Privacy-preserving similarity coefficients for binary data

Article *in* Computers & Mathematics with Applications · May 2013

DOI: 10.1016/j.camwa.2012.02.028

2 authors:

Kok-Seng Wong
VinUniversity
61 PUBLICATIONS **211** CITATIONS
SEE PROFILE

Myung Ho Kim
Soongsil University
49 PUBLICATIONS **183** CITATIONS
SEE PROFILE

Computers and Mathematics with Applications 65 (2013) 1280–1290

Contents lists available at SciVerse ScienceDirect

Computers and Mathematics with Applications

journal homepage: www.elsevier.com/locate/camwa

Privacy-preserving similarity coefficients for binary data

Kok-Seng Wong, Myung Ho Kim *

School of Computer Science and Engineering, Soongsil University, Sangdo-Dong Dongjak-Gu, 156-743, Seoul, South Korea

ARTICLE INFO

Keywords:
Similarity coefficients
Association coefficients
Binary data similarity measures
Similarity computation
Privacy preserving similarity test

ABSTRACT

Similarity coefficients (also known as coefficients of association) are important measurement techniques used to quantify the extent to which objects resemble one another. Due to privacy concerns, the data owner might not want to participate in any similarity measurement if the original dataset will be revealed or could be derived from the final output. There are many different measurements used for numerical, structural and binary data. In this paper, we particularly consider the computation of similarity coefficients for binary data. A large number of studies related to similarity coefficients have been performed. Our objective in this paper is not to design a specific similarity coefficient. Rather, we are demonstrating how to compute similarity coefficients in a secure and privacy preserved environment. In our protocol, a client and a server jointly participate in the computation. At the end of the protocol, the client will obtain all summation variables needed for the computation while the server learns nothing. We incorporate cryptographic methods in our protocol to protect the original dataset and all other intermediate results. Note that our protocol also supports dissimilarity coefficients.

1. Introduction

Informally, a similarity coefficient can be defined as a function used to quantify the extent to which objects resemble one another. It is frequently used to study the coexistence of objects and the similarity of objects.

A large number of similarity coefficients have been proposed in relevant literature. Sokal–Michener [1], Jaccard [2,3], Sørensen [4], Dice [5], and Sokal–Sneath [6] coefficients are commonly used measurements to decide on how the similarity of two objects is calculated.

Similarity coefficients play an important role in many application domains. In the context of ecology and biogeography research, similarity coefficients are frequently used in the measurement of inter-species association analysis, particularly to study the coexistence of species and the similarity of sampling sites [7–10]. In the privacy preserving data mining (PPDM) applications such as clustering [11,12], the similarity coefficient is used to assign a set of observations or data into subsets called clusters. Recently, the similarity coefficient has also been applied in biometric areas to solve identification problems such as iris and fingerprint recognition [13].

There are many measurements used for numerical, structural (i.e., sequences and graphs), and binary data (i.e., presence or absence of the data). In this paper, we particularly consider the similarity coefficient for binary data.

Binary data has been widely used to represent a variety of data. For example, binary data were used in the medical test to indicate whether a patient is allergic or not to a particular medication; in the context of ecology and biogeography research, binary data were used to reflect whether a species of animal or plant is found or not in a certain habitat.

* Corresponding author. Tel.: +82 2 820 0915; fax: +82 2 817 8961.
E-mail addresses: kswong@ssu.ac.kr (K.-S. Wong), kmh@ssu.ac.kr (M.H. Kim).

0898-1221/$ – see front matter © 2012 Elsevier Ltd. All rights reserved.
doi:10.1016/j.camwa.2012.02.028

Table 1
Contingency table.

	1	0	Total
1	a	b	$a+b$
0	c	d	$c+d$
Total	$a+c$	$b+d$	n

1.1. Motivations and contributions

The motivations behind our work in this paper are based on several considerations. Summation variables (as defined in Section 2.1) for binary data are being used in most similarity coefficients. In order to provide a secure and privacy preserved computation, these variables should be computed without the leakage of the original dataset.

To the best of our knowledge, no similar protocol has been proposed to compute summation variables for binary data in a privacy preserved environment. With these summation variables, coefficients such as similarity and dissimilarity can be computed easily. In this paper, we only consider the measurement of similarity coefficients, but, the same summation variables can be used for dissimilarity measurements.

Our protocol allows the client to compute more than one similarity coefficient from a single protocol. The client is able to compute different similarity coefficients without modifying the protocol. The server participates in the computation without learning any information at the end of the protocol.

1.2. Paper organization

The rest of this paper is organized as follows: The background and related works for this research is in Section 2 and the technical preliminaries are described in Section 3. We present our secure computation protocol in Section 4 followed by the analysis and the discussions in Section 5. Our conclusion is in Section 6.

2. Background and related works

This section describes the background for our research in this paper. In Section 2.1, we discuss the summation variables for binary data follows with a brief introduction to similarity coefficients in Section 2.2. Related works will be discussed in Section 2.3.

2.1. Summation variables for binary data

In general, binary data is a representation of the presence or absence of an attribute in the given objects. The value "1" is used to show the presence of an attribute while "0" is used to represent the absence of an attribute. Hence, a binary dataset is composed of a series of strings with "1" and "0".

