A Privacy Preserving Scalable Architecture For Collaborative Event Correlation

Hani Qusa, Roberto Baldoni, Roberto Beraldi
Dipartimento di Ingegneria Informatica Automatica e Gestionale "Antonio Ruberti"
Università degli Studi di Roma, Via Ariosto 25 - 00185 Roma, Italy
$\{$*qusa,baldoni,beraldi*$\}$*@dis.uniroma1.it*

***Abstract*—We propose an efficient software architecture for private collaborative event processing, enabling information sharing and processing among administratively and geographically disjoint organizations over the Internet. The architecture is capable of aggregating and correlating events coming from the organizations in near real-time, while preserving the privacy of sensitive data items even in the case of coalition of attackers. Although there is a rich literature in the field of secure multi party computation techniques that preserve the privacy in a distributed systems, the ability of such systems to scale up horizontally (number of participants) and vertically (dataset per participant) is still limited.**

The key novelty of the architecture is the usage of a pseudo-random oracle functionality distributed among the organizations participating to the system for obfuscating the data, that allows for achieving a good level of privacy while guaranteing scalability in both dimensions. Some preliminary performance results are provided.

***Keywords*-Privacy-preserving, secure multiparty computation, collaborative environments**

I. Introduction

Data level collaboration among different organizations is a key factor for increasing their productivity with consequent benefits for customers, such as improving competitiveness and cost reduction [1]. In addition, sharing information allows for a deeper analysis in the context of the organization's security. For example, organizations that suffer from distributed denial of service (DDoS) attack [2], know that they have been attacked, but they cannot easily distinguish the group of IP addresses that commit this attack alone. Therefore, an essential correlation of malicious activities from different monitoring networks can help in extracting robust attack signature and prevent such an attack in advance [3], [4].

Hence, these systems often consider the privacy issues as one of the main requirements to be satisfied during the analysis and the processing of the collected data. Secure Multiparty Computation (SMC) techniques have been studied for designing a system where a set of data distributed among several organizations who are interested in jointly running a computation over these data, while at the same time preserve the data privacy without relying on a trusted third party. However, even though SMC provides a strong privacy and security guarantees, designing a practical solution using these techniques in terms of time, computation and communication, is still considered as non-trivial issue.

The problem we are discussing in this paper is refereed in the literature as *privacy-preserving data aggregation*. We are particularly interested in collaborative environments where a set of participants agree on a common form of data and want to compute a specific function over the collected data from all of them. For example, the participant's input is represented as a key-value pair event. This event indicates a specific measurement in the network such as a suspicious events occurred in these organizations. The function to be computed usually includes aggregation of the data. In this context, aggregation has the meaning of event correlation, where the values of the same key are added together.

The risk of the privacy comes from the behavior of some participants, where all of them are interested in getting the results of collaboration by following the protocol assigned to them step by step, but some of them try to get more information about the others by getting their private data illegally. This breach of the private data can lead to financial loss or a degradation in the reputation of some participants in the environment. The appropriate description and modeling of this behavior is well-known as semi-honest adversary model (or honest but curious adversary model) [5].

Furthermore, the risk of the privacy becomes more serious when a set of semi-honest participants, a coalition, collude together after the execution of the protocol and attempt to deduce additional information from non-colluding participants. This behavior is considered as malicious behavior, where there is an intention of one participant to share information illegally with another participant in the environment in order to get more information about others. The risk of the violation of the privacy of some participants can limit their contribution and affect the efficiency of the collaboration. In particular, we consider only the risk raised by the semi-honest adversary model while other types like malicious adversary model is not covered in this work.

A reference scenario: the financial infrastructure

A number of service providers (e.g., banks) make available a set of on-line services to their customers. Service provision relationships are regulated by means of *contracts* as depicted in figure 1. A contract defines the rights and obligations the involved parties have to comply with. Dependability requirements can be specified in the contracts and include, among the others, *privacy guarantees* customers may require and service providers can ensure. For example a contract specifies clauses related to the non disclosure of sensitive data (e.g., customers' identities) to any third party unless the customer is doing suspicious activities[1]. In the latter case, a bank is relieved by any privacy obligation and can disclose information[2].

