Arabian Journal for Science and Engineering (2018) 43:7747–7756
https://doi.org/10.1007/s13369-018-3122-5

RESEARCH ARTICLE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

Privacy-Preserving Secure Multiparty Computation on Electronic Medical Records for Star Exchange Topology

Ahmed M. Tawfik[1] · Sahar F. Sabbeh[2,3] · Tarek EL-Shishtawy[3]

Received: 7 August 2017 / Accepted: 4 February 2018 / Published online: 12 March 2018

Abstract
Nowadays, a huge amount of data are available and shared in collaborative scenarios. These scenarios exist due to the need for joint computations of cooperating data owners for the purpose of making analysis and knowledge extraction. This requirement comes together with some privacy issues. One major issue is how to enable query execution, while no party is allowed to see the entire dataset (computational privacy). Thus, secure multiparty computation protocols allow a group of distrustful data owners to jointly cooperate in executing analytical queries against their data while revealing nothing about the entire dataset. In this paper, we propose a technique that enables a privacy-preserving query processing on horizontally partitioned electronic medical records among a set of hospitals, which have no desire to share their confidential data; however, they all need to cooperate to answer global queries about patients' medical history. The proposed technique depends on a bucketization technique to reduce computational costs. It relies on a head party, which acts as a mediator between the authorized users and the cooperating parties, which are arranged in a star exchange topology. It ensures that the head party learns nothing about the sensitive data. Our experimental results prove that our technique provides a smaller computational cost and better privacy without the need for a trusted third party.

Keywords Privacy preservation · Electronic medical records · Secure multiparty computation · Star exchange topology · Bucketization technique · Trusted third party

1 Introduction

The revolution of web technologies, cloud computing, and distributed data processing made hospitals and medical institutions more aware of the benefits of transitioning toward a digital version of paper-based medical records (EMRs) [1] for a higher quality of patient care.

EMRs contain both the medical and the treatment history of the patient to enable patients' medical data sharing whenever it is needed. They are used as tools for building new medical technologies (e.g., drug repositioning and genome-wide association study) and the patient recommendation systems [2]. However, patients' information at one institution should not be revealed to the public or even to the other participants (Privacy) [3]. Thus, privacy is a major issue that should be considered in the shared environment scenarios. As, in the last few decades, security and privacy issues [4,5] are becoming a major concern in many trends (i.e., data mining, confidential data analysis, cloud computing, and Internet of things (IoT)…, etc.).
For this purpose, secure multiparty computation (SMC) is intended to allow a group of parties to evaluate a common function in a secure manner, i.e., the input of each party remains private and not disclosed by any of the other parties. Research work focuses on the protocols, which support SMC while preserving data privacy (privacy-preserving SMC protocols).

These protocols in this literature are classified into two categories: (a) the real model, where parties cooperate without the need for a trusted third party (TTP), and (b) the ideal model, where parties rely on a centralized TTP for computations [6].

✉ Ahmed M. Tawfik
ahmed.tawfek@fci.bu.edu.eg

1 Information Technology Department, Faculty of Computers and Informatics, Benha University, Benha, Egypt

2 Information Systems Department, Faculty of Computers and Information Technology, King Abdulaziz University, Jeddah, Kingdom of Saudi Arabia

3 Information Systems Department, Faculty of Computers and Informatics, Benha University, Benha, Egypt

The major contributions of this paper are: (a) to propose the real model of privacy-preserving SMC on horizontally partitioned EMRs, based on a bucketization technique and (b) to adopt and extend the work of Maryam et al. [7] by moving from the ring to star exchange topology to support an end-to-end encryption and better computational results.

The proposed technique uses the commutative encryption [8] to encrypt the patients' searchable attributes and symmetric encryption scheme to encrypt patients' sensitive data.

The remainder of the paper is organized as follows: Sect. 2 introduces the related studies; Sect. 3 describes the proposed system; experimental results are discussed in Sect. 4; and finally, our paper is concluded and our future work is presented in Sect. 5.

2 Related Work

This section sheds some light on state-of-the-art and related work of privacy-preserving SMC protocols.

Privacy-preserving SMC protocols enable a group of distrustful parties to compute any function over distributed inputs, revealing nothing about the entire dataset. The research work related to privacy-preserving SMC can be based on either garbled circuits [9–19], secret sharing [20–26], oblivious polynomial evaluation [27–30], homomorphic encryption [31–36], or commutative encryption [8,37–39].

Garbled circuit protocol [9–19] , which has been developed by Andrew Yao, is an encrypted version of the binary circuit for preserving the privacy of function inputs. It relies on symmetric cryptography and an oblivious transfer protocol. One party (the generator) generates the garbled circuit by building the binary circuit, choosing a pair of encryption keys for every wire of the circuit, and encrypting the output wire keys using the keys of the input wires. The generator then sends the garbled circuit and the input keys that correspond to his inputs to the second party (the evaluator). The evaluator obtains the keys corresponding to his inputs by engaging in an oblivious transfer with the generator. Using the obtained input keys, the evaluator can then decrypt the garbled circuit to obtain the result while learning no intermediary information. However, Yao's protocol has a high communication complexity and requires the function and input sizes to be known in advance to allow pre-computation [25], To be more specific, the oblivious transfer stage requires one exponentiation (e.g., public key encryption) per bit of input.

Secret-sharing protocol [20–26] allows the private data to be split into a set of encrypted shares and divides them between the participants. It requires some random encrypted data in preprocessing phase to be combined with encrypted shares during the computation. These shares are processed in privacy manner and then accumulated to get the output result. Goldreich–Micali–Wigderson (GMW) protocol [20], which is similar to garbled circuit protocol, uses a binary circuit representation of the function but performs the security evaluation on shares rather than using encrypted gates. The parties secretly share their inputs using an XOR secret-sharing scheme. To evaluate an XOR gate, the parties simply XOR the shares of the input wires. To evaluate an AND gate, the parties perform an oblivious transfer: One party pre-computes all possible outputs of the gate and the other party obliviously obtains the output that corresponds to its input shares. To obtain the output of the circuit, the parties exchange the shares of the output wires. The GMW protocol allows the pre-computation of all symmetric cryptographic operations before the function, or the inputs to the function are known. It requires less communication per AND gate than Yao's garbled circuit protocol. However, the GMW protocol requires a number of communication rounds that are linear in the depth of the circuit [25].

