Middle-East Journal of Scientific Research 17 (7): 992-997, 2013
ISSN 1990-9233

DOI: 10.5829/idosi.mejsr.2013.17.07.12145

Privacy-preserving Protocol over Vertically Partitioned Data in Multiparty K-means Clustering

Vadim Gennadyevich Zhukov and Alexey Vyacheslavovich Vashkevich

Siberian State Aerospace University named after ac. M.F. Reshetnyov, Krasnoyarsk, Russia

Abstract: The article examines and analyzes existing solutions of privacy-preserving computations over vertical partitioning in multiparty clustering by k-means. The modifications of cryptographic primitives to eliminate detected deficiencies are offered. The results of comparative analysis of advanced cryptographic primitives are given. The protocol is developed that preserves data privacy with few parties considering their possible collusion. The conclusions on the effectiveness of the improved cryptographic primitives are made and the recommendations how to use the developed protocol are given.

Key words: Secure multiparty computations . clustering . k-means . secure dot product . secure sum

INTRODUCTION

With vertically partitioned data in multiparty clustering of k-means (a description of the algorithm can be found in the work of Hartigan and Wong) [1] each party has a set of attributes for every object (parties have different data columns). The centers of the clusters are randomly generated and are not private. Protected are such steps of k-means as the calculation of distances from an object to centers, the choice of the minimal distance and also checking the conditions of stopping.

To preserve privacy two approaches are offered:

1. First, between the "particular" parties the distance is distributed from the object to all the clusters, they find the shortest distance. However, they know the distance from the object to the clusters in a rearranged order. Next they report the "rearranged" number of the shortest distance to the party who knows the rearrangement and it sends out the real number of the smallest distance. In the algorithm special parties are required who conduct additional operations and must not collude with each other [2, 3].
2. All the parties do not summarize the distance from the object to the clusters; they summarize the difference between the distances from the object to the clusters. Therefore, after the summation it is sufficient to determine if the difference is more than zero or not, to calculate the sum is not necessary. The algorithm does not have particular parties [4, 5].

In all the algorithms the collusion of several (depending on the algorithm) parties creates the problem of security: the data of the victims of collusion, in some degree, will be disclosed. Thus, the aim of this work is to improve the protocol of preserving data privacy for vertical partitioning.

The analysis of existing solutions: In the solution of Vaidya and Clifton [2] the permutation algorithm offered by Du and Attala [6] was used to find the nearest cluster and homomorphic encryption was used to get the distances to the cluster. The protocol requires three particular parties. One of them knows the permutation, the other two know the results of comparing the distances between the dot and the centers of the clusters, but know the "rearranged" results, i.e. do not know what distance is for what cluster.

Doganay *et al*. [3] repeated the scheme of Vaidya and Clifton, but instead of homomorphic cryptosystems they used secret random pieces. Four particular parties are used instead of three (two of the parties generate the rearrangement and two make comparisons). However, the result of the collusion of just two parties is less critical than that according to Vaidya and Clifton.

In order not to have particular parties, Samet *et al*. [4] used the cryptographic primitive "Secure Sum" offered by Clifton and colleagues [7], which also requires the presence of parties who do not collude (parties i and i+2). But secure sum is not suitable when there are only two parties, so Samet *et al*. offered to use secure dot product (Secure Dot Product, SDP) [5].

The protocol offered by Samet [4] does not require the presence of particular parties, let us take a closer

Corresponding Author: Vadim Gennadyevich Zhukov, Siberian State Aerospace University named after ac. M.F. Reshetnyov, Avenue named after "Krasnoyarskiy Rabochiy", 31, 660014, Krasnoyarsk, Russia.

look. For several (more than two) of the parties the following scheme is used:

Let us denote the set of attributes that P_i has (the party number i) as

$$A_i = \{a_{i1}, a_{i2}, ..., a_{im}\}$$

P_i owns the set of attributes for each of the centers of the clusters

$$m_{ji} = \{m_{ji1}, m_{ji2}, ..., m_{jim}\}$$

Next, each party sums up the distance between the corresponding dimensions. Thus, the portion of the distance between the object and the center of the cluster μ_{ji} in party P_i will be as follows (example for the square Euclidean distance):

$$d_{ji} = (a_{il} - m_{ji1})^2 + (a_{i2} - m_{ji2})^2 + ... + (a_{im} - m_{jim})^2$$