To protect themselves from frauds and cyber attacks, banks share information on top of the internet through a *collaborative processing system* for timely discovery of customers performing cyber attacks, and thus mitigating the risks those attacks can cause to bank systems (e.g., unavailability of the services, economic losses[3], damage to reputation)[8]. For this purpose, we assume that banks use a short form of NetFlow data model [9] for this early detection process, where the form comprises the source IP address and the traffic amount occurred by this IP address in a specific period of time. However, the IP address can lead to some information about the costumers of the banks. This knowledge can be exploited by the other competitors in the collaborative environment for their advantage in the case of sharing this information in clear. For instance, bank A gives an IP_A of one of its costumers to bank B, then bank B can use some services provided by advertising companies like Phorm [10] which has an agreement with ISPs, and runs deep packet inspection in order to extract the habit of that costumer. After that, bank B exploits this information to advertise more attractive offers making that costumer moves from bank A to bank B. Thus, there is a need to obfuscate the source IP address, before sharing it with the others. While, the rest of the data, known as the value part, could include more than one field and can be shared in clear to preserve the accuracy of the computation. The knowledge of the value part reflects very little information about the costumers of a specific bank without the knowledge of the source IP address.

Practically, as not all customers misbehave; then sensitive data (IP address) of honest customers are to be kept secret during the collaborative computation, with respect to the other competitive banks while the sensitive data of misbehaving costumers must be revealed to all without harming the reputation of the bank that hosts that costumer.

Our Contribution. In this paper, we design, implement, and evaluate a system that provides event correlation through different participants in a scalable way that can support the architecture shown in figure 1. In doing this, the system preserves the privacy of the data and the privacy of the participants, despite of a set of colluding participants. In the proposed architecture, each participant of the system has an obfuscated copy generated in a pseudo-random way of the

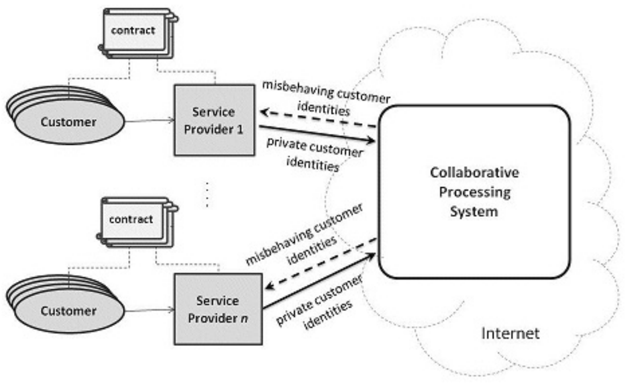

contract
Customer
Service Provider 1
misbehaving customer identities
private customer identities
contract
Customer
Service Provider n
misbehaving customer identities
private customer identities
Collaborative Processing System
Internet

Figure 1. Contract-based privacy preserving architecture

whole dataset collected from all the participants. Nothing can be inferred from these datasets separately. However, the original data can only be reconstructed by combining all the local dataset together. This local copy allows for each participant to perform event correlation locally, thus avoiding costly distributed event correlation operations. Hence, the system achieves a good level of privacy while guaranteing scalability in horizontal (number of participants) and vertical (dataset per participant) dimensions.

The rest of the paper is organized as follow: Section II presents the related works. Section III discusses System properties and assumptions. Section IV presents the privacy preserving architecture. Section V discuss the privacy analysis and the proof of the privacy properties. Section VI shows the experimental results and performance evaluation of the privacy preserving architecture and finally Section VII concludes the paper.

II. Related works

Aggregating and analyzing distributed data collected from different participants has been studied in several ways. We categorize these ways under three main approaches: centralized, fully decentralized, and semi-centralized. Many of the existing solutions rely on centralized approach for the aggregation like depending on the existence of trusted third party(TTP) [11], [12]. In this approach, TTP aggregates the data from different participants, analyzes and processes it, thereby learns the identity and the data of all the participants. Hashing is considered as one of the main techniques for preserving the privacy of sensitive data of the clients in centralized solutions, where the clients hash their sensitive fields (for example, using SHA-256) before sending it to the TTP server, so the server can see only the hashed values of the sensitive data [13], [14], [15]. This approach is not recommended in the case if the participants are rivals in the same market (i.e. like a set of banking systems or internet service providers) for two main reasons: 1) Colluding between any participant and the TTP intentionally or unintentionally (i.e. one participant can attack the TTP) will lead to a privacy breach of all the participants. 2) finding

[1]Suspicious activities have a different meaning from country to country according to country's legislation.

