Privacy-Preserving Shared-Additive-Inverse Protocols and Their Applications

Huafei Zhu, Tiejian Li, and Feng Bao

Department of Information Security
Institute for Infocomm Research, A-Star, Singapore 119613
{huafei, litiejian, baofeng}@i2r.a-star.edu.sg

Abstract. Privacy-preserving clustering algorithms group similar databases populated at distributed locations to improve data qualities and enable accurate data analysis and thus provide fundamental security components for distributed data mining with privacy concerns. This paper makes three contributions regarding shared k -means clustering algorithms. First, a new notion called shared-additive-inverse (SAI) protocols – a building block for efficient implementation of shared k -means clustering protocols within the arbitrarily partitioned database model, is introduced and formalized. Second, a generic implementation of SAI protocols from shared-scalar-product (SSP) protocols is proposed which is provably secure in the semi-honest model assuming that any underlying SSP protocol is privacy-preserving. Finally, we propose an immediate application of SAI protocols for privacy-preserving computation of shared cluster means – a crucial step in the shared k -means clustering algorithms. To the best of our knowledge, this is the first implementation of shared k -means clustering algorithms with provable security from SAI protocols which in turn are derived from SSP protocols.

1 Introduction

Privacy-preserving clustering algorithms group similar databases populated at distributed locations to improve data qualities and enable accurate data analysis and thus provide fundamental security components for distributed data mining with privacy concerns. The solutions to privacy-preserving clustering algorithms have been presented with respect to horizontally, vertically and arbitrarily partition databases. For instance, Kantarcioglu and Clifton [12] describe algorithms for mining association rules in horizontally partitioned data and Vaidya and Clifton [16] develop analogous algorithms for vertically partitioned data. Jagannathan and Wright [11] provide the extensions for doing secure clustering protocols in arbitrarily partitioned data using well known k -means clustering algorithms.

k -means clustering algorithms have many applications, such as bio-medical and DNA data analysis, financial data analysis, targeted marketing, forensics (see [11, 12, 16] for further reference). Informally, a k -means clustering algorithm is an iterative algorithm that successively refines potential cluster in an attempt to minimize the k -means objective function. The main idea for k -means clustering algorithms is to define k cluster centers, one for each cluster. The k cluster centers should be placed in a cunning way because of different location causes different result. So, the better choice is

Please use the following format when citing this chapter:

Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 201, Security and Privacy in Dynamic Environments, eds. Fischer-Hubner, S., Rannenberg, K., Yngstrom, L., Lindskog, S., (Boston: Springer), pp. [insert page numbers].

to place them as much as possible far away from each other. There is some research on picking the initial cluster means [2]. The next step is to take each point belonging to a given data set and associate it to the nearest center. These two steps are alternated until a stopping criterion is met, i.e., when there is no further change in the assignment of the data points. In order to create a privacy-preserving version of k -means, we have to devise a privacy-preserving protocol to compute the cluster means (see Section 5 for more details). For instance, Alice has her input (x, a) and Bob has his input (y, b) when we consider a two-party k -means clustering algorithm. Both participants wish to jointly compute the cluster mean $\frac{x+y}{a+b}$, the output of the participants can be the following three cases, where $s_a + s_b = (a+b)^{-1} \bmod m$ and $s_a \in Z_m$ and $s_b \in Z_m$ are the random shares of Alice and Bob respectively:

Case 1: either Alice or Bob learns the cluster mean $\frac{x+y}{a+b}$ (this is a special situation of the case 3 when $s_a = 0$ or $s_b = 0$);

Case 2: Alice and Bob learn $\frac{x+y}{a+b}$ (this is a special situation of the case 3 when Alice and Bob learn s_a and s_b simultaneously);

Case 3: Alice learns s_a and Bob learns s_b such that $s_a + s_b = \frac{x+y}{a+b}$ (a general case in which we are interested in this paper).

In the next section, we will briefly sketch previous works for k -means clustering algorithms and at the same time we will state the problems regarding these implementations.

