Experimental Evaluation of Privacy-Preserving Aggregation Schemes on PlanetLab

Francesco Randazzo[†], Daniele Croce[*], Ilenia Tinnirello[*], Cettina Barcellona[*], Maria Luisa Merani[‡],
[†][*]Dipartimento Energia, Ingegneria dell'Informazione e Modelli Matematici, University of Palermo, Palermo, Italy
E-mail: [†] francesco.randazzo05@community.unipa.it - [*] *name.surname*@unipa.it
[‡]Dipartimento di Ingegneria "Enzo Ferrari", University of Modena and Reggio Emilia, Modena, Italy
E-mail: marialuisa.merani@unimore.it

***Abstract*—New pervasive technologies often reveal many sensitive information about users' habits, seriously compromising the privacy and sometimes even the personal security of people. To cope with this problem, researchers have developed the idea of privacy-preserving data mining which refers to the possibility of releasing aggregate information about the data provided by multiple users, without any information leakage about individual data. These techniques have different privacy levels and communication costs, but all of them can suffer when some users' data becomes inaccessible during the operation of the privacy preserving protocols. It is thus interesting to validate the applicability of such architectures in real-world scenarios. In this paper we experimentally evaluate two promising privacy-preserving techniques on PlanetLab, analyzing the execution time and the failure rate that each scheme exhibits.**

***Index Terms*—secret sharing, privacy, data mining, secure multi-party computation.**

I. Introduction

Privacy-preserving data mining refers to the possibility to release aggregate information about the data provided by multiple users, without any information leakage about individual data [1]. This possibility is particularly significant in the emerging scenario of pervasive technologies that constantly track users' location, energy consumption, visited web sites, music and movie downloads: undoubtedly, data correlation can reveal many sensitive information about users' habits.

Two main approaches have been proposed so far for privacy-preserving data mining: altering the data before delivering them to the data miner in such a way that the aggregation results are not compromised, or relying upon more sites that have to cooperate to get the mining results. Data alteration solutions may introduce mining errors, if the alteration is based on random noise [2], and may result extremely complex when homomorphic data encryption is employed [3]. We therefore choose to follow the second approach, exploiting multi-party computation protocols.

Multi-party computation protocols have been designed to compute a function of the data collected from multiple users, without revealing any information except for the final value of the function. Typical architectures for multi-party computation rely on multiple (independent) computation sites, which process randomized versions of the users' data provided by the so called input peers. Different solutions for distributing data among the computation sites exist, based on logic circuits [4], arithmetic circuits [5], or linear secret sharing schemes [6]. Solutions based on linear secret shares computed over small fields, such as Sharemind [7], SEPIA [8], P4P [9] and [10], have been recently demonstrated to effectively scale to large numbers of users. In [11] we put forth two promising secure sum protocols for privacy preserving data mining and analytically quantified the privacy degree and the communication cost that they achieve, as well as their reliability in unstable scenarios, that is, when users unpredictably depart and their data are no longer available. The examined architecture was based on a central data mining server and on users grouped in logical clouds where their data are preliminarily aggregated. The current contribution complements the previous theoretical work, exploring the performance of the proposed schemes in the real testbed environment of PlanetLab.

II. Scenario and Proposal

A. Background

To protect the users' sensitive data during the data mining process, we rely upon the Shamir's secret sharing (SSS) scheme [13]. For this scheme the dealer, who owns a secret s, splits it into n pieces, called shares: the knowledge of any k, with $k \leq n$, or more shares allows the reconstruction of s, whereas the knowledge of fewer than k shares leaves the secret undetermined. This is a typical (k, n)-threshold scheme. To generate the shares, the dealer randomly selects the $(k-1)$ coefficients of a polynomial $p(x)$ whose degree is $k-1$ and sets the known term $p(0)$ equal to the secret itself; then the n shares are determined as: $sh_1(s) = [x_1, p(x_1) \ mod \ q], sh_2(s) = [x_2, p(x_2) \ mod \ q], \cdots, sh_n(s) = [x_n, p(x_n) \ mod \ q]$, where $x_1, x_2, ..., x_n$ are arbitrary integers, or seeds, and q is a prime number larger than both s and n. Interpolating k or more shares it is possible to reconstruct the polynomial coefficients and then obtain s as $p(0)$.

