Secure Similarity Coefficients Computation with Malicious Adversaries

Bo Zhang and Fangguo Zhang

School of Information Science and Technology,
Sun Yat-Sen University, Guangzhou 510006, P. R. China
ldzhbo@hotmail.com, isszhfg@mail.sysu.edu.cn

Abstract. Similarity coefficients play an important role in many application aspects. Recently, a privacy-preserving similarity coefficients protocol for binary data was proposed by Wong and Kim (Computers and Mathematics with Application 2012). In this paper, we show that their protocol is not secure, even in the semi-honest model, since the client can retrieve the input of the server without deviating from the protocol. Also we propose a secure similarity coefficients computation in the presence of malicious adversaries, and prove it using the standard simulation-based security definitions for secure two-party computation. We also discuss several extensions of our protocol for settling other problems. Technical tools in our protocol include zero-knowledge proofs and distributed ElGamal encryption.

Keywords: Similarity coefficients, Distributed ElGamal encryption, Zero-knowledge proof, Secure two-party computation, Malicious adversary.

1 Introduction

Similarity coefficients (SC) aim at quantifying the extent to which objects resemble each other. Many application domains need this parameter to analyze the data, such as ecology and biogeography [7, 18, 16, 19], privacy-preserving data mining [12, 36], biometric areas [34], etc. The description about the importance of similarity coefficients can be found in [35].

The functionality of the privacy-preserving similarity coefficients for binary data in [35] (denoted by $\mathcal{F}_{SC}$) can be described as follows. Consider that user P_1 has a binary dataset $X = \{x_1, x_2, \cdots, x_n\}$, similarly P_2 has $Y = \{y_1, y_2, \cdots, y_n\}$, where $x_i, y_i \in \{0, 1\}$. After the computation, P_1 gets the result: $n_{11}, n_{10}, n_{01}, n_{00}$, where n_{11} denotes the number of (x_i, y_i) when $x_i = y_i = 1$ in the set $C_{XY} = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, similarly n_{10} (n_{01}, and n_{00}) denotes the number about the case $x_i = 1, y_i = 0$ ($x_i = 0, y_i = 1$, and $x_i = 0, y_i = 0$, respectively).

There are many variations of similarity coefficients. For instance, (1). SC with n_{00}: Rusell-Rao Index [31] n_{11}/n, Sokal-Michener [33] $(n_{11}+n_{00})/n$, etc.; (2). SC without considering n_{00}: Jaccard's index [21] $n_{11}/(n_{11}+n_{10}+n_{01})$, etc., where $n = n_{11}+n_{10}+n_{01}+n_{00}$. All these similarity coefficients can be correctly computed using $n_{11}, n_{10}, n_{01}, n_{00}$. In case n_{11}/n, P_1 can compute it only with n_{11}. However, in case $n_{11}/(n_{11} + n_{10} + n_{01})$, P_1 needs to know n_{11} and $n_{10} + n_{01}$ at least, since our protocols only output the numbers of $n_{11}, n_{10}, n_{01}, n_{00}$, or the sum of elements in the subset of them, rather than similarity coefficients directly.

Secure two-party computation allows two parties to jointly compute some functions with their private inputs, while preserving some kinds of properties. Research on the general functionality of secure computation in the computational model was first proposed in [37]

in the semi-honest model, extended in [17, 27, 25, 26] in the malicious model. However, these general protocols are inefficient for practical uses (because these protocols were constructed based on the boolean circuit or the arithmetic circuit of the functionality, and do not utilize the properties of specific functionality). Therefore, these protocols are not practical to compute $\mathcal{F}_{SC}$ in the malicious model.

We note that protocol in [35] can not protect P_2's privacy, since P_1 can retrieve P_2's input from the communication transcript. More detailed analysis about this attack will be given in Section 3. Also adversaries considered in [35] are semi-honest, which means that P_1 and P_2 have to correctly behave the protocol. We argue that this level of security is not sufficient in the application.

Our results. In this paper, we first analyze the security of the protocol in [35], and give out an attack on it, by which P_1 can retrieve P_2's input without deviating from the protocol. Also we note that the correctness of the protocol in [35] is not sound, since the result of the multiplication of some parameters may overflow from the plaintext space.

Next, we construct a secure protocol (denoted by Π_{SC})computing the functionality $\mathcal{F}_{SC}$ in the presence of malicious adversaries, and also give out its standard simulation-based security proof. We also construct another protocol (denoted by Π_{SC1}) by modifying our first protocol Π_{SC}. In Π_{SC1}, P_1 can only obtain the number n_{11}, rather than that P_1 obtains four numbers in Π_{SC}. This can perfectly protect P_2's privacy if P_1 only needs the similarity coefficient n_{11}/n [31]. Similarly, a simple extension can be made, in order that P_1 obtains the sum of elements in the subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$. Not only for the binary data, our protocol can also be modified to settle the other kinds of data, such as hex number system.

Our main technical tools include distributed ElGamal encryption and zeo-knowledge proofs of knowledge. Our protocol is similar with a pattern matching protocol in [20]. However, we can not directly use a secure pattern matching protocol to settle the similarity coefficients problem.

Related works. Wong and Kim [35] gave out the first solution to compute $\mathcal{F}_{SC}$ in the semi-honest model. However, their protocol is not secure. Also the security model in [35] is semi-honest model, which is not sufficient.

Private equality test protocol [23, 13] can only output the result that whether two inputs of P_1 and P_2 are equal or not without leaking any extra information. Whereas, in $\mathcal{F}_{SC}$, P_1 needs to know $n_{11}, n_{10}, n_{01}, n_{00}$, rather than $n_{11}+n_{00}$. Also the modification should be made to protect the privacy of P_2, since bit-by-bit equality test can leak the information about P_2's input. Therefore, Private equality test protocol is not suitable to settle the similarity coefficients computation.

Scalar product protocol [8, 10] only outputs the scalar product of two vectors owned by P_1 and P_2 respectively. Therefore, it only reveals the number n_{11} with the inputs X, Y. Pattern matching protocol [20] is used to find the positions that one pattern p appears in the other long text t. Therefore, with inputs X, Y, it only outputs the number n_{11} or n_{00} separately. Also, it is not suitable here. Meanwhile, as discussed in [35], private matching [1], private similarity search [30, 24], and similarity-based text retrieval [22, 38] are also not suitable.

Therefore, we can not directly use these prior works to settle this similarity coefficients computation, except these inefficient general results. Also we note that most of the above protocols are only secure in the semi-honest model.

Organization. This paper is organized as follows: In Section 2, we present the security definition and tools used in our protocol. Review of the protocol [35] and its analysis will be given in Section 3. In Section 4, we construct a secure similarity coefficients computation protocol Π_{SC} in the presence of malicious adversaries, and give out its standard simulation-based security proof. In Section 5, we will give out the protocol Π_{SC1} by modifying Π_{SC}, and discuss its extensions. The conclusion is in Section 6.