1.1 Previous works and problem statement

Jagannathan and Wright [11] first introduced the notion of arbitrarily partitioned data and then proposed a simple solution to shared two-party k -means clustering protocols (with respect to the Case 3). Their solution is to first approximate the function $\frac{x+y}{a+b}$ by a circuit C , and then use the heuristic implementations [1, 6, 7, 17] to construct a privacy-preserving protocol. Although, the architecture for privacy-preserving clustering protocols is attractive, their approach is heuristic. Consequently, a less heuristic approach would be a great step forward. The recent work of Jha, Kruger and McDaniel [10] can be viewed as such a forward step.

At ESORICS 2005, [10], Jha, Kruger and McDaniel presented two solutions to the Case 2 protocols (Alice and Bob learn $\frac{x+y}{a+b}$ simultaneously) – the first one is based on oblivious polynomial evaluation and the second one is based on homomorphic encryption. We would like to provide the following observation on their solutions: oblivious polynomial evaluation first introduced and formalized by Naor and Pinkas [13], is a protocol involving two parties, a sender whose input is a polynomial $f(x) \in F[x]$, and a receiver whose input is a value $a \in F$, where F is a finite field. At the end of the protocol the receiver learns $f(a)$ and the sender learns nothing. Two oblivious polynomial evaluation (OPE) protocols have been proposed by Naor and Pinkas [13] – the first construction is based on a conjecture that given a randomly chosen input to the noisy polynomial interpolation problem, the value of the polynomial P at $x = 0$ is pseudo-random; – the second construction is more efficient one based on a stronger assumption that the value of the polynomial at $x = 0$ is pseudo-random even given

some additional hints about the location of the values of the polynomial. The security of the protocols is based on the assumptions that the noisy polynomial interpolation problem¹ was as hard as the polynomial reconstruction problem². Unfortunately, Naor and Pinkas's construction were shown to be weaker than expected by Bleichenbacher and Nguyen [3]. More precisely, Bleichenbacher and Nguyen presented new methods to solve noisy polynomial interpolation problem which do not apply to the polynomial reconstruction problem. In particular they show that the noisy polynomial interpolation problem can be transformed into a lattice shortest vector problem with high probability, provided that the parameters satisfy a certain condition that is explicitly stated in [3]. The method is similar to the well-known lattice-based methods to solve the subset sum problem. They further show that most practical instances of the noisy polynomial interpolation problem with small m can be solved, thus the noisy polynomial interpolation problem is much easier than expected and should be used cautiously as an intractability assumption. It follows that the cryptographic protocols should be based on the polynomial reconstruction problem rather than the noisy polynomial interpolation problem (we refer the reader to [3] for further reference). Since Naor and Pinkas's construction is based on the hardness assumption of the noisy polynomial interpolation problem rather than the polynomial reconstruction problem, it follows that the k -means clustering algorithm protocol proposed by Jha, Kruger and McDaniel inherently suffers from this weakness. Thus any more satisfactory construction of k -means clustering protocols is certainly welcome.

The second approach is based on any semantically secure homomorphic encryption, and thus their second methodology avoids using oblivious polynomial evaluation protocols as building blocks. We stress that there is no secret sharing mechanism deployed within their constructions (both the OPE-based and non-OPE based constructions, i.e., the authors only consider the implementation of the Case 2 in their paper), it follows that their method cannot be extended for the construction of k -means clustering protocols with respect to the general case (the Case 3 where secure computation of shared k -means of clustering protocols is processed within the arbitrarily database model). Since no result is known regarding the construction of general k -means clustering protocols from semantically secure homomorphic encryptions (to the best of our knowledge), we thus provide an interesting research problem below.

Research problem: How to construct shared k -means clustering algorithms from semantically secure homomorphic encryptions in the arbitrarily partitioned database model?

¹ The noisy polynomial interpolation problem is the following thing: let P be a k -degree polynomial over a finite field F , given $n > k + 1$ sets $S_1, \dots, S_n$ and n distinct elements $x_1, \dots, x_n \in F$ such that each $S_i = \{y_{i,j}\}_{1 \leq j \leq m}$ contains $m - 1$ random elements and $P(x_i)$, recover the polynomial P , provided that the solution is unique.