An oblivious polynomial evaluation (OPE) protocol [27–30] has been developed by Naor and Pinkas. The sender, who has a polynomial function F, and the receiver, who has an input x, want to jointly compute $F(x)$ so that the sender learns nothing about x, and the receiver learns nothing except the output of $F(x)$. To compare two items x and y by using OPE protocol, the receiver and sender generate random linear polynomial functions P and Q, respectively, to compute the two values so the receiver computes $R = P(x) + Q(y)$, and the sender computes $S = P(y) + Q(x)$. If the two computed values of R and S are the same, then $x = y$; otherwise, they are different with high probability. In case the receiver and the sender have a list of z inputs, the protocol requires each party to perform z oblivious evaluations of a polynomial of degree n with the communication cost of $O(n)$. This protocol is considered too expensive to implement in the multiparty computation [40].

Homomorphic encryption allows a set of computations to be executed on a ciphertext, and then these computations will generate an encrypted result, which matches the result of operations performed on the plaintext when decrypted. It has been classified as a partial homomorphic encryption and fully homomorphic encryption. The partial homomorphic encryption performs one type of operation (addition or multiplication), for computing specific-purpose functions [32–36]. Fully homomorphic encryption (FHE) was proposed by Rivest et al. [31] to perform a set of homomorphic operations, such as addition and multiplication. Homomorphic encryption requires computationally expensive public-key operations that scale very inefficiently for larger security parameters [25]. As mentioned in [7], while this technique offers strong privacy guarantee, it does not scale well for large-size data because of using heavy weight cryptographic operation among parties.

Commutative encryption [8,37–39], which has been developed by Agrawal et al., enables a plaintext to be encrypted more than once using different users' public keys. The resulted ciphertext can be decrypted without considering the order of public keys used in the encryption/re-encryption processes. In other words, the order of keys, used in encryption and in decryption processes, does not affect the computational result. If there is a pair of encryption functions F and G to encrypt a value v, the result of the encryption scheme will be $F(G(v)) = G(F(v))$. Thus, by using the combination $F(G(v))$ or $G(F(v))$ to encrypt v, we can ensure that one data owner cannot compute the encryption/decryption of the value v without the help of the other data owner.

The proposed work aims to handle privacy-preserving SMC for computing set-intersection queries. Early SMC protocols only supported an environment setting that includes two parties only [9,20]. The relationship between SMC and privacy-preserving SMC set-operations protocols came later, as a development of the idea of using MPC for computing set intersections in the privacy-preserving manner [7,8,37,41–44]. The evaluation of set-intersection queries has been addressed in many studies.

Freedman et al. [41] proposed a technique for set-intersection queries, based on homomorphic encryption. In their protocol, each party P_i $\forall i \in \{1, , m-1\}$ sends a polynomial F_i to P_m. The F_i polynomial has degree n and is rooted in P_i items. Then, P_m, for each item x in his list, sends $n(m-1)$ matrix that is built in the point x of polynomials previously received from other parties. Receiving parties decrypt and combine the evaluations to determine whether their items belong to the intersection.

Kissner and Song [43] designed a technique for set-intersection problem using OPE. Their protocol relies on randomly selected polynomials zeroing at the data values. Each data owner encrypts its polynomials and sends them to other data owners. Then, all data owners jointly perform a decryption process to compute the operation result. While Kissner–Song's polynomial technique looks effective, using it to support a privacy-preserving SMC for the set-intersection query poses some hard scalability problems [7].

Sang et al. [42] adopted a distinct technique using OPE protocol. It yet provides lower computation and communication costs with respect to Freedman et al. approach [45].

Vaidya and Clifton [37] proposed SMC technique for the set-intersection queries based on commutative encryption as an extension to Agrawal et al. protocol. This technique has lower complexity than [41,42].

Sepehri et al. [45] proposed the time complexity comparison of solutions [8,41–43] for set-intersection queries and found that the Agrawal et al. protocol has provided lower complexity than the others, as shown in Table 1.

Table 1 Time complexity comparison of solutions for set-operation problem (n is the Paillier cryptosystem RSA modulus, c is a suitable constant, and m is the number of participants)

Method	Efficiency requirements	
	Computation cost	Communication cost
Freedman et al. [41]	$O(s(s+m)\lg N)$	$10s(m-1)2\lg N$
Kissner et al. [43]	$O(s^2 \lg N))$	$2m(5s+2)\lg N$
Sang et al. [42]	$O(sm\lg N) + c$	$2m(4s+5)\lg N$
Agrawal et al. [8]	$O(sm\lg N)$	$((m-1)s+(m-2)s+1+(m-1))\lg N$

Li et al. [44] proposed a protocol that involves a TTP to compare the values held by two other parties. The protocol is based on homomorphic encryption scheme. Although the protocol is faster than OPE as mentioned in [45], it has two main drawbacks: (1) The TTP should be trusted by all parties; (2) When the number of parties increases, the solution does not scale because of the communication and computation bottleneck created at TTP.

Maryam et al. [7] proposed a privacy-preserving SMC technique for set-intersection queries, which are executed on horizontally partitioned data. These data are held by different data owners, which are arranged in the ring exchange topology. Data owners use the commutative encryption protocol to compute user queries on entire relations without sharing their private partitions. Although the work has succeeded to reduce the cost of both communication and computation through the usage of bucketization, it has two main drawbacks: (1) This technique uses a TTP to begin the protocol execution and (2) the system supposed a ring exchange topology in which all parties communicate with each other in a closed loop. Each party communicates only with two adjacent parties on either side. In such environment, data travel in one direction.