Having summed up these distances for all of the parties, we get $d_{j1}+d_{j2}+\ldots+d_{jr}$ where r is the number of the parties. For the center of another cluster μ_q, we have the formula $d_{q1}+d_{q2}+\ldots+d_{qr}$. To determine which of the clusters (j or q) is closer it is necessary to sum up the values

$$\sum_{i=1}^{r} d_{ji} - d_{qi} = \sum_{i=1}^{r} d_i$$

If the result is positive, μ_q is closer, otherwise μ_j is closer. This step is repeated k-1 times to find the minimal of the distances from the dot to the center of the cluster.

This summation is made using Secure Sum algorithm, offered by Clifton. P_1 generates a random number x, sums it up with d_1 and sends to the party P_2. Each subsequent party sums that number up with its d_i and sends further (the last party sends it to the first one). When the circle is complete, P_1 subtracts x from the sum and gets $\sum_{i=1}^{r} d_i$. Then checks $\sum_{i=1}^{r} d_i > 0$ and sends the result to all.

There are two important points:

1. If there are only two parties, Secure Sum is not suitable due to the fact that P_1 immediately knows the meaning of d_2, simply subtracting from the received value $(x+d_1+d_2)$ sent $(x+d_1)$.
2. If P_i and P_{i+2} collude then they can find the meaning of d_{i+1} similarly to the case when there are only two parties.

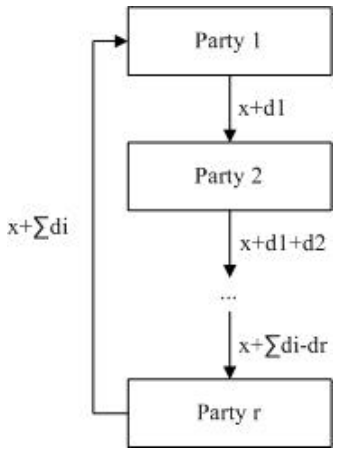

Fig. 1: The diagram of secure sum algorithm

Samet *et al*. offered (but did not describe the algorithm) to split up d_i for several random pieces and sum them up in different routes, so that the neighbors do not recur. The routes and the number of pieces can be varied, but there is one more thing that the Samet *et al*. did not consider- with three parties the routes cannot be changed and when r>3 to disclose information r-1 parties are enough and with small values of r the possibility of collusion cannot be neglected. Thus, when there are few parties another algorithm of summation should be used.

For the case with two parties Samet *et al*. offered to use the protocol based on secure dot product:

1. P_1 randomly selects a nonzero number l_1 and creates the vector $X = \left(\frac{d_1}{l_1}; \frac{1}{l_1}\right)$ and P_2 creates the vector $Y = (1; d_2)$.
2. P_1 and P_2 use secure dot product described by Malek and Miri [5] and P_2 gets the number l_2 in the way $l_1 \cdot l_2 = d_1+d_2$.
3. P_2 sends l_2 sign. If $l_2 = 0$ the distance to both clusters is the same, if $l_2 \neq 0$, l_2 sign shows the cluster the distance to which is less.

In the offered protocol the vector length of X and Y equals two and as it will be shown later, it leads to the disclosure of the data of P_1.

Thus, the algorithm [4] has two drawbacks that are mandatory for elimination:

1) When r = 2 the data of one of the parties will be disclosed;
2) When r is small the collusion of parties r -1 is possible.

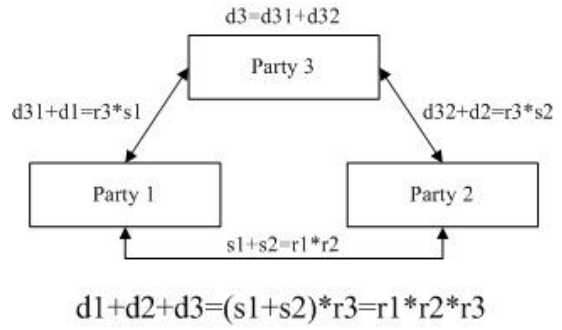

Fig. 2: The example of SDP for three parties

One possible solution for the case when there are few parties is to use the same secure dot product. This requires from the party to have a session of SDP with every party. The example for three parties is shown in Fig. 2.