For simplicity, let $X = \{x_i | i = 1, 2, \ldots, n\}$ and $Y = \{y_i | i = 1, 2, \ldots, n\}$ are two binary datasets, where $x_i,\ y_i \in \{0,\ 1\}$, and C_{XY} is defined as follows:

$$C_{XY} = \{(x_i, y_i) | i = 1, 2, \ldots, n\}. \tag{1}$$

We further specify the following summation variables:

- a is the number of (x_i, y_i) where both elements have the value "1" in C_{XY}.
- b is the number of (x_i, y_i) where x_i is "1" and y_i is "0" in C_{XY}.
- c is the number of (x_i, y_i) where x_i is "0" and y_i is "1" in C_{XY}.
- d is the number of (x_i, y_i) where both elements have the value "0" in C_{XY}.

These summation variables can be represented in a contingency table as shown in Table 1.

Note that the size of the dataset X (and Y) is given by $n = a + b + c + d$.

2.2. Similarity coefficients

The computation of similarity coefficients are based on the size of a, b, c and d (as defined in the previous section). In the relevant literature [14–16], a is known as a "positive match", b and c are known as "mismatch", and d is referred to as a "negative match".

Similarity coefficient choice is based on some criterion. An important consideration is the inclusion or exclusion of the negative match d in the computation. For some data, the absence of an element in both objects would indicate similarity, but in certain cases, this might not be true. Hence, the similarity coefficients are often categorized into two types [16,17].

The first type takes into consideration negative matches. For example, Russell and Rao [18] introduced the similarity coefficient of this type which can be expressed as follows:

$$\frac{a}{a+b+c+d}. \tag{2}$$

Table 2
Type 1 similarity coefficients (with negative matches).

No.	Similarity coefficient	Computation of $S(X, Y)$
1	Rusell–Rao Index [18] (RR)	$\frac{a}{a+b+c+d}$
2	Sokal–Michener [1] (SM)	$\frac{a+d}{a+b+c+d}$
3	Rogers–Tanimoto [23] (RT)	$\frac{a+d}{a+2b+2c+d}$
4	Yule–Kendall [24] (YK)	$\frac{ad}{ad+bc}$
5	Sokal–Sneath 1 [6] (SS1)	$\frac{a+d}{a+\frac{1}{2}(b+c)+d}$

Table 3
Type 2 similarity coefficients (without considering negative matches).

No.	Similarity coefficient	Computation of $S(X, Y)$
1	Jaccard's Index [2] (JI)	$\frac{a}{a+b+c}$
2	Sokal–Sneath 2 [16] (SS2)	$\frac{a}{a+2(b+c)}$
3	Dice [5] (SD)	$\frac{2a}{2a+b+c}$
4	Kulszynski [25] (Ku)	$\frac{1}{2}(\frac{a}{a+b}+\frac{a}{a+c})$
5	Ochiai [26] (Och)	$\frac{a}{\sqrt{(a+b)}\sqrt{(a+c)}}$

This similarity coefficient represents the proportion of positive matches in the dataset. Note that the denominator in Eq. (2) is actually the size of the dataset, n.

In the second type, negative matches will not be considered during the computation. For example, the Jaccard's coefficient [2] can be calculated as follows:

$$\frac{a}{a+b+c}. \tag{3}$$

As shown in Eq. (3), the Jaccard's coefficient is independent of the summation variable d. In the asymmetric type of binary data, the positive matches are usually more significant than the negative matches [19,20]. However, the inclusion or exclusion of negative matches in the similarity coefficients are still an ongoing issue in many research areas [6,15,16,21].

Some commonly used similarity coefficients for type 1 and type 2 are shown in Tables 2 and 3, respectively. We refer readers to [22] for a more comprehensive list (the authors compiled a list of 76 binary similarity coefficients).

In this paper, we particularly consider the similarity coefficients for binary data. But, with the correct size of each summation variable in Section 2.1, the client is able to compute dissimilarity coefficients of two datasets (i.e., X and Y). We do not discuss further about dissimilarity coefficients in this paper, but, we would like to stress that our protocol can also be applied to dissimilarity coefficients computations.

2.3. Related works

Various procedures and protocols for finding the similarity (or homogeneity) of two or more datasets have been proposed in the relevant literature. Private matching is a practical problem that is used to determine common entries for two or more inputs. In private matching, two or more parties want to find common data from the joint databases without revealing any private information to any party. The general approach was studied by Agrawal, Evfimievski, and Srikant in [27] which has motivated many researchers to find efficient solutions to address the private matching problem.

Private equality test (PET) is a scheme used to compare two or more inputs (from two or more parties) without leaking extra information to any party [28,29]. The test result is "1" if and only if all inputs are equal and "0", otherwise. The equality test problem is a part of a class of problems where two or more parties, who have varying degrees of mutual trust, hold secrets.

Similarity search in large databases for a variety of applications such as medical history database attracts significant attention from researchers and practitioners in view of privacy protection. Private similarity search (PSS) has been proposed to protect the privacy for both querier and the database owner without revealing any information [30,31]. In PSS, the similarity is determined by computing the distance function (score) between the query and the database [31]. Similarity-based text or document retrieval is another important research area which requires the computation of the similarity coefficient [32,33]. Other research areas such as network verification [34] and trusted computing [35] also can benefit from the similarity computation.