Note that when $k = n$, the shares can be created in a much simpler way, following the so-called trivial secret sharing scheme: in this case the first $(n-1)$ shares are randomly picked in $[0, q]$ in accordance to a uniform distribution, then

This work has been partially supported by the Italian national research project PON 04 i-NEXT.

978-1-4799-5344-8/15/$31.00 ©2015 IEEE

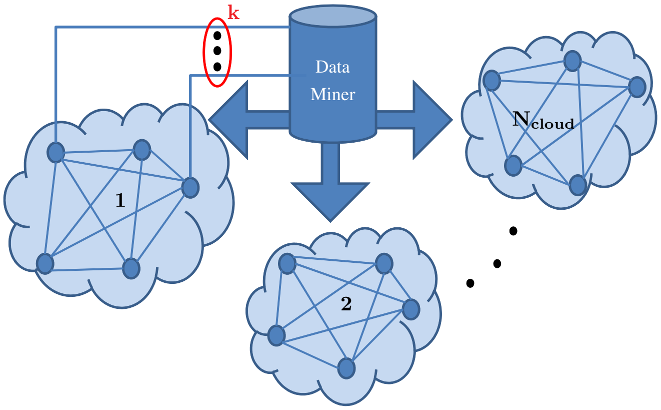

Fig. 1. Multi-Cloud Scenario

the last share is computed as $(s - sh_1(s) - sh_2(s) - \cdots - sh_{n-1}(s))\ mod\ q$.

Both schemes exhibit the *homomorphic property*: when two (or more) secrets are considered, e.g., s_1 and s_2, the sum of two shares of s_1 and s_2 is a share of $s_1 + s_2$, the sum of the secrets. In the specific case of SSS, for this property to hold it is required that the shares of different secrets be evaluated for the same seed y: so, given shares $sh(s_1) = [y, p_1(y)]$ and $sh(s_2) = [y, p_2(y)]$, $sh(s_1) + sh(s_2)$ is a share of $s_1 + s_2$.

B. Framework and Hypotheses

We next describe the partly distributed, privacy preserving architecture proposed to mine the data of a group of users respecting their privacy. We focus on those statistical learning strategies whose update laws require linear operations such as vector addition: on one hand, this allows to exploit the homomorphic property that the previously introduced secret sharing strategies display; on the other, the linear feature is exhibited by several popular data mining algorithms, and therefore does not represent a severe restriction.

We assume that a central unit is interested in knowing the aggregate behaviour of the users, grouped in proper clusters, and that this knowledge has to be acquired without disclosing the single user data. The output of the privacy-preserving data mining process will therefore turn out very useful for both the data miner and the users whose data are being mined; no privacy leakage will occur, an attractive feature given the mining technique typically collects personal, sensitive data.

Rather than the large field operations required by public-key cryptography or homomorphic encryption, we will rely upon SSS scheme to protect users' data from external and internal attackers. Although SSS natively supports both secure addition and multiplication, we limit our analysis to secure sums because a large number of popular data mining and machine learning algorithms can be decomposed and parallelized in simple additions [9].

Let us therefore refer to a multi-cloud scenario where, as depicted in Fig. 1, we have:

- one central unit, also called profiling server, that acts as the data miner and takes on the role of clocking the mining protocol;
- N users (interchangeably termed nodes in what follows) grouped in multiple clouds on the basis of their geographical location: we will denote by N_{cloud} the number of clouds and by $N_{node} = N/N_{cloud}$ the number of nodes in each cloud (with no loss in generality, N is assumed to be a multiple of N_{cloud}).

In this model the profiling server is honest-but-curious, so it follows the protocol without cheating, but it is not totally trusted: it might want to access the users' data for its own purposes. Users will be modeled as honest-but-curious parties too: each of them can collude with other nodes within the same cloud and/or with the server, against one or more victims. Moreover, nodes are not permanently connected to the network: there is a non-null probability p that the node status be *off*, due to either intermittent network connectivity or sudden departure of the user from the network; finally, users independently fail. Examples are wireless users, or users of a peer-to-peer overlay.

The unstable network scenario we just depicted can severely impair the effectiveness of the proposed architectures: we therefore evaluate not only the time needed for each scheme to be completed, but also their robustness against host failures and loss of users' data.