The main drawback of the algorithm is that the amount of data transmitted over the network increases in proportion to the square of the number of parties. Because of it this algorithm is not suitable for clustering with a large number of parties because it is used in every iteration for every object of the data k-1 times.

Thus, when the number of parties is small the use of SDP is obligatory, as it guarantees the absence of the disclosure of these parties' data. When the number of parties is large one still cannot give up secure sum, as it works much faster due to the smaller amount of interactions between the parties. However, one should also improve the protocol of secure sum to avoid any disclosure of information in case of the collusion of a small number of parties.

Main part. For a small number of parties existing solutions do not guarantee the preservation of privacy in case of collusion between several parties. It is necessary to develop or choose from the existing ones such a primitive that would ensure privacy even in case of collusion of all the parties, except one and would not disclose any party's data.

Samet *et al*. for two parties use secure dot product: suppose two parties (let us call them Alice and Bob) each has an n-dimensional vector: Alice has

$$\bar{x} = \{x_1, x_2, ..., x_n\}$$

and Bob has

$$\bar{y} = \{y_1, y_2, ..., y_n\}$$

Each dimension is a finite field F_p and the vector space is the final n-dimensional extension F_p^n of the finite field F_p.

Alice and Bob want to count $\bar{x} \cdot \bar{y}$ keeping privacy. The result of the protocol should be known only to Alice. The input data of each of the parties do not have to be disclosed. Alice and Bob perform the following protocol:

1) Alice randomly selects the vector $\bar{\varphi} \in F_p^n$ and $a,b,c,d \in F_p$, $(ad-bc)^{-1} \neq 0$. Alice computes the following vectors:

$$\bar{u} = a\bar{x} + b\bar{j}$$

$$\bar{v} = c\bar{x} + d\bar{j}$$

2) Then Alice sends the vectors $\bar{u}$ and $\bar{v}$ to Bob. Knowing his vector $\bar{y}$, Bob computes $\bar{y} \cdot \bar{u}$ and $\bar{u} \cdot \bar{y}$. Then Bob sends the results back to Alice.

3) Alice computes $(ad-bc)^{-1}\left(d\left(\bar{y} \cdot \bar{u}\right) - b\left(\bar{y} \cdot \bar{v}\right)\right)$ that is equivalent to $\bar{x} \cdot \bar{y}$.

Samet *et al*. used SDP to find the nearest cluster. Alice and Bob create vectors $(1;d_2)$ and $\left(\frac{d_1}{l_1};\frac{1}{l_1}\right)$ where l_1 is a random number generated by Bob. At the output Alice gets l_2 so that $l_1 \cdot l_2 = d_1 + d_2$. Then, to determine the sign of the sum d_1+d_2 the numbers l_1 and l_2 are multiplied.

It is necessary to note two things. First, the primitive of Malek and Miri is made for finite fields. Thus, one should not use the signs of numbers l_1 and l_2 for the reliable determination of the sign of the sum d_1+d_2 after using the primitive.

The example that proves this statement. There is a finite field {-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5}, $d_1 = -1$, $d_2=-2$. Suppose $l_1 = 2$. Then, after using secure dot product $l_2 = 4$ is obtained as in this finite field $2 \cdot 4 = -1-2$. The signs of the numbers l_1 and l_2 are both positive and the sign of the sum is negative-the protocol does not always work correctly in finite fields. Also, the use of finite fields decreases the accuracy of the analysis, since it is necessary to "impose" the range of permissible values on a finite field, which produces the quantization error. The solution is to use a set of real numbers instead of finite fields.