2 Preliminaries and Tools

2.1 Definitions

Notations. We denote the security parameter by n. A function $\mu(\cdot)$ is negligible in n, if for every polynomial $p(\cdot)$ and all sufficiently large n, it holds that $\mu(n) < \frac{1}{p(n)}$. Let $X = \{X(a,n)\}_{n\in\mathbb{N},a\in\{0,1\}^*}$ and $Y = \{Y(b,n)\}_{n\in\mathbb{N},b\in\{0,1\}^*}$ be distribution ensembles. Then, we say that X and Y are computationally indistinguishable, denote $X \stackrel{c}{\equiv} Y$, if for every polynomial non-uniform distinguisher D, there exists a negligible function $\mu(\cdot)$ such that for every $a \in \{0,1\}^*$,

$$|Pr[D(X(a,n)) = 1] - Pr[D(Y(a,n)) = 1]| < \mu(n).$$

Security definitions. Details about malicious adversary model in the two-party computation setting can be found in [14]. Please refer to Appendix A for the formal description about this model.

Hybrid Model. In this paper, we regard zero-knowledge proof of knowledge as a subprotocol, and use the hybrid model in our protocol. Results in [4] show that we could analyze the security of our protocol in a hybrid model. This means that in our protocol, parties interact with each other and have access to a trusted party that computes the zero-knowledge proof of knowledge for them. Therefore, in the real and ideal model simulation proof, we could cut the proof shorter. We note that the composition theorem of [4] holds for the case that the subprotocol executions are all run sequentially. For more details, please refer to [4].

2.2 ElGamal Encryption Scheme

In our protocol, we use an additively homomorphic variation of ElGamal encryption [11] with distributed decryption over a Group $\mathbb{G}_q$ in which DDH is hard, i.e., $En_{pk}(m,r) = (g^r, g^m h^r)$. This is a modified version of Brandt [2], in which the ElGamal encryption format with distributed decryption is $En_{pk}(m,r) = (g^r, mh^r)$. For ciphertexts $C = (\alpha, \beta)$, $C' = (\alpha', \beta')$, and $r \in \mathbb{Z}_q$, we use C^r and C/C' to denote (α^r, β^r) and $(\alpha/\alpha', \beta/\beta')$, respectively.

2.3 Zero-knowledge Proofs

For the purpose of preventing malicious behavior, we use zero-knowledge proof in the protocol to confirm that all the parties behave the protocol correctly. We will use Σ-protocols made secure against malicious verifiers with standard techniques as [20]. We denote these

associated functionalities by $\mathcal{F}_{DL}$, $\mathcal{F}_{EqDL}$, $\mathcal{F}_{isBit}$, and $\mathcal{F}_{perm}$. All of them are proved in the cyclic group $\mathbb{G}_q$ with prime order q, where q is the parameter in the ElGamal Encryption. Next, we simply describe the associated zero-knowledge protocols: π_{DL}, π_{EqDL}, π_{isBit}, and π_{perm}.

π_{DL} [32]. The prover can prove to the verifier that he knows the knowledge of the solution x to a discrete logarithm.

$$\mathcal{R}_{DL} = \{((\mathbb{G}_q, q, g, h), x) |\ h = g^x\}.$$

π_{EqDL} [6]. The prover can prove to the verifier that the solutions of two discrete logarithm problems are equal.

$$\mathcal{R}_{EqDL} = \{((\mathbb{G}_q, q, g, g_1, g_2, g_3), x) |\ g_1 = g^x \wedge\ g_3 = g_2^x\}.$$

π_{isBit}. The prover can prove to the verifier that the encrypted message m of ciphertext C equals to 0 or 1. This can be obtained from π_{EqDL} using the technique of Cramer *et al.* [5].

$$\mathcal{R}_{isBit} = \{(\mathbb{G}_q, q, g, h, C = (\alpha, \beta)), (m, r) | (\alpha, \beta) = (g^r, g^m h^r) \wedge m \in \{0, 1\}.\}$$

π_{perm}. [15] The prover can prove to the verifier that a set of ciphertexts is a permutation and rerandomization of another set of ciphertexts.

$$\mathcal{R}_{perm} = \{((pk, \{C_i\}_i, \{C_i'\}_i), (\pi, \{r_i\}_i))\ s.t.\ (\alpha_i', \beta_i') = (\alpha_{\pi(i)} g^{r_i}, \beta_{\pi(i)} h^{r_i})\}.$$

2.4 Distributed ElGamal Encryption

We adopt the distributed ElGamal Encryption in our protocol, rather than ElGamal encryption [11] and Paillier encryption [28]. Details about the distributed ElGamal Encryption can be found in [20]. Diffie-Hellman key exchange [9] is required to modify the private key generation algorithm on the original ElGamal scheme, in order to let the two parties additively share the private key [29]. Here we simply describe the protocol.

In a distributed ElGamal Encryption protocol, the parties P_1 and P_2 first agree on the parameter $\mathbb{G}_q$ and g. P_1 randomly chooses s_1, and sends g^{s_1} to P_2. Then P_1 proves to P_2 that he has s_1 via π_{DL}. Similarly, P_2 randomly chooses s_2 and sends g^{s_2} to P_1, then proves the knowledge about s_2. Thus, the public key is $pk = (\mathbb{G}_q, q, g, h = g^{s_1+s_2})$, the private key is $(sk_1, sk_2) = (s_1, s_2)$, shared by P_1 and P_2. We denote this protocol as π_{KeyGen}, and its functionality as $\mathcal{F}_{KeyGen}(1^k, 1^k) = ((pk, sk_1), (pk, sk_2))$.

When they want to decrypt $C = (\alpha, \beta)$, P_2 sends α^{s_2} to P_1, and proves that $(g, g^{s_2}, \alpha, \alpha^{s_2})$ is a DH-tuple via π_{EqDL}. At last, P_1 obtains g^m via $c_2/(c_1)^{s_1+s_2}$, then computes out m after computing the discrete log of g^m base g. We denote this part as π_{dec}, and its functionality as $\mathcal{F}_{dec}$. If $0 \leq m \leq T$, this takes expected time $O(\sqrt{T})$ using Pollard's Lambda method. Thus, the plaintext space should be limited. A similar decryption algorithm can be found in [3]. For simplicity, we use $C = En(g^m)$ to denote the encryption of g^m.

