Brief Announcement: Privacy Preserving Mining of Distributed Data Using a Trusted and Partitioned Third Party

Nir Maoz and Ehud Gudes(✉)

Department of Mathematics and Computer Science, The Open University,
1 University Road, 43537 Ra'anana, Israel
ntaizi@yahoo.com

1 Introduction

We like to discuss the usability of new architecture of partitioned third party, offered in [1] for conducting a new protocols for data mining algorithms over shared data base between multiple data holders. Current solution for data mining over partitioned data base are: Data anonimization [4], homomorphic encryption [5], trusted third party [2] or secure multiparty computation algorithms [3]. Current solutions suffer from different problems such as expensive algorithms in terms of computation overhead and required communication rounds, revealing private information to third party. The new architecture offered by Sherman et al. allow the data holders to use simple masking techniques that are not expensive in computation nor assume trust in the third party, yet allow to perform simple and complex data mining algorithms between multiple data owners while private data is not revealed. That come with the assumption of no collude between the two parts of the PTTP. In the PTTP architecture offered by Sherman et al. [1] the trusted third party is divided into two parts CE the Computer Engine which does the data mining and mathematical calculation on behalf of the participants and to the Randomizer R which generates random numbers and permutations needed for the protocol, and share them securely with the participants. All communication between data base holders and the CE or R is assumed to be private e.g. using symmetric encryption with private key shared between the two sides broadcasting each other. In this paper, we show one basic data mining algorithms for calculating union/intersection, to show the power of this architecture. We developed few more basic and complex algorithms, for calculating aggregation functions, Min/Max and association rules. Although, some of these operations like union/intersection were discussed in [1], we developed different and simpler protocols than those suggested there.

2 Intersection/Union

In this section we describe a PTTP protocol for computing intersection and union of private sets. The set is separated horizontally between the different

S. Dolev and S. Lodha (Eds.): CSCML 2017, LNCS 10332, pp. 193–195, 2017.
DOI: 10.1007/978-3-319-60080-2_14

databases holders. In the setting that we consider here, each of the private databases includes a column in which the entries are taken from some groundset Ω. For example, if the column includes the age of the data subject, in whole years, then we may take $\Omega = \{0, 1, \ldots, 120\}$. Our protocol enables the different database holders to compute the union or intersection of their columns; The protocol uses a PTTP. The main idea of the protocol, is to create for each DB_i two Boolean vectors. Both are of the length of the groundset. All DB_i will sort the groundset in the same order (e.g. alphabetic order). Each DB_i will create one vector that have 1 for each index in the vector that represent the position of the item it has in its DB and 0 for the rest of the indexes. Then each DB creates another vector which has a 0 for each index in the vector that represent the position of the item it has in its DB and 1 for the rest of the indexes (Exactly the inverse vector). Than it will concatenate both vectors, and apply a permutation, shared by all participants, on the result vector. After applying the permutation, no one can know, without knowing the permutation, which item a specified DB holds or not even how many items there are in each DB. The reason is, that in the way we build the concatenated vector, there is exactly the same number of 1's and 0's (Ω times) which prevent the knowledge of how many items each DB holds. And since the vector is permuted, there is no way to know which 1's belong to the first half of the vectors (index of item the participant hold) and which 1's belong to the second half (index of item the participants doesn't hold). The parties in our protocol are as follows:

- Q is the querier who issues the query to be answered.
- D_i, $1 \leq i \leq M$, are the databases.
- CE (Computation Engine) and R (Randomizer) are the two parts of the PTTP.

In the protocol, $V = V_\Omega$ is a vector that includes all values in the groundset Ω. Protocol 1 shows the algorithm in detail.

3 Conclusions

We show in this paper how one can use the new PTTP architecture proposed by Sherman et al. to create simple algorithms like union/intersection over distributed database, without the need for strong cryptography techniques or the use of hash functions. Using PTTP also results in less communication rounds and in some cases also reduces the size of the messages. In most cases the new algorithms can also protect against malicious coalitions. That said, more research is still needed in order to shift more privacy preserving responsibilities from the two parts of the the PTTP back to the database holders, so that even coalition which involve both parts of the PTTP won't allow to reveal significant information. Finally, more research is also needed to show the utility of the PTTP architecture for other data mining tasks such as: clustering or decision trees construction.

Protocol 1. A PTTP protocol for Computing Set Operations.

1: Q sends the query to DB_i, $1 \leq i \leq M$, and the query type (either intersection or union) to R.
2: Q sends R which private column participate in the query (e.g. the age column).
3: R generates a random permutation σ on the set of integers $\{1, \ldots, 2|\Omega|\}$ and sends it to DB_i, $1 \leq i \leq M$.
4: **for all** $1 \leq i \leq M$ **do**
5: DB_i sets a Boolean vector V_i=$\{v_{i,1},\ldots,v_{i,|\Omega|}\}$ where

$$v_{i,j} = \begin{cases} 1 \; if DB_i \; hold \; private \; value \; j \\ 0 \; \text{otherwise} \end{cases},$$

6: DB_i sets a Boolean vector $\neg V_i$=$\{\neg v_{i,1},\ldots,\neg v_{i,|\Omega|}\}$ where

$$\neg v_{i,j} = \begin{cases} 0 \; if DB_i \; hold \; private \; value \; j \\ 1 \; \text{otherwise} \end{cases},$$

7: DB_i sets a new vector that is the concatenation of the two previous mentioned vectors, $CV = V_i \parallel \neg V_i$
8: DB_i calculate $PV_i = \sigma(CV) = \sigma(V_i \parallel \neg V_i) = \sigma(\{v_{i,1},\ldots,v_{i,|\Omega|},\neg v_{i,1},\ldots,\neg v_{i,|\Omega|}\})$
9: DB_i sends its vector to CE.
10: CE computes the intersection or union of all vectors received from the M databases and send the result vector RV to R.
11: R calculate the final vector $FV = \sigma^{-1}(RV)$
12: R throws away the second half of the vector and output it.

Acknowledgment. The authors would like to thank Tassa Tamir, for providing very helpful comments on the algorithms presented here.

References

1. Chow, S.S.M., Lee, J.-H., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: NDSS 2009
2. Ghosh, J., Reiter, J.P., Karr, A.F.: Secure computation with horizontally partitioned data using adaptive regression splines. Comput. Stat. Data Anal. **51**(12), 5813–5820 (2007)
3. Tassa, T.: Secure mining of association rules in horizontally distributed databases. IEEE Trans. Knowl. Data Eng. **26**(4), 970–983 (2014)
4. Tassa, T., Gudes, E.: Secure distributed computation of anonymized views of shared databases. ACM Trans. Database Syst. (TODS) **37**(2), 11 (2012)
5. Zhong, S.: Privacy-preserving algorithms for distributed mining of frequent itemsets. Inf. Sci. **177**(2), 490–503 (2007)