[2]Protection of the financial infrastructure has been deeply investigated in the context of the EU CoMiFin project [6].

[3]Recent studies evaluate around 6 millions dollars per day the tangible loss for a utility company of a down of an e-service [7].

a TTP that all participants can send their sensitive data to it with a high certainty that their sensitive data will not be revealed illegally is not a practical and easy issue. These reasons can prevent someone from participating and gaining the benefits from such collaboration or at least minimize the contribution in a way that makes this collaboration insufficient and useless.

Another approach is for fully decentralized systems. The theoretical cryptographic solutions provided in this approach satisfy a very strong notion of the privacy and the security. In general, these tools are not efficient enough to be used in practice. Secure multi-party computation (SMC) is the formal description of these tools and techniques. In SMC, few systems have been implemented [16], [17], [18] rely on secret sharing scheme for developing multi-purpose private computation. [19] is another SMC system that rely on garbled circuits for running a set of functions over distributed private data. Secure set intersection [20], [21] has been considered as more efficient solution when the number of participants is small. However, while most of these solutions have a strong guarantee in the context of the privacy and some of them achieved a quiet good efficiency in a small set of participants, non of them achieves a practical efficiency in our setting where the number of participants is increasing and generating a large amount of data to be processed in a reasonable time.

A new approach, semi-centralized approach, provides a privacy guarantee when a small set of participants try intentionally to collude in order to deduce more information during computation. In [22], Authors proposed a solution that has this guarantee of privacy. The environment includes a set of participants and additional two components called *proxy* and *database*. Database is responsible for correlating encrypted data coming from all participants through the proxy. Proxy is responsible for blinding participant's input and forwarding it to the database for correlation. Authors proved that the system preserves the privacy of the data and the privacy of the participants and anti-colluding in the case if there are a coalition between participants among themselves, coalition between participants and the proxy, or a coalition between the participants and the database. In this system, there is a possibility for some participants to have an access to the proxy and to capture the data passing through it, which could lead to a privacy breach, especially, the privacy of the participants.

III. System properties and assumptions

The collaborative environment is composed of n uniquely identified participants that agree on a common data format consisting of a set of key-value records. We assume that the key is the sensitive part of the data and we want to aggregate in a private way all the records of the data by summing the value part of the same keys and to reveal the keys that have a total sum value greater than a specific threshold.

A. Privacy properties

Our system guarantees that no one of the participants can know or link any key of the other participants in the collaborative environment, unless a specific data pattern is satisfied.

Therefore, generally, we define the privacy properties to be guaranteed in these systems by the following points:

- ***Data Privacy:*** At the end of the computation, no one can deduce any additional information about the private data of the other participants in the collaborative environment.
- ***Participant Privacy:*** At the end of the computation, no one can link between the disclosed data, that represents a specific behavior, and the original owner of this data.
- ***Coalition Resistance:*** In the presence of a set of colluding participants, a coalition, the system must guarantee the data privacy and participant privacy of the non-colluding participants.

More details about these properties will be provided later in privacy analysis section.

B. Threat Model

We consider semi-honest (known also as Honest-but-Curious) adversary model for describing the behavior of the adversary in the environment. A semi-honest adversary is assumed to faithfully follow all protocol specifications. However, it attempts to infer additional information from the local views and the intermediate messages obtained during or after protocol's execution.

Furthermore, we focus our attention on a specific malicious behavior where a set of semi-honest participants, a coalition, collude together after the protocol's execution by exchanging their local views and intermediate messages in order to deduce additional information about the private data of non-colluding participants.

C. Cryptosystem schemes

Two main cryptosystem schemes are used in the proposed privacy-preserving architecture, (i) Shamir's secret sharing which is used to preserve the privacy of the key part during the correlation process and (ii) Oblivious Pseudo Random Function (OPRF) helps in getting the same secret values of the same keys for all participants and keeps the comparability of the private data. These two techniques are summarized next.