Second, the analysis of the primitive showed that the use of two-dimensional vectors $(1; d_2)$ and $\left(\frac{d_1}{l_1};\frac{1}{l_1}\right)$ discloses Bob's data to Alice. The reason is in step 2 of the primitive when Bob sends Alice the results of dot products $\bar{y} \cdot \bar{u}$ and $\bar{y} \cdot \bar{v}$. Alice gets two equations with two unknowns:

$$y_1u_1 + y_2u_2 = \overline{y} \cdot \overline{u}$$

$$y_1v_1 + y_2v_2 = \overline{y} \cdot \overline{v}$$

Vectors $\overline{u}$ and $\overline{v}$ are known to Alice already and the results of dot products $\overline{y} \cdot \overline{u}$ and $\overline{y} \cdot \overline{v}$ she got from Bob. Having solved the equations, Alice computes d_1/l_1 and $1/l_1$ and gets d_1. Bob's data becomes known to Alice.

This problem can be solved by splitting two dimensions of the vectors (1; d_2) and $\left(\frac{d_1}{l_1};\frac{1}{l_1}\right)$ into three. After splitting the vectors look as follows:

- The vector of Alice (1;1; d_2),
- The vector of Bob $\left(\frac{d_1+z}{l_1};\frac{-z}{l_1};\frac{1}{l_1}\right)$

where z is a random number generated by Bob. The dot product of the vectors will remain the same, but now in two equations there will be three unknowns $\frac{d_1+z}{l_1}$, $-z/l_1$ and $1/l_1$, from which "curious" Alice will not in the general case get the value d_1 of Bob. But in certain parameters of the vectors $\overline{u}$ and $\overline{v}$ the disclosure of data is still possible, so Bob should check the vectors $\overline{u}$ and $\overline{v}$ sent from Alice for the following conditions (which can be performed separately but not simultaneously):

$$u_1 = u_2,\ v_1 + v_2$$

If both of these conditions of the equations are fulfilled one can find $\frac{d_1+z}{l_1} + \left(\frac{-z}{l_1}\right)$ and $1/l_1$ separately and so get

$$\left(\frac{d_1+z}{l_1} + \left(\frac{-z}{l_1}\right)\right) \cdot l_1 = d_1$$

Having eliminated unsatisfactory features, one can start using secure dot product to determine the closest cluster. However, when there is a large number of a parties and / or data object, it leads to a large volume of traffic. Therefore, the use of SDP is recommended for a small amount of data objects or parties when there is a risk of collusion and providing privacy to the detriment of the speed of the algorithm.

Table 1: The dependence of the maximal possible number of parties in collusion from the total number of parties

r	N	n	The possible number of parties in collusion
3	1	1	1
4	1	1	1
5	1,2	2	3
6	1	1	1
7	1,2,3	3	5
8	1,3	2	3
9	1,2,4	3	5
10	1,3	2	3
11	1,2,3,4,5	5	9
12	1,5	2	3
13	1,2,3,4,5,6	6	11
14	1,3,5	3	5

With a large amount of data when the volume of traffic is important, it is necessary to provide a smaller amount of data transfers. One can apply the improved algorithm of secure sum.

When using the original secure sum offered by Clifton *et al*. [7] private information of any party (except the first one) may be disclosed by its neighbors through the protocol if they collude with each other. The solution of the problem is "The modified ck-secure sum" [8] where the data of every party is splitted into k random pieces, there is k of summation rounds, in every of which neighbors are rearranged. It is guaranteed that no two parties having colluded will get the data of the party between them. Now, however, may be not only two, but just three parties who colluded to disclose confidential information are enough. Also in this protocol the data is divided into segments that increase the cost of communication.

The existing modernizations of "secure sum" (before "Modified ck- secure sum" there were protocols [9, 10]), do not have a special advantage in speed comparing to dot product as they split the data into the number of pieces equal to the number of parties and also do not preserve privacy even with three parties who are in collusion. The algorithm is required that allows to split the data into minimal number of pieces so as to protect the privacy for the specified maximal number of parties in collusion.

Theorem: Suppose, the number of participants is r. Then we define the family of operations +N in a set {1,2,…,r}, so that any element of $Z_r\backslash\{0\}$ after r usages of the operation of the family +N will return into itself. To do this, all the factors of the number N should be mutually simple with all the factors of the number r (or N = 1).