The semantic security of this scheme follows from the hardness of decisional Diffie-Hellman (DDH) in $\mathbb{G}_q$. For more details, please refer to [20]. The additively homomorphic can be computed easily, e.g., for $C_1 = En(g^{m_1}) = (g^{r_1}, g^{m_1} h^{r_1})$ and $C_2 = En(g^{m_2}) = (g^{r_2}, g^{m_2} h^{r_2})$, thus $C_1 C_2 = En(g^{m_1+m_2}) = (g^{r_1+r_2}, g^{m_1+m_2} h^{r_1+r_2})$.

3 Review of the Protocol in [35] and its Security Analysis

In this section, we first review the protocol computing similarity coefficients in [35], then give out an attack method, by which P_1 can retrieve P_2's input.

3.1 Review of the Protocol in [35]

Wong *et al.* [35] adopted the original ElGamal encryption [11]- $En_{pk}(m) = (g^r, mh^r)$ in their protocol. This encryption has the multiplicative homomorphic property, i.e., for ciphertext $C_1 = En(m_1)$, $C_2 = En(m_2)$, we have $C_1C_2 = En(m_1m_2)$. Next is the protocol:

- **Inputs:** The input of P_1 is a binary set with order n: $X = \{x_1, x_2, \cdots, x_n\}$. Similarly, P_2's input is $Y = \{y_1, y_2, \cdots, y_n\}$.
- **Auxiliary inputs:** The parties share a security parameter 1^k.
- **The protocol:**
  1. Both P_1 and P_2 generate a pair of cryptography keys (i.e., $(pk_1, sk_1), (pk_2, sk_2)$), send the public key pk_1, pk_2 to each other, and keep the secret key privately.
  2. P_1 randomly chooses non-zero numbers $t, s, u_1, u_2, \cdots, u_n \in \mathbb{Z}_q$, conditioned on that $u_1, u_2, \cdots, u_n$ are not multiples of t or s. For each $x_i \in X$, P_1 replaces it with u_it or u_is as follows: if $x_i = 1$, define $x'_i = u_it$; otherwise, define $x'_i = u_is$.
  3. P_1 encrypts $x'_1, x'_2, \cdots, x'_n$ with his public key pk_1:
  $$En_1(X') = \{En_1(x'_1), En_1(x'_2), \cdots, En_1(x'_n)\}.^1$$
  Then P_1 sends $En_2(t)$, $En_2(s)$, and $En_1(X)$ to P_2.
  4. P_2 decrypts $En_2(t)$ and $En_2(s)$ to obtain t and s using sk_2, and randomly chooses non-zero numbers $v_1, v_2, \cdots, v_n \in \mathbb{Z}_q$, conditioned on that they are not multiples of t or s. For each $y_i \in Y$, P_2 replaces it with v_i or v_is as follows: if $y_i = 1$, define $y'_i = v_i$; otherwise, define $y'_i = v_is$.
  5. P_2 encrypts $y'_1, y'_2, \cdots, y'_n$ using pk_1: $En_1(Y') = \{En_1(y'_1), En_1(y'_2), \cdots, En_1(y'_n)\}$, then multiples $En_1(X')$ and $En_1(Y')$ using the multiplicative homomorphic property. Then P_2 has:
  $$En_1(X'Y') = \{En_1(x'_1y'_1), En_1(x'_2y'_2), \cdots, En_1(x'_ny'_n)\}.$$
  Next P_2 uses a shuffle protocol to rerandomize and permute the ciphertexts $En_1(X'Y')$, and sends the permuted ciphertexts to P_1. We denote the permuted result as $En_1(X''Y'')$.[2]
  6. P_1 decrypts $En_1(X''Y'')$ with sk_1, then obtains the messages $x''_1y''_1, x''_2y''_2, \cdots, x''_ny''_n$. Next, P_1 computes three modulus functions for each $x''_iy''_i$ with respect to t, ts, and s^2 as follows: $p_i = (x''_iy''_i)$ **mod** t, $q_i = (x''_iy''_i)$ **mod** ts, $r_i = (x''_iy''_i)$ **mod** s^2. At last, P_1 evaluates these values as follows: for $i \in \{1, 2, \cdots, n\}$,
  (a) if $p_i = 0, q_i \neq 0, r_i \neq 0$, then $\hat{x}_i = 1$ and $\hat{y}_i = 1$;
  (b) if $p_i = 0, q_i = 0, r_i \neq 0$, then $\hat{x}_i = 1$ and $\hat{y}_i = 0$;
  (c) if $p_i \neq 0, q_i \neq 0, r_i \neq 0$, then $\hat{x}_i = 0$ and $\hat{y}_i = 1$;
  (d) if $p_i \neq 0, q_i \neq 0, r_i = 0$, then $\hat{x}_i = 0$ and $\hat{y}_i = 0$.
  Then P_1 obtains the numbers $n_{11}, n_{10}, n_{01}, n_{00}$ after counting the numbers of $(1, 1)$, $(1, 0)$, $(0, 1)$, $(0, 0)$ in the set: $\{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \cdots, (\hat{x}_n, \hat{y}_n)\}$.

[1] We use $En_1(m)$ to denote the encryption of m using the public key pk_1. Similarly, $En_2(m)$ denotes the ciphertext using pk_2.

[2] The order of the ciphertexts is shuffled. However, the plaintexts set of them are not changed.

3.2 Security Analysis

In this section, we analyze the above protocol in two parts: (1). the result of the multiplication $x_i''y_i''$ may overflow the plaintext space; (2). P_1 can retrieve P_2's input Y without deviating from the protocol.

In order to make sure that the modulus functions $p_i = (x_i''y_i'') \bmod t$, $q_i = (x_i''y_i'') \bmod ts$, $r_i = (x_i''y_i'') \bmod s^2$ output the correct values, $x_i''y_i''$ should not overflow the plaintext space of the ElGamal encryption. Or else, the modular algorithms of p_i, q_i, and r_i are not sound.

For instance, if $\hat{x}_i = 1, \hat{y}_i = 0$, then $x_i''y_i'' = u_itv_is$. We assume that the plaintext space of their ElGamal encryption is $\mathbb{Z}_q$. Thus, if $x_i''y_i'' = u_itv_is > q$, P_1 can get the result $x_i''y_i'' = u_itv_is \bmod q$ after the decryption. However, the outputs p_i, q_i, r_i of $x_i''y_i''$ do not satisfy the case $p_i = 0, q_i = 0, r_i \neq 0$ after modular over q. Therefore, protocol in [35] should add a restriction on the random numbers that the value $u_itv_is < q$, for $i \in \{1, 2, \cdots, n\}$, e.g., assume that $u_i, t, v_i, s < q^{1/4}$, for $i \in \{1, 2, \cdots, n\}$.