The most related work to our protocol design in this paper is the privacy preserving scalar product method which is used to securely compute the size of intersection between two binary datasets. It is viewed as an important method in most privacy preserving data mining operations [36,37]. The output of the scalar product only reveals the number of (x_i, y_i) where

both elements have the value "1" (in our protocol, this is the summation variable a). Other than this, no other information can be gained from the said protocol. Hence, it is not suitable to be used in our protocol.

There is no solution that has been proposed previously to compute more than one similarity coefficient under a single protocol. All the previously mentioned solutions are designed for solving a specific similarity coefficient. They cannot be used to compute another coefficient using the same protocol.

3. Technical preliminaries

In this section, we describe some technical preliminaries for our protocol design.

3.1. Problem definition

In this paper, we particularly consider two-party (client–server) similarity measurements on binary data under distributed environment. We defined the problem for distributed similarity measurement as follows.

Let $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ be two binary datasets belonging to a client and a server, respectively. We assume that the client wants to measure the similarity between dataset X and Y. However, the client does not want to reveal X to the server. At the same time, the server is willing to participate in the computation if Y will not be revealed to any party and no extra information could be derived from the final output. In order to compute any of the similarity coefficients in Tables 2 and 3, the client must obtain all the summation variables (a, b, c, and d) from dataset X and Y. Hence, our objective is to output all the summation variables to the client without compromising the data privacy of either party.

In this paper, we assume that the client and the server are two semi-honest parties. In the semi-honest model, both parties follow the prescribed action in the protocol, but might attempt to discover extra information from the intermediate results during the computation.

We do not consider a fully-honest model in our protocol because no computation protocol is able to prevent a participating party from changing its input before a protocol begins, or aborting after it receives the desired results [38]. Furthermore, no party can prevent another party from performing additional operations to learn extra information from the intermediate results.

3.2. Privacy and security definition

In an ideal computation, the client and the server send their original dataset to a trusted third party (TTP) for the summation variables computation. Then, the TTP broadcasts the value of summation variables to the client without revealing any extra information to either party. Under the secure two-party computation, all computations were carried out by only two parties without the involvement of others. The formal objective for the two-party computation is to perform secure computations in the absence of a TTP.

In our protocol, only the client is allowed to learn the final output. Hence, we require that the server cannot distinguish between the different data used by the client in the computation. At the end of the protocol, the server should learn nothing. As compared to the ideal computation, it is essential to ensure that the client will not learn any extra information besides the final outputs. Without the TTP, the client learns the same information as in the ideal computation.

In a privacy concerned environment, the original dataset used in the computation should not be revealed to any party before encryption. Only the data owner has access to the original dataset. The final outputs need to be protected such that no original dataset can be derived from it.

In terms of security, the protocol is secure if both parties participate in the protocol without colluding with external parties.

3.3. Cryptographic primitives

The primary cryptographic tool we use to construct our protocol is the homomorphic semantically-secure scheme (i.e., ElGamal encryption Scheme [39]). Let $E_{pk}(m)$ denote the encryption of m with an encryption key, E_{pk}. Given two ciphertexts $E_{pk}(m_1)$ and $E_{pk}(m_2)$, there exists an efficient algorithm $\cdot_h$ to compute

$$E_{pk}(m_1) \cdot_h E_{pk}(m_2) = E_{pk}(m_1 \cdot m_2). \tag{4}$$

Note that Eq. (4) can be performed without the knowledge of the decryption key. In this paper, we will utilize this multiplicative property in our protocol.

3.4. Notations used

In Table 4, we summarize all the notations used hereafter in this paper.

Table 4
Common notations used.

Notation	Definition
X	Binary dataset from the client
Y	Binary dataset from the server
x_i	i-th element of X
y_i	i-th element of Y
n	Size of the binary dataset
t	Random prime number used to perturb value "1" by the client
s	Random prime number used to perturb value "0" by both parties
u_i	Temporary random non-zero number generated by the client
v_i	Temporary random non-zero number generated by the server
E_c	Encryption key from the client
D_c	Decryption key from the client
E_s	Encryption key from the server
D_s	Decryption key from the server
M'	Permutation of elements in M
$E_{pk}(\cdot)$	Encryption operation by using E_{pk}
$D_{pk}(\cdot)$	Decryption operation by using D_{pk}
mod	Modulo operation
p	Modulo function with respects to t
q	Modulo function with respects to ts
r	Modulo function with respects to s^2
$S(X, Y)$	Similarity coefficient for dataset X and Y

4. Our solution

4.1. Protocol idea

The main idea behind our computation protocol is to compute all the summation variables without compromising the privacy of data for both parties. Let $X = \{x_1,\ x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_n\}$ represent two binary datasets (such that $x_i,\ y_i \in \{0,\ 1\}$) from the server and a client, respectively.

All intermediate results should be protected and no extra information could be derived from the final outputs. The outputs of the protocol are accurate if each party contributes their private input honestly. The server participates in the computation without learning anything at the end of the protocol.