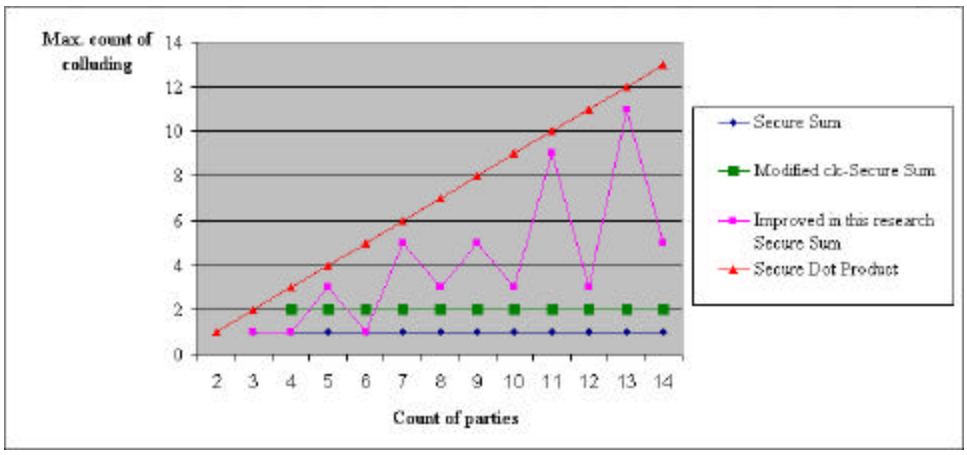

Fig. 3: The dependence of the maximal possible amount of parties in collusion from the total number of parties

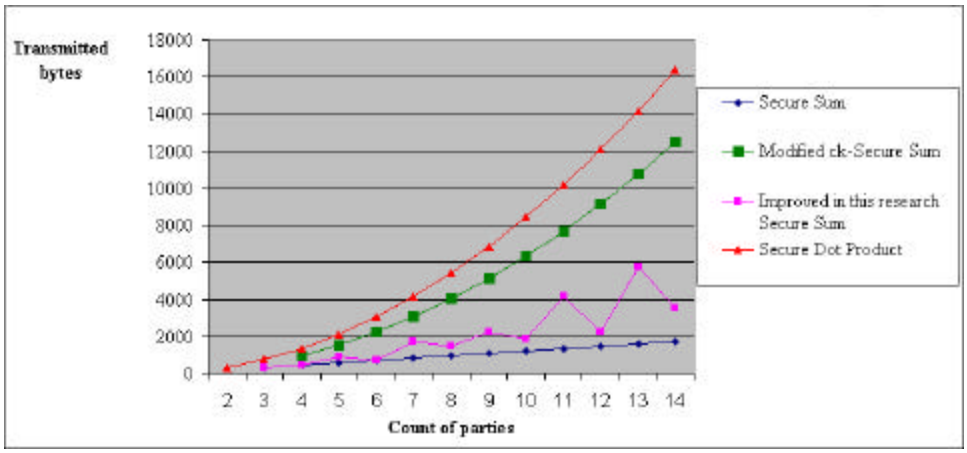

Fig. 4: The dependence of the number of bytes transferred in a single operation of comparing distances from the number of parties

If we take the operations of this family, when N<r/2 it is possible to create the routes of summation in such a way that every time there will be different neighbors. Let us denote the number of such operations from +N as n. Then the maximal possible number of parties in collusion that will still provide privacy equals n·2-1. With simple N the possible number of parties in collusion is maximal and equals N-2 (Table 1).

For 5 and starting from 7 parties, the offered algorithm allows more parties in collusion than the algorithms that existed earlier.

This modification allows to change the neighbors according to the algorithm and to maintain privacy even for some, though not very large, number of parties in collusion. The maximal amount of those in collusion depends on how many operations from the family +N can be chosen.

It is also not necessary to use all the possible routes, if the model of the adversary will include fewer parties in collusion than the possible maximum, so one can split the data into fewer pieces and, accordingly, sum up in fewer routes, which will reduce the amount of network traffic.