The most important problem of the above protocol is the privacy. Now we give out a way to retrieve P_2's input using the communication transcripts of P_1. We assume that in the last step P_1 has the decryption results: $\{x_1''y_1'', x_2''y_2'', \cdots, x_n''y_n''\}$. Thus the communication transcripts of P_1 include t, s, $\{u_1, u_2, \cdots, u_n\}$, and $\{x_1''y_1'', x_2''y_2'', \cdots, x_n''y_n''\}$. Next is the simple retrieving method. For $i \in \{1, 2, \cdots, n\}$,

1. P_1 first checks the parameters p_i, q_i, r_i of $x_i''y_i''$, and determines the pair $\hat{x}_i, \hat{y}_i$.
2. P_1 checks that which parameter u_j can divide $x_i''y_i''$, i.e. $x_i''y_i'' \bmod u_j = 0$, for $j \in \{1, 2, \cdots, n\}$.
3. P_1 obtains that the jth input of P_2 equals to $\hat{y}_i$, if $x_i''y_i'' \bmod u_j = 0$ and $x_j = \hat{x}_i$.

If any two numbers of $\{u_1, u_2, \cdots, u_n\}$ are coprime, and u_i does not divide v_i,[3] for $i \in \{1, 2, \cdots, n\}$, the above algorithm will always uniquely retrieve P_2's input. Also if we only assume that any two numbers of $\{u_1, u_2, \cdots, u_n\}$ are coprime, the above algorithm will always retrieve one bit of P_2's input with high probability. We note that if P_1 could correctly retrieve one bit from the above method, the protocol is not secure anymore.

Now, we give out a simple example. Assume that $p_1 = 0, q_1 \neq 0, r_1 \neq 0$ for $x_1''y_1''$, which means that there exists $\hat{x}_1 = 1, \hat{y}_1 = 1$ in the set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$. Here we assume that only u_2 divides $x_1''y_1''$ and $x_2 = \hat{x}_1$, which means that $x_1''y_1'' = u_2tv_2$, since $\hat{x}_1 = 1, \hat{y}_1 = 1$. Therefore, before shuffling, $En_1(x_1''y_1'')$ is at the second place of the set $En_1(X'Y')$. Hence, the second input of P_2 is $\hat{y}_1 = 1$. In the end, after executing all the values $x_i''y_i''$, for $i \in \{1, 2, \cdots, n\}$, P_1 retrieves P_2's input bit-by-bit.

Since the users in the protocol [35] are semi-honest (they all correctly follow the protocol), and $u_1, u_2, \cdots, u_n$ and $v_1, v_2, \cdots, v_n$ are all not multiples of t or s, the above retrieving method will always break the privacy of P_2. Therefore, their protocol is not secure. Also, we can not give out a modification on their protocol to avoid this attack.

[3] As the spaces of u_i, t, v_i, s are restricted for the correctness (e.g., $u_i, t, v_i, s < q^{1/4}$), and P_2 correctly follows the protocol in the semi-honest model, u_i can be selected largely in order that any multiples of u_i will overflow the space of v_i. Hence u_i does not divide v_i.

4 Secure Similarity Coefficients Computation in the Presence of Malicious Adversaries

In this section, we give out the secure similarity coefficients computation protocol (Π_{SC}) in the presence of malicious adversaries. First, we give out the ideal functionality of similarity coefficients $\mathcal{F}_{SC}$ as follows:

$$((x_1, x_2, \cdots, x_n), (y_1, y_2, \cdots, y_n)) \mapsto ((n_{11}, n_{10}, n_{01}, n_{00}), \lambda),$$

where λ means P_2 gets nothing after the computation, and $n_{b_1 b_2}$ denotes the numbers of the pair (b_1, b_2) in the set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$, for $b_1, b_2 \in \{0, 1\}$. In the ideal model, P_1 sends the binary dataset $\{x_1, x_2, \cdots, x_n\}$ to the trusted party, similarly P_2 sends $\{y_1, y_2, \cdots, y_n\}$ to the trusted party. At last, the trusted party sends $(n_{11}, n_{10}, n_{01}, n_{00})$ back to P_1, and nothing to P_2.

The main tools in our protocol include distributed ElGamal encryption (rather than the original ElGamal encryption used in [35]) and zero-knowledge proofs. Next is the protocol, Π_{SC}.

- **Inputs:** The input of P_1 is a binary set with order n: $X = \{x_1, x_2, \cdots, x_n\}$. Similarly, P_2's input is $Y = \{y_1, y_2, \cdots, y_n\}$.
- **Auxiliary inputs:** Both parties have the security parameter 1^k.
- **The protocol:**
  1. P_1 and P_2 engage in the protocol $\pi_{KeyGen}(1^k, 1^k)$ to generate the public key $pk = (\mathbb{G}_q, g, h = g^{s_1+s_2})$, and the private keys s_1 and s_2, shared by P_1 and P_2 respectively.
  2. P_1 computes $C_{x_i} = En(x_i, r_{x_i})$, $i \in \{1, 2, \cdots, n\}$, and sends them to P_2. Then the parties run the zero-knowledge proof of knowledge π_{isBit}, allowing P_2 to verify that the plaintext of C_{x_i} is a bit known to P_1, for $i \in \{1, 2, \cdots, n\}$.
  3. Similarly, P_2 computes $C_{y_i} = En(y_i, r_{y_i})$, $i \in \{1, 2, \cdots, n\}$, and sends them to P_1. Then the parties run the zero-knowledge proof of knowledge π_{isBit}, allowing P_1 to verify that the plaintext of C_{y_i} is a bit known to P_2, for $i \in \{1, 2, \cdots, n\}$.
  4. Both parties compute: for $i \in \{1, 2, \cdots, n\}$,
  $$C_i = C_{x_i}^2 C_{y_i} = En(2x_i + y_i, 2r_{x_i} + r_{y_i}).$$
  5. P_2 picks a uniformly random permutation $\pi : \mathbb{Z}_n \to \mathbb{Z}_n$ and applies π to the set $\{C_1, C_2, \cdots, C_n\}$,
  $$\{C'_1, C'_2, \cdots, C'_n\} \leftarrow \pi\{C_1, C_2, \cdots, C_n\},$$
  and rerandomizes all the encryptions, $C''_i \leftarrow C'_i \cdot En(0, r_i)$ for every $i \in \{1, 2, \cdots, n\}$, where $r_1, r_2, \cdots, r_n$ are randomly chosen from $\mathbb{Z}_q$. Then P_2 sends $\{C''_1, C''_2, \cdots, C''_n\}$ to P_1.
  6. P_1 and P_2 execute π_{perm} on $(\{C_1, C_2, \cdots, C_n\}, \{C''_1, C''_2, \cdots, C''_n\})$ allowing P_1 to verify that the plaintexts of $\{C''_1, C''_2, \cdots, C''_n\}$ correspond to those of $\{C_1, C_2, \cdots, C_n\}$.
  7. Finally, P_1 and P_2 execute π_{dec} on each ciphertext in $\{C''_1, C''_2, \cdots, C''_n\}$. P_1 obtains the result $\{g^{m_1}, g^{m_2}, \cdots, g^{m_n}\}$ of $\{C''_1, C''_2, \cdots, C''_n\}$ after distributed decryption. Note that $g^{m_i} \in \{g^3, g^2, g^1, g^0\}$. At last, P_1 obtains the parameters $n_{11}, n_{10}, n_{01}, n_{00}$, where $n_{11}, n_{10}, n_{01}, n_{00}$ equal to the number of g^3, g^2, g^1, g^0 in $\{g^{m_1}, g^{m_2}, \cdots, g^{m_n}\}$, respectively.

