Toward Secure Clustered Multi-Party Computation: A Privacy-Preserving Clustering Protocol

Sedigheh Abbasi, Stelvio Cimato, and Ernesto Damiani

DI - Università degli Studi di Milano, 26013 Crema, Italy
firstname.lastname@unimi.it

Abstract. Despite a large amount of research work has been done and a large number of results produced, the deployment of Secure Multi-party Computation (SMC) protocols for solving practical problems in real world scenarios is still an issue. This is mainly due to the complexity of the SMC-based solutions and to the needed assumptions that are not easy to fit to the considered problem. In this paper we propose an innovative approach for the deployment of SMC, providing a tradeoff between efficiency and privacy. In the Secure Clustered Multi-Party Computation (SCMC) approach, a function is more efficiently computed through reducing the number of participants to the SMC protocol by clustering, such that a reasonable privacy leakage inside the cluster is allowed. Toward this direction, this paper verifies the impact and the feasibility of applying different clustering techniques over the participants to a SMC protocol and proposes an effective specifically-tailored clustering protocol.

Keywords: Secure Multi-Party Computation, Privacy-Preserving Clustering, Privacy and Efficiency Tradeoff.

1 Introduction

Secure computation was introduced by A. Yao [6] in 1982 by presenting the solution to the millionaires' problem as a two-party computation protocol and successively generalized to include more parties [4]. In general, secure multi-party computation (SMC) refers to protocols where multiple participants want to jointly compute the value of a public function over their individually-held secret bits of information without revealing the secrets to the other participants. In such a setting, the security of the protocol is said to be preserved if no participant can learn more from the description of the public function and the result of the global calculation than what he could learn from his own input (under particular conditions and depending on the model used). In other words, in a SMC protocol the only information that should be revealed is what can be reasonably deduced by knowing the actual result of the function. As a fundamental theorem, it has been proved that any multi-party functionality can be securely computed [1, 3, 4, 6].

However, in many cases, these significant achievements are not so efficiently scalable and adaptable to provide a solution to real-world problems. Indeed, few practical applications of SMC protocols are reported in literature, such as the first large-scale

K. Mustofa et al. (Eds.): ICT-EurAsia 2013, LNCS 7804, pp. 447–452, 2013.
IFIP International Federation for Information Processing 2013

practical experiment, where SMC is used to implement a secure sugar-beet auction in Denmark [2] and the usage of SMC to supply chain management, where the shared optimization problem is solved by maintaining the privacy of the user inputs [5]. In both cases, the complexity of the solutions is huge and the completion time of the protocol is directly proportional to the number of participants, limiting the applicability of the proposed solutions.

In order to achieve a more scalable solution, in this paper, we propose to execute SMC protocols over a significantly-reduced number of participants by appropriately clustering the participants. The paid price is in terms of privacy, since intra-cluster computations will rely on a relaxed (or none) privacy model while the SMC protocol will be executed over clusters' representatives among which privacy assumptions will be preserved. This approach in which a SMC protocol is executed over the participants' clusters rather than individual participants is called Secure Clustered Multiparty Computation (SCMC) and is depicted in Fig. 1.

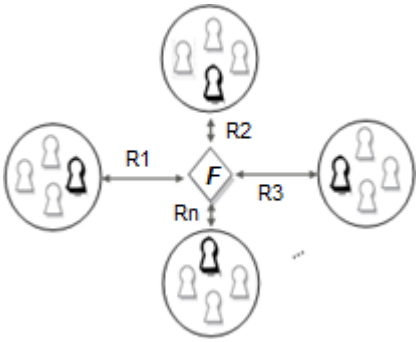

Fig. 1. An abstract view of a SCMC protocol. Bold symbols show clusters' representatives and R1 to Rn denote the results of intra-cluster computations.