The developed modernized algorithm of secure sum allows to reduce the amount of the transferred data both compared to SDP and in comparison with the analogues. In this case, the algorithm ensures the preservation of privacy with more parties in collusion than for the similar ones. However, for few parties (in particular, for 2, 3, 4 and 6) it is still recommended to use SDP.

Figure 3 and 4 present the graphs of dependence of the possible maximal number of parties in collusion and the bytes transferred during a single operation of comparison of distances from the total number of parties. We considered that in every package beside useful information the network protocol headers were present (Ethernet, IP, UDP) [11].

The function of the traffic with SDP grows faster than the function of possible parties in collusion. As the number of bytes transferred in SDP is more than in the improved secure sum and the traffic with a lot of parties can be several tens of gigabytes, secure sum is more preferable in use with a lot of parties or objects. One should consider that k-means, as a rule, should be started several times with different numbers of clusters, so the traffic will increase many-fold. When using this protocol, one should define the expected volume of traffic and the number of possible parties in collusion and on the basis of these parameters select a cryptographic primitive underlying the work of the protocol.

CONCLUSION

The result of the study is the protocol preserving privacy in multiparty k-means clustering for vertical partitioning and the given model of the adversary. The analysis of the protocol showed its advantages over similar protocols. The correct use of the cryptographic primitive “Secure Dot Product” in the protocol guarantees privacy even with the collusion of all the parties against one. With a large number of parties of cluster analysis instead of "Secure Dot Product "the modernized during this work primitive “Secure Sum” is used, which unlike the original “Secure Sum” and other versions of the primitive guarantees privacy with a large number of parties in collusion.

ACKNOWLEDGMENTS

This work was supported by the grant MK- 473.2013.9 of the President to young PhD.

REFERENCES

1. Hartigan, J. and M. Wong, 1979. Algorithm AS 136: A K-Means Clustering Algorithm. Journal of the Royal Statistical Society. Applied Statistics, 28 (1): 100-108.
2. Vaidya, J. and C. Clifton, 2003. Privacy-Preserving K-Means Clustering Over Vertically Partitioned Data. In the Proceedings of the 9th ACM SIGKDD International Conference on Knowledge discovery and Data Mining. Washington, USA, pp: 206-215.
3. Doganay, M., T. Pederson, Y. Saygin, E. Savas and A. Levi, 2008. Distributed Privacy Preserving Clustering with Additive Secret Sharing. In the Proceedings of the 2008 International Workshop on Privacy and Anonymity in Information Society, pp: 6003-6011.
4. Samet, S., A. Miri and L. Orozco-Barbosa, 2007. Privacy-Preserving K-Means Clustering in Multi-Party Environment. In the Proceedings of the 2007 International Conference on Security and Cryptography, Barcelona, Spain, pp: 381-385.
5. Malek, B. and A. Miri, 2006. Secure dot-product protocol using trace functions. In the Proceedings of the 2006 IEEE International Symposium on Information Theory, pp: 927-931.
6. Du, W. and M. Atallah, 2001. Privacy-Preserving Cooperative Statistical Analysis. In the Proceedings of 17th Annual Computer Security Applications Conference, USA, pp: 102-110.
7. Clifton, C., M. Kantarcioglu, J. Vaidya, X. Iin and M. Zhu, 2002. Tools for privacy preserving data mining. SIGKDD Explorations, 4 (2): 28-34.
8. Sheikh, R., B. Kumar and D. Mishra, 2010. A Modified ck-Secure Sum Protocol for Multi-Party Computation. Journal of Computing, 2: 62-66.
9. Sheikh, R., B. Kumar and D. Mishra, 2009. Privacy-Preserving k-Secure Sum Protocol. International Journal of Computer Science and Information Security, 6: 184-188.
10. Sheikh, R., B. Kumar and D. Mishra, 2010. Changing Neighbors k-Secure Sum Protocol for Secure Multi-Party Computation. International Journal of Computer Science and Information Security, 7: 239-243.
11. Zhukov, V. and A. Vashkevich, 2013. Privacy-preserving Clustering in Vertical Partitioning. Proceedings of the 13th International Scientific Conference “Information Security”, Taganrog, Russia, pp: 191-198.