In most of the computation protocols, a fair exchange is needed in order to ensure that both parties will receive the same result. A protocol is said to be symmetric if the same information or output is received by both parties. In our protocol, we deal with an asymmetric case, where only the client learns all the summation variables.

4.2. Our protocol

In this section, we will explain the details on how to compute the summation variables in a secure and privacy preserved environment.

Before the computation begins, both the client and the server generate a pair of cryptography keys and exchange the encryption key with each other. We assume that there exists a secure channel between the client and the server for the key exchange and also data transmission. All the decryption keys should be kept secret to its owner throughout the protocol.

A. Phase 1

The client first generates a pair of distinct random prime numbers: t and s. Next, it generates n different random non-zero numbers u_i $(1 \leq i \leq n)$, which are not multiples of t or s. For each $x_i \in X$, the client replaces it with $u_i t$ or $u_i s$. The operation is done as follows:

$$x'_i = \begin{cases} u_i t, & \text{if } x_i = 1 \\ u_i s, & \text{if } x_i = 0. \end{cases} \tag{5}$$

The client encrypts all x'_i in Eq. (5) with its encryption key, E_c to obtain:

$$E_c(X') = \{E_c(x'_1),\ E_c(x'_2), \ldots, E_c(x'_n)\}. \tag{6}$$

In Eq. (6), all the encrypted data $E_c(x'_i)$ have the same sequence order as its original data. By using the encryption key from the server E_s, the client encrypts t and s and transmits $E_c(X')$, $E_s(t)$ and $E_s(s)$ to the server.

B. Phase 2

The server decrypts $E_s(t)$ and $E_s(s)$ to obtain t and s. Then, it generates n different random non-zero numbers, $v_i (1 \leq i \leq n)$ which are not multiples of t or s. The server replaces each $y_i \in Y$ with the following operation:

$$y'_i = \begin{cases} v_i, & \text{if } y_i = 1 \\ v_i s, & \text{if } y_i = 0. \end{cases} \tag{7}$$

Table 5
Actions for each set of (p_i, q_i, r_i).

Case	p	q	r	Action
1	0	$\neq 0$	$\neq 0$	Increases a by 1
2	0	0	$\neq 0$	Increases b by 1
3	$\neq 0$	$\neq 0$	$\neq 0$	Increases c by 1
4	$\neq 0$	$\neq 0$	0	Increases d by 1

Next, the server encrypts y'_i in Eq. (7) with the client's encryption key E_c to produce:

$$E_c(Y') = \{E_c(y'_1), E_c(y'_2), \ldots, E_c(y'_n)\}. \tag{8}$$

The sequence of all encrypted data $E_c(y'_i)$ in Eq. (8) will be the same as its sequence order in the original dataset.

C. Phase 3

In this phase, the server multiplies $E_c(X')$ and $E_c(Y')$ by using the homomorphic multiplicative property in Eq. (4). The multiplication is done in accordance to the sequence i of x'_i and y'_i. The computation result is as follows:

$$E_c(X') \cdot_h E_c(Y') = E_c(X' \cdot Y') = \{E_c(x'_1 \cdot y'_1), E_c(x'_2 \cdot y'_2), \ldots, E_c(x'_n \cdot y'_n)\}. \tag{9}$$

Before sending this result to the client, the server shuffles elements in Eq. (9) randomly. We assume that there exists an efficient shuffle protocol which randomly changes the sequence of elements in $E_c(X' \cdot Y')$. For simplicity, let $M = E_c(X' \cdot Y')$ and $M' = \mathit{shuffle}(M)$. The server transmits M' to the client without revealing M to any party.

D. Phase 4

After receiving M' from the server, the client decrypts each $m'_i \in M'$ with its decryption key D_c as follows:

$$D_c(E_c(x'_i \cdot y'_i)) = (x'_i \cdot y'_i). \tag{10}$$

Next, the client computes three modulus functions for each $(x'_i \cdot y'_i) \in (X' \cdot Y')$ with respect to t, ts and s^2 as follows:

$$p_i = (x'_i \cdot y'_i) \ \mathbf{mod}\ t \tag{11}$$

$$q_i = (x'_i \cdot y'_i) \ \mathbf{mod}\ ts \tag{12}$$

$$r_i = (x'_i \cdot y'_i) \ \mathbf{mod}\ s^2. \tag{13}$$

Results from Eqs. (11)–(13) will be used by the client to evaluate the summation variable a, b, c and d. For each set of (p_i, q_i, r_i), the client uses Table 5 to determine the increment of each summation variable.

At the end of this phase, the client obtains all the summation variables needed to compute the similarity coefficient.

The pseudo-code of computation phases by the client and the server is shown in Figs. 1 and 2, respectively.

4.3. Assumptions

In order to guarantee that our protocol outputs the correct summation variables to the client, we make the following assumptions:

- There exists a secure channel between the client and the server for key exchange and data transmission.
- We deal with an asymmetric case, where only the client has all the outputs from the protocol.
- The size of datasets from both parties is the same.
- We assume there exists an efficient shuffle protocol which randomly shuffles elements in M.

5. Analysis and discussions

The proof of security and privacy for our protocol depends on how much information has been revealed during the execution of the computation protocol. In this section, we will provide correctness, privacy and security analysis for our protocol.