In short, the overall execution of a SCMC protocol can be divided in two phases: clustering phase and computation phase. During the clustering phase participants are clustered in such a way that a more efficient intra-cluster computation is possible, by applying a relaxed privacy model. Throughout this paper, the relaxed privacy model we adopt is a trust-based model, where privacy leakages from a confiding participant to a trustee are allowed. Therefore, in its simplest representation, it can be assumed that all the clustering decisions are made based on a square trust matrix (T) in which the element $T_{i,j}$ denotes the amount of trust of participant P_i to the participant P_j. On the other hand, the clustering algorithm itself should be privacy-preserving; because, it uses pair-wise trust values as inputs that are private to the participants. Then, the computation phase starts with intra-cluster computation to evaluate partial results of the clusters and ends with a SMC protocol execution run over the clusters' representatives as participants, holding clusters' partial results, as private inputs.

The purpose of this paper is to address the issues related to the participants' clustering in a SCMC protocol. Clearly, these issues are independent of the specific SMC protocol that is run in the last phase of the protocol and therefore all discussions about

different SMC options, their underlying assumptions and definitions remain unchanged and are outside the scope of our study.

2 A Privacy-Preserving Clustering Protocol in SCMC

In order to achieve an efficient clustering, we propose a general privacy-preserving clustering protocol that is particularly designed to work with the asymmetric trust matrix as input and do not impose any limitation over the representation of trust values. In other words, each participant is independently free to evaluate her trust to the other participants. This protocol, which is similar to a voting system, securely defines the clusters in terms of balanced hierarchies. During the execution of the protocol, the participants are prioritized and the owners of the highest priorities are introduced to the others as candidates for the current election. On the other hand, the protocol utilizes a Trusted Third-Party (TTP) to improve efficiency and prevent possible privacy leakages.

2.1 TTP-Based Prioritized Voting Clustering Protocol

As mentioned above, the Prioritized Voting Clustering (PVC) protocol is in fact a prioritized voting system in which an election is run during each iteration for one or more participants with the highest priorities as the current candidates. A candidate is elected by a participant only if the participant agrees to share her private input with the candidate in the computation phase of the SCMC protocol. In this situation, it is said that the participant has chosen to be a follower of the candidate. Such a successful election leads to the formation of a parent-child relation in the clustering hierarchy in which the parent represents the leader. The candidates are in fact chosen in such an order that favors the creation of the minimum number of balanced clusters, taking into account two factors affecting the participants' priorities: the probability of becoming a follower (a child) and the current level in the clustering hierarchy (all the participants start the protocol at level 0). The first probability can be simply estimated by considering the number of trustees of each participant (the less the number of trustees is, the more probably the participant is a parent). Also, in order to prevent skew clustering, the participants in the lower levels of the clustering hierarchy (closer to the root) are also given higher priorities. Therefore, the priority of a participant is defined as the sum of her level in the clustering hierarchy and her number of trustees: the less the result of this sum is, the higher priority the participant has. Note that, for the sake of simplicity, here a uniform binary trust matrix has been assumed (even if our protocol does not put any limitation on the trust representations used by different participants). On the other hand, since a cluster hierarchy is not allowed to have any cycle, a participant cannot vote for one of its ancestors in the clustering hierarchy. For this reason, all the participants are required to keep track of their followers.

This algorithm preserves the privacy of the private trust vectors in multiple ways: Firstly, by introducing a TTP, secondly by using one of the participants (namely P0) as the inductor and thirdly by utilizing a public-key cryptosystem that is used to hide the status of each participant from the other participants as well as from the TTP. In fact, the algorithm benefits from the TTP not only to prevent the disclosure of the trust values among the participants, but also to create a central coordinator that helps

to improve the efficiency of the clustering by reducing the overall number of needed elections (since more than one candidate in an iteration is considered when possible). On the other hand, in order to preserve the participants' privacy against the TTP itself, the inductor is used: any information from the participants is sent to the TTP after being collected by the inductor, while the results from the TTP are directly broadcast to all the participants. As a result, due to this obfuscation, the participants are kept anonymous to the TTP.