1) Shamir secret sharing: Shamir's (k, n) secret sharing scheme [23] permits to keep data d secret among n participants in a way that the secret (data) can be easily reconstructed if and only if any k out of the n participants make their local data (called the *shares*) available, where $k \leq n$ provides the strength of the scheme. This is achieved by generating a random polynomial f of degree $k-1$ defined over a prime field $\mathbb{Z}_p$, with $p > d$ and such that $f(0) = d$. The polynomial is used to generate n different shares, $d_1, d_2, .., d_n$, where $d_i = f(i)$. The vector of shares is denoted by $s[d]$, i.e., $s[d] = s_1[d], s_2[d], .., s_n[d]$, whereas the i-th share is denoted as $s_i[d]$. The secret is reconstructed by exploiting the Lagrange interpolation technique.

2) Oblivious Pseudo Random Function: An oblivious pseudo-random function (OPRF) [24], [25] is a protocol executed between two parties: a client C and a server S. OPRF is used when a client has an input k and wishes to obtain a blinded version of k. At the end of the OPRF protocol, the participant learns $F_s(k)$ and nothing else, and the server learns nothing. Basically, let G_g be a multiplicative group with a generator g, F_s be a function that contains a vector of m values $\{s_1, s_2, ..., s_m\}$ selected according to the seed s known only by the server, and k be an array of m-bits = $\{x_1, x_2, ..., x_m\}$, then $F_s(k) = g^{\prod_{x_i=1} s_i}$. Oblivious Transfer (OT) protocol is used in order to make the server unable to recognize the sequence of the selected values from the m-vector according to the 1-bit in the input key k.

IV. Privacy preserving architecture

The architecture of our collaborative environment consists of n participants. Each participant has three main components 1) The Edge Gateway, 2) The Processing Unit and 3) The Anonymizer Proxy. The participant's components are used in three different phases of the data processing. The edge gateway executes the first level of aggregating, filtering, and anonymizing of the input data with a help from all anonymizer proxies in all participants, whereas processing unit aggregates the private data. The phases work in a pipeline as described next.

A. Pre-processing phase

This phase is carried out by the Edge Gateway component. The component is responsible for (i) encrypting sensitive data items, and (ii) injecting encrypted data to the Collaborative Processing System. The Edge Gateway embodies two modules, namely the *privacy-enabled pre-processing* and *data dissemination* modules.

1) Privacy-enabled pre-processing module: This module transforms raw data (e.g., received from a log file of a web server) into a set of m key-value pairs (k, v) following a predefined format. The key (which represents the sensitive part) of each pair is then used as the secret of the Shamir's protocol, so that each participant generates a vector of shares $s[k]$ for each key k they have. The number of generated shares in the vector is equal to the number of participants in the environment, where later, each participant will hold one share from the vector. In order to perform correlation process correctly, all participants need to produce the same list of shares for the same key k. For this reason, the Shamir's protocol in our system uses a pseudo-random generator initialized with the same value, called seed, in order to generate the same set of coefficients required for the polynomial function which are used to generate the shares in the Shamir's protocol. Using the input key k as the seed makes the system vulnerable against dictionary attack. Hence, we use OPRF protocol described before in order to compute the seed in the following way: the participant divides the input key k into a set of parts equal to the number of participants in the system $part_1(k), \ldots, part_n(k)$ where each part comprises a set of bits ($part_i(k) \in \{0,1\}^*$). After that, the participant starts running OPRF protocol for each part $part_i(k)$ by contacting the anonymizer proxy component in each participant where the edge gateway in the participant represents the client side of the OPRF protocol with input $part_1(k), \ldots, part_n(k)$ and the anonymizer proxy in each participant represents the server side of OPRF protocol. At the end of this process, the seed can be computed using the following equation:

$$seed(k) = F_s(k) = \prod_{i=1}^{n} F_{s_i}(part_i(k)) \qquad (1)$$

where s_i is the seed of the anonymizer proxy of the participant P_i. It's worth to mention that each participant will contact at most a number of proxies equal to the number of bits in the input key. This leads to the fact that in case if the number of participants is greater than the number of bits in the input key, then some proxies will have the same seeds.