Correctness. Since the zero-knowledge proof π_{isBit} and π_{perm} make P_1 and P_2 behave the protocol correctly, each plaintext of $\{C_1'', C_2'', \cdots, C_n''\}$ belong to the set $\{0, 1, 2, 3\}$. In other words, $0 \leq 2x_i + y_i \leq 3$, where $x_i, y_i \in \{0, 1\}$, for $i \in \{1, 2, \cdots, n\}$. We note that the rerandomization and permutation over the set $\{C_1, C_2, \cdots, C_n\}$ just randomize the ciphertexts and obfuscate the order of the ciphertexts, and do not change the plaintexts in it. Therefore, P_1 always obtains the correct results.

Next, we give out the security proof of the above protocol.

Theorem 1. *Assume that π_{KeyGen}, π_{dec}, π_{isBit} and π_{perm} are as described in Section 2 and that (Gen, En, Dec) is the ElGamal scheme. Then Π_{SC} securely computes $\mathcal{F}_{SC}$ in the presence of malicious adversaries.*

Proof. We prove this theorem in the hybrid model, where a trusted party is used to compute the ideal functionality $\mathcal{F}_{KeyGen}$, $\mathcal{F}_{dec}$, $\mathcal{F}_{isBit}$ and $\mathcal{F}_{perm}$. We also separately analyze the case "P_1 is corrupted" and the case "P_2 is corrupted".

P_1 is Corrupted. Assume that P_1 is corrupted by adversary $\mathcal{A}$ with the auxiliary input z in the real model, we construct a simulator $\mathcal{S}$, who runs in the ideal model with the trusted party computing the functionality $\mathcal{F}_{SC}$. $\mathcal{S}$ works as follows:

1. $\mathcal{S}$ is given $\mathcal{A}$'s input and auxiliary input, and invokes $\mathcal{A}$ on these values.
2. $\mathcal{S}$ first emulates the trusted party for π_{keyGen} as follows. It first chooses two random elements $s_1, s_2 \in \mathbb{Z}_q$, and hands $\mathcal{A}$ s_1 and the public key $(\mathbb{G}_q, q, g, h = g^{s_1+s_2})$.
3. $\mathcal{S}$ receives from $\mathcal{A}$ n encryptions and $\mathcal{A}$'s input for the trusted party for $\mathcal{F}_{isBit}$, then defines $\mathcal{A}$'s inputs as X.
4. Then, $\mathcal{S}$ sends X to the trusted party computing $\mathcal{F}_{SC}$ to complete the simulation in the ideal model. Let $(n_{11}, n_{10}, n_{01}, n_{00})$ be the returned numbers from the trusted party.
5. Next, $\mathcal{S}$ randomly chooses $Y' = \{y_1', y_2', \cdots, y_n'\}$, conditioned on that the numbers of $(1,1), (1,0), (0,1), (0,0)$ in the set $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ equal to $(n_{11}, n_{10}, n_{01}, n_{00})$. $\mathcal{S}$ completes the execution as the honest P_2 would on inputs Y'.
6. If at any step, $\mathcal{A}$ sends an invalid message, $\mathcal{S}$ aborts, sends $\perp$ to the trusted party for $\mathcal{F}_{SC}$. Otherwise it outputs whatever $\mathcal{A}$ does.

The different part between the above simulation and the real hybrid model is that $\mathcal{S}$ who does not have the real P_2's input Y, simulates following steps with the randomly chosen parameter Y' under the condition that the outputs of them $(n_{11}, n_{10}, n_{01}, n_{00})$ are the same. The computationally indistinguishability of them can be deduced from the semantic security of ElGamal encryption. In other words, if $\mathcal{A}$ can distinguish the simulation from the real execution, we can construct a distinguisher $\mathcal{D}$ to attack the semantic security of ElGamal encryption. Since this reduction has been proved in [20], we omit it here. Next, the distribution of the zero-knowledge proofs can be assured by the definition of the zero-knowledge proof. Therefore, the distribution of the above simulation is computationally indistinguishable from the hybrid model. Since the distribution of hybrid model is also indistinguishable from the real model [4], therefore we prove the security in the case "P_1 is corrupted".

P_2 is corrupted. The proof of this part is similar with the above. We construct a simulator $\mathcal{S}$ in the ideal model, based on the real adversary $\mathcal{A}$ in the real model.

1. $\mathcal{S}$ is given $\mathcal{A}$'s input and auxiliary input, and invokes $\mathcal{A}$ on these values.
2. $\mathcal{S}$ first emulates the trusted party for π_{keyGen} as follows. It first chooses two random elements $s_1, s_2 \in \mathbb{Z}_q$, and hands $\mathcal{A}$ s_2 and the public key $(\mathbb{G}_q, q, g, h = g^{s_1+s_2})$.
3. $\mathcal{S}$ randomly chooses $X' = \{x'_1, x'_2, \cdots, x'_n\}$, where $x_i \in \{0, 1\}$, for $i \in \{1, 2, \cdots, n\}$, then encrypts them using the public key.
4. Next, $\mathcal{S}$ sends these ciphertexts to $\mathcal{A}$, and proves to $\mathcal{A}$ that all the plaintexts of them belong to the set $\{0, 1\}$ using π_{isBit}.
5. $\mathcal{S}$ receives from $\mathcal{A}$ n encryptions and $\mathcal{A}$'s input to the trusted party for $\mathcal{F}_{isBit}$, then defines $\mathcal{A}$'s inputs as Y.
6. Then, $\mathcal{S}$ completes the next steps as the honest P_1.
7. If at any step, $\mathcal{A}$ sends an invalid message, $\mathcal{S}$ aborts, and sends $\perp$ to the trusted party for $\mathcal{F}_{SC}$. Otherwise $\mathcal{S}$ sends Y to the trusted party computing $\mathcal{F}_{SC}$, and outputs whatever $\mathcal{A}$ does.

Similar to the case P_1 *is corrupted*, the difference between the simulation and the real model is that $\mathcal{S}$ uses X' as P_1's inputs. However, X' is encrypted by the public key of a semantic secure ElGamal encryption. Same as the above, the analysis of this simulation distribution can be assured by the zero-knowledge proof definition and semantic security of ElGamal encryption.