Algorithm 1 illustrates the initialization and the execution phases of the protocol. Note that, this algorithm requires that all the participants are identified by a unique ID that is known to the other participants and kept unknown to the TTP. In this way, the TTP can access the public keys, but it cannot recognize whom a particular key belongs to.

Algorithm 1. TTP-Based PVC Algorithm

Initialization Phase:

1) Each participant marks itself as $Not\ Connected$.
2) All the participants send a tuple $< Enc_{PK}(ID), Enc_{TK}(NOT), PK >$ to the inductor, where PK is the public key of the participant, $Enc_{PK}(ID)$ denotes the encryption of its ID under its public key, and $Enc_{TK}(NOT)$ is the encryption of the number of its trustees under the public key of the TTP.
3) The inductor prepares a randomly permuted list collecting the tuples from all the participants, including him and sends the list to the TTP.
4) The TTP initializes n disjoint sets representing the initial clusters, one for each of the received tuples, using $Enc_{PK}(ID)$ as index. For each tuple he initializes the following variables:

$$ID = Enc_{PK}(ID), PublicKey = PK, Level = 0, Priority = NOT + Level, Elected = 0$$

Execution Phase:

1) The TTP finds the participants with the highest priorities whose $Elected = 0$ (as the candidates of the current election) and performs the following steps:
   1-1) For all the candidates sets $Elected$ to 1.
   1-2) Runs an election by broadcasting the list of candidates' $Enc_{PK}(ID)$s.
2) All the participants do as the following:
   2-1) If he is marked as $Not\ Connected$ and he trusts any of the candidates, he selects one of the candidates and sets her vote as $V = Enc_{PK}(ID')$ where ID' is the identifier of the selected candidate; otherwise the vote is set to $NULL$.
   2-2) The tuple $<Enc_{PK}(ID), Enc_{TK}(V) >$ containing the encrypted vote is sent to the inductor.
3) The inductor collects all the tuples from the participants as well as her own, perform a random permutation and sends the list to the TTP.
4) The TTP decrypts the second element of each pair in the list and extracts the vote (V) corresponding to a particular $Enc_{PK}(ID)$. Then the following operations are done:
   4-1) If $V = NULL$, the result R is set to $NULL$, otherwise $V = Enc_{PK}(ID')$. If $Enc_{PK}(ID')$ and $Enc_{PK}(ID)$ belong to different clusters R is set to $Enc_{PK}(ID')$, the two clusters are merged and $Level$ and $Priority$ variables related to $Enc_{PK}(ID)$ are updated.
   4-2) For each of the received pairs an ordered-pair in the form of $(Enc_{PK}(ID), Enc_{PK}(R))$ is created in which $Enc_{PK}(R)$ is the result encrypted under the corresponding $PublicKey$. These ordered-pairs form the $Results$ list that is broadcast to all the participants.
5) Each participant who has voted in the current election decrypts the corresponding $Enc_{PK}(R)$ using her private key and if R is not $NULL$, he can retrieve the ID that identifies the her leader (her parent in the hierarchy) and marks himself as $Connected$; otherwise the message is ignored by the participant.

This protocol completes when for all the participants $Elected = 1$. In this situation, the TTP broadcasts a $Terminate$ message to all the participants. Also, the TTP may decide to terminate the protocol sooner if it recognizes that a sufficient number of clusters have been achieved or the clustering will not be changed anymore.

2.2 Algorithm Analysis

From the viewpoint of efficiency, in the worst case the protocol is completed after n iterations. This situation occurs when in each iteration only one participant with the highest priority exists. On the other hand, in the best case, the protocol needs only one iteration, that is when the trust matrix leads to the same initial priority for all the participants and, moreover, there is no conflict among the selected candidates. During each iteration $O(n)$ number of $O(nk)$ bit messages are transferred, where k is the security parameter of the underlying field or ring and n denotes the number of participants. Therefore, the overall number of bits communicated during the protocol is $O(n^3k)$ and $O(n^2k)$ in the worst and the best case, respectively. From the computation point of view, a constant number of encryptions and/or decryptions are required in each iteration.