Unfortunately, this doesn't prevent that two shares belonging to two different keys from being the same. Therefore, a perfect hash function is applied to the concatenation of all the shares, so that each share can correctly be associated to its original key (this is required by the processing phase, as detailed next).

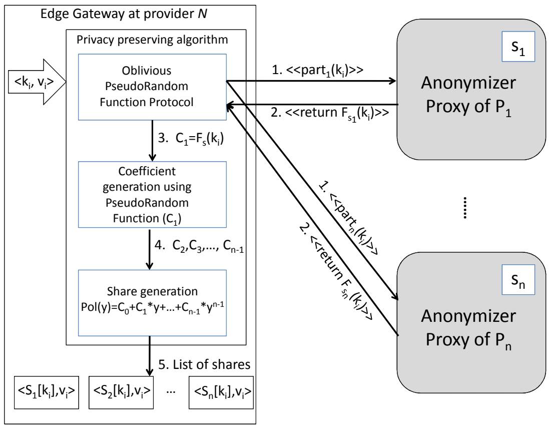


Figure 2. Privacy preserving algorithm

In more details, as depicted in figure 2, the module performs the following operations for each key-value pair in the input file:

1) The participant divides the input key k into parts

and starts running OPRF protocol with each proxy component for each part in order to generate $F_s(k)$.

2) $F_s(k)$ is used to initialize a pseudo-random function that generates the coefficients of the share generation polynomial.
3) The set of shares $s[k]$ is generated using Shamir's secret scheme using the polynomial prepared in the previous step, where $f(0) = k$.
4) For each generated share $s_i[k]$, a triple $(hash(s[k]), s_i[k], v)$ is prepared and added into a list L_i. It's worth to note that the hash value is one-to-one mapping between the original and obfuscated key, and it's used in the aggregation phase while the share $s_i[k]$ is used only in the reconstruction phase.

2) Data Dissemination Module: This module is in charge of disseminating the previously generated lists to all the participants. The dissemination occurs periodically, i.e., every fix time window. The beginning and end of each period is demarcated through special signaling messages. We denote L_{ij} to be the List L_i produced by P_j.

Participants are connected as a logical ring. Participant P_i starts the collection of lists L_{i*} from all the other participants by sending a start-collection message in the form of an empty token, T_i. The token is circulated along the ring until all the participants have appended their lists to it. After that, P_i removes the token. To avoid link-ability, the first participant that puts the list inside the token is determined at random. Specifically, when participant P_j (where $i \neq j$) receives T_i for the first time then, if the token is empty it adds its list L_{ij} to T_i with a probability p_{start}; otherwise, it adds the list for sure. P_i removes the token from the ring when the token passes through it for the second time nonempty and unmodified. This ensure that all participants have added to the token their lists. P_i then sends the list $\Lambda_i = L_{i1}, L_{i2}, \ldots, L_{in}$ to private processing module and the next private processing phase begins. It's worth to note that L_{ii} is never sent to any participants and it's added to Λ_i after delivering the token.

B. Private processing phase

During this phase each participant processes the received list Λ_i in order to check for anomalies. This phase is implemented by the private processing unit. The unit uses the MapReduce framework [26]. The processing logic is represented by high level query language which is compiled into a series of MapReduce jobs. Specifically, the language supports SQL-like query (e.g.,we use HIVE [27]) constructs that specifies the *data pattern* to be discovered on the set of input data. A query engine inside each private processing unit retrieves the data in the storage elements and aggregates them according to one or more SQL-like queries. The output of the query is a subset of Λ_i, denoted as Λ_i^*.

C. Reconstruction phase

In this phase, the secret associated to Λ_i^* has to be retrieved. Each reconstruction unit sends Λ_i^* and waits for receiving a similar list from all the other participants. Each unit then applies the Lagrange interpolation algorithm to reconstruct the original secret. The reconstruction algorithm is organized as sequence of reconstructions. The first interpolation is applied using the first share in the lists received from the participants, the second interpolation is applied using the shares in the second position, etc.

