages, and $\odot$ is an arithmetic multiplication operation in the encrypted domain under the same public key.

In contrast to the traditional homomorphic cryptosystem (such as in Paillier [20] or ElGamal [4]), the BCP cryptosystem supports arithmetic operations in the encrypted domain under multiple parties' public keys. This can be done using the cloud server generating a public parameter. After which, every party generates its key pairs based on the public parameter. All the parties' public keys can then be transformed into a uniform public key, and the ciphertext under each party's

public key can be transformed into the ciphertext under the uniform public key.

The BCP cryptosystem employs two noncolluding, semi-honest cloud servers (i.e., C and S). Server S only generates the public parameter PP, while server C can compute with limits of interaction of clients. The cryptosystem is constructed as follows:

$Setup(\lambda)$: Given a security parameter λ, server S chooses $N = pq$ of bit length λ, where $p = 2p' + 1$ and $q = 2q' + 1$ for distinct primes p' and q', respectively. S then selects a random element $g \in \mathbb{Z}^*_{N^2}$, such that $g^{p'q'} \mod N^2 = 1 + \lambda N$ for $\lambda \in [1, N-1]$. Subsequently, S generates the public parameter PP and the master key MK as follows:

$$PP = (N, \lambda, g) \text{ and } MK = (p', q').$$

$KeyGen(PP)$: After each party obtains PP, it selects a random $a_i \in \mathbb{Z}_{N^2}$ and generates its public key pk_i and private key sk_i as follows:

$$pk_i = g^{a_i}, sk_i = a_i$$

$Enc_{pk_i}(m_i)$: Each party selects a random $r_i \in \mathbb{Z}_{N^2}$ and outputs the ciphtertext (A_i, B_i) as follows, given a plaintext $m_i \in \mathbb{Z}_N$:

$$A_i = g^{r_i} \mod N^2, B_i = g^{a_i r_i}(1 + m_i N) \mod N^2.$$

$Dec_{sk_i}(A_i, B_i)$: Each party can output the plaintext m_i as follows, given a ciphertext (A_i, B_i) and a secret key $sk_i = a_i$:

$$m_i = \frac{B_i/(A_i^{a_i}) - 1 \mod N^2}{N}.$$

We refer the interested reader to Refs. [2] and [22] for the detailed construction of the BCP cryptosystem.

4.3 Outsourced privacy preserving weighted average protocol

In this section, an outsourced privacy preserving weighted average protocol (OPPWAP) is proposed as follows:

The privacy preserving weighted average problem (PPWAP) [9] is defined as follows:

Definition 1 Party 1 has a pair (x, a), where x is a real number, and a is a positive integer. Similarly, Party 2 has pair (y, b). They intend to jointly compute $(x+y)/(a+b)$, while there is no need to disclose their private data to the counterpart.

We modify it for the multi-party situation in our scheme in a cloud-computing environment. We use the BCP algorithm described in Sect. 4.3. Our solutions for the OPPWAP are outlined in the paragraphs that follow.

Suppose there are n parties $\{P_i \mid 1 \le i \le n\}$, and each of whom has private input x_i and a_i, without knowing the other parties' x_i and a_i. At the end, each P_i will obtain the same average number of $\sum_{i=1}^{n} x_i / \sum_{i=1}^{n} a_i$. We suppose that all of the parties and the cloud servers (i.e., C and S) are semi-honest, and that, no party obtains information about the other's input, except the output value.

Algorithm 1 describes the detailed protocol.

Algorithm 1: OPPWAP

Require: $\{P_i \mid 1 \le i \le n\}$, each of whom has a private input x_i and a_i.

1. Server S chooses the BCP homomorphic encryption scheme, and distributes its public parameters $PP = (N, k, g)$ to C and P_i.
2. After this initial setup, P_i can generate its key pair of (pk_i, sk_i) based on the received PP. It then sends public keys pk_i to C and other parties.
3. C then calculates the product Prod.pk $:= \prod_{i=1}^{n} pk_i \mod N^2$, which can encrypt all P_i's data under a single public key. C then returns Prod.pk to each client.
4. P_i encrypts its private data x_i and a_i with $Prod.pk$ and obtains the result of (A_i, B_i) and (A'_i, B'_i).
5. Subsequently, P_i selects random numbers ρ_i and ρ'_i, and re-calculates the encryptions as shown follows:

$$\begin{aligned} \bar{A}_i &= A_i \cdot g^{\rho_i} \mod N^2 \\ \bar{B}_i &= B_i \cdot \text{Prod.pk}^{\rho_i} \mod N^2 \\ \bar{A}'_i &= A_i \cdot g^{\rho'_i} \mod N^2 \\ \bar{B}'_i &= B'_i \cdot \text{Prod.pk}^{\rho'_i} \mod N^2 \end{aligned}$$

It then sends the encryptions to C.