² The polynomial reconstruction problem is the following thing: given as input integers k, t and n points $(x_1, y_1), \dots, (x_n, y_n) \in F^2$, output all polynomials P of degree at most k such that $y_i = P(x_i)$ for at least t values of i .

1.2 This work

In this paper, a new notion called shared-additive-inverse protocol is introduced and formalized as a building block for k -means of clustering protocols. Informally, a shared-additive-inverse (SAI) protocol is the following thing – there are two participants Alice with her input $inp_A = a$ and Bob with his $inp_B = b$ who wish to compute $s_a + s_b = (a + b)^{-1} \bmod m$, where $s_a \in Z_m$ and $s_b \in Z_m$ are the random shares of Alice and Bob respectively. The output of Alice is $s_a \in Z_m$ while the output of Bob is $s_b \in Z_m$. We then provide a generic construction of SAI protocols from shared-scalar-product (SSP) protocols. Our implementation of SAI protocols is provably secure in the semi-honest model assuming that the underlying SSP is privacy-preserving. Using any privacy-preserving SAI protocol a build block, we can simply compute a shared cluster mean $\mu_x + \mu_y = \frac{x+y}{a+b} = (x+y)(s_a + s_b)$ by means of any secure shared-scalar-product protocol so that Alice learns μ_x and Bob learns μ_y . As a result, a secure k -means clustering algorithm for distributed data mining is derived as an immediate application of our SAI protocol.

In summary, this paper makes three contributions regarding privacy-preserving clustering algorithms. First, a new notion called shared-additive-inverse (SAI) protocols is introduced and formalized. Second, a generic implementation of SAI protocols from shared-scalar-product (SSP) protocols is proposed which is provably secure in the semi-honest model assuming that the underlying SSP protocol is privacy-preserving. Finally, we transform any secure SAI protocol to privacy-preserving computation of shared cluster means protocol in the semi-honest model – a crucial step in the shared k -means clustering algorithms. To the best of our knowledge, this is the first implementation of shared k -means clustering algorithms with provable security from SAI protocols which in turn are derived from SSP protocols.

Road map The rest of paper is organized as follows: In Section 2, syntax, functionality and security definition for shared-additive-inverse protocols are presented; And we briefly introduce the building block for SAI protocols in Section 3. Our implementation of SAI protocols and the proof of security are presented in Section 4. An application of SAI protocols to k -means clustering protocols is presented in Section 5. We conclude our works in Section 6.

2 Syntax, functionality and security definition

2.1 Syntax

A shared-additive-inverse protocol consists of the following two probabilistic polynomial time (PPT) Turing machines.

- On input a system parameter m (throughout the paper, we assume that Z_m is a finite field. Furthermore, we ignore the trivial case when $a=0$ or $b=0$ since the general technique presented in Section 4 can be tailored to solve the trivial case), a PPT Turing machine A (say, Alice), chooses $a \in_r Z_m$ uniformly at random. The initial input of Alice is denoted by $inp_A = a$;

- On input a system parameters m , a PPT Turing machine B (say, Bob), chooses $b \in Z_m$ uniformly at random. The initial input of Bob is denoted by $inp_B=b$;
- On inputs inp_A and inp_B , Alice and Bob jointly compute $s_a + s_b = (a + b)^{-1} \bmod m$, where $s_a \in Z_m$ and $s_b \in Z_m$ are random shares of Alice and Bob respectively;
- The output of Alice is $s_a \in Z_m$ while the output of Bob is $s_b \in Z_m$.

