Global Scientific JOURNALS

GSJ: Volume 11, Issue 12, December 2023, Online: ISSN 2320-9186
www.globalscientificjournal.com

CONFIDENTIAL FILE SHARING BETWEEN UNTRUSTED DATA-BASES

Solomon SARPONG

Department of Physical and Mathematical Sciences, University of Environment and Sustainable Development, Somanya, Ghana
ssarpong@uesd.edu.gh

KeyWords
Cryptographic computation, honest-but-curious, inter-connectivity, intersection protocol, multi-party computation.

© GSJ

ABSTRACT

The inter-connectivity of human activities has made isolationism not a popular practice and, in some instances, impractical. Inter-connectivity leads to person(s) wanting to access information from others (persons or databases) and in some cases share information. The accessing and or sharing of information has brought into question the security of data and the identity of the data owner(s). This paper proposes protocols that will enable the sharing of information securely among untrusted databases.

Introduction

Data sharing has become unavoidable and important in recent times. The use of cloud computing and other databases have made it virtually impossible to prevent sharing of information between two or more databases. The sharing of information most often than not is necessitated by: (1). a database needing information from another. This maybe mutual or on demand; (2). when databases want to verify the information they have in common. The later brings to the fore the computation of the intersection between the databases. The problem in the computation of intersection between databases is that, none of the databases want to make their information known. That is the databases would like to compute the intersection securely without disclosing the information they possess. This problem has brought to the fore the use of trusted third party and secure multi-party computation.

Trusted Third Party

This is a domain that can be trusted to offer identity/data verification and facilitate the interaction between two parties. The trusted third party should be "trusted" by the two parties and must be secure against attacks. The parties who need to compute the intersection of their attributes send them to a trusted third party to do the computation of their intersection. The trusted third party informs both parties about the content of the intersection. In some scenarios, the trusted third party oversees the computation of the intersection between the databases without taking part in it.

Secure/multi-party Computation

Secure/multiparty computation (S/MPC) as proposed by [1] is a cryptographic protocol that enables computation involving many parties where none of the parties involved can see the others' data. Secure Multiparty Computation refers to cryptographic protocols that allow for the distributed computation of a function over distributed inputs without revealing additional information about the inputs, [2]. Let databases $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$. The motive of the databases is to compute the intersection of the content of the attributes without disclosing what they possess. Hence, the aim is to computationally find the intersection

of the contents of their attributes such that the databases know only the intersection but nothing else. In secure multiparty computation, the basic requirements are privacy and correctness.
Privacy – databases should know the output of the computation of the intersection and nothing else.
Correctness – each database should receive the actual output of the computation of the intersection. It must be noted that, an adversarial attack on secure multiparty computation may be to know the private information of the databases or cause the result of the computation by the databases to be incorrect. Hence, with the use of secure multiparty computation the adversary will not be able to interfere with the computation of the two databases.

The motive for the computation of the intersection of attributes maybe necessitated by: (1). In a market survey, two companies (*A* and *B*) may want to know the customers that patronize the goods and services of both companies. Each company does not want to disclose the list of its customers and the type of goods and services they patronize. In such a scenario, the information from the two companies should be shared such that; (i) neither party learns more than their own data and must obtain the intersection (if one exists), while neither should learn anything about the others' set and (ii) they should learn the results of the intersection mutually. (2). SMPC can be used when there is the need to compare the DNA of an individual against a database of the DNA of patients suffering from a terminal disease. The goal is to check if the individual is in a high-risk group for a certain type of the terminal disease.
The need to find the intersection between private datasets that each database does not want to make public arises often in interaction protocols. This task becomes complicated when the databases are suspicious of each other (untrusted databases). It is in lieu of this that this paper proposes the sharing of information securely between untrusted databases. Thus, the protocols in this paper enables the sharing of information securely between databases. The rest of the paper is organized as follows; a review of previous research works; the protocol for the secure computation of the intersection between the databases; the proposed protocol; the security analysis of the protocol and the conclusion.

Related Research

Private set intersection is a secure multiparty cryptographic computation technique that can enable two parties to compute the intersection of the encrypted contents of their datasets. In this computation, nothing is revealed to each other except the content of their intersection. Plain and authorized private set intersection (PSI and APSI) find application if there is the need for client authorization and/or server unlinkability, as well as on server's ability to engage in precomputation, [3]. In order to securely and privately compute the intersection of data sets, [4] proposed the use of homomorphic encryption in PSI.