The bulk of the aforementioned studies have been tailored to the scenario of the multiparty setting where each one has a different query type. These studies have involved a TTP to prevent data leakage in query computation.

Table 2 is formed to differentiate the SMC techniques for each paper mentioned in the related work section. In more detail, in Table 2 could observe independently for each related review that has been studied useful information related to the authors, SMC technique, query type, exchange topology, SMC paradigms, and adversary model for each paper.

In the remainder of this paper, we will focus on a privacy-preserving SMC technique where the data owners are arranged in star exchange topology and hold some rows from the horizontally partitioned EMRs. The proposed work is based on the commutative encryption technique, which provides a smaller computational cost and better privacy, to handle SMC for computing set-intersection queries without the need for a TTP.

Table 2 The research work related to privacy-preserving SMC

Authors	SMC technique	Query type	Exchange topology	SMC paradigm	Adversary model
Freedman et al. [41]	Homomorphic encryption	Set-intersection	N/A	Ideal model	Semi-honest and malicious models
Kissner et al. [43]	Oblivious polynomial evaluation	Union and set-intersection	N/A	Ideal model	Honest-but-curious model
Sang et al. [42]	Oblivious polynomial evaluation	Set intersection and set matching	N/A	N/A	Semi-honest model
Vaidya and Clifton [37]	Commutative encryption	Set intersection	Tree	Ideal model	Semi-honest model
Li et al. [44]	Homomorphic encryption	Equality test	N/A	Ideal model	Semi-honest model
Maryam et al. [7]	Commutative encryption	Selection and equi-join	Ring	Ideal model	Semi-honest and malicious model

3 Proposed System

In this section, we propose the real model privacy-preserving SMC technique. In this technique, the data owners are arranged in a star exchange topology environment. The proposed system is based on a bucketization technique and targets set-intersection queries. We adopted the work of Maryam et al. [7], where it has two main drawbacks that we try to overcome:

1. First, the system supposed a ring exchange topology in which all parties communicate with each other in a closed loop. Each party communicates only with two adjacent parties on either side. In such environment, data travel in one direction. Such environment suffers from being slow, where if the number of the parties increases, the execution time of the system increases, due to the communication cost between all parties. Moreover, one malfunctioning party can affect all the other parties.
2. Second, the parties, in the beginning of the protocol execution, depend on a TTP, which may become a malicious or even honest but curious (semi-honest) attacker. Moreover, the TTP builds an interchange matrix W of the data owners' permutations to apply a bucketization technique and sends the row vectors of W to the data owners in order to select the buckets which correspond to user value, but this strategy is risky because disclosing the private input of the data owners even to the TTP should be avoided.

In our proposed system, we try to overcome these drawbacks by arranging the data owners in a star exchange topology instead of the ring exchange topology. In the proposed technique, there is a head party, which acts as a mediator between all the participants. In the star exchange topology, we do not need to build an interchange matrix of participants' permutations to apply the bucketization technique, but we could apply that through the head party, where

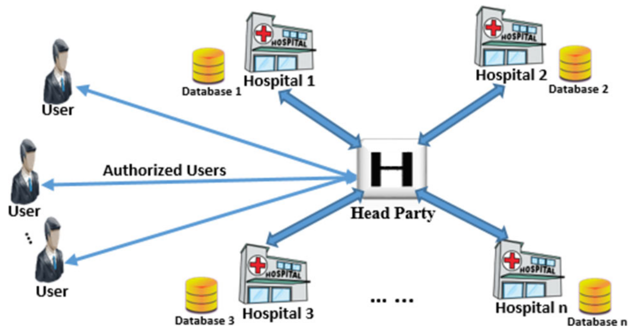


Fig. 1 System scenario

each party directly communicates with it without revealing its inputs. This topology can overcome the traditional shortcomings of the ring topology and provides more stability to the multiparty environment.

Moreover, in the proposed system, the participants no longer need to rely on the TTP, while the system provides a head party, which acts as a mediator between the data owners and the authorized users without revealing either the sensitive data, or the buckets' numbers of the data owners to provide end-to-end encryption (i.e., each node communicates with the head party in a secure manner).

Our proposed system involves a set of data owners (Hospitals), as illustrated in Fig. 1, with a common database that has been partitioned horizontally among them, and a set of authorized users[1] who are able to query the database.

One of our major goals is to achieve both data and query privacy, where the user knows only the query results while data owners learn nothing about the query.

[1] Authentication and access control are not the main focus in this paper, and we assume that authorizations between data owners and users are appropriately managed.

3.1 Secure Multiparty Computation for Set-Intersection Queries

In the proposed technique, the sensitive data of each data owner include one searchable attribute $T_{i,A}$ that includes a set of values V_A and one sensitive attribute $T_{i,B}$ that contains a group of values V_B. Our protocol adopts a bucketization technique on the searchable attribute to ensure scalability.

Buckets are defined by dividing the domain of each searchable attribute A into S buckets of size L as defined by the following Eq. (1):

$$L = \frac{A_{\max} - A_{\min}}{S} \tag{1}$$

Public bucketization $(BU) = \{B_1 : [A_{\min}, L], \ldots, B_S : (L_{(S-1)}, A_{\max}]\}$. It is accessible to all data owners and authorized users as well, where $A_{\max}$ and $A_{\min}$ are the maximum and minimum values in the domain of each searchable attribute A, respectively. The buckets are assigned by a unique identifier (ID) as defined in BU. The bucketization improves the efficiency of our protocol by reducing the search space, which only considers the records that are relative to the user's searching value.