6. C calculates the result of $\bar{A} = \prod_{i=1}^{n} \bar{A}_i$, $\bar{B} = \prod_{i=1}^{n} \bar{B}_i$, $\bar{A}' = \prod_{i=1}^{n} \bar{A}'_i$, $\bar{B}' = \prod_{i=1}^{n} \bar{B}'_i$, and sends the $\bar{A}$ and $\bar{A}'$ back to the clients.
7. P_i chooses other random numbers r_i and r'_i, then calculates the data of $X_i = r_i \cdot \bar{A}^{sk_i} \mod N^2$ and $X'_i = r'_i \cdot \bar{A}'^{sk_i} \mod N^2$, and sends X_i and X'_i to C. After which, P_i encrypts the r_i and r'_i with the other parties' public key $pk_j (1 \le j \le n, j \ne i)$, and sends them to the other parties.
8. C obtains the data of X_i and X'_i and calculates $K = \prod_{i=1}^{n} X_i \mod N^2$ and $K' = \prod_{i=1}^{n} X'_i \mod N^2$. It then chooses a random number τ and returns K, K', $\tau\bar{B}$ and $\tau\bar{B}'$ to each party.
9. Each party P_i first uses its private key to decrypt each parties' random number r_j and r_j, then calculates $R = \prod_{i=1}^{n} r_i$ and $R' = \prod_{i=1}^{n} r'_i$. Each can then obtain the average number as follows.

$$\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} a_i} = \frac{\tau\bar{B}R/K - 1 \mod N^2}{\tau\bar{B}'R'/K' - 1 \mod N^2}$$

Output: Each $P_i \leftarrow \sum_{i=1}^{n} x_i / \sum_{i=1}^{n} a_i$.

4.3.1 Security analysis

The security analysis considers only the semi-honest model. The initial setup of the BCP cryptosystem is presented in Step 1. After which, each party P_i generates its keys, whereby no private information is exchanged. Server C simply calculates the $Prod.pk$, and returns it to each party in Step 3. In Steps 4 and 5, each party encrypts its private data and recalculates them with random numbers, whereby no private information is exchanged either. Server then multiplies these encrypted data. Server C does not have the master key and there being no collusion between server C and S, C cannot obtain any sensitive information from the encrypted data. When $\bar{A}$ and $\bar{A}'$ are sent to each party, they cannot decrypt them and obtain any sensitive information. In Step 8, server C simply calculates K and K' and sends them to each party. Therafter, each party can obtain the result, but cannot obtain other parties' private information.

4.4 Outsourced secure set intersection protocol

In this section, an outsourced secure set intersection protocol (OPPWAP) is proposed as follows:

Secure set intersection problem (SSIP) [31] can be defined as follow:

Definition 2 Suppose there are n parties $\{\mathrm{P}_i \mid 1 \leq i \leq n\}$. Each party P_i has set V_i chosen from a common global universe. The problem is to securely compute the size of the intersection set $|\bigcap_{i=1}^{n} V_i|$.

We modify it for the outsourcing computation in our scheme as OSSIP. This scheme is presented as follows:

4.4.1 The subprotocol of KeyProd

The purpose of this protocol is to transform the encryptions of all the participating clients P_1,...,P_n into encryptions under a single public key called $Prod.pk = \prod_{i=1}^{n} pk_i \mod N^2$ in which the underlying plaintexts of all involved public keys are unchanged.

1. C stores $((A_i, B_i), pk_i)$, $i = 1, ..., n$, and picks a random number τ_i for every input ciphertext (A_i, B_i). It then calculates $(C_i, D_i) \leftarrow Add((A_i, B_i), Enc_{pk_i}(-\tau_i))$, and sends all the $((C_i, D_i), pk_i)$ to S.
2. S decrypts them using the master key to obtain $z_i \leftarrow mDec_{(pk_i,MK)}(C_i, D_i)$, and calculates $Prod.pk = \prod_{i=1}^{n} pk_i \mod N^2$. It then re-encrypts each (C_i, D_i) with $(W_i, Z_i) \leftarrow Enc_{Prod.pk}(\prod_{i=1}^{n} z_i)$, and sends (W_i, Z_i), $i = 1, ..., n$ back to C.
3. C calculates $(\bar{A}_i, \bar{B}_i) \leftarrow ((W_i, Z_i), Enc_{Prod.pk}(-\tau_i))$ and outputs the ciphertexts $(\bar{A}_i, \bar{B}_i)$.

Algorithm 2: OSSIP

Input: P_i's $(1 \leq i \leq k-1)$ input is a vector $V_i = \{x_{i1}, x_{i2}, ..., x_{in}\}$, while P_k's class vector $T = \{c_1, c_2, ..., c_n\}$.
Output: SN, The set intersection number.
1. P_i performs the following:
(a) chooses the BCP homomorphic encryption scheme and generates its keys.
(b) uses interpolation to compute the coefficients of the polynomial $P(z) = \sum_{i=1}^{k} a_i z^i$ such that $P(x_i) = 0$ for all x_i; and
(c) encrypts each of the n coefficients that uses the key of $\mathrm{pk_i}$ as $\{Enc_{pk_i}(a_i)\}$, and uses the $KeyProd$ protocol to transform ciphertexts under the key of Prod.pk, then sends the cloud server the resulting set of $\{Enc_{Prod.pk}(a_i)\}$.
2. P_k performs the following:
(a) uses the key of Prod.pk to evaluate the encrypted data of $\{Enc_{Prod.pk}(c_i^j) | 1 \leq j \leq k\}$ and send them to the cloud server C.
3. The cloud server C computes the polynomial at $Enc_{Prod.pk}(SN) = \prod_{i=1}^{n} Enc_{Prod.pk}(P(c_i) + c_i) = \prod_{i=1}^{n} [Enc_{Prod.pk}(\sum_{j=1}^{k} a_j c_i^j) \cdot Enc_{Prod.pk}(c_i)]$ using the Add protocol and the $Mult.$ protocols. It uses use the $TransDec$ protocol to transform it back to n encryptions of the same plaintext under each different client's public key pk_i, then sends the result back to the client.
4. P_i can decrypt all these ciphertexts received by using its corresponding private key sk_i, and can obtain the final result.