The need for users to cryptographically protect their data (both at rest and in transit) becomes apparent when public cloud services are being used. The use of public cloud services for storage and confidentially sharing the data makes it more apparent the need for end users to cryptographically protect the data. In some cases, data protection also involves hiding the identity of the data owner from both the data users and the cloud provider. In order to provide anonymity for the data owner and access to data confidentially, [5] proposed A-SKY. In order to enhance security and privacy of authorized data users, [6] proposed a keyword search mechanism where each data user has keys to encrypt and decrypt the data. The protocol in their paper enables queries to the server that meets certain criteria to be sent to the enquiry (user) without it being decrypted. Protecting data in remote stores and be able to perform operations on them securely is to adopt cryptographic schemes. This has brought to the fore the problem of the management of secret keys by the database owners. Hence, [7] proposed a distributed key management system (DKMS) that is zero-knowledge to the data to manage the keys of data-owners and handle data sharing requests. The uncertainty of the integrity, security and confidentiality of data with cloud service providers is of great concern to data owners. Based on these, [8] proposed a system that imperatively imposes access control policies of data owners hence, preventing the cloud storage providers from unauthorized access and making illegal authorization.

Protocol

The privacy requirements for this protocol are: (i). the databases in the protocol learn nothing about each other's dataset except for the final outcome of the computation; (ii). before the exchange of their secret keys, the databases in the protocol get to know only the total number of matching records in their databases and nothing else and (iii). at the end of the protocol, only the databases in the protocol get to know the actual records they have in common.

Setup

The trusted third party (TTP) has a hash function, $h_{TP}()$, and a global public-key that allows registered *Users* to communicate with it. The trusted third party is passive hence it only oversees the protocol.
Each database generates an identity, *ID*, for ease of identification. Database *A* generates ID_A; likewise, database *B* generates ID_B.

Each database generates an RSA keypair. The RSA keypair for database *A* and *B* are (e_{A_1}, d_{A_1}) and (e_{B_1}, d_{B_1}) respectively. In order to use this protocol, the database(s) need to register with the TTP. The database(s) registers with the TTP by sending the *ID* and the public key of the RSA keypair to the it. Hence, the database *A* sends $Enc_{e_{TP}}[(h_{TP}(ID_{A_1}), ID_{A_1}), e_{A_1}]$. Database *B* also registers with the TTP by sending $Enc_{e_{TP}}[(h_{TP}(ID_{B_1}), ID_{B_1}), e_{B_1}]$. Furthermore, each database generates an RSA keypair: (e_{A_2}, d_{A_2}) and (e_{B_2}, d_{B_2}) for databases *A* and *B* respectively.

The Intersection Protocol

Let database *A* and database *B* have private data $A = \{a_1, a_2, \dots, a_n\}$ and $B = \{b_1, b_2, \dots, b_m\}$ respectively. The databases *A* and *B* agree on a hash function, $h()$. The databases hash the content of their files. Database *A* computes $h(A) = \{h(a_1), h(a_2), \dots, h(a_n)\}$. Database *B* also computes $h(B) = \{h(b_1), h(b_2), \dots, h(b_m)\}$. Each database encrypts the hash of the content of their files and randomly arranges the encrypted files. The databases exchange the encrypted randomized files. Database *A* sends $Random\{Enc_{e_{A_1}}[h(A)]\}$ to database *B.* On the other hand, database *B* sends $Random\{E_{e_{B_1}}[h(B)]\}$ to database *A.* Database *A* re-encrypts the encrypted files received from database *B* using the public key e_{A_2}. The re-encrypted file is randomized. The encrypted files received from database *B* and the randomized re-encrypted file by database *A* are concatenated and sent to database *B.* Database *A* sends $\{Random\{Enc_{e_{B_1}}[h(B)]\} || Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(B)]\}]\}$ to database *B.* Also, database *B* re-encrypts the encrypted files received from database *A* using the public-key e_{B_2}. The re-encrypted file is randomized. The encrypted files received from database *A* and re-encrypted by database *B* are concatenated and sent to database *A.* Likewise, database *B* sends $\{Random\{Enc_{e_{A_1}}[h(A)]\} || Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(A)]\}]\}$ to database *A.* Each database computes the intersection of information received. Each database computes the intersection $I_{AB} \in Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(B)]\}] \cap Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(A)]\}]$.