Considering privacy, the security of this algorithm depends on the hardness of the underlying cryptosystem. In other words the protocol itself does not disclose any information about the trust values neither to the other participants, nor to the TTP. The only information revealed from the participants to the TTP is the number of trusted participants each participant has, without revealing which participant particular information belongs to (this information can be utilized by the TTP to recognize when no more connection is possible among the participants and terminate the protocol in fewer rounds).

2.3 Computation Phase

As mentioned earlier, the computation phase of a SCMC protocol can be divided in two stages: *intra-cluster* and *inter-cluster* computation. The first stage is in fact a bottom-up computation in which the participants' private inputs flow up the clustering hierarchies toward their direct parents (their selected leaders). For a parent, then, there are two different strategies: *aggregation* and *shuffling*. In the former, the parent is responsible for computing the function over the inputs collected from its children as well as its input (aggregation) and sending the result to its direct parent if any. In other words, in this computation model partial results of the function are gradually computed and propagated in the clustering hierarchy, and are finally aggregated in the clusters' roots. In the shuffling model a non-root parent does not participate in the actual computation but he is simply responsible for shuffling and forwarding a list containing the inputs as well as their children's. In this way, no non-root participant is aware of the function that is supposed to be computed in the SMC phase. The main purpose of aggregation and shuffling is to prevent *unwanted* inputs' disclosure especially in the case of non-transitive trust relations.

In any case, the intra-cluster computation is over when all the roots receive the partial results or lists of inputs from all their ancestors. Then, the roots do a local computation to evaluate the partial results of the function over the inputs included in their clusters and take part to a SMC protocol as the representatives of their followers.

3 Conclusions

In this paper we analyzed the problem of trust-based privacy-preserving clustering of participants in a SCMC protocol. After studying some of the recently-proposed solutions, we showed that none of the existing clustering protocols are fully consistent with the requirements of SCMC. In future works we plan to improve our proposed clustering protocol by eliminating the TTP and replacing it with a secure distributed protocol. In this way, it would be applicable in other contexts when access to a TTP is not possible.

References

1. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation (Extended Abstract). In: The Proceedings of the 20th Annual ACM Symposium on Theory of Computing (STOC), pp. 1–10 (1988)
2. Bogetoft, P., Christensen, D.L., Damgård, I., Geisler, M., Jakobsen, T., Krøigaard, M., Nielsen, J.D., Nielsen, J.B., Nielsen, K., Pagter, J., Schwartzbach, M., Toft, T.: Secure Multiparty Computation Goes Live. In: Dingledine, R., Golle, P. (eds.) FC 2009. LNCS, vol. 5628, pp. 325–343. Springer, Heidelberg (2009)
3. Chaum, D., Crépeau, C., Damgard, I.: Multiparty Unconditionally Secure Protocols. In: The Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing (STOC), pp. 11–19 (1988)
4. Goldreich, O., Micali, S., Wigderson, A.: How to Play any Mental Game or a Completeness Theorem for Protocols with Honest Majority. In: The Proceedings of the 19th Annual ACM Symposium on Theory of Computing (STOC), pp. 218–229 (1987)
5. Kerschbaum, F., Schropfer, A., Zilli, A., Pibernik, R., Catrina, O., de Hoogh, S., Schoenmakers, B., Cimato, S., Damiani, E.: Secure Collaborative Supply-Chain Management. IEEE Computer 9, 38–43 (2011)
6. Yao, A.C.: Protocols for Secure Computations (Extended Abstract). In: The Proceedings of the 23rd Annual Symposium on Foundations of Computer Science (FOCS), pp. 160–164 (1982)