4.4.2 The subprotocol of Add

The underlying BCP cryptosystem is additively homomorphic for encryptions under the same public key. Hence, all the client's encryption inputs are encrypted under the same public key and can easily be an additive if we compute all the data under product $Prod.pk$.

4.4.3 The subprotocol of Mult

However, a multiplication gate must be interactively computed with server S as follows:

(1) C picks random numbers ρ_1 and ρ_2 for the input ciphertexts (A_1, B_1) and (A_2, B_2) under Prod.pk. It then calculates $(C_i, D_i) \leftarrow Add((A_i, B_i), Enc_{Prod.pk}(-\rho_i))$, $i = \{1, 2\}$ and sends them to S.
(2) S decrypts them using the master key and obtains $z_i \leftarrow mDec_{(Prod.pk,MK)}(C_i, D_i)$, it then re-encrypts them with $(Z_1, Z_2) \leftarrow Enc_{Prod.pk}(\prod_{i=1}^{n} z_i)$. And sends Z_1, Z_2 back to C
(3) C calculates $(T_1, T_2) \leftarrow Enc_{Prod.pk}(-\rho_1\rho_2)$ and can obtain the multiplication result as $(\bar{A}, \bar{B}) \leftarrow (Z_1 A_1^{\rho_2} A_2^{\rho_1} T_1 \mod N^2, Z_2 B_1^{\rho_2} B_2^{\rho_1} T_2 \mod N^2)$.

4.4.4 The subprotocol of TransDec

The task of subprotocol *TransDec* is to take the result encrypted under *Prod.pk* and transform it back to n encryptions of the same plaintext.

(1) C picks a random number ρ for the input ciphertext (A, B) under Prod.pk. It then calculates $(C, D) \leftarrow Add((A, B), Enc_{Prod.pk}(\rho))$, and sends them to S.
(2) S decrypts them using the master key and obtains $z \leftarrow mDec_{(Prod.pk,MK)}(C, D)$, it then re-encrypts them with $(X_i, Y_i) \leftarrow Enc_{pk_i}(z)$ under each different client's public key pk_i. And sends (X_i, Y_i) back to C.
(3) C calculates $Enc_{pk_i}(-\rho)$ and obtains the following result:

$$(\bar{A}, \bar{B}) \leftarrow Add((X_i, Y_i), Enc_{pk_i}(-\rho))$$

It then outputs the result to each party.

4.4.5 Security analysis

Subprotocols Add, $Mult$, and $TransDec$ were proven secure under the semi-honest model in [22].

In the OSSIP protocol, after generating the public parameters, each party P_i generates its keys and computes the coefficients of the polynomial $P(z)$. Each then encrypts the n coefficients and sends them to the cloud. No private information is exchanged in this step. Subsequently, P_k encrypts its class vector using $Prod.pk$ and sends them to cloud server C. C does not have the master key. Therefore, it cannot decrypt the ciphertext. After which, server C uses subprotocols *KeyProd*, $Mult$, and Add to calculate the ciphertext of $Enc_{Prod.pk}(SN)$. Subprotcols Add, $Mult$, and *TransDec* were proven secure in [22]. Hence, we hold the opinion that Step 3 is secure. C uses *TransDec* to transform them back into the encryption under each P_i's public key. Each P_i can use its private key to decrypt the encrypted data and obtain the result. Each party cannot obtain other parties' private information because *TransDec* is secure, and each party only obtains a part of the whole ciphertext. Thus, we hold the opinion that the OSSIP protocol is secure under the semi-honest model.

5 Outsourced privacy preserving C4.5 protocol

In this section, the outsourced privacy-preserving C4.5 algorithm over horizontally partitioned data and vertically partitioned data algorithms are proposed based on the OPPWAP in Sect. 4.3 and OSSIP in Sect. 4.4.

5.1 Decision tree and the C4.5 algorithm

C4.5 [23] is an extension of the ID3 algorithm [24] used to generate decision trees. C4.5 produces a decision tree from an example set in a topdown manner. At each node of the tree, C4.5 uses the **information gain ratio** to choose the data attribute that most effectively splits its set of samples into subsets. The attribute with the highest **information gain ratio** is selected as the best attribute.

The information gain ratio of an attribute A is defined as follows:

$$GainRatio(S, A) = \frac{Gain(S, A)}{SplitI(S, A)} \tag{2}$$

$$Gain(S, A) = Entropy(S) - \sum_{v \in A} \frac{|S_v|}{|S|} Entropy(S_v) \tag{3}$$

$$SplitI(S, A) = -\sum_{v \in A} \frac{|S_v|}{|S|} \log_2 \frac{|S_v|}{|S|} \tag{4}$$

where S_v is the subset of S with tuples having value v for attribute A.

5.2 Outsourced privacy-preserving C4.5 algorithm over horizontally partitioned data

The parties for the horizontally distributed dataset shared the set of general attributes $A = A_1 \cup A_2 \cup ... \cup A_n$ and the class attribute $C = c_1, ..., c_m$. However, the number of records in the databases $|S_i|$ and the true record values $v_i \in A$ are unknown to the other parties. The whole protocol contains the six following steps:

1. All the parties jointly compute the information gain $Gain(S, A)$ and $SplitI(S, A)$ for each attribute $A_i \in A$. The C4.5 algorithm only needs to determine the name of the attribute that maximizes $Gain(S, A)$, as presented in Eq. 5. To this end, two key components exist: $Entropy(S)$ and $Entropy(S_v)$.