In summary, we complete the proof of Π_{SC} in the presence of malicious adversaries. □

Complexity of Π_{SC}. In our protocol, since the round of the zero-knowledge is constant, and the batched technology can be used to reduce the communication round (e.g., P_1 (P_2) can prove all his ciphertexts by using n zero-knowledge proofs for π_{isBit} in parallel), the round complexity is constant. Further, the number of group elements exchanged is bounded by $O(2n)$, as the length of the input and the number of zero-knowledge proofs are all equal to n. Also, Π_{SC} requires $O(2n)$ exponentiations and $O(2n)$ multiplications.

5 Extensions to Other Similarity Coefficients Computation

In this section, we give out several extensions of Π_{SC}, in order to settle the next problems:

- P_1 can only obtain the number n_{11};
- P_1 can only obtain the sum of the elements in the subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$;
- settling the same problem of other kinds of data, such as hex number system.

Next, we give out the constructions to settling the above problems.

5.1 Protocol Π_{SC1}

In protocol Π_{SC1}, P_1 can only obtain the number n_{11}, rather than $\{n_{11}, n_{10}, n_{01}, n_{00}\}$, as in some applications (e.g., [31]), P_1 only needs to know n_{11}/n. Privacy of P_2 will be leaked if we let P_1 get $\{n_{11}, n_{10}, n_{01}, n_{00}\}$. Therefore, we construct the protocol Π_{SC1}. This type of property can be realized by deleting the other kinds of pairs in $\{C_1, C_2, \cdots, C_n\}$ of Π_{SC}. After the protocol, we will give out a method to modify Π_{SC1} in order that P_1 obtains anyone of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$.

Next is the protocol, Π_{SC1}.

– **Inputs:** The input of P_1 is a binary set with order n: $X = \{x_1, x_2, \cdots, x_n\}$. Similarly, P_2's input is $Y = \{y_1, y_2, \cdots, y_n\}$.
– **Auxiliary inputs:** Both parties have the security parameter 1^k.
– **The protocol:**
1. P_1 and P_2 do the first four steps of Π_{SC}. Both parties have: for $i \in \{1, 2, \cdots, n\}$,

$$C_i = C_{x_i}^2 C_{y_i} = En(2x_i + y_i, 2r_{x_i} + r_{y_i}).$$

2. Assuming $C_i = (C_{i1}, C_{i2})$, both parties compute: for $i \in \{1, 2, \cdots, n\}$,

$$\hat{C}_i = (C_{i1}, C_{i2}/g^3) = (g^{2r_{x_i}+r_{y_i}}, g^{2x_i+y_i-3}h^{2r_{x_i}+r_{y_i}}) = En(2x_i + y_i - 3, 2r_{x_i} + r_{y_i}).$$

3. P_2 randomly chooses $k_1, k_2, \cdots, k_n$ from $\mathbb{Z}_q$, and computes $\hat{C}_i^{\,k_i}$, for $i \in \{1, 2, \cdots, n\}$. Then send them to P_1. Both parties execute π_{DL} on $(\hat{C}_i, \hat{C}_i^{k_i})$ allowing P_1 to verify that P_2 owns the solution k_i, for $i \in \{1, 2, \cdots, n\}$. For simplicity, we define $\bar{C}_i := \hat{C}_i^{\,k_i}$, for $i \in \{1, 2, \cdots, n\}$.
4. Next, P_2 rerandomizes and permutates $\{\bar{C}_1, \bar{C}_2, \cdots, \bar{C}_n\}$ as P_2 does in step 5 in Π_{SC}, and obtains $\{C_1'', C_2'', \cdots, C_n''\}$. Also P_1 and P_2 executes π_{perm} on $(\{\bar{C}_1, \bar{C}_2, \cdots, \bar{C}_n\}, \{C_1'', C_2'', \cdots, C_n''\})$ after sending $\{C_1'', C_2'', \cdots, C_n''\}$ to P_1.
5. Finally, P_1 and P_2 execute π_{dec} on each ciphertext in $\{C_1'', C_2'', \cdots, C_n''\}$. P_1 obtains the format of plaintexts $\{g^{m_1}, g^{m_2}, \cdots, g^{m_n}\}$ of $\{C_1'', C_2'', \cdots, C_n''\}$, and then denotes n_{11} as the number of $g^{m_i} = g^0 = 1$ in $\{g^{m_1}, g^{m_2}, \cdots, g^{m_n}\}$. At last, P_1 obtains the parameters n_{11}.

Correctness. All the plaintexts of $\{\bar{C}_1, \bar{C}_2, \cdots, \bar{C}_n\}$ are shuffled, except the plaintext 0. Thus P_1 only obtains the number of 0 in the plaintexts in the last step, which equals to the number of 3 in $\{2x_1 + y_1, 2x_2 + y_2, \cdots, 2x_n + y_n\}$. Therefore, this protocol can correctly compute out the parameter n_{11}, and securely protect the other types of data (numbers of $(1,0), (0,1), (0,0)$).
Remark. Parameters $k_1, k_2, \cdots, k_n$ only obfuscate the plaintexts of $\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_n$, except plaintext 0. For instance, assuming $C = En(m, r) = (g^r, g^m h^r)$, we have that $C^k = (g^{rk}, g^{mk}h^{rk})$. Thus if $m = 0$, the plaintext of C^k is still 0. However, if $m \neq 0$, the plaintext of C^k is uniformly distributed, as k is randomly selected. This is similar as π_{dec0} in [20]. We advance this exponentiation ahead, in order to reduce the computation for rerandomization.

Theorem 2. *Assume that π_{KeyGen}, π_{dec}, π_{DL}, π_{isBit} and π_{perm} are as described in Section 2 and that (Gen, En, Dec) is the ElGamal scheme. Then Π_{SC1} securely computes $\mathcal{F}_{SC1}$ in the presence of malicious adversaries.*

Formal simulation is analogous to that in Section 3. We omit it here.
Remark. In order to let P_1 only obtain n_{10}, we can modify step 2 of Π_{SC1}:

$$\hat{C}_i = (C_{i1}, C_{i2}/g^2) = (g^{2r_{x_i}+r_{y_i}}, g^{2x_i+y_i-2}h^{2r_{x_i}+r_{y_i}}) = En(2x_i + y_i - 2, 2r_{x_i} + r_{y_i}).$$

Similarly, P_1 can obtains n_{01} or n_{00} with the same modification. Therefore, the above protocol can perfectly settle the problem that P_1 only needs to obtain one parameter in $\{n_{11}, n_{10}, n_{01}, n_{00}\}$.