Note that compared to other solutions, we do not rely on trusted third parties or "privacy peers" (as in SEPIA), thus completely eliminating the risk of third party collusion. As we will show, this property comes at the cost of slower performance because the secure protocol involves generic peers in the network without any control on their computing capacity or load. This is the price to be paid to guarantee this desired privacy degree.

C. Base Scheme

In this scheme we propose to employ SSS technique and take advantage of its homomorphic property separately in each cloud. During the distribution phase (DP), shares are created and distributed among the users belonging to the same cloud; during the collection phase (CP), k sums of shares will be delivered to the server, as detailed below. The DP starts when the server randomly triggers a node within each cloud: let us denote this node as node i. Node i of a generic cloud:

1) makes N_{node} shares of its secret data d_i following Shamir (k, N_{node})-threshold scheme, using the nodes identifiers, $j = 0, 1, \ldots, i, \cdots, (N_{node} - 1)$, as seeds;
2) keeps for itself share $sh_i(d_i) = [i, p_i(i)\ mod\ q]$;
3) sends share $sh_j(d_i) = [j, p_i(j)\ mod\ q]$ to node j of its cloud, $j = 0, 1, \ldots, (N_{node} - 1)$, $j \neq i$.

When receiving node i share, every node in the cloud learns that it is time to compute the shares of its secret: it therefore behaves as node i, retaining the share computed with its identifier as seed and sending all other shares to the proper node within the cloud.

Once node i has received the $(N_{node} - 1)$ shares from the nodes within its same cloud, it sums them up and determines

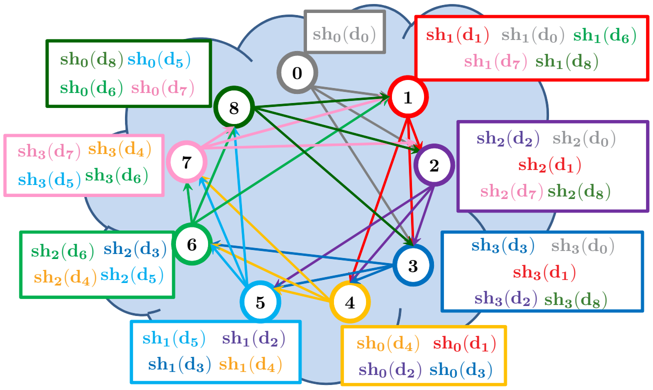
sh$_0$(d$_8$) sh$_0$(d$_5$)
sh$_0$(d$_6$) sh$_0$(d$_7$)
sh$_0$(d$_0$)
sh$_1$(d$_1$) sh$_1$(d$_0$) sh$_1$(d$_6$)
sh$_1$(d$_7$) sh$_1$(d$_8$)
sh$_3$(d$_7$) sh$_3$(d$_4$)
sh$_3$(d$_5$) sh$_3$(d$_6$)
sh$_2$(d$_2$) sh$_2$(d$_0$)
sh$_2$(d$_1$)
sh$_2$(d$_7$) sh$_2$(d$_8$)
sh$_2$(d$_6$) sh$_2$(d$_3$)
sh$_2$(d$_4$) sh$_2$(d$_5$)
sh$_3$(d$_3$) sh$_3$(d$_0$)
sh$_3$(d$_1$)
sh$_3$(d$_2$) sh$_3$(d$_8$)
sh$_1$(d$_5$) sh$_1$(d$_2$)
sh$_1$(d$_3$) sh$_1$(d$_4$)
sh$_0$(d$_4$) sh$_0$(d$_1$)
sh$_0$(d$_2$) sh$_0$(d$_3$)
0
1
2
3
4
5
6
7
8