(1) Each party P_i calculates the value of $Entropy(S)$.

$$Entropy(S) = \sum_{k=1}^{d} -(\frac{|S(c_k)|}{|S|}) \log_2(\frac{|S(c_k)|}{|S|}). \tag{5}$$

Eq. 5 suggests that $Entropy(S)$ can be computed if the $|S(c_k)|$ and $|S|$ values are known. Note that in this case, $|S(c_k)|$ and $|S|$ should be computed over all data

sources. Then Eq. 5 can be denoted as:

$$Entropy(S) = \sum_{k=1}^{d} -\left(\frac{\sum_{i=1}^{n} |S(c_k)|_i}{\sum_{i=1}^{n} |S|_i}\right) \cdot \log_2\left(\frac{\sum_{i=1}^{n} |S(c_k)|_i}{\sum_{i=1}^{n} |S|_i}\right). \tag{6}$$

From that point, each party can count the number of $|S(c_k)|_i$, $|S|_i$ in its individual database. Thus, Eq. 6 can be written in the format of

$$\frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} a_i} \log_2 \frac{\sum_{i=1}^{n} x_i}{\sum_{i=1}^{n} a_i} \tag{7}$$

and the calculation of the logarithmic operation can be performed in each party's database. To be calculated is the value of $\sum_{i=1}^{n} x_i / \sum_{i=1}^{n} a_i$ which can be easily determined by utilizing our OPPWAP protocol as in Sect. 4.3. Then all the parties can independently calculate the value of $Entropy(S)$.

(2) Each party P_i calculates the value of $Entropy(S_v)$.

$$Entropy(S_v) = \sum_{k=1}^{d} -\frac{\sum_{i=1}^{n} |S_v(c_k)|_i}{\sum_{i=1}^{n} |S_v|_i} \cdot log_2\left(\frac{\sum_{i=1}^{n} |S_v(c_k)|_i}{\sum_{i=1}^{n} |S_v|_i}\right). \tag{8}$$

The value of $Entropy(S_v)$ is then calculated by utilizing our OPPWAP protocol, as presented in Sect. 4.3.

(3) Each party P_i calculates the value of $SplitI(S, A)$, which is calculated as Eq. 8:

$$SplitI(S, A) = \sum_{k=1}^{d} -\frac{\sum_{i=1}^{n} |S_v|_i}{\sum_{i=1}^{n} |S|_i} \cdot log_2\left(\frac{\sum_{i=1}^{n} |S_v|_i}{\sum_{i=1}^{n} |S|_i}\right). \tag{9}$$

Eq. 8 can be written in the $(\sum_{i=1}^{n} x_i / \sum_{i=1}^{n} a_i)$ format, which can be determined by utilizing our OPPWAP protocol.

After which, each party can obtain the corresponding conditional entropy value for each attribute A. In addition, the value of $|S_v|/|S|$ can be written as $\sum_{i=1}^{n} |S_v|_i / \sum_{i=1}^{n} |S|_i$, which can also be calculated by utilizing our OPPWAP protocol. Moreover, each of them can calculated the value of $Gain(S, A)$ for each attribute.

3. Each party can identify the attribute with the highest information gain ratio. It can then create a new node and set that attribute as the root node.
4. Each party is notified of the result and can thus they partition their data accordingly.
5. We then return to Step 1 and recursively construct the other nodes of the tree in a similar manner.

5.2.1 Security analysis

In our privacy-preserving C4.5 protocol over the horizontally partitioned data, each party can count the number of $|S(c_k)|_i$, $|S|_i$, $|S_v(c_k)|_i$ and $|S_v|_i$ in its individual database. The party then calculates the average value as compared to those of the other parties by utilizing our OPPWAP protocol. No other information is exposed to the other parties during this process. Our OPPWAP protocol can be secure in the semi-honest model, and no party obtains any information about the other's input, except the output value. Thus, our privacy-preserving C4.5 protocol proved security in the semi-honest model.

5.3 Outsourced privacy preserving C4.5 algorithm over vertically partitioned data

For the vertically distributed data set, each party P_i has a database S_i, $(1 \leq i \leq n)$. We note the whole database as S, $S = S_1 \cup \cdots \cup S_n$. The parties share the number of records in the databases $|S_i|$ and the same order of the records. However, they hold different sets of attributes A_i for the same set of objects. Moreover, class attribute C is known only by one party. The whole protocol contains the six following steps:

1. We first check if R is empty. This can be calculated by the Add protocol, as presented in Sect. 4.4. Each party P_i has a flag AR_i (i.e., $AR_i = 0$ if there are no remaining attributes, and $AR_i = 1$ if P_i has remaining attributes). We then check if the sum of AR_i is equal to zero. An "equal to zero" result implies no remaining attribute.
2. All the parties jointly compute the information gain $Gain(S, A)$ and $SplitI(S, A)$ for each attribute $A_i \in A$. As presented in Eq. 7, $Entropy(S)$ and $Entropy(S_v)$ can be computed if the $|S(c_k)|$, $|S|$, $|S_v(c_k)|$ and $|S_v|$ values are known. However, in the vertically partitioned dataset, the value of $|S|$ can be known to each party. The key information we need to calculate when only one party knows the class are $|S_v(c_k)|$. This is accomplished with a OSSIP, as explained in Sect. 4.4.

We formalize this by introducing the notion of a constraint vector. Each of these vectors corresponds to a particular value

of A. For instance, $A = a_i$. Hence, the vector for the dataset of P_i can be denoted as $V_{a_i} = \{x_1, x_2, ..., x_m\}$, where m is the number of transactions in the dataset of P_i:

- $x_i = 1$ if (i) attribute A's value in the i-th transaction is held by the other parties or (ii) attribute A's value is a_i and is held by P_i.
- $x_i = 0$ if attribute A's value of held by P_i and is not a_i.