2.2 Functionality

The functionality $\mathcal{F}_{SAI}$ of shared-additive-inverse protocols (SAI) can be abstracted as follows:

- A player (say Alice) has her initial input $inp_A=a \in Z_m$; Another player (say Bob) has his initial input $inp_B=b \in Z_m$; Each participant sends the corresponding input to TTP – an imaginary trusted third party in the ideal world via a secure and private channel.
 - Upon receiving inp_A and inp_B , TTP tests whether $a \in Z_m$ and $b \in Z_m$.
 - if the conditions are satisfied, then TTP chooses $s_a \in Z_m$ uniformly at random and computes s_b from the equation $s_a + s_b = (a + b)^{-1} \bmod m$.
 - if $a \notin Z_m$, then TTP chooses an element $a' \in_r Z_m$ and substitutes a with a' .
- Similarly, if $b \notin Z_m$, then TTP chooses an element $b' \in_r Z_m$ and substitutes b with b' . By $inp_A=a$ (using the same notation of the initial input of Alice), we denote the valid input of Alice which may be modified by TTP; By $inp_B=b$ (again using the same notation of the initial input of Bob), we denote the valid input Bob which may be modified by TTP. Once given valid inputs inp_A and inp_B , TTP chooses $s_a \in Z_m$ uniformly at random and computes s_b from the equation $s_a + s_b = (a + b)^{-1} \bmod m$.
- The output of Alice is $s_a \in Z_m$ which is sent by TTP to Alice via the specified secure and private channel between them. The output of Bob is $s_b \in Z_m$ which is sent by TTP to Bob via the specified secure and private channel between them such that $s_a + s_b = (a + b)^{-1} \bmod m$.

2.3 The definition of security

Privacy-preserving protocols are designed in order to preserve privacy even in the presence of the adversarial participants that attempt to gather information about the inputs of their inputs. There are however different levels of adversarial behavior – a semi-honest adversarial behavior and a malicious adversarial behavior (see [5] for more details). This paper however, concerns SAI protocols in the semi-honest model (the security definition of k -means clustering algorithms is inherently within the semi-honest model therefore).

Let $f: \{0, 1\}^* \mapsto \{0, 1\}^* \times \{0, 1\}^*$ be probabilistic, polynomial time functionality, where $f_1(x, y)$ (respectively, $f_2(x, y)$) denotes the first (resp., second) element of $f(x, y)$; and let π be two-party protocol for computing f . Let the view of the first (resp., second) party during an execution of the protocol π on input (x, y) , denoted $view_1^\pi(x, y)$ (resp., $view_2^\pi(x, y)$), be $(x, r_1, m_1, \dots, m_t)$ (resp., $(y, r_2, m_1, \dots, m_t)$).

r_1 represents the outcome of the first (resp., r_2) party's internal coin tosses, and m_i represents the i -th message od it has received. The output of the first (resp., second) party during the an execution of π on (x, y) is denoted $output_1^\pi(x, y)$ (resp., $output_2^\pi(x, y)$) and is implicit in the party's view of the execution.

Definition 1. *Two distribution ensembles $X = \{X_s\}_s \in S$ and $Y = \{Y_s\}_s \in S$ are computationally indistinguishable if for any probabilistic polynomial time distinguisher D and all sufficiently large s , there is a negligible function ν such that $|Pr(D(X_s) = 1) - Pr(D(Y_s) = 1)| \leq \nu$.*

Definition 2. *π privately computes f if there exist probabilistic polynomial time algorithms, denotes S_1 and S_2 such that*

$$\{S_1(x, f_1(x, y), f_2(x, y))\}_{x, y \in \{0,1\}^*} \equiv \{(view_1^\pi(x, y), output_2^\pi(x, y))\}_{x, y \in \{0,1\}^*}$$

$$\{(f_1(x, y), S_2(x, f_2(x, y)))\}_{x, y \in \{0,1\}^*} \equiv \{output_1^\pi(x, y), (view_2^\pi(x, y))\}_{x, y \in \{0,1\}^*}$$

where $\equiv$ denotes computational indistinguishability.

Definition 3. *An oracle-aided protocol is a protocol augmented by a pair of oracle types, per each party, an oracle-call steps defined as follows: Each of the parties may send a special oracle request message, to the other party, after writing a string called the query on its write-only oracle tape. In response, the other party writes a string, its query, on its own oracle tape and respond to the first party with a oracle call message. At this point the oracle is invoked and the result is that a string, not necessary the same, is written by the oracle on the read-only oracle tape of each party. This pair of strings is called the oracle answer.*

Definition 4. *An oracle-aided protocol is said to privately reduce g to f when using the oracle functionality f . In such a case, we say that g is privately reducible to f .*

Composition theorem for the semi-honest model [5]: Suppose that g is privately reducible to f and that there exists a protocol for privately computing f , then there exists a protocol for privately computing g .