5.2 The Sum of Elements in Subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$

We can modify the protocol Π_{SC1} with a simple extension, in order that P_1 obtains the sum of elements in subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$, rather than only one number as Π_{SC1}.

For simplicity, we assume that P_1 wants to obtain $n_{11} + n_{10}$. Here we only give out the simple description about the modification. We modify step 2 of Π_{SC1}: for $i \in \{1, 2, \cdots, n\}$,

$$\hat{C}_i = (C_{i1}, C_{i2}/g^3) = En(2x_i + y_i - 3, 2r_{x_i} + r_{y_i}),$$
$$\hat{C}'_i = (C_{i1}, C_{i2}/g^2) = En(2x_i + y_i - 2, 2r_{x_i} + r_{y_i}).$$

Thus, both parties have $\{\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_n; \hat{C}'_1, \hat{C}'_2, \cdots, \hat{C}'_n\}$. Next steps are the same as Π_{SC1}, except that the permutation, rerandomization and other operations are over the above set $\{\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_n; \hat{C}'_1, \hat{C}'_2, \cdots, \hat{C}'_n\}$ with length $2n$ (rather than n).

Remark that the correctness and security are analogous to Π_{SC1}. The disadvantage of this modification is that the length has to extend to $2n$. Thus complexity is higher. However, this protocol can fully settle this problem that P_1 can only obtain $n_{11} + n_{10}$ in the presence of malicious adversaries.

Meanwhile, we can still make some exchanges on step 2, in order that P_1 obtains the other sum of elements in subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$. By extending the result of the step 2 to $\{\hat{C}_1, \hat{C}_2, \cdots, \hat{C}_n; \hat{C}'_1, \hat{C}'_2, \cdots, \hat{C}'_n; \hat{C}'_1, \hat{C}'_2, \cdots, \hat{C}'_n\}$, P_1 obtains the number $n_{11} + 2n_{10}$. Therefore, all the linear combinations of $n_{11}, n_{10}, n_{01}, n_{00}$ can be privately computed with the similar extension.

5.3 Settling the Other Kinds of Data

Our protocol can be easily extended to settle the same problem of other kinds of data (which can be represented by the binary number system), rather than that protocol in [35] only considers the similarity coefficients for binary data. Also it is hard to make a simple extension on protocol in [35] for settling other kinds of data.

Here, we give out a simple modification settling the hex number system. We assume that P_1 owns the set $X = \{x_1, x_2, \cdots, x_n\}$, P_2 owns the set $Y = \{y_1, y_2, \cdots, y_n\}$, where $x_i, y_i \in \{0, 1, \cdots, 9, A, B, \cdots, F\}$, for $i \in \{1, 2, \cdots, n\}$. In this case, x_i, y_i can be transformed to 4 bits, i.e., $x_i = x_{i4}||x_{i3}||x_{i2}||x_{i1}$, $y_i = y_{i4}||y_{i3}||y_{i2}||y_{i1}$, where $||$ denotes bit concatenation. Thus in step 4 of Π_{SC}, P_1 and P_2 computes

$$C_i = En(2^4(2^3 x_{i4} + 2^2 x_{i3} + 2x_{i2} + x_{i1}) + (2^3 y_{i_4} + 2^2 y_{i3} + 2y_{i2} + y_{i1}), \hat{r}),$$

where $\hat{r}$ denotes the corresponding random parameter. The other executions are the same with Π_{SC}.

With the similar method as the above, other kinds of data, which can be represented by the binary number system, can be privately computed.

6 Conclusion

In this paper, we first analyze the security of the protocol in [35], and propose an attack method retrieving P_2's input without deviating from the protocol. Also we point out that

the correctness of the protocol in [35] is not sound. Meanwhile, we construct a secure protocol Π_{SC} computing the functionality $\mathcal{F}_{SC}$ in the presence of malicious adversaries, and also give out its standard simulation-based security proof. Extensions of Π_{SC} are also given, in order to protect the privacy of P_2 while P_1 only needs to obtain the sum of element in the subset of $\{n_{11}, n_{10}, n_{01}, n_{00}\}$. For other kinds of data (can be expressed by binary number system), we also give out a simple solution.

References

1. 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.
2. Felix Brandt. Efficient cryptographic protocol design based on distributed ElGamal encryption. In ICISC, pp. 32-47, 2005.
3. D. Boneh, E. Goh and K. Nissim. Evaluating 2-DNF formulas on ciphertexts. TCC 2005, LNCS 3378, pp. 325-342, Springer, Heidelberg (2005).
4. R. Canetti. Security and Composition of Multiparty Cryptographic Protocols. Journal of Cryptology, 13(1):143-202, 2000.
5. Ronald Cramer, Rosario Gennaro, and Berry Schoenmakers. A secure and optimally efficient multi-authority election scheme. In EUROCRYPT, pp. 103-118, 1997.
6. D. Chaum and T.P. Pedersen. Wallet databases with observers. CRYPTO 92, LNCS 740, pp. 89-105, Springer, Heidelberg (1992).
7. E.F. Connor, D. Simberloff, Intraspecific competition and species co-occurrence patterns on Islands, Oikos 41 (1983) 455-465.
8. W. Du and M.J. Atallah. Privacy-Preserving Cooperative Scientific Computations, IEEE Computer Security Foundations Workshop, pp. 273-282, 2001.
9. W. Diffie and M.E. Hellman. New directions in cryptography. IEEE Transctions on Information Theory, 22(6):644-654, November 1976.
10. 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.
11. T. El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. CRYPTO 85. LNCS 482, pp. 10-18, Springer, Heidelberg (1985).
12. D.H. Fisher, Knowledge acquisition via incremental conceptual clustering, Machine Learning 2 (1987) 139-172.
13. R. Fagin, M. Naor, P. Winkler, Comparing information without leaking it, Communications of the ACM 39 (1996) 77-85.
14. O. Goldreich. Basic Application. Foundations of Cryptography, vol.2, Cambridge University Press, 2004.
15. Jens Groth. A verifiable secret shuffle of homomorphic encryptions. In Public Key Cryptography, pp. 145-160, 2003.
16. M.E. Gilpin, J.M. Diamond, Factors contributing to non-randomness in species Co-occurrences on Islands, Oecologia 52 (1982) 75-84.
17. O. Goldreich, S. Micali, and A. Wigderson. How to play Any mental game. STOC 1987, pp. 218-229, 1987.
18. M.E. Hohn, Binary coefficients: a theoretical and empirical study, Mathematical Geology 8 (1976) 137-150.
19. Z. Hubáek, Coefficient of association and similarity: based on binary (presenceCabsence) data: an evaluation, Biological Reviews 57 (1982) 669-689.
20. C. Hazay and T. Toft. Computationally Secure Pattern Matching in the Presence of Malicious Adversaries. ASIACRYPT 2010. LNCS 6477, pp. 195-212. Springer, Heidelberg (2010).
21. P. Jaccard, Nouvelles recherches sur la distribution florale, Bulletin de la Socite Vaudense des Sciences Naturelles 44 (1908) 223-270.