From that point, each party can independently obtain the values of $|S_v(c_k)|$ and calculate the value of $Entropy(S_v)$. Moreover, each one can obtain the values of $Gain(S, A)$ and $SplitI(S, A)$ using the protocol of Add. Each party can then independently calculate the **highest information gain ratio**.

3. Each party can identify the attribute with the highest information gain ratio, create a new node, and set that attribute as the root node.
4. Each party is notified of the result and can, thus, partition their data accordingly.
5. We then return to Step 1 and recursively construct the other nodes of the tree in a similar manner.

5.3.1 Security analysis

Each party in our privacy-preserving C4.5 protocol over the vertically partitioned data has the value of $|S|_i$ and $|S_v|_i$. The party then formalizes the constraint vector in its individual database and calculates the set intersection values with those of the other parties by utilizing our OSSIP protocol, which can be obtained as the values of $|S(c_k)|_i$ and $|S_v(c_k)|_i$. Subsequently, each party can calculate the value of $Entropy(S_v)$, $Entropy(S)$ and $SplitI(S, A)$ using the protocol of Add. After which, they can obtain the highest information gain ratio by themselves. No other information is exposed to the other parties during this process. Our OSSIP protocol can be secure in the semi-honest model, and no party obtains any information about the others input, except the output value. In other words, our privacy-preserving C4.5 protocol can obtain security in the semi-honest model.

6 Performance analysis

6.1 Complexity and communication analysis

The performance of our contribution depends on security parameter k, number of clients n, number of additions and multiplications performed, and network performance. The data owners should encrypt their own data before computing. They should also transfer the data to the server to compute. The server then needs to compute the encrypted data. We suppose that n parties and m data objects exist in each iteration.

Table 2 Complexity and runtime analysis of the protocol

Algorithm	Time	Traffic in bits
Setup	$O(k^5/\log k^2)$ on S	$4k$
Add	$O(k^2)$ on C	$2k$
Mult	$O(k^3)$ on C	$12k$
	$O(k^3)$ on S	
TransDec	$O(nk^3)$ on C	$4(n+1)k$
	$O(nk^3)$ on S	
OPPWAP	$O(k^3)$ on client	$4(n+1)k$
	$O(nk^3)$ on C	
	$O(nk^5/\log k^2)$ on S	
OSSIP	$O(k^3)$ on client	$6(n+1)k$
	$O(tk^2)$ on C	
	$O(tk^3)$ on S	
PPC4.5 over	$O(tk^2)$ on client	$12(n+1)k$
Horizontal dataset	$O(nmtk^3)$ on C*	
	$O(nmk^3)$ on S	
PPC4.5 over	$O(tmk^3)$ on client	$18(n+1)k$
Vertical dataset	$O(nm^2t^2k^2)$ on C	
	$O(ntk^3)$ on S	

Each data object has t different attributes. The setup step will do the computation for $O(k^5/\log k^2)$. Each party and cloud server in our PPC4.5 over the horizontally partitioned data algorithm will use the OPPWAP protocol to compute $(\sum_{i=1}^{n} x_i / \sum_{i=1}^{n} a_i)$ for the $n*m$ times. Moreover, each party and cloud server in our PPC4.5 over the vertically partitioned data will use the OSSIP protocol to compute $Entropy(S_v)$, $Entropy(S)$ and $SplitI(S, A)$ for $n*t$ times. Table 2 shows a full overview of the complexity of each protocol.

6.2 Performance evaluation

We implemented our method for privacy-preserving C4.5 decision trees over horizontally and vertically partitioned data for multiple parties using Java in a prototype. In this section, we evaluate the performance of our method by comparing the computation costs for the client and the server provider.

We implemented our OPPWAP and OSSIP protocols in C++. The speed of the implementation mainly depends on the speed of the bignum library. Hence, we used the GMP library. The experiments were conducted with the client and the server on a Dell XPS 430 running Windows 8.1, with an Intel Core 2 Duo CPU having a 4 GB RAM and a 2.33 GHz speed.

We initialized the BCP cryptosystem at equivalent levels of security with modules having a size of 1024 bits. And we obtained 54.639 s as the average key generation time for

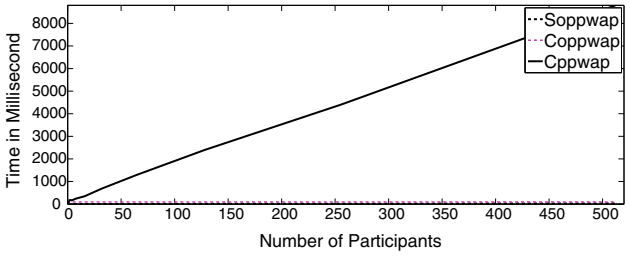


Fig. 4 Performance measurements for our OPPWAP and PPWAP with different numbers of participants

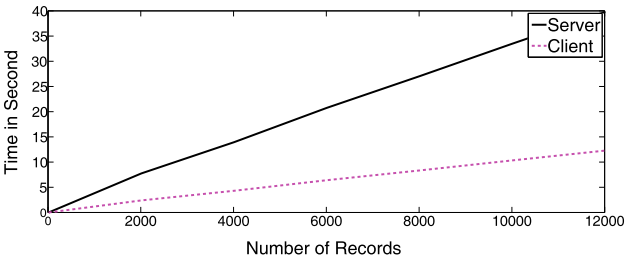


Fig. 5 Performance measurements for our OPPC4.5 over horizontally partitioned data with different numbers of records

the cloud. We then measured our OPPWAP with different numbers of participants and run all clients and both servers on the same host, such that the network latency can be ignored. Then we compare with the PPWAP [9] based on the Paillier crytosystem (with the same security parameter as BCP, 1024 bits) without any outsourcing computation. Figure 4 shows the client time cost and the cloud time cost.