3 Building blocks

Privacy preserving data mining is a new and rapidly emerging research area, where data mining algorithms are analyzed for the side-effects they incur in data privacy. Secure scalar product protocols are the fundamental cryptographic build blocks for building secure data mining protocols [14]. Several private (shared) scalar-product protocols have been proposed in the context of privacy-preserving data ming [4, 8, 9, 16]. Informally, a scalar-protocol is the following things: there are two participants Alice who holds her input vector $(x_1, \dots, x_l)$ and Bob who holds his input vector $(y_1, \dots, y_l)$. They wish to compute $\sum_{i=1}^l x_i y_i$ such that at the end of the protocol, Alice holds the value $\sum_{i=1}^l x_i y_i$, while Bob holds nothing. A sibling notion is shared-scalar-product protocols (SSP). An SSP protocol is the following thing: there are two participants Alice who holds her input

vector $(x_1, \dots, x_l)$ and Bob who holds his input vector $(y_1, \dots, y_l)$. They wish to compute random shares of scalar product s_a and s_b such that $\sum_{i=1}^l x_i y_i = s_a + s_b$. At the end of the protocol, Alice holds the value s_a while Bob holds s_b . Clearly, a scalar-product protocol is a special case of a shared-scalar-product protocol when $s_b=0$.

Since the SSP protocol due to [8] will be used as a building block for our construction, we sketch the protocol below:

- Private input of Alice: $inp_A = (x_1, \dots, x_l)$, $x_i \in Z_m$ for $i = 1, \dots, l$;
- Private input of Bob: $inp_B = (y_1, \dots, y_l)$, $y_i \in Z_m$ for $i = 1, \dots, l$;
- Private output of Alice (resp., Bob) is $s_a \in Z_m$ (resp., $s_b \in Z_m$) such that $s_a + s_b = \sum_{i=1}^l x_i y_i$.

System setup: Alice generates a public key and private key pair (sk, pk) of Paillier's encryption scheme [15] (i.e., $sk=(P, Q)$, P and Q are large safe primes, $pk=(g, N)$, where $N=PQ$, $g = (1 + N)$).

Step 1: for $i=1$ to l , Alice generates a random string r_i and sends $c_i = \text{Enc}_{pk}(x_i, r_i)$ to Bob; Step 2: Bob computes $w = \prod_{i=1}^l c_i^{y_i} \text{Enc}_{pk}(-1)^{s_b}$ and sends w to Alice; Step 3: Alice decrypts w and obtains $s_a \in Z_m$.

Lemma 1. (due to [8]) *The protocol is secure in the semi-honest model assuming that the underlying Paillier's encryption scheme is semantically secure and $\ln^2 < |N|$.*

4 Implementation and proof of security

We propose an efficient implementation of SAI protocols in this section based on the privacy-preserving shared-scalar-product protocols in the semi-honest model.

4.1 Our implementation

Our implementation of SAI protocols consists of the following steps:

Step 1: on input $inp_A = x \in Z_m$ and $y \in Z_m$, Alice and Bob then choose $r_x \in_r Z_m$ and $r_y \in_r Z_m$ uniformly at random. Two participants then jointly compute the following value $z \in Z_m$:

$$\begin{aligned} z &= (x + y)(r_x + r_y) \\ &= xr_x + (xr_y + r_x y) + yr_y \\ &= (xr_x + t_x) + (yr_y + t_y) \\ &= z_x + z_y, \end{aligned}$$

where $z_x := xr_x + t_x$ and $z_y := yr_y + t_y$ and the intermediate shares $t_x \in Z_m$ and $t_y \in Z_m$ satisfying the equation $t_x + t_y = xr_y + r_x y$ are computed from any privacy-preserving shared-scalar-product protocol defined over the finite field Z_m such that Alice learns $t_x \in Z_m$ while Bob learns $t_y \in Z_m$;

Step 2: Alice sends z_x to Bob, and at the same time Bob sends z_y to Alice;

Step 3: once given z_x and z_y , both Alice and Bob can compute the inverse of z individually;

Step 4: the output of Alice is $s_x = r_x z^{-1} \bmod m$ while the output of Bob is $s_y = r_y z^{-1} \bmod m$.