V. Privacy analysis

In this section, we discuss the privacy properties of the proposed system:

Property 1 (Data privacy). Let k_i and k_j be two input keys owned by P_i and P_j respectively. Then, P_i can reconstruct the input key k_j of the other participant P_j if and only if k_j appears in the short list, or $k_j = k_i$. In other words, P_i can know nothing about k_j unless: (i) $s_i[k_j] \in \Lambda_i^*$ or (ii) $s_i[k_j] \in \Lambda_i$ and $k_i = k_j$.

In order to show that, let $s[k_j]$ be the vector of n-shares for the key k_j produced by participant P_j and let DP be the data pattern applied during the processing of the data. Now, each participant will hold one share of $s[k_j]$ and all the participants will apply the same data pattern DP over this data. If the share $s_i[k_j]$ that is held by participant P_i satisfies the data pattern DP, then it will appear in the short list Λ_i^* (that must be reconstructed) of the participant P_i as well as for any other participant. This means that, eventually, P_i will reconstruct and know the value of input key k_j.

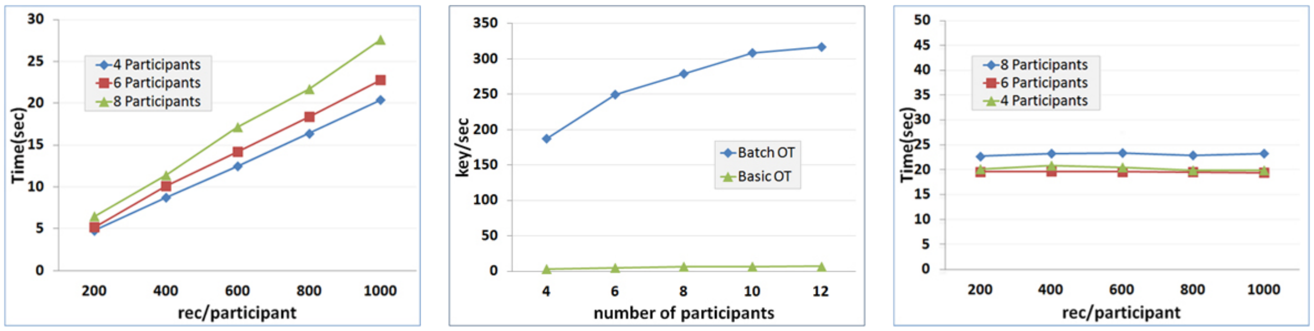
However, if participant P_i has the same key ($k_i = k_j$), then P_i can deduce from the share $s_i[k_j]$ that some one in the environment has the same input key k_i, even if this share doesn't appear in Λ_i^*, but it cannot know exactly who is that one according to the participant privacy guarantee.

Property 2 (Participant privacy). Any participant P_i is unable to link any input key k_j with the owner P_j.

In fact, for P_i being able to link k_j to P_j, P_i must know the insertion order of shares in the token T_i as well as the number of shares that each participant added to the token. However, this is not possible as the first participant that starts adding shares in the token T_i is random (recall that a participant begins the data dissemination phase with probability p_{start}) and a participant can add an arbitrary number of shares.

Property 3 (Coalition resistance). A coalition of $n-1$ participants cannot break data privacy by revealing the original value k_i of a non-colluding participant P_i.

This is because P_i will only distribute $(n-1)$-shares, and keep one share for its use. As the system uses *(n-out-of-n)*



(a) (b) (c)

Figure 3. (a)Privacy-preserving mechanism overhead , (b)Scaling number of keys/sec., (c)Processing unit throughput.

Shamir secret sharing, so a collection of $n-1$ shares is not enough to reconstruct the original value k_i.

VI. Experimental results

The proposed architecture has been implemented in Java and run on a cluster of 4 quad core 2.8 Ghz dual processor physical machines, equipped with 24GB of RAM. The physical machines are connected to a LAN of 10Gbit, running ubuntu linux.

A. *Privacy-preserving mechanism analysis*

In order to asses the time overhead added by the proposed privacy-preserving mechanism with respect to the processing of the collected data without the privacy-preserving mechanism, we analyzed the time required by all the modules involved in this mechanism. Practically, the total time of the privacy-preserving mechanism is the sum of the time required by privacy-enabled preprocessing module(OPRF), the shares generation and the reconstruction modules.