5.1. Correctness analysis

Our protocol outputs the summation variables correctly if both parties follow the protocol faithfully.

Theorem 1. *Our protocol correctly computes the size of each summation variable: a, b, c, and d without revealing extra information to any party.*

Proof. Let us assume both parties follow the protocol correctly. In phase 1, two distinct prime numbers, t and s are generated by the client (where $t \neq s$). The client replaces each $x_i \in X$ with $u_i t$ (if $x_i = 1$) or $u_i s$ (if $x_i = 0$). Note that u_i are temporary random numbers which are not multiples of t or s.

Computation phases for the Client
Inputs: $X = \{x_i | i = 1,2,\ldots,n\}$, $x_i \in \{0,1\}$
Output: size of summation variable a, b, c, and d

/*Phase 1*/
Initialise: $a = 0$, $b = 0$, $c = 0$ and $d = 0$;
Generates two distinct random prime numbers: t and s;
Encrypts t and s with E_s (e.g., $E_s(t)$ and $E_s(s)$);
for $i = 1$ **to** n **do**
  // u_i is a temporary random non-zero number.
  // u_i is not a multiple of t or s.
  Generates random u_i;
  if $x_i = 1$ **then**
    $x'_i = u_i t$;
  else
    $x'_i = u_i s$;
  end if
  Encrypts x'_i with E_c (e.g., $E_c(x'_i)$);
end for
Sends $E_c(X') = \{E_c(x'_1), E_c(x'_2), \ldots, E_c(x'_n)\}$. , $E_s(t)$ and $E_s(s)$ to the server;

/*Phase 4*/
Receives M' from the server (result from the phase 3);
for $i = 1$ **to** n **do**
  Computes (i) $p_i = D_c(m'_i)$ ***mod*** t;
  (ii) $q_i = D_c(m'_i)$ ***mod*** ts;
  (iii) $r_i = D_c(m'_i)$ ***mod*** s^2;

  if p_i =0, $q_i \neq 0$ and $r_i \neq 0$ **then**
    a increases by 1;
  else if p_i =0, q_i =0 and $r_i \neq 0$ **then**
    b increases by 1;
  else if $p_i \neq 0$, $q_i \neq 0$ and $r_i \neq 0$ **then**
    c increases by 1;
  else if $p_i \neq 0$, $q_i \neq 0$ and $r_i = 0$ **then**
    d increases by 1;
  else
    stop (error);
  end if
end for

Fig. 1. Client's computation phases.

Table 6
Inputs and outputs of the protocol (phase 1 to 4).

x_i	y_i	Client's result in phase 1	Server's result in phase 2	Result in phase 3	p	q	r
1	1	$E_c(u_i t)$	$E_c(v_i)$	$E_c(u_i v_i t)$	$(u_i v_i t)$ ***mod*** $t = 0$	$(u_i v_i t)$ ***mod*** $ts \neq 0$	$(u_i v_i t)$ ***mod*** $s^2 \neq 0$
1	0	$E_c(u_i t)$	$E_c(v_i s)$	$E_c(u_i v_i ts)$	$(u_i v_i ts)$ ***mod*** $t = 0$	$(u_i v_i ts)$ ***mod*** $ts = 0$	$(u_i v_i ts)$ ***mod*** $s^2 \neq 0$
0	1	$E_c(u_i s)$	$E_c(v_i)$	$E_c(u_i v_i s)$	$(u_i v_i s)$ ***mod*** $t \neq 0$	$(u_i v_i s)$ ***mod*** $ts \neq 0$	$(u_i v_i s)$ ***mod*** $s^2 \neq 0$
0	0	$E_c(u_i s)$	$E_c(v_i s)$	$E_c(u_i v_i s^2)$	$(u_i v_i s^2)$ ***mod*** $t \neq 0$	$(u_i v_i s^2)$ ***mod*** $ts \neq 0$	$(u_i v_i s^2)$ ***mod*** $s^2 = 0$

In phase 2, the server replaces each $y_i \in Y$ with v_i (if $y_i = 1$) and $v_i s$ (if $y_i = 0$). Although the server did not use t in phase 2, it is necessary to inform the server about the value t so that the temporary random numbers v_i generated are not multiples of t or s. Also, both u_i and v_i will not be kept by its generator. The generations of t, s, u_i and v_i are essential to ensure that the modulus operations in phase 4 will give the appropriate results for p, q and r.