3.2 The Protocol

Our proposed protocol has two phases:

Phase 1: Computation of permutation vectors

Input	Set of data owners O, number of buckets S
Output	Owner permutation vector W_i, Head permutation vector H
Step 1	Each data owner O_i, $1 \leq i \leq M$, selects its private permutation $\pi_i = (\pi_{i1}, \ldots, \pi_{iS})$ of bucket indexes $(1, 2, \ldots, S)$, where S is the number of buckets and M is the number of data owners
Step 2	Each data owner O_i separately computes its private permutations $(B_{\pi_{i1}}, \ldots, B_{\pi_{iS}})$ for the searchable attribute of the public bucketization scheme
Step 3	The head party chooses its own permutation H to send it to the data owners
Step 4	Each data owner O_i computes his vector W where the vector elements are defined by the following Eq. (2). In the following equation, we denote by $H^{-1}(j)$ the position in vector H^{-1} that contains value j $W(j) = \pi_\delta \quad \forall j \in \{1, 2, , S\}$, $\delta = H^{-1}(j)$ (2)
Step 5	The head party sends its permutation H to the user who is initiating the query

Phase 2: Query protocol

Input	User query value $= V_R$. Set of buckets S with values $\langle V_A, V_B \rangle$ for each data owner, where V_A is the searchable attribute and V_B is the extra attribute
Output	Values of V_B for $V_A \cap V_R$
Step 1	Both user and data owners O apply hash function h to their values, $V' = h(V_R)$ and $T'_{i,A} = h(V_{i,A}) \forall i \in \{1, \ldots, M\}$, respectively, where M is the number of data owners, V' is the user hashed value, and $T'_{i,A}$ is the searchable attribute hashed value for each data owner
Step 2	User and data owners randomly choose commutative encryption keys, k_r and $\langle k_i, k'_i \rangle$, respectively
Step 3	User encrypts his hashed value and then sends his encrypted hash value $Y_R = f_{k_r}(V')$ to the head party
Step 4	Each one of the data owners O_i, $1 \leq i \leq M$, performs the following: 1. Computes $f_{k_i}(T'_{i,A}) = Y_i = y_i = f_{k_i}(x) \mid x \in V'_{i,A}$ 2. Generates new symmetric keys, one for each value of the sensitive attribute B, as $K_i^B = \{k_{i_x} = f_{k'_i}(x) \mid x \in V_{i,A'}\}$ 3. Encrypts each value x in $T_{i,B}$ with the corresponding key k_{i_x} to obtain $Y_i^B = \{E_{k_x^i}(u) \mid u \in V_{i,B}\}$ 4. Computes $I_i = \{f_{k'_i}(Y_R)\}$ for the purpose of decrypting the values of the sensitive attribute B at the user site 5. O_i randomly reorders the tuples Y_i and Y_i^B
Step 5	The user determines his B_{H_k} from the public bucketization and sends it to the head party to span it to all data owners, where k is the index of a bucket which contains the user's query value
Step 6	At this step, each data owner O_i selects his bucket which corresponds to B_{H_k} of the head party
Step 7	Each data owner sends $\langle Y_i, Y_i^B, I_i \rangle$ after randomly reordering to the head party in the star exchange topology environment
Step 8	Head party receives all tuples and then initiates Agrawal's two-party set-intersections protocol between the user and the head party
Step 9	Head party passes Y_R through the star exchange topology environment to encrypt it by all encryption keys $k_1, \ldots, k_m$ for obtaining $Y'_{R_i} = f_{k_i}(f_{k_r}(V'))$, and after that, the head party sends Y'_{R_i} together with $\langle Y_i, Y_i^B, I_i \rangle$ to the user who initiated the query, for each i where $1 \leq i \leq M$
Step 10	The user then decrypts Y'_{R_i} using his commutative key k_r to get $Y''_{R_i} = f_{k_i}(V'_{i,A})$
Step 11	For each i where $1 \leq i \leq M$, the user performs the following: 1. Finds all the tuples in Y_i which are equal to the value of Y''_{R_i} 2. Determines the sensitive attributes which correspond to those tuples 3. Decrypts I_i with his own commutative key k_r, to obtain $f_{k'_i}(V'_{i,A})$ 4. Uses $f_{k'_i}(V'_{i,A})$ to decrypt the sensitive attributes in Y_i^B that correspond to Y_i in which their values are equal to query value of the user

3.3 Example

We suppose that there are four data owners/hospitals, arranged in a star exchange topology, where each data owner has one searchable attribute (e.g., patient age) and one sensitive attribute (e.g., patient disease). We assume that the domain of the patients' age includes a group of values in the range [1, 100]. We consider the range of the values divided into five buckets of the same size. Thus, the protocol creates five buckets as $BU = \{B_1 : [1, 20], B_2 : (20, 40], B_3 : (40, 60], B_4 : (60, 80], B_5 : (80, 100]\}$.

– BU is accessible to all data owners and authorized users as well.
– We assume that each data owner privately chooses his permutation, which is a random vector of BU's positions, as the following:

$$\pi_1 = (3, 2, 1, 4, 5)$$
$$\pi_2 = (5, 2, 1, 3, 4)$$
$$\pi_3 = (3, 2, 1, 5, 4)$$
$$\pi_4 = (1, 4, 3, 5, 2)$$

– For instance, π_1 shows that data owner O_1 selects his private permutation as $B_1 : (40, 60]$, $B_2 : (20, 40]$, $B_3 : [1, 20]$, $B_4 : (60, 80]$, and $B_5 : (80, 100]$ from BU boundaries; moreover, π_2 shows that the second data owner O_2 selects his private permutation as $B_1 : (80, 100]$, $B_2 : (20, 40]$, $B_3 : [1, 20]$, $B_4 : (40, 60]$, and $B_5 : (60, 80]$ and the other data owners do the same.
– The head party works as a mediator (not as a TTP) between the data owners and the authorized users, so it does not know the private permutations of the data owners. We suppose that the head party randomly chooses its permutation H as the following:

$$H = (5, 3, 1, 4, 2)$$

– Each data owner computes a permutation of vector W, which works as a link between the head party permutation H and the private permutation of the data owner, by applying Eq. (2), which is defined in phase 1 of the protocol.
– The rationale behind Eq. (2) is generating the elements of the vector W by determining the corresponding element of the private permutation vector π. The vector W of each data owner is obtained by looking for the position of index j in the H permutation vector.
– For instance, to compute the first element of vector W_1, the protocol follows this procedure: $W_1(j = 1) = \pi_{1,\delta}, \quad \forall\, \delta = H^{-1}(j = 1) = 3$.
– Hence, $W_1(j = 1) = \pi_{1,3} = 1$, which refers to the 3^{rd} element in the vector π_1. To make (2) more clear, the second element of vector W_1 is calculated by applying this procedure:
$W_1(j = 2) = \pi_{1,\delta}, \quad \forall\, \delta = H^{-1}(j = 2) = 5$. Hence, $W_1(j = 2) = \pi_{1,5} = 5$, which refers to the 5th element in the vector π_1.
– Each data owner computes his W permutation vector, and then, the head party sends its permutation to the user.