Figure 4 depicts an obvious increase on the time cost of the cloud server when the number of participants increased. However, the client's time cost was much more stable because most of the calculations in the OPPWAP protocol were conducted on the cloud side. Hence, the time cost of the cloud linearly increased with the number of participants. The client's time cost will be larger than that of the cloud when the number of participants becomes less than 2000. However, a significant increase for the cloud side was observed when the number of participants increased. Meanwhile, no significant increase for the client side was found as the number of participants increased because there are only the encrypt and decrypt operations in the client side.

We also implemented our OPPC4.5 over the horizontally partitioned data. These experiments were performed with 1024-bit secret keys, 20 participants, and various dataset sizes. Fig. 5 shows our results (Table 3).

We then measured our OSSIP with different numbers of records, also compare with SSIP [31] based on the Paillier cryptosystem. The results are as shown in Fig. 5.

Figure 6 illustrates an obvious increase on the time cost of the cloud server when the number of records increased. However, the time cost of the clients much slowly increased.

Table 3 Time cost of the OPPWAP and PPWAP protocols

Number of parties	C_{OPPWAP} (ms)	S_{OPPWAP} (ms)	C_{PPWAP} (ms)
2	94	2	181
4	95	3	169
8	93	2	246
16	94	3	348
32	95	3	687
64	94	7	1282
128	95	11	2394
256	96	17	4412
512	95	32	8797

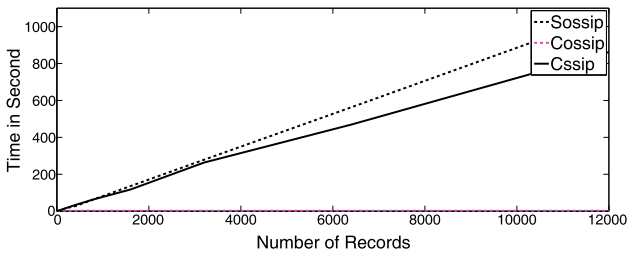


Fig. 6 Performance measurements for our OSSIP and the SSIP with different numbers of records

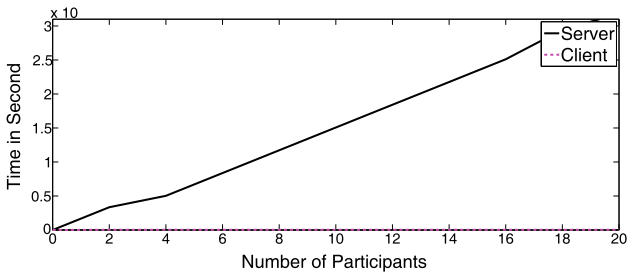


Fig. 7 Performance measurements for our OPPC4.5 over vertically partitioned data with different participants

The cloud's time cost for the OSSIP was also larger than that in OPPWAP because there were much more encrypt and decrypt operations in the cloud side. In practical application, not so many attributes exist for the records. Hence, the number of participants was limited. We can utilize some parallel processing architectures, such as Hadoop or Spark, or use the GPU to reduce the time cost.

We also implemented our OPPC4.5 over the vertically partitioned data. These experiments were performed with 1024-bit secret keys, the sizes of datasets of 3600. And each participant have two attribute, every attribute has three different attribute values. Figure 7 shows our results, from which we can prove that a sublinear relationship exists between the computational cost of the user side and the number of participating parties (Table 4).

Table 4 Time cost of the OSSIP and SSIP protocols

Number of records	C_{OSSIP} (s)	S_{OSSIP} (s)	C_{SSIP} (s)
200	2.13	16.76	16.76
400	2.07	27.04	33.50
800	2.14	63.8	64.24
1600	2.09	135.95	117.49
3200	2.10	278.90	263.65
6400	2.17	562.94	469.23
12800	2.24	1136.51	918.42

7 Conclusion

We have presented two efficient algorithms for generating two outsourced privacy-preserving C4.5 decision tree algorithms over horizontally and vertically partitioned data, respectively, for multiple parties. In addition, we concluded that the intensive computing tasks can be performed via outsourcing to cloud server.

We emphasize that, although the total computation cost both on the client and server side of the proposed protocol grows, the computation at client side decreases greatly. That is exactly our aim to outsource most of the computation to server side. To further decrease the computation cost at server side, parallel computation techniques should be deployed, as one of the future work.

In addition, we intend to extend our algorithm to arbitrary partitioning in the presence of a semi-honest model. We can also extend all of the three partition datasets in the presence of a malicious model. We are finding our way to extending the OPPWAP and OSSIP protocols into a general multi-party, privacy-preserving framework suitable for other useful schemes, like random decision tree, Bayes, SVM, secure deduplication [11] and forgery detection scheme [13]. Our solution can be used in the health-care filed, for example, there are many companies that provide automatic medical assessments and risk profiles for various diseases by evaluating a user's responses to an online questionnaire, or by analyzing a user's DNA profile [36]. Or in the finance area, some banks who want to make deep analysis of their data, but concerns the privacy of the data.