4.2 The proof of security

The correctness of the protocol can be easily verified:

$$\begin{aligned}
 s_x + s_y &= r_x z^{-1} + r_y z^{-1} \\
 &= (r_x + r_y) z^{-1} \\
 &= (r_x + r_y)(r_x + r_y)^{-1}(x + y)^{-1} \\
 &= (x + y)^{-1},
 \end{aligned}$$

Lemma 2. *Our implementation of SAI protocols securely reduce SAI protocols to SSP protocols in the semi-honest model.*

Proof: The first party Alice, sends her input $inp_A = (x, r_x)$ to the SSP oracle $\mathcal{F}_{SSP}$ while the second party, Bob sends his input $inp_B = (y, r_y)$ to $\mathcal{F}_{SSP}$. $\mathcal{F}_{SSP}$ outputs (t_x, t_y) so that Alice learns t_x while Bob learns t_y . As a result, the distribution $view_1^\pi(x, y)$ can be defined as $\{x, r_x, t_x, z_x, z_y, s_x\}$, where $z_x \leftarrow xr_x + t_x, z_y \in_r Z_m, s_x \leftarrow r_x(z_x + z_y)^{-1}$. We then define the distribution of $output_2^\pi(inp_A, inp_B)$. By the implementation, we have equations: $z_y = yr_y + t_y$ and $t_x + t_y = xr_y + yr_x$. Consequently, we have the equation $z_y = yr_y + t_x - xr_y - yr_x$. Furthermore, for fixed variables (x, r_x, t_x, z_y) and y , there is a unique solution of r_y such that $r_y = (y - x)^{-1}(yt_x + z_y - t_x)$ if $x \neq y$. Since $s_y = r_y z^{-1} = r_y(z_x + z_y)^{-1}$, it follows that s_y is uniformly distributed over Z_m assuming that y is uniformly distributed over the finite field Z_m . Finally, we need to show that there exists a simulator S_1 for the honest-but-curious party Alice. This can be done as follows: we choose $s_y \in_r Z_m$ uniformly at random and set $f_2(inp_A, inp_B) \leftarrow s_y$. We then define $S_1(inp_A, s_y)$ according to the following steps:

- on input $inp_A = (x, r_x)$, S_1 runs the sub-simulator for the (sub-protocol) SSP protocol. Such a sub-simulator exists since the underlying SSP protocol is secure in the semi-honest model. The distribution of the sub-simulator is denoted by (t_x, t_y) ;
- S_1 then computes z_x from the equation $z_x = xr_x + t_x$, where S_1 learns t_x from the distribution of the simulator (we stress that S_1 does not learn t_y). And then S_1 chooses $z_y \in_r Z_m$;
- S_1 then computes s_x from the equation $s_x = r_x(z_x + z_y)^{-1}$;
- The distribution of S_1 is $(x, r_x, t_x, z_x, z_y, s_x)$.

Obviously, the following two distributions are computationally indistinguishable:

$$\begin{aligned}
 &\{S_1(inp_A, f_1(inp_A, inp_B), f_2(inp_A, inp_B))\}_{inp_A, inp_B \in \{0,1\}^*} \\
 &= \{(x, r_x, t_x, z_x, z_y, s_x), s_y\} \\
 &= \{(view_1^\pi(inp_A, inp_B), output_2^\pi(inp_A, inp_B))\}_{inp_A, inp_B \in \{0,1\}^*}.
 \end{aligned}$$

Similarly, we can define a simulator S_2 for the honest-but-curious party Bob satisfying the security definition. It follows that our implementation is securely reduce SAI protocols to SSP protocols in the semi-honest model.