$$W_1 = (1, 5, 2, 4, 3)$$
$$W_2 = (1, 4, 2, 3, 5)$$
$$W_3 = (1, 4, 2, 5, 3)$$
$$W_4 = (3, 2, 4, 5, 1)$$
$$H = (5, 3, 1, 4, 2) \rightarrow u$$

– The user determines his bucket number from the BU and sends it to the head party to span it to all data owners, and then, the data owners get their bucket ID which corresponds to user value, as shown in Fig. 2.
– Each data owner selects his bucket based on the user's searching value, as shown in Fig. 3.
– The head party collects all encrypted tuples from data owners, as shown in Fig. 4.
– The head party passes Y_R through the star exchange topology environment in order to have it encrypted with all the keys of the data owners and then gets Y'_R, as shown in Fig. 5.
– The user decrypts Y'_R with his commutative decryption key, to get Y''_R, so he could get all the tuples in which their values of the encrypted searchable attributes are equal to Y''_R as mentioned in these steps 10 and 11 of query protocol, as shown in Fig. 6.

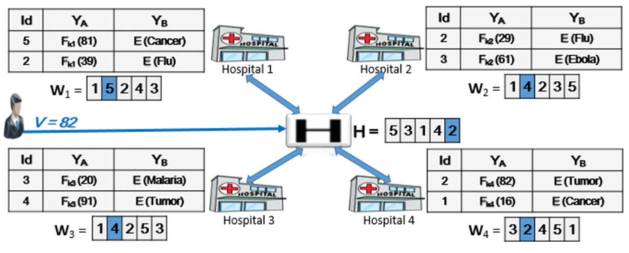


Fig. 2 Data owners get their bucket IDs that correspond to user value

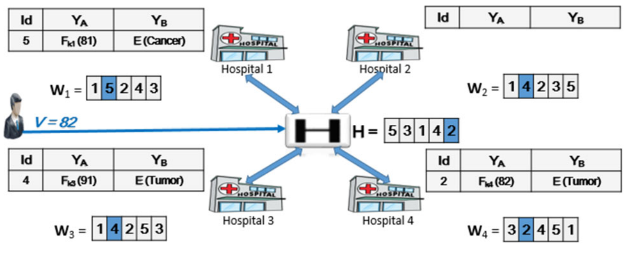
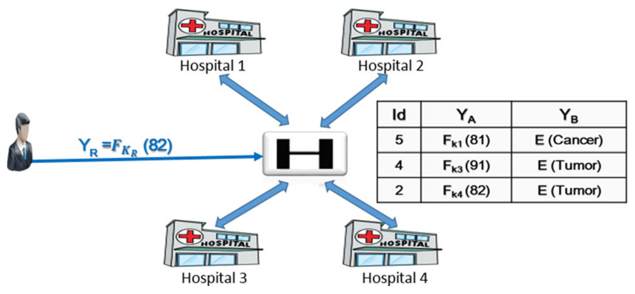
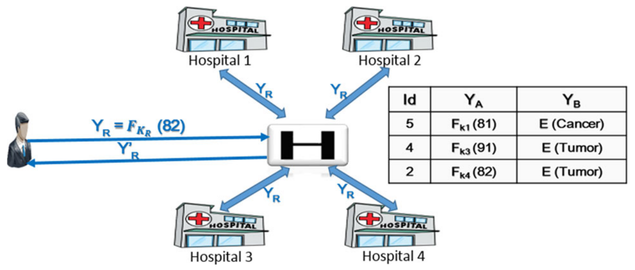


Fig. 3 Each data owner selects his bucket that correspond to user value



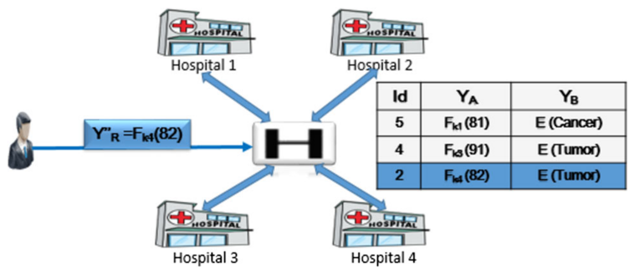
Id	Y_A	Y_B
5	F_{K1} (81)	E (Cancer)
4	F_{K3} (91)	E (Tumor)
2	F_{K4} (82)	E (Tumor)

Fig. 4 The head party collects the encrypted tuples that correspond to user value



Id	Y_A	Y_B
5	F_{K1} (81)	E (Cancer)
4	F_{K3} (91)	E (Tumor)
2	F_{K4} (82)	E (Tumor)

Fig. 5 The user value is encrypted by all the keys of the data owners



Id	Y_A	Y_B
5	F_{k1} (81)	E (Cancer)
4	F_{k3} (91)	E (Tumor)
2	F_{k4} (82)	E (Tumor)

Fig. 6 The user finds the diseases that correspond to user value

3.4 Correctness

We show that our protocol correctly computes the query results by proving that each data owner chooses the correct bucket that includes the query value all the time (Step 6 in query protocol).