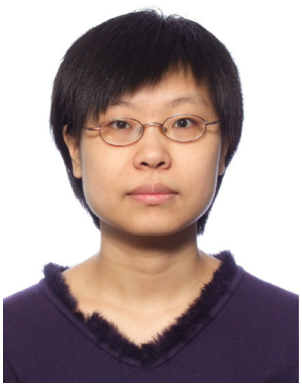
Acknowledgements This work is supported by National High Technology Research and Development Program of China (No. 2015AA016 008), National Natural Science Foundation of China (No. 61402136), Natural Science Foundation of Guangdong Province, China (No. 2014A030313697), National Natural Science Foundation of China (No. 61472091), Natural Science Foundation of Guangdong Province for Distinguished Young Scholars (2014A030306020), Guangzhou scholars project for universities of Guangzhou (No. 1201561613), Science and Technology Planning Project of Guangdong Province, China (2015B01012 9015) and Guangdong Province Key Laboratory of High Performance Computing (No. [2013]82).

References

1. A. Yao.: How to generate and exchange secrets. In: Proceedings of Annual Symposium on Foundations of Computer Science, pp. 162–167 (1986)
2. Bresson, E., Catalano, D., Pointcheval. A simple public-key cryptosystem with a double trapdoor decryption mechanism and its applications. In: Advances in Cryptology—ASIACRYPT 2003, Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30–December 4, 2003, vol. 2894, pp. 37–54. (2003)
3. Liu, D., Bertino, E., Yi, X.: Privacy of outsourced K-means clustering. In: Proceedings of ACM Symposium on Information, Computer and Communications Security, pp. 123–134. (2014)
4. Elgamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: CRYPTO 1984: Proceedings of Advances in Cryptology, pp. 10–18. (1985)
5. Emekci, F., Sahin, O.D., et al.: Privacy preserving decision tree learning over multiple parties. Data Knowl Eng **63**(2), 348–361 (2007)
6. Fu, Z., Huang, F., Sun, X., et al.: Enabling semantic search based on conceptual graphs over encrypted outsourced data. IEEE Trans. Serv. Comput. (2016). doi:10.1109/TSC.2016.2622697
7. Gangrade, A., Patel, R.: Building privacy-preserving C4.5 decision tree classifier on multi-parties. Int. J. Comput. Sci. Eng. **1**(3), 199–205 (2009)
8. Gupta, B.B., Agrawal, D.P., Yamaguchi, S.: Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security. IGI Global Publisher, Hershey (2016)
9. G. Jagannathan, Wright, R.N.: Privacy-preserving distributed Kmeans clustering over arbitrarily partitioned data. In: Proceedings of ACM International Conference on Knowledge Discovery, pp. 593–599. (2005)
10. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Lecture Notes in Computer Science, vol. 3378, pp. 264–282. (2005)
11. Li, Jin, Chen, Xiaofeng, Li, Mingqiang, Li, Jingwei, Lee, Patrick, Lou, Wenjing: Secure deduplication with efficient and reliable convergent key management. IEEE Trans. Parallel Distrib. Syst. **25**(6), 1615–1625 (2014)
12. Li, Jin, Li, Jingwei, Chen, Xiaofeng, Jia, C., Lou, W.: Identity-based encryption with outsourced revocation in cloud computing. IEEE Trans. Comput. **64**(2), 425–437 (2015)
13. Li, Jian, Li, Xiaolong, Yang, Bin, Sun, Xingming: Segmentation-based image copy-move forgery detection scheme. IEEE Trans. Inf. Forensics Secur. **10**(3), 507–518 (2015)
14. Zhan, J., Matwin, S., et al.: Privacy preserving decision tree classification over horizontally partitioned data. In: Proceedings of International Conference on Electronic Business, pp. 470–476. (2005)
15. Kamara, S., Mohassel, P., Raykova, M.: Outsourcing multi-party computation. IACR Cryptol. Eprint Arch. **2011**(3), 435–451 (2011)
16. Keonsoo, L., et al.: A comparative evaluation of atrial fibrillation detection methods in Koreans based on optical recordings using a smartphone. In: IEEE Access. (2017). doi:10.1109/ACCESS.2017.2700488
17. Li, J., Yan, H., Liu, Z., et al.: Location-sharing systems with enhanced privacy in mobile online social networks. IEEE Syst. J. **99**, 1–10 (2015)
18. Malek, M.S.B.A., Ahmadon, M.A.B., Yamaguchi, S., et al.: On privacy verification in the IoT service based on PN2. In: Global Conference on Consumer Electronics, 2016 IEEE. (2016)
19. Xiao, M., Huang, L., et al.: Privacy preserving ID3 algorithm over horizontally partitioned data. In: Proceedings of International Con-