In order to asses the overhead introduced by this OPRF module, we have first implemented a basic 3-way oblivious transfer (OT_1^2), which comprises one encryption in the gateway component side and one decryption on the proxy component side. We used RSA implementation for encryption and decryption. All the encryption and decryption used 1024-bit key. The proxy required 35 ms to perform RSA decryption, while the participant needs 9 ms for RSA encryption. The computation of the seed of one IP address (the input key), then requires $(35+9) \times 32 \approx 1400$ msec, regardless the time needed for the communications.

In order to reduce this time, we implemented batch oblivious transfer(b-OT_1^2, where b is the size of the batch) based on batch RSA proposed in [28]. Basically, only one decryption has to be computed for every set of small size of public exponents encryptions. In our experimentation, we use a batch of size b=2 which requires just two small encryptions. This leads to a very short decryption time. Furthermore, we adopt offline computation technique, where only one 3-way communication is needed instead of using multiple 3-way communications used in the basic OT_1^2. Hence, the computation time of the seed of one IP address was reduced to 40 ms.

The computation time required to generate the shares for one IP is about 0.12 ms while the reconstruction phase takes about 0.2 ms per IP. The share generation and reconstruction times vary very little according to the number of exponentiations, which are determined by the number of participants in the environment.

Figure 3(a) shows the total time overhead of the whole privacy-preserving mechanism as a function of the number of input key/participant and a different number of participants. We notice that the relationship of the increasing number of input key per each participant (vertical scalability) is linear in time.

Figure 3(b) shows the throughput of the system (total number of keys per sec) as a function of the number of participants (horizontal scalability). We can see that throughput increases with the number of participants and reaches more than 300 keys per second for 12 participants. Clearly, the load of each proxy increases with the number of participants and this will eventually limit the throughput. However, proxies can be duplicated as soon as their number becomes greater than the size of the input key.

B. *Data processing analysis*

Inside the processing unit, the data is processed by a collection of Hadoop's MapReduce jobs inside participant. The processing logic, which represents our data pattern, is expressed in HIVE-QL. Figure 3(c). the throughput of this phase. The time for processing the data is an average of ≈22 sec in all cases. This is because Hive is used mainly for processing large dataset and it requires a high start-up time, while in our experiments, the dataset comprises 12000 records in case of 12 participants with 1000 record per each one which is not big enough in terms of Hive.

In fact, the processing time of the data in this unit is not influenced by the privacy-preserving mechanism. In other word, the time of processing the obfuscated data is the same time of processing non-obfuscated data.

VII. Conclusions and future work

In this paper, we presented the design, implementation, and the evaluation of privacy-preserving mechanism for a scalable collaborative environment that can scale up in input data as well as in the number of participants. The proposed mechanism avoids the time complexity of the fully decentralized system and doesnt rely on a TTP. In our vision, we introduced a solution that can survive and preserve the privacy in the existence of a coalition between participants.

In our future work, we intend to use C++ cryptography library in order to improve our throughput as, currently, we use a fully java implementation which affects the performance of system. We are planing to carry out an intensive experimental evaluation in order to assess more the cost of our privacy mechanism in terms of throughput and accuracy. We also plan to study privacy preserving mechanism while increasing the complexity of the adversary, i.e., passing from colluding honest but curious processes to byzantine ones.

References

[1] F. Cate, M. Staten, and G. Ivanov, "The value of information sharing," 2000.

[2] C. V. Zhou, C. Leckie, and S. Karunasekera, "A survey of coordinated attacks and collaborative intrusion detection," *Computers & Security*, vol. 29, no. 1, pp. 124 – 140, 2010.

[3] G. Zhang and M. Parashar, "Cooperative detection and protection against network attacks using decentralized information sharing," *Cluster Computing*, vol. 13, no. 1, pp. 67–86, Mar. 2010.

[4] Y. Xie, V. Sekar, M. Reiter, and H. Zhang, "Forensic analysis for epidemic attacks in federated networks," in *Proceedings of the Proceedings of the 2006 IEEE International Conference on Network Protocols*, ser. ICNP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 43–53.