1. General scheme
   Each data owner O privately computes his W vector based on head party H permutation. The user searches for a value, which falls in bucket 'x' of the public bucketization.
   The head party sends H_x to all the data owners. Then, the data owners receive H_x and compute $b = W(H_x)$ to select bucket numbers. Finally, they select their tuples $Y(b)$ of the selected buckets.
2. Proof of correctness
   Each data owner, who has the data Y, computes $b = W(H_x)$ and selects his tuples $Y(b)$.
   Since, by definition $b = W(H_x) = \pi_\delta$, where $\delta = H^{-1}(H_x)$. Hence, $b = \pi_x$, which indicates that the correct bucket is chosen by the data owners.

3.5 Privacy Analysis

The algorithm shows that the data owners choose the bucket ID that is relative to the user's query value. We have:

1. Indistinguishability of data distribution
   Data owners privately choose their buckets, which correspond to the user's query value. The distribution of values that are hashed and also encrypted using commutative encryption is indistinguishable than the uniform distribution.
2. Elimination of bucket inference
   Data owners privately compute their W permutation vectors based on the head party H permutation vector, which is known for both the users and the data owners. Each data owner learns nothing about the permutation vectors of the other data owners. Each data owner directly communicates with the head party. The head party learns nothing about the data owners' buckets and the query value of the user.

4 Experimental Cost Analysis

In this section, we compare the results of implementing privacy-preserving SMC technique, which applies a bucketization technique for processing set-intersection queries, for both the ring and star exchange topologies. The results have been compared using the computational cost.

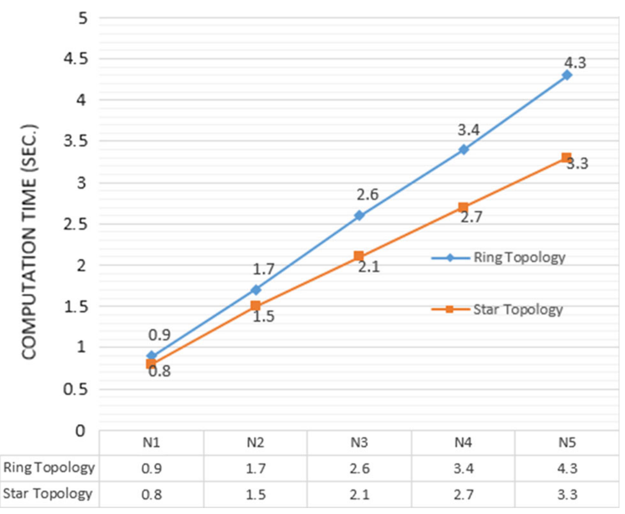
We developed our technique using the .Net framework on a Windows 8 platform with core i3 processor and 4GB RAM. We created five instances, in a star exchange topology environment, including three instances for the three hospitals, one instance for the head party, and another one for the authorized user.

Data owners hold a table with one searchable attribute and one sensitive attribute for set-intersection queries. We implemented a hash function on the entire values of the data owners and the authorized user and applied a commutative encryption protocol based on exponentiation modulo p to encrypt the hashed searchable attributes of the data owners, and then applied symmetric encryption protocol (Advanced Encryption Standard) to encrypt the sensitive attribute.

We tested a five different data with a number of records of the common dataset as $N_1 = 1000$, $N_2 = 2000$, $N_3 = 3000$, $N_4 = 4000$ and $N_5 = 5000$. The experimental results, which are shown in Table 3 and Fig. 7, illustrate the difference in computational time for both the ring and star exchange topologies.

Table 3 The computational time for the ring and star exchange topologies with $M = 3$ data owners and $S = 5$ buckets

Number of records (N)	Computation time(s)–ring exchange topology	Computation time(s)–star exchange topology
1000	0.9	0.8
2000	1.7	1.5
3000	2.6	2.1
4000	3.4	2.7
5000	4.3	3.3



	N1	N2	N3	N4	N5
Ring Topology	0.9	1.7	2.6	3.4	4.3
Star Topology	0.8	1.5	2.1	2.7	3.3

Fig. 7 The computational time of our protocol based on the ring and star topologies ($M = 3$, $S = 5$)

We notice that the difference, in the computational cost between the ring and star topology lines, is approximately the same when the number of records is relatively small, but it differs as the number of records increases as shown in the graph in Fig. 7. The results confirm the fact that the star exchange topology is effective in reducing the computation time when the number of records increases rather than the ring exchange topology.

Moreover, the star exchange topology provides better privacy since it does not need to rely on a TTP where each data owner can separately compute its local permutations using the BU of the head party without the need for building an interchange matrix between them. Additionally, data do not need to traverse through the ring due to star topology end-to-end communication support.

Furthermore, we tested the impact of incrementing the number of the buckets on the computational costs with the number of rows $N = 5000$ and changed the number of buckets from 1 to 5. We notice that when the number of buckets changed in an increasing order, the computation time changed in a decreasing order, as shown in Table 4 and Fig. 8.

Table 4 The impact of incrementing the number of buckets on computational costs for the ring and star topologies ($N = 5000$ records, $M = 3, 1 \leq S \leq 5$)

Number of buckets (S)	Computation time(s)–ring exchange topology	Computation time(s)–star exchange topology
1	5.5	4.1
2	5.1	3.8
3	4.7	3.7
4	4.6	3.6
5	4.2	3.3