For instance, consider case 1 in Table 6 where both elements are "1". The client's result in phase 1 is $E_c(u_i t)$ while the server computes $E_c(v_i)$ in phase 2. Then, the result of homomorphic multiplication computed by the server is as follows:

$$E_c(u_i t) \cdot_h E_c(v_i) = E_c(u_i v_i t). \tag{14}$$

After the decryption, the client obtains:

$$D_c(E_c(u_i v_i t)) = u_i v_i t. \tag{15}$$

Computation phases for the Server
Inputs: $Y = \{y_i | i = 1,2,\ldots,n\}$, $y_i \in \{0,1\}$
Output: M'

/*Phase 2*/
Receives $E_c(X')$, $E_s(t)$ and $E_s(s)$ from the client (results from the phase 1);
Computes $D_s(E_s(t))$ and $D_s(E_s(s))$ to obtain t and s;
for $i = 1$ **to** n **do**
  //v_i is a temporary random non-zero number.
  //v_i is not a multiple of t or s.
  Generates random v_i;
  if $y_i = 1$ **then**
    $y_i' = v_i$;
  else
    $y_i' = v_i s$;
  end if
  Encrypts y_i' with E_c (e.g., $E_c(y_i')$);
end for

/*Phase 3*/
// computation of $E_c(X') \cdot_h E_c(Y') = E_c(X' \cdot Y')$.
for $i = 1$ **to** n **do**
//homormophic multiplicative property
  Computes $E_c(x_i') \cdot_h E_c(y_i')$ (e.g., $E_c(x_i' \cdot y_i')$);
end for
Let $M = E_c(X' \cdot Y')$
Shuffles M such that $M' = shuffle(M)$;
Sends M' to the client;

Fig. 2. Server's computation phases.

If the client computes the modulus operations for Eq. (15) with respect to t, ts and s^2, only the modulus operation with respect to t will give the value "0". This is true if and only if the prime number t is present. The result for other modulus operations (with respect to ts and s^2) will not be "0" because ts and s^2 are not present in Eq. (15). Hence, in phase 4, the client obtains $p = 0$, $q \neq 0$ and $r \neq 0$. After referring to Table 5, the client increases a by 1.

As shown in Table 6, the client will encounter four possible sets of modulus results. Each set of results enable the client to determine which summation variable should be increased by 1. After the evaluations of all possible set of modulus operations, the client can obtain all the summation variables. Hence, our protocol correctly computes the size of each summation variable if both parties follow the protocol faithfully. □

5.2. Privacy analysis

The proof of privacy protection for our protocol depends on what information has been revealed and how to prevent the leakage of extra information. Each party should ensure that only the required data can be revealed during the computation protocol.

Theorem 2. *The server cannot distinguish between the elements (x_i) in the client's dataset (X). The server only has the knowledge about t and s.*

Proof. In our protocol design, the server knows the value of t and s since it has the decryption key, D_s. Other than t and s, the server has no knowledge about the other data received from the client (i.e., u_i and $x_i \in X$).

In phase 4, the client performs three modulus operations without informing the server what functions are being used (i.e., with respect to t or s^2). Furthermore, once the server has completed the phase 3, it has no knowledge about any information from the client. Therefore, the server is not able to determine any element in X. □

Theorem 3. *The client cannot identify the elements in the server's dataset (Y). At the end of the protocol, the client only learns the size of each summation variable.*

Proof. In our protocol, we assumed that the size of the dataset for both parties is the same. Each element in X and Y (according to their order) will be used for finding the size of each summation variable. Therefore, the order of all elements is essential in order to guarantee that the final output is correct. However, if the result in phase 3 is sent to the client in their original order, the modulus operations will reveal extra information to the client (i.e., if $p = 0$, $q \neq 0$ and $r \neq 0$, the client knows that y_i must be "1"). In order to prevent such leakage, the server randomly shuffles elements in M. We assume there exists an efficient shuffles protocol which randomly shuffles elements in M.

Table 7
Computation costs for both client and server.

Cost	Client	Server
Encryption	nT_E	nT_E
Decryption	nT_D	–
Homomorphic multiplication	–	nT_H
Total	$nT_E + nT_D$	$nT_E + nT_H$

Although the same s is also used by the server, the client is not able to determine the server's original dataset because the client has no knowledge about the temporary random non-zero numbers (v_i) generated by the server. In phase 2, the server uses difference v_i for each y_i. The reason is to avoid a malicious client who uses the same u_i or remembers u_i being used in phase 1 to learn the element in Y.

At the end of the computation, the client will know the size of each summation variable. Other than these, the client is not able to gain extra information from the protocol. Hence, our protocol prevents the client from identify elements in Y and only the required data have been revealed. □

5.3. Security analysis

To guarantee that our protocol is secure, we require the protocol to enable the client to detect the dishonest server.

Theorem 4. *Our protocol is secure against dishonest server. The client is able to detect and prevent any adversary activity from the dishonest server during the execution of the protocol.*

Proof. In the preliminary phase, both parties exchange the encryption key with each other via a secure channel. The decryption key is then kept secret and will not be revealed to any party.

In phase 1, the client uses the encryption key from the server to encrypt t and s. Since the decryption key D_s is only known by the server, no other parties can know the value of t or s. Assuming that the server follows the protocol: (i) uses s in phase 2, (ii) generates difference v_i for each y_i, and (iii) encrypts y_i' with the client's encryption key E_c. In phase 4, only the client can decrypt M' because other parties have no knowledge about the decryption key D_c.

In the worst case scenario, if both decryption keys (D_c and D_s) have been compromised, all elements in dataset X and Y are still secure from the malicious parties because the original elements have been perturbed by the temporary non-zero random numbers, u_i and v_i.

In phase 4, if the client obtains different sets of modulus results (other than the four possible sets in Table 6), the client can conclude that there might be a risk for some attacks. Note that in order to protect the summation variables computed from our protocol, the client should not inform the server or any party about the functions being used to compute the modulus operations (i.e., with respect to t, ts, or s^2).