Combining the Lemma 1 and the Lemma 2, we have the main statement below (the composition theorem for the semi-honest model):

Theorem 1. *Our implementation of SAI protocols is secure in the semi-honest model assuming that the Paillier's encryption scheme is semantically secure.*

5 Applications

SAI protocols can be used to compute clustering means in the horizontally partitioned database model, and vertically partitioned database model. We also would like to provide an interesting application of SAI protocols to the general case of the privacy-preserving distributed k -means clustering algorithms for the arbitrarily partitioned database model [11]. Let $D=\{d_1, d_2, \dots, d_n\}$ be a database set consists of n objects. Each object d_i is described by the value of the l numerical attributes. That is each d_i is partitioned into two disjoint subsets d_i^A and d_i^B . Let μ_j^A (resp. μ_j^B) for $1 \leq j \leq k$ denote Alice's (resp., Bob's) share of the j -th mean. The candidate cluster center are given by $\mu_j^A + \mu_j^B$ for $1 \leq j \leq k$. For each object d_i , Alice and Bob securely compute the distance $\text{dist}(d_i, \mu_j)$ for $1 \leq j \leq k$, between the object and each of the k cluster centers. The result of the distance calculation is learned as random shares between Alice and Bob. Using the random shares, they then securely compute the closest cluster for each object in the database (all computations are performed over a finite field Z_m). At the end of each iteration, Alice and Bob learns his or her share of the cluster centers. The next iteration requires the recomputing each of the k -cluster centers. Suppose that Alice has objects $d_{i_1}^A, \dots, d_{i_p}^A$ and Bob has objects $d_{i_1}^B, \dots, d_{i_q}^B$, for $1 \leq j \leq k$. Each of the d_i^A and d_i^B is an l -tuple, where $d_{i,j}^A$ (resp., $d_{i,j}^B$) denotes the j -th coordinate of the corresponding l -tuple.

Alice (resp., Bob) calculates the shares s_j and n_j (resp. t_j and m_j) for $1 \leq j \leq l$, where $s_j = \sum_{r=1}^p d_{i_r,j}^A$ (resp., $t_j = \sum_{r=1}^q d_{i_r,j}^B$), and n_j (resp., m_j) denotes the number of objects in $d_{i_1}^A, \dots, d_{i_p}^A$ (resp., $d_{i_1}^B, \dots, d_{i_q}^B$) for which Alice (resp., Bob) has the values for the attribute A_j . If Alice (resp., Bob) does not have the value for the attribute j for some object d_i^A (resp., d_i^B), she (resp., he) treats it as zero. Thus the j -th components of i -th cluster center is given by $\mu_{i,j} = \frac{s_j + t_j}{m_j + n_j}$. For convenience, we simply assume that Alice has her input $\text{inp}_A = (a, r_a)$ while Bob has his input $\text{inp}_B = (b, r_b)$. They wish to securely compute $s_a + s_b = \frac{a+b}{r_a+r_b}$ over Z_m . We now can provide alternative solution to the problem below³:

- Private input of Alice: $\text{inp}_A = (a, r_a) \in Z_m \times Z_m$;
- Private input of Bob: $\text{inp}_B = (b, r_b) \in Z_m \times Z_m$;
- Private output of Alice (resp., Bob) is $s_a \in Z_m$ (resp., $s_b \in Z_m$) such that $s_a + s_b = \frac{a+b}{r_a+r_b}$.

Step 1: On inputs $\text{inp}_A = r_a$ and $\text{inp}_B = b$, Alice and Bob jointly run the SAI protocol to compute t_a and t_b such that $t_a + t_b = (r_a + r_b)^{-1}$, where Alice learns $t_a \in Z_m$ and Bob learns $t_b \in Z_m$;

Step 2: On inputs $\text{inp}_A = (a, t_a) \in Z_m \times Z_m$ and $\text{inp}_B = (b, t_b) \in Z_m \times Z_m$, Alice and Bob jointly run any privacy-preserving shared-scalar-product protocol to compute $w_a \in Z_m$ and $w_b \in Z_m$ such that $w_a + w_b = at_b + bt_a$;

Step 3: Alice computes $s_a \leftarrow at_a + w_a$ and Bob computes $s_b \leftarrow bt_b + w_b$;

Step 4: The output of Alice is $s_a \in Z_m$ and while the output of Bob is $s_b \in Z_m$.