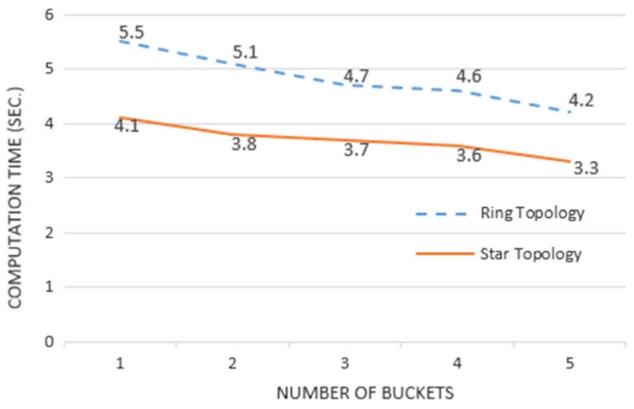
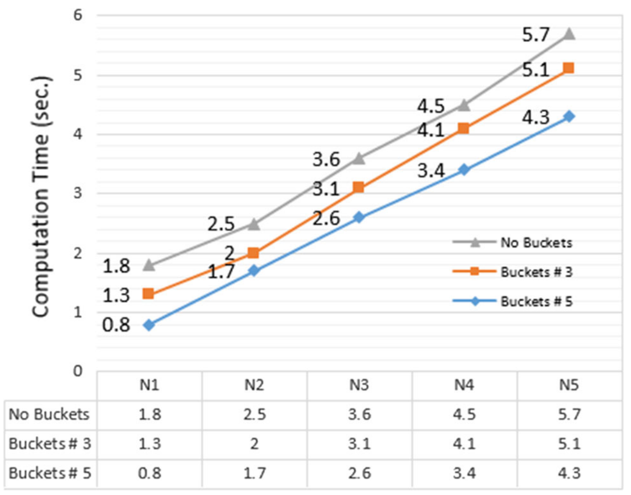


Fig. 8 The impact of incrementing the number of buckets on the computational costs for the ring and star topologies ($N = 5000$ rows, $M = 3, 1 \leq S \leq 5$)



	N1	N2	N3	N4	N5
No Buckets	1.8	2.5	3.6	4.5	5.7
Buckets # 3	1.3	2	3.1	4.1	5.1
Buckets # 5	0.8	1.7	2.6	3.4	4.3

Fig. 9 The impact of incrementing the buckets number on computational costs for set-intersection queries ($M = 3$, $S = \{1, 3 \& 5\}$)

We also tested the impact of incrementing the number of buckets on the computational costs for set-intersection queries, with a number of records $N = \{1000, 2000, 3000, 4000$ and 5000 records and a number of buckets $S = \{1, 3$ and 5 buckets$\}$, for the star exchange topology as shown in Table 5 and Fig. 9.

Table 5 The impact of incrementing the number of buckets for the star exchange topology ($M = 3$ data owners and $S = \{1, 3 \& 5\}$)

Number of records (N)	Computation time (s)		
	Buckets = 5	Buckets = 3	No buckets
1000	0.8	1.3	1.8
2000	1.7	2	2.5
3000	2.6	3.1	3.6
4000	3.4	4.1	4.5
5000	4.3	5.1	5.7

5 Conclusion

In this paper, we presented the real model of privacy-preserving SMC technique in a star exchange topology environment. The proposed technique targets the set-intersection queries and is based on a bucketization technique to provide scalability without the help of a TTP. It uses a commutative encryption scheme to encrypt the patients' searchable attributes and symmetric encryption scheme to encrypt patients' sensitive data to provide end-to-end encryption. It achieves both the data privacy and query privacy, where the user knows only the query results while data owners do not learn the query. The experimental results confirm the fact that the star exchange topology is effective in reducing the computational time when the number of records increases rather than the ring exchange topology. Moreover, the star exchange topology provides better privacy since it does not need to rely on a TTP. The bucketization technique improves the efficiency of the protocol by considering only the records, which are relative to the buckets containing the user query value. As a future work, we plan to adopt our system to support range and inner-join queries.

References

1. Wu, D.: Research on patient privacy protection for medical data in cloud computing. J. Netw. **8**(11), 2678–2684 (2013)
2. Lee, K.H.; Lee, D.: Electronic medical records privacy preservation through k -anonymity clustering method. In: IEEE, pp. 1119–1124 (2012)
3. Sabbeh, S.F.: Privacy preservation in the cloud: current solutions and open issues. Int. J. Comput. Trends Technol. (IJCTT). (2017). https://doi.org/10.14445/22312803/IJCTT-V51P102
4. Zhaolong, G.; Yamaguchi, S.; Gupta, B.B.: Analysis of various security issues and challenges in cloud computing environment: a survey. In: Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security (2016)
5. Omkar Badve, B.B.; Gupta, S.G.: Reviewing the security features in contemporary security policies and models for multiple platforms. In: Handbook of Research on Modern Cryptographic Solutions for Computer and Cyber Security (2016)
6. Sheikh, R.; Mishra, D.K.: Secure sum computation for insecure networks. In: Proceeding ICTCS '16 Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies Article No. 102 , ACM ICTCS '16 Proceedings of the Second International Conference on Information and Communication (2016)
7. Sepehri, M.; Cimato, S.; Damiani, E.: Privacy-preserving query processing by multi-party computation. Comput. J. Secur. Comput. Syst. Netw. **58**(10), 2195–2212 (2015)
8. Agrawal, R.; Evfimievski, A.; Srikant, R.: Information sharing across private databases. In: Proceedings of the 2003 ACM SIGMOD International Conference on on Management of Data, p. 86 (2003)
9. Yao, A.C.C.: How to generate and exchange secrets. In: Proceedings of 27th Annual Symposium on Foundations of Computer Science, IEEE Computer Society (1), pp. 162–167 (1986)
10. Lindell, Y.: A proof of security of Yao's protocol for two-party computation. J. Cryptol. **22**, 161–188 (2009)
11. Kruger, L.; Jha, S.; Goh, E.J.; Boneh, D.: Secure function evaluation with ordered binary decision diagrams. In: Proceedings of the 13th ACM Conference on Computer and Communications Security—CCS '06, p. 410 (2006)
12. Iliev, A.; Smith, S.W.: Small, stupid, and scalable: secure computing with faerieplay. In: The ACM Workshop on Scalable Trusted Computing, pp. 41–51 (2010)
13. Evans, D.; Katz, J.: Faster secure two-party computation using garbled circuits. In: 20th USENIX Security Symposium (August), pp. 8–12 (2011)
14. Jha, S.; Kruger, L.; Shmatikov, V.: Towards practical privacy for genomic computation. In: Proceedings—IEEE Symposium on Security and Privacy, pp. 216–230 (2008)
15. Bellare, M.; Hoang, V.T.; Rogaway, P.: Foundations of garbled circuits. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security—CCS '12, p. 784 (2012)
16. Malkhi, D.; Nisan, N.; Pinkas, B.; Sella, Y.: Fairplay—-Secure Two-Party Computation System. USENIX Security Symposium, pp. 287–302 (2004)
17. Mohassel, P.; Franklin, M.: Efficiency tradeoffs for malicious two-party computation. Proc. Public Key Cryptogr. Conf. **3958 LNCS**, 458–473 (2006)
18. Huang, Y.; Katz, J.; Evans, D.: Efficient secure two-party computation using symmetric cut-and-choose. Crypto **8043 LNCS**(PART 2), 18–35 (2013)
19. Woodruff, D.P.: Revisiting the efficiency of malicious two-party computation. In: Advances in Cryptology—EUROCRYPT 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20–24, 2007, Proceedings **4515**, pp. 79–96 (2007)
20. Goldreich, O.; Micali, S.; Wigderson, A.: How to play any mental game. In: Symposium on Theory of Computing, pp. 218–229 (1987)
21. Beaver, D.: Efficient multiparty protocols using circuit randomization. Crypto **576**(814), 420–432 (1991)
22. Atallah, M.; Bykova, M.; Li, J.; Frikken, K.; Topkara, M.: Private collaborative forecasting and benchmarking. In: Proceedings of the 2004 ACM Workshop on Privacy in the Electronic Society—WPES '04, p. 103 (2004)
23. Pullonen, P.; Bogdanov, D.; Schneider, T.: Institute of information security the design and implementation of a two-party protocol suite for Sharemind 3. CYBERNETICA Institute of Information Security, Institute of Information Security (2013)
24. Damgård, I.; Orlandi, C.: Multiparty computation for dishonest majority: from passive to active security at low cost. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **6223 LNCS**, pp. 558–576 (2010)
25. Bringer, J.; Chabanne, H.; Favre, M.; Patey, A.; Schneider, T.; Zohner, M.: GSHADE: faster privacy-preserving distance com-