ference on Parallel and Distributed Computing, Applications and Technologies, pp. 239–243. (2005)
20. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: EUROCRYPT 1999 Proceedings, pp. 223–238. (1999)
21. Lory, P.: Enhancing the efficiency in privacy preserving learning of decision trees in partitioned databases. In: Proceedings of International Conference on Privacy in Statistical Databases, pp. 322–334. (2012)
22. Peter, A., Tews, E., Katzenbeisser, S.: Efficiently outsourcing multiparty computation under multiple keys. IEEE Trans. Inf. Forensics Secur. **8**(12), 2046–2058 (2013)
23. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, Burlington (1993)
24. Quinlan, J.R.: Induction of decision trees. Mach. Learn. **1**(1), 81–106 (1986)
25. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 439–450. (2000)
26. Samet, S., Miri, A.: Privacy preserving ID3 using Gini index over horizontally partitioned data. In: Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, pp. 645–651. (2008)
27. Shen, Y., Shao, H., Yang, L.: Privacy preserving C4.5 algorithm over vertically distributed datasets. In: Proceedings of IEEE International Conference on Networks Security, Wireless Communications and Trusted Computing, pp. 446–448. (2009)
28. Stergiou, C., Psannis, K.E., Kim, B.G., et al.: Secure integration of IoT and cloud computing. Future Gener. Comput. Syst. (2016)
29. Vaidya, J., Clifton, C.: Privacy-preserving decision trees over vertically partitioned data. In: Lecture Notes in Computer Science, vol. 2, pp. 139–152. (2005)
30. Vaidya J, Clifton C.: Privacy preserving association rule mining in vertically partitioned data. In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 639–644. (2002)
31. Vaidya, J., Clifton, C.: Secure set intersection cardinality with application to association rule mining. J. Comput. Secur. **13**(4), 593–622 (2005)
32. Vaidya, J., Shafiq, B., Fan, W., et al.: A random decision tree framework for privacy-preserving data mining. IEEE Trans. Dependable Secur. Comput. **11**(5), 399–411 (2014)
33. Veluru, S., Gupta, B.B., Rahulamathavan, Y., et al.: Privacy preserving text analytics: research challenges and strategies in name analysis. In: Handbook of Research on Securing Cloud-Based Databases with Biometric Applications, pp. 67–92 (2015)
34. Wang, Z., Gu, T. and Cheung, S.: A theoretical framework for distributed secure outsourced computing using secret sharing. In: Proceedings of IEEE International Workshop on Information Forensics and Security. (2014)
35. Fang, W., Yang, B.: Privacy preserving decision tree learning over vertically partitioned data. In: Proceedings of IEEE International Conference on Computer Science and Software Engineering, pp. 1049–1052. (2008)
36. Wu, D.J., Feng, T., Naehrig, M., et al.: Privately evaluating decision trees and random forests. In: Proceedings on Privacy Enhancing Technologies, vol. (4). (2016)
37. Liu, X., Jiang, Z.L., Yiu, S.M., Wang, X..: Outsourcing two-party privacy preserving K-means clustering protocol in wireless sensor networks. In: Proceedings of International Conference on Mobile Ad-Hoc and Sensor Networks, pp. 124–133. (2015)
38. Xiao, M.J., Han, K., Huang, L.S., et al.: Privacy preserving C4.5 algorithm over horizontally partitioned data. In: Proceedings of International Conference on Grid and Cooperative Computing, pp. 78–85. (2006)
39. Jararweh, Y., Alsmirat, M., Al-Ayyoub, M., et al.: Software defined system support for enabling ubiquitous mobile edge computing. Comput. J. Oxf. (2017). doi:10.1093/comjnl/bxx019
40. Lindell, Y., Pinkas, B.: Privacy preserving data mining. J. Cryptol. **15**(3), 177–206 (2002)
41. Lindell, Y., Pinkas, B.: Secure multi-party computation for privacy-preserving data mining. J. Privacy Confid. **25**(2), 59–98 (2009)
42. Zhangjie, Fu, Xinle, Wu, Guan, Chaowen, Sun, Xingming, Ren, Kui: Toward efficient multi-keyword fuzzy search over encrypted outsourced data with accuracy improvement. IEEE Trans. Inf. Forensics Secur. **11**(12), 2706–2716 (2016)
43. Zhangjie, Fu, Ren, Kui, Shu, Jiangang, Sun, Xingming, Huang, Fengxiao: Enabling personalized search over encrypted outsourced data with efficiency improvement. IEEE Tran. Parallel Distrib. Syst. **27**(9), 2546–2559 (2016)
44. Xia, Zhihua, Wang, Xinhui, Sun, Xingming, Wang, Qian: A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. IEEE Trans. Parallel Distrib. Syst. **27**(2), 340–352 (2015)

Ye Li is a Ph.D. student in the School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China. His research interests include cryptography, computer security and privacy.



Zoe L. Jiang received the Ph.D. degree from The University of Hong Kong, Hong Kong, in 2010. She is currently an Assistant Professor with School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, China. Her research interests include cryptography and digital forensics.

Lin Yao is a Ph.D., Lecturer of Computer Science and Technology School of Harbin institute of technology, Director of software engineering laboratory. She received a master's degree in computer science from University of Southern California. In 2011, she obtained a Ph.D. degree in computer science from Harbin institute of technology and worked as a post-doctoral in Peking University from 2012 to 2015. In terms of scientific research, She has been engaged

in the research of artificial intelligence, natural language processing and biological information.

Xuan Wang received his M.S. and Ph.D. degrees in Computer Sciences from Harbin Institute of Technology in 1994 and 1997 respectively. He is a professor and Ph.D. supervisor in the Computer Application Research Center, Harbin Institute of Technology Shenzhen Graduate School. His main research interests include artificial intelligence, computer vision, computer network security and computational linguistics.

S. M. Yiu received the PhD degree in computer science from the Department of Computer Science, The University of Hong Kong, in 1996. He is currently an associate professor in the same department. His research interests include information security, cryptography, and bioinformatics.

Zhengan Huang received his B.S. and M.S. degrees from Department of Mathematics, Sun Yat-sen University in 2009 and 2011, respectively, and his Ph.D. degree from Department of Computer Science and Engineering, Shanghai Jiao Tong University in 2015. He served as a security engineer in Huawei Technologies Co. Ltd. from 2015 to 2016. Currently, he is a postdoctoral researcher in Guangzhou University. His research interests include public-key cryptography and information security.