The computation of $I_{AB} \in Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(B)]\}] \cap Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(A)]\}]$ by each database gives the intersection size between the two databases. From the intersection computed, it means $I_{AB} \in$ database *A* and also $I_{AB} \in$ database *B*. Each of the databases sends the intersection I_{AB} to the trusted third-party. Let $Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(I_{AB})]\}]$ and $Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(I_{AB})]\}]$ be the intersection computed by databases *B* and *A* respectively and sent to the TTP. The TTP checks if $Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(I_{AB})]\}]$ from database *A* and $Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(I_{AB})]\}]$ from *B* are the same. If they are not the same, then one of the databases cheated hence, the protocol is terminated. Database *A* sends d_{A_2} to the trusted third-party by sending $Enc_{e_{TP}}[d_{A_2} || h_{TP}(d_{A_2})]$. Likewise, database *B* sends d_{B_2} to the trusted third-party by sending $Enc_{e_{TP}}[d_{B_2} || h_{TP}(d_{B_2})]$. With the knowledge of d_{B_2} from database *B,* the TTP computes $Decrypt_{d_{B_2}}\langle Enc_{e_{B_2}}[Random\{Enc_{e_{A_1}}[h(I_{AB})]\}]\rangle$. After the decryption, the trusted third-party sends $Random\{E_{e_{A_1}}[h(I_{AB})]\}$ to database *A.*

Also, with the knowledge of d_{A_2} from database *A,* the TTP computes $Decrypt_{d_A}\langle Enc_{e_{A_2}}[Random\{Enc_{e_{B_1}}[h(I_{AB})]\}]\rangle$. After the decryption, the trusted third-party sends $Random\{E_{e_{B_1}}[h(I_{AB})]\}$ to database *B.*

The decrypted intersection received from the trusted third-party is the encrypted hash of the files the databases have in common. For the databases to know the actual files they have in common, the intersection received from the trusted third-party is decrypted. These decrypted files are then compared with the hashed files of their individual databases. The final decrypted files in the intersection computed by database *A* has $I_{AB} \in h(a_i), i = 1, \dots, p$ where $p \ll n$. Also, the final decrypted files in the intersection computed by database *B* has $I_{AB} \in h(b_i), i = 1, \dots, p$ where $p \ll m$. Database *A* compares hash of the files obtained from the intersection computed $h(a_1, a_2, \dots, a_p)$ and the hash of the original files $h(a_1, a_2, \dots, a_n)$ in order to know the actual files that are in common with the other database. Likewise, database *B* compares hash of the files obtained from the intersection computed $h(b_1, b_2, \dots, b_p)$ and the hash of the original files $h(b_1, b_2, \dots, b_m)$ in order to know the actual files that are in common with the other database. Two hashed files are the same if they are semantically equal.

Security

The protocols in this paper are designed to prevent both honest-but-curious and external attacks. The protocols are resistant against external attackers however, honest-but-curious attacks need to be prevented. The files in the databases are hashed and encrypted before they sent to the other database. The cryptographic hash function and the encryption will prevent both honest-but-curious and external attackers from knowing the content of the files of the database.

During the protocol, database *A* sends $\left\{Random\left\{Enc_{e_{B_1}}[h(B)]\right\}||Enc_{e_{A_2}}\left[Random\left\{Enc_{e_{B_1}}[h(B)]\right\}\right]\right\}$ to database *B.* Database *B* also sends $\left\{Random\left\{Enc_{e_{A_1}}[h(A)]\right\}||Enc_{e_{B_2}}\left[Random\left\{Enc_{e_{A_1}}[h(A)]\right\}\right]\right\}$ to database *A*. The sending of $\left\{Random\left\{Enc_{e_{B_1}}[h(B)]\right\}||Enc_{e_{A_2}}\left[Random\left\{Enc_{e_{B_1}}[h(B)]\right\}\right]\right\}$ and $\left\{Random\left\{Enc_{e_{A_1}}[h(A)]\right\}||Enc_{e_{B_2}}\left[Random\left\{Enc_{e_{A_1}}[h(A)]\right\}\right]\right\}$ by databases *B* and *A* respectively serves as a check against tempering by attackers. If say database *A* observes that, $Random\left\{Enc_{e_{A_1}}[h(A)]\right\}$ received from database *B* is different from what was sent to database *B,* database *A* terminates the protocol and reports to the trusted third-party. The randomization of the second part of the concatenation prevents a database from undertaking known cyphertext attack on the protocol.
In order to prevent honest-but-curious attacks, the protocol is symmetric – databases *A* and *B* compute the intersection simultaneously. Furthermore, the content of the files in the databases should be large in order to prevent dictionary attacks. The databases send the computed intersection to the trusted third-party. The trusted third-party on receiving I_{AB} knows it is the intersection computed from the protocol between databases *A* and *B.* The trusted third-party verifies whether I_{AB} from database *A* is the same as I_{AB} from database *B.* The protocol is terminated if the intersection received from the databases are not the same.