putation and biometric identification. In: Proceedings of the 2nd ACM Workshop on Information Hiding and Multimedia Security, pp. 187–198 (2014)

26. Gilboa, N.: Two party RSA key generation. In: Advances in CryptologyCRYPTO'99, pp. 116–129 (1999)
27. Naor, M.; Pinkas, B.: Oblivious polynomial evaluation. SIAM J. Comput. **35**(5), 1254–1281 (2006)
28. Özarar, M.; Özgit, A.: Secure multiparty overall mean computation via oblivious polynomial evaluation. In: Proceedings of the First International Conference on Security and Networks (2008)
29. Chang, Y.C.; Lu, C.J.: Oblivious polynomial evaluation and oblivious neural learning. Theor. Comput. Sci. **341**(1–3), 39–54 (2005)
30. Luyao, L.; Zongtao, D.; Qinglong, W.: Unconditionally secure oblivious polynomial evaluation protocol. In: International Conference on Advanced Information and Communication Technology for Education (Icaicte), pp. 579–583 (2013)
31. Rivest, R.L.; Dertouzos, M.L.: On data banks and privacy homomorphisms. In: Proceedings of IEEE Annual Symposium on Foundations of Computer Science (1978)
32. Osadchy, M.; Pinkas, B.; Jarrous, A.; Moskovich, B.: SCiFI—a system for secure face identification. In: 2010 IEEE Symposium on Security a Privacy (2), pp. 239–254 (2010)
33. Carter, H.; Amrutkar, C.; Dacosta, I.; Traynor, P.: For your phone only: custom protocols for efficient secure function evaluation on mobile devices. Security and Communication Networks **7**(7), 1165–1176 (2014)
34. Brickell, J.; Shmatikov, V.: Privacy-preserving graph algorithms in the semi-honest model. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) **3788 LNCS**, pp. 236–252 (2005)
35. Hazay, C.; Lindell, Y.: Efficient protocols for set intersection and pattern matching with security against malicious and covert adversaries. J. Cryptol. **23**(3), 422–456 (2010)
36. Miyaji, A.; Rahman, M.S.: Privacy-preserving data mining in presence of covert adversaries. In: Adma 2010 **1**(LNCS 6440), pp. 429–440 (2010)
37. Vaidya, J.; Clifton, C.: Secure set intersection cardinality with application to association rule mining. J. Comput. Secur. **13**, 593–622 (2004)
38. Huang, K.; Tso, R.: A commutative encryption scheme based on ElGamal encryption. In: Proceedings—3rd International Conference on Information Security and Intelligent Control, ISIC 2012, pp. 156–159 (2012)
39. Khayat, S.H.: Using commutative encryption to share a secret key idea proposed scheme description. Electrical Engineering, pp. 1–6 (2008)
40. Ishai, Y.; Kilian, J.; Nissim, K.; Petrank, E.: Extending oblivious transfer efficiently. In: Advances in Cryptology-CRYPTO **2003**, pp. 145–161 (2003)
41. Freedman, M.J.; Nissim, K.; Pinkas, B.: Efficient private matching and set intersection. Eurocrypto 2004 (i), 1–19 (2004)
42. Sang, Y.; Shen, H.; Tan, Y.; Xiong, N.: Efficient protocols for privacy preserving matching against distributed datasets. In: Proceedings of Information and Communications Security, vol. **4307**, pp. 210–227 (2006)
43. Kissner, L.; Song, D.X.: Privacy-preserving set operations. Crypto 2005 **3621** (February 2005), pp. 241–257 (2005)
44. Li, R.; Wu, C.K.: Co-operative private equality test. Int. J. Netw. Secur. **1**(3), 149–153 (2005)
45. Sepehri, M.: Privacy-preserving query processing by multi-party computation and encrypted data outsourcing. Ph.D. thesis (2012)