[5] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries." Secaucus, NJ, USA: Springer-Verlag New York, Inc., Apr. 2010, vol. 23, no. 2, pp. 281–343.

[6] R. Baldoni and G. Chockler, *Collaborative Financial Infrastructure Protection*. Springer, 2011.

[7] S. Baker and S. Waterman, "In the crossfire: Critical infrastructure in the age of cyber war," 2010.

[8] G. Lodi, L. Querzoni, R. Baldoni, M. Marchetti, M. Colajanni, V. Bortnikov, G. Chockler, E. Dekel, G. Laventman, and A. Roytman, "Defending financial infrastructures through early warning systems: the intelligence cloud approach," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research*, ser. CSIIRW '09. New York, NY, USA: ACM, 2009, pp. 18:1–18:4.

[9] "Introduction to cisco ios netflow," 2007.

[10] "Phorm: profiling of a person's web surfing habits," http://www.phorm.com/, 2012.

[11] "Dshield: Cooperative network security community - internet security," http://www.dshield.org/indexd.html/, 2011.

[12] G. Vigna and R. A. Kemmerer, "Netstat: a network-based intrusion detection system," *J. Comput. Secur.*, vol. 7, no. 1, pp. 37–71, Jan. 1999.

[13] M. Allman, E. Blanton, V. Paxson, and S. Shenker, "Fighting coordinated attackers with cross-organizational information sharing," in *In Proc. Fifth Workshop on Hot Topics in Networks (HotNets-V*, 2006.

[14] A. J. Lee, P. Tabriz, and N. Borisov, "A privacy-preserving interdomain audit framework," in *Proceedings of the 5th ACM workshop on Privacy in electronic society*, ser. WPES '06. New York, NY, USA: ACM, 2006, pp. 99–108.

[15] P. Lincoln, P. Porras, and V. Shmatikov, "Privacy-preserving sharing and correction of security alerts," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 17–17.

[16] M. Burkhart, M. Strasser, D. Many, and X. Dimitropoulos, "Sepia: privacy-preserving aggregation of multi-domain network events and statistics," in *Proceedings of the 19th USENIX conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 15–15.

[17] D. Bogdanov, S. Laur, and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *Proceedings of the 13th European Symposium on Research in Computer Security: Computer Security*, ser. ESORICS '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 192–206.

[18] I. Damgård, M. Geisler, M. Krøigaard, and J. B. Nielsen, "Asynchronous multiparty computation: Theory and implementation," in *Proceedings of the 12th International Conference on Practice and Theory in Public Key Cryptography: PKC '09*, ser. Irvine. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 160–179.

[19] A. Ben-David, N. Nisan, and B. Pinkas, "Fairplaymp: a system for secure multi-party computation," in *Proceedings of the 15th ACM conference on Computer and communications security*, ser. CCS '08. New York, NY, USA: ACM, 2008, pp. 257–266.

[20] M. J. Freedman, K. Nissim, and B. Pinkas, "Efficient private matching and set intersection." Springer-Verlag, 2004, pp. 1–19.

[21] L. Kissner and D. Song, "Privacy-preserving set operations," in *IN ADVANCES IN CRYPTOLOGY - CRYPTO 2005, LNCS*. Springer, 2005, pp. 241–257.

[22] B. Applebaum, H. Ringberg, M. J. Freedman, M. Caesar, and J. Rexford, "Collaborative, privacy-preserving data aggregation at scale," in *Proceedings of the 10th international conference on Privacy enhancing technologies*, ser. PETS'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 56–74.

[23] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.

[24] M. Naor and O. Reingold, "Number-theoretic constructions of efficient pseudo-random functions," vol. 51, no. 2. New York, NY, USA: ACM, Mar. 2004, pp. 231–262.

[25] M. J. Freedman, Y. Ishai, B. Pinkas, and O. Reingold, "Keyword search and oblivious pseudorandom functions," in *Proceedings of the Second international conference on Theory of Cryptography*, ser. TCC'05. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 303–324.

[26] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.

[27] "Hive," http://wiki.apache.org/hadoop/Hive, 2011.

[28] A. Fiat, "Batch rsa," New York, NY, USA, pp. 175–185, 1989.