With the knowledge of d_{A_2} and d_{B_2} from databases *A* and *B* respectively, the trusted third-party computes $Decrypt_{d_{A_2}}\langle Enc_{e_{A_2}}\left[Random\left\{E_{e_{B_1}}[h(I_{AB})]\right\}\right]\rangle$ and sends $Random\left\{Enc_{e_{B_1}}[h(I_{AB})]\right\}$ to database *B.* Likewise, the trusted third-party computes $Decrypt_{d_{B_2}}\langle Enc_{e_{B_2}}\left[Random\left\{Enc_{e_{A_1}}[h(I_{AB})]\right\}\right]\rangle$ and sends $Random\left\{E_{e_{A_1}}[h(I_{AB})]\right\}$ to database *A.* At this point, the trusted third-party does not know the content of the actual files the databases have in common. Even if the trusted third-party become honest-but-curious, it can know only the hash of the files the databases have in common. Likewise, if the TTP is compromised it is only the hash of the files that will be exposed. Hence the protocol in this paper ensures: (1) guaranteed output delivery – an adversary or honest-but-curious database will not be able to prevent the other database from receiving its output; (2). fairness: the databases receive the same output simultaneously.

Conclusion

This paper has proposed a protocol that is secure against external and honest-but-curious attacks. Hence the protocol can be used for the comparison of the contents of databases over an untrusted network. The ability of the protocols to prevent even the trusted third-party from knowing the content of the files of a database shows its idealness in protecting information of users of the protocol. Furthermore, there is guaranteed output delivery and fairness in the proposed protocol.

© GSJ

REFERENCES

[1] A. C. Yao, 'PROTOCOLS FOR SECURE COMPUTATIONS.', *Annual Symposium on Foundations of Computer Science - Proceedings*, pp. 160–164, 1982, doi: 10.1109/SFCS.1982.38.
[2] K. B. Frikken, 'Secure Multiparty Computation (SMC)', in *Encyclopedia of Cryptography and Security*, Springer, Boston, MA, 2011, pp. 1121–1123. doi: 10.1007/978-1-4419-5906-5_766.
[3] E. De Cristofaro, J. Kim, and G. Tsudik, 'Linear-Complexity Private Set Intersection Protocols Secure in Malicious Model', in *Advances in Cryptology - ASIACRYPT 2010*, M. Abe, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 213–231.
[4] H. Chen, K. Laine, and P. Rindal, 'Fast Private Set Intersection from Homomorphic Encryption', in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, in CCS '17. New York, NY, USA: Association for Computing Machinery, 2017, pp. 1243–1255. doi: 10.1145/3133956.3134061.
[5] S. Contiu, S. Vaucher, R. Pires, M. Pasin, P. Felber, and L. Réveillère, 'Anonymous and confidential file sharing over untrusted clouds', Jul. 2019, doi: 10.1109/SRDS47363.2019.00013.
[6] C. Dong, G. Russello, and N. Dulay, 'Shared and searchable encrypted data for untrusted servers', in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, pp. 127–143. doi: 10.1007/978-3-540-70567-3_10.
[7] X. Pei, X. Wu, and L. Sun, 'Practical Data Sharing at Untrusted Stores', *2020 10th Annual Computing and Communication Workshop and Conference, CCWC 2020*, pp. 506–510, Jan. 2020, doi: 10.1109/CCWC47524.2020.9031216.
[8] G. Zhao, C. Rong, J. Li, F. Zhang, and Y. Tang, 'Trusted data sharing over untrusted cloud storage providers', *Proceedings - 2nd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2010*, pp. 97–103, 2010, doi: 10.1109/CLOUDCOM.2010.36.