³ There are two solutions suggested in [11]: one is based on Yao's circuit evaluation protocol [17]; another solution is based on Bar-Ilan and Beaver's protocol [1]. Unfortunately, no detailed solution is available in [11] yet.

As an immediate application of the Theorem 1, we have the following corollary:

Corollary 1. *The recomputing k -means protocol is secure assuming that the underlying SAI protocol is secure in the semi-honest model.*

6 Conclusion

In this paper, a notion called shared-additive-inverse (SAI) protocols has been introduced and formalized. We then have proposed a novel construction of SAI protocols from any secure shared-scalar-product (SSP) protocols, and shown that the implementation is secure assuming that the underlying SSP protocol is secure in the semi-honest model. Finally, an immediate application of SAI protocols to the k -means cluster recomputing algorithm has been presented. To the best of our knowledge, our algorithm is the first practical solution to the k -means cluster recomputing algorithm suitable for the arbitrarily partitioned database model.

Acknowledgment

The research of the first author is partially supported by National Natural Science Foundation of CHINA under the Project number 60273058.

References

1. Judit Bar-Ilan, Donald Beaver: Non-Cryptographic Fault-Tolerant Computing in Constant Number of Rounds of Interaction. PODC 1989: 201–209.
2. P.S. Bardley and U.M. Fayyad. Refining initial points for k -means clustering. In Proceedings of 15th International Conference on Machine Learning (ICML), pages 91–99, 1998.
3. Daniel Bleichenbacher and Phong Q. Nguyen. Noisy Polynomial Interpolation and Noisy Chinese Remaindering. EUROCRYPT 2000, LNCS 1807, pp. 53–69, 2000.
4. Wenliang Du and Zhijun Zhan, Building decision tree classifier on private data, In Proceedings of the IEEE ICDM Workshop on Privacy, Security and Data Mining (2002).
5. Oded Goldreich. Foundations of Cryptography, Volume 2, Cambridge University Press, 2004.
6. O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game – a completeness theorem for protocols with honest majority. In 19th Symposium on Theory of Computer Science, pages 218–229, 1987.
7. O. Goldreich, S. Micali, and A. Wigderson. Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. Journal of the ACM, 38(1):691–729, 1991.
8. Bart Goethals, Sven Laur, Helger Lipmaa, Taneli Mielikäinen: On Private Scalar Product Computation for Privacy-Preserving Data Mining. ICISC 2004: 104–120
9. Ioannis Ioannidis, Ananth Grama, and Mikhail Atallah, A secure protocol for computing dot products in clustered and distributed environments, In Proceedings of the International Conference on Parallel Processing (2002). Geetha KDD 2005: 593–599.
10. Somesh Jha, Louis Kruger, Patrick McDaniel: Privacy Preserving Clustering. ESORICS 2005: 397–417

11. Geetha Jagannathan, Rebecca N. Wright: Privacy-preserving distributed k-means clustering over arbitrarily partitioned data. KDD 2005: 593–599.
12. M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *The ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (DMKD'02)*, June 2, 2002.
13. Moni Naor, Benny Pinkas: Oblivious Transfer and Polynomial Evaluation. STOC 1999: 245–254
14. Benny Pinkas, Cryptographic Techniques for Privacy-Preserving Data Mining. Newsletter of the ACM Special Interest Group on Knowledge Discovery and Data Mining, January 2003.
15. Pascal Paillier: Public-Key Cryptosystems Based on Composite Degree Residuosity Classes. Proc. of EUROCRYPT 1999: 223–238, Springer Verlag.
16. Jaideep Vaidya and Chris Clifton, Privacy preserving association rule mining in vertically partitioned data, In the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2002), 639–644.
17. Andrew Chi-Chih Yao: How to Generate and Exchange Secrets (Extended Abstract) FOCS 1986: 162–167.