The leakage of modulus results in phase 4 (i.e., the value of p, q and r) will not reveal any information because only the client knows what are the functions being used in the computations. Hence, our protocol is secure against the dishonest server. □

5.4. Efficiency analysis

Our protocol only requires one round communication. The overall communication cost is measured based on the amount of data needed to be transferred in the protocol. In our protocol, the client sends n encrypted data to the server. In return, the server responds with n permuted data. The communication complexity incurred by both parties will be $O(n)$.

In our protocol, two random prime numbers (t and s) and $2n$ temporary random numbers (u_i and v_i) will be generated. The generation of u_i and v_i will not incur much computation overhead because they are just temporary numbers used in our protocol. Both parties will not keep these random numbers once they have been used. The following contributes to the computation costs in our protocol:

1. Total number of $2n$ encryptions by both parties.
2. Computation of n homomorphic multiplications by the server.
3. Total number of n decryptions by the client.

Let T_E, T_D and T_H denote the encryption, decryption and homomorphic multiplication cost, respectively. The total computation costs for each party are listed in Table 7.

The total communication costs from both parties are $2nT_E + nT_H + nT_D$. Note that the encryption and decryption cost is almost the same, and the cost for the homomorphic multiplicative operation is smaller than the decryption cost. Therefore, our computation overhead is $O(nT_E + nT_D)$, or we can say $O(nT_E)$ or $O(nT_D)$.

5.5. Computation for multi-dataset

So far, all the discussions and analysis for our protocol are based on single-dataset scenario, where each party (client and server) has only one binary dataset. In many cases, the server might hold more than one dataset. For example, a client with a binary dataset X would like to know the similarity coefficient of X with the datasets in $Y = \{Y^1, Y^2, \ldots, Y^N\}$. This scenario has been defined in most clustering and classification problems under the distributed environment.

To compute multi-dataset cases, the client performs the same operation as in phase 1. In phase 2, the server computes encrypted dataset for each $Y^j \in Y$ as follows:

$$E_c(Y^{j'}) = \{E_c(y_1^{j'}), E_c(y_2^{j'}), \ldots, E_c(y_n^{j'})\}. \tag{16}$$

In this phase, the server will obtain N encrypted datasets. Next, the server computes the homomorphic multiplication between $E_c(X')$ and each $E_c(Y^{j'})$ in phase 3. We can denote the result in phase 3 as a matrix representation such that,

$$E_c(X' \cdot Y_i^{j'}) = \begin{pmatrix} E_a(x'_1 y_1^{1'}) & \cdots & E_a(x'_1 y_1^{N'}) \\ \vdots & \ddots & \vdots \\ E_a(x'_n y_n^{1'}) & \cdots & E_a(x'_n y_n^{N'}) \end{pmatrix}. \tag{17}$$

In phase 4, the client computes the modulus operations (p, q and r) for each of the element in $E_c(X' \cdot Y_i^{j'})$. Hence, the client is able to obtain all the summation variables for each pair of datasets between X and Y (i.e., X and Y^1 or X and Y^2). From the similarity coefficients computed, the client will be able to determine which dataset (or cluster) in the server is most similar to its private dataset X.

6. Conclusion

In this paper, we proposed a privacy preserving protocol to compute the summation variables for binary datasets under a semi-honest model. These summation variables are an important component for similarity coefficient measurements which have been used in many research domains such as data mining, cloud computing, and bioinformatics. Our protocol is based on the semantically-secure cryptography system. The design of our protocol allows the client to obtain the necessary summation variables without the leakage of extra information before and after the protocol execution.

We would like to remark that our objective in this paper is not to design a specific similarity coefficient. Rather, we are demonstrating how to compute the similarity coefficients under a secure and privacy preserving environment. Note that our protocol also supports dissimilarity coefficients.

Acknowledgments

This research was supported by the Soongsil University Research Fund 2011. We would also like to thank anonymous reviewers for their constructive comments and helpful advice.