22. 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.
23. H. Lipmaa, Verifiable homomorphic oblivious transfer and private equality test, in: Advances in Cryptology, Springer-Verlag, Taipei, Taiwan, 2003, pp. 416-433.
24. 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.
25. Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adveraries. EURPCRYPT 2007, LNCS 4515, pp. 52-78, 2007.
26. Y. Lindell and B. Pinkas. Secure two-party computation via cut-and-choose oblivious transfer. TCC 2011, LNCS 6597, pp. 329-346, 2011. Full version in Cryptology ePrint Archive, report 2010/284.
27. D. Malkhi, N. Nisan, B. Pinkas and Y. Sella. Fairplay - a secure two-party computation system. USENIX Security Symposium, pp. 287-302, 2004.
28. P. Paillier. Public-key cryptosystems based on composite degree residuosity classes. EUROCRYPTO 91, LNCS 1592, pp. 223-238, Springer Heidelberg (1999).
29. Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In CRYPTO, pages 129-140, 1991.
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. 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.
32. C.P. Schnorr. Efficient identification and signatures for smart cards. CRYPTO 89, pp. 239-252, Springer Heidelberg (1989).
33. R.R. Sokal, C.D. Michener, A statistical method for evaluating systematic relationships, University of Kansas Science Bulletin 38 (1958) 1409-1438.
34. P. Willett, Similarity-based approaches to virtual screening, Biochemical Society Transactions 31 (2003) 603-606.
35. K.-S. Wong, M.H. Kim, Privacy-Preserving similarity coefficients for binary data, Computers and Mathematics with Applications (2012), doi: 10.1016/j.camwa.2012.02.028.
36. 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.
37. A.C. Yao. How to generate and exchange secrets, FOCS 1986, pp. 162-167, 1986.
38. 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.

A Security in the Presence of Malicious Adversaries

In this section, we briefly give out the standard security definition in the presence of malicious adversary for two-party computation. For more details, please refer to [14].

Two-party Computation. A two-party computation could be simply described as a random process that maps two random inputs to pairs of outputs. We use a function to denote the process as $f : \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^* \times \{0,1\}^*$, where $f = (f_1, f_2)$. That is, for every pairs of inputs (x, y), the process outputs $(f_1(x, y), f_2(x, y))$, where P_1 gets $f_1(x, y)$, P_2 gets $f_2(x, y)$.

Adversary behavior. Loosely speaking, the aim of secure two-party computation is to protect an honest party against a dishonest party. In a malicious model, the corrupted party could do anything during the protocol. It may abort the protocol at any time, may send

the wrong input, etc. Here we consider the malicious adversary, in order to give out the security definition in the presence of malicious adversary.

Security of Protocols (Informal). According to the comparison what an adversary can do in a real protocol execution to what it can do in an ideal scenario, we analyze the security of the protocol. In the ideal scenario, there exists a trusted party, to whom parties P_1 and P_2 send their inputs. After getting these inputs, the trusted party computes correctly and sends back the outputs for them. A protocol is secure if any adversary interacting in the real protocol can do no more harm than if it was involved in the above described ideal computation. Next, we describe the detailed definition of malicious security.

Execution in the Ideal Model.

- **Inputs:** P_1 obtains his input x, P_2 obtains his input y ($|x| = |y|$). The adversary $\mathcal{A}$ obtains the auxiliary input z.
- **Sends Inputs to the trusted party:** An honest party correctly sends his inputs to the trusted party. The corrupted party (e.g., P_1) controlled by $\mathcal{A}$ may, depending on his input (e.g., x) and z, either abort or send another input (e.g., x', $|x'| = |x|$) to the trusted party.
- **The trusted party answers the first party:** In case it has obtained an input pair (x, y), then the trusted party first reply to the first party $f_1(x, y)$. Otherwise (i.e., in case it receives only one input), the trusted party replies to both parties with a special symbol $\perp$.
- **The trusted party answers the second party:** In case the first party is malicious, it may decide to stop the the trusted party by sending $\perp$ after receiving his output, depending on its input and the trusted party's answers. In this case, the trusted party sends $\perp$ to the second party. Otherwise, the trusted party sends $f_2(x, y)$ to the second party.
- **Outputs:** An honest party always outputs the message it has obtained from the trusted party. A malicious party may output an arbitrary function of its input, the auxiliary input z and the message obtained from the the trusted party.

Let f be a two-party functionality, where $f = (f_1, f_2)$, let $\mathcal{A}$ be a nonuniform PPT machine, and let $I \subseteq [2]$ be the set of corrupted parties, i.e., at least one party is honest. The **ideal execution** of f on inputs (x, y), auxiliary input z to $\mathcal{A}$ and security parameter l, denoted $IDEAL_{f,\mathcal{A}(z),I}(x, y, l)$, is defined as the output pair of the honest party and the adversary $\mathcal{A}$ from the above ideal execution.

Execution in the Real Model. The malicious adversary may follow an arbitrary feasible strategy in the real model. Let f be as above and Π be as a two-party protocol for computing f. Furthermore, let $\mathcal{A}$ be a nonuniform PPT machine and I be the corrupted party. Then the **real execution** of Π on inputs (x, y), auxiliary input z to $\mathcal{A}$ and security parameter l, denoted by $REAL_{\Pi,\mathcal{A}(z),I}(x, y, l)$, is defined as the output vector of the honest parties and the adversary $\mathcal{A}$ from the real execution of Π.

Security as Emulation of a Real Execution in the Ideal Model. Having defined the ideal and real models, we can define security of protocols. The definition asserts that

a secure party protocol (in the real model) emulates the ideal model (in which the trusted party exists). This is formulated by saying that adversaries in the ideal model are able to simulate execution of the real model protocol.

Definition 1. *Let f and Π be as above. Protocol Π is said to* ***securely compute*** *f* ***with abort in the presence of malicious adversaries*** *if for every nonuniform probabilistic polynomial-time adversary $\mathcal{A}$ for the real model, there exists a nonuniform probabilistic polynomial-time adversary $\mathcal{S}$ for the ideal model such that for every $I \subseteq [2]$,*

$$\{IDEAL_{f,\mathcal{S}(z),I}(x,y,n)\}_{x,y,z\in\{0,1\}^*,n\in\mathbb{N}} \stackrel{c}{\equiv} \{REAL_{\Pi,\mathcal{A}(z),I}(x,y,n)\}_{x,y,z\in\{0,1\}^*,n\in\mathbb{N}},$$

where $|x| = |y|$.