References

[1] R.R. Sokal, C.D. Michener, A statistical method for evaluating systematic relationships, University of Kansas Science Bulletin 38 (1958) 1409–1438.
[2] P. Jaccard, Nouvelles recherches sur la distribution florale, Bulletin de la Sociète Vaudense des Sciences Naturelles 44 (1908) 223–270.
[3] P. Jaccard, The distribution of the flora in the alpine zone, New Phytologist 11 (1912) 37–50.
[4] T. SØRENSON, A method of stabilizing groups of equivalent amplitude in plant sociology based on the similarity of species content and its application to analyses of the vegetation on danish commons, Kongelige Danske Videnskabernes Selskab Biologiske Skrifter 5 (1948) 1–34.
[5] L.R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (1945) 297–302.
[6] R.R. Sokal, P.H.A. Sneath, Principles of Numeric Taxonomy, Freeman, W.H., San Francisco, 1963.
[7] E.F. Connor, D. Simberloff, Intraspecific competition and species co-occurrence patterns on Islands, Oikos 41 (1983) 455–465.
[8] M.E. Hohn, Binary coefficients: a theoretical and empirical study, Mathematical Geology 8 (1976) 137–150.
[9] M.E. Gilpin, J.M. Diamond, Factors contributing to non-randomness in species Co-occurrences on Islands, Oecologia 52 (1982) 75–84.
[10] Z. HubÁLek, Coefficient of association and similarity: based on binary (presence–absence) data: an evaluation, Biological Reviews 57 (1982) 669–689.
[11] D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (1987) 139–172.
[12] H. Wang, W. Wang, J. Yang, P.S. Yu, Clustering by pattern similarity in large data sets, in: Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, ACM, Madison, Wisconsin, 2002, pp. 394–405.
[13] P. Willett, Similarity-based approaches to virtual screening, Biochemical Society Transactions 31 (2003) 603–606.
[14] G.K. Gilbert, Finely's Tornado predictions, The American Meteorological Journal 1 (1884) 66–72.
[15] L.A. Goodman, W.H. Kruskal, Measures of association for cross classifications. II: further discussion and references, Journal of the American Statistical Association 54 (1959) 123–163.
[16] P.H.A. Sneath, R.R. Sokal, Numerical Taxonomy: The Principles and Practice of Numerical Classification, W.H. Freeman, San Francisco, 1973.
[17] H.T. Clifford, W. Stephenson, An Introduction to Numerical Classification, Academic Press, New York, 1975.
[18] P.F. Russel, T.R. Rao, On habitat and association of species of Anophe-line larvae in South-eastern Madras, Journal of Malaria Institute India 3 (1940) 153–178.
[19] C. Baroni-Urbani, M.W. Buser, Similarity of binary data, Systematic Zoology 25 (1976) 251–259.
[20] D.P. Faith, Asymmetric binary similarity measures, Oecologia 57 (1983) 287–290.
[21] L.A. Goodman, W.H. Kruskal, Measures of association for cross classifications, Journal of the American Statistical Association 49 (1954) 732–764.

[22] S.S. Choi, Correlation Analysis of Binary Similarity and Dissimilarity Measures, Pace University, 2008.
[23] D.J. Rogers, T.T. Tanimoto, A computer program for classifying plants, Science 132 (1960) 1115–1118.
[24] G.U. Yule, M.G. Kendall, An Introduction to the Theory of Statistics, fourteen ed., Hafner, New York, 1950.
[25] S. Kulczynski, Classe des sciences mathématiques et naturelles, Bulletin International de l'Académie Polonaise des Sciences et des Lettres (1927) 57–230.
[26] A. Ochiai, Zoogeographic studies on the soleoid fishes found in Japan and its neighbouring regions, Bulletin of the Japanese Society for Science and Fisheries 22 (1957) 526–530.
[27] R. Agrawal, A. Evfimievski, R. Srikant, Information sharing across private databases, in: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data, ACM, San Diego, California, 2003, pp. 86–97.
[28] H. Lipmaa, Verifiable homomorphic oblivious transfer and private equality test, in: Advances in Cryptology, Springer-Verlag, Taipei, Taiwan, 2003, pp. 416–433.
[29] R. Fagin, M. Naor, P. Winkler, Comparing information without leaking it, Communications of the ACM 39 (1996) 77–85.
[30] H.A. Park, B.H. Kim, D.H. Lee, Y.D. Chung, J. Zhan, Secure similarity search, in: IEEE International Conference on Granular Computing, Silicon Valley, USA, 2007, pp. 598–604.
[31] S. Laur, H. Lipmaa, On private similarity search protocols, in: Proceedings of the Ninth Nordic Workshop on Secure IT Systems, NordSec 2004, Citeseer, Helsinki, 2004, pp. 73–77.
[32] V. Klyuev, V. Oleshchuk, Semantic retrieval: an approach to representing, searching, and summarizing text documents, International Journal of Information Technology, Communications and Convergence 1 (2011) 221–234.
[33] Y. Yunming, L. Xutao, W. Biao, L. Yan, A comparative study of feature weighting methods for document co-clustering, International Journal of Information Technology, Communications and Convergence 1 (2010) 206–220.
[34] M. Somayeh, A. Mehmet, B. Christoph, Runtime verification in distributed computing, Journal of Convergence 2 (2011) 1–10.
[35] L. Tong, F. Yu, Y. Lin, X. Kong, Y. Yu, Trusted computing dynamic attestation using a static analysis based behavior model, Journal of Convergence 2 (2011) 61–68.
[36] W. Du, M. Atallah, Privacy-preserving cooperative statistical analysis, in: Proceedings of the 17th Annual Computer Security Applications Conference, IEEE Computer Society, New Orleans, Louisiana, 2001, pp. 102–110.
[37] W. Du, Z. Zhan, Building decision tree classifier on private data, in: Proceedings of the IEEE International Conference on Privacy, Security and Data Mining, vol. 14, Australian Computer Society, Inc., Maebashi City, Japan, 2002, pp. 1–8.
[38] Y. Li, J.D. Tygar, J.m. Hellerstein, Private matching, in: Computer Security in the 21st Century, Springer Science + Business Media, New York, 2005, pp. 25–50.
[39] T.E. Gamal, A public key cryptosystem and a signature scheme based on discrete logarithms, IEEE Transactions on Information Theory 31 (1985) 469–472.