Fig. 2. Distribution Phase example with $N_{node} = 9$ and $z = 4$

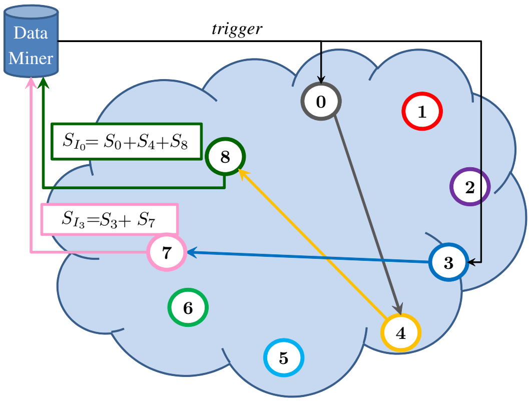

Fig. 3. Collecting Phase example with $N_{node} = 9$, $z = 4$ and $k' = 2$

$S_i = \sum_{j=0}^{N_{node}} sh_i(d_j) = [i, \sum_{j=0}^{N_{node}-1} p_j(i) \ mod \ q]$, which owing to the homomorphic property, is a share of the sum of the secrets. Now the CP begins: during this phase k uniformly randomly designated nodes within each cloud send the server the partial sum they computed along with their identifier: in detail, node h sends the (h, S_h) pair, the server collects k of such contributions, by interpolation finds the polynomial and then the sum of the secrets for that cloud, $S_{cloud} = \sum_{i=1}^{N_{node}} d_i$. Summing together the contributions from all clouds, the server obtains the sum of all users' secrets, as desired.

Note that during the DP and CP each user owns N_{node} shares of N_{node} different secrets: not enough to recover any valuable information about other users. The server only knows partial sums and cannot recover the data of the single user either. However, if k nodes in a cloud are corrupted and form a coalition, they can collect k shares of the other users in the cloud and disclose their secret data: it is therefore important to properly set $k \cdot N_{node}$ to a sufficiently high value.

D. Enhanced Scheme

When the number of nodes within each cloud grows, the distribution phase of the base scheme becomes considerably burdensome. To relieve it, we observe that it is not necessary that in each cloud the generic node sends its secret data shares to every other node.

Accordingly, we modify the previous proposal and employ a (k', z)-threshold scheme, with $z < N_{node}$ and $k' \leq z$, as better detailed next.

Within each cloud, we divide all nodes in z different sets I_r, $r = 0, 1, \ldots, (z-1)$, based on the node identifier, j, so that set I_r includes all nodes such that their identifier satisfies the condition

$$I_r = \{\text{all nodes with identifier } j \,|\, j \ mod \, z = r\} \qquad (1)$$

As an example, with $N_{node} = 9$ and $z = 4$, we have $I_0 = \{0, 4, 8\}$, $I_1 = \{1, 5\}$, $I_2 = \{2, 6\}$ and $I_3 = \{3, 7\}$. Now we require node i to interact with a reduced number of nodes, namely, node i

1) makes z shares based on the (k', z)-threshold scheme, using $0 \leq r \leq (z-1)$ as seed: $sh_r(d_i) = [r, p_i(r) \ mod \ q]$;
2) keeps the share evaluated in $r = i \ mod \ z$;
3) sends $sh_r(d_i)$ to a randomly selected node within set I_r, $\forall r \in [0, (z-1)]$ and $r \neq i \ mod \ z$;

Resuming the previous example, node 1 of Fig. 2 keeps for itself $sh_1(d_1)$ and might choose to send the remaining $z - 1 = 3$ shares as follows: share $sh_0(d_1)$ to node 4 in I_0, share $sh_2(d_1)$ to node 2 in I_2 and share $sh_3(d_1)$ to node 3 in I_3, as the red lines in Fig. 2 indicate; this is simply an example, any combination fulfilling the previous constraints is equally acceptable. At the end of the DP, nodes within the same set I_r will possess only shares evaluated in $r = i \ mod \ z$: note that not all the nodes within a set might receive shares (as it happens for node 0 in Fig. 2). Next, the CP begins: within set I_r, each node sums to the share it kept for itself the shares that it might have received, to compute a partial sum, S_i for node i. Then such partial sums S_i's are collected, in order to determine $S_{I_r} = \sum_{i \in I_r} S_i = \sum_{i=0}^{N_{node}-1} sh_r(d_i)$: a possible choice is to accumulate them in a round robin manner, starting from a specific node triggered by the server, that transmits its contribution to the next downstream node, and gradually covering all other nodes within the set. For the toy example we introduced earlier (see Fig. 3), given that within set I_0 node 0 is triggered, this node sends S_0 (which is the sum of its share $sh_0(d_0)$ plus the shares that it might have received during the previous DP) to node 4; node 4 then adds S_4 and forwards everything to node 8, that adds S_8. It is up to node 8 to deliver S_{I_0} that it just completed to the server. Now it is sufficient that any $k^{'}$ of such sums of shares computed in $k^{'}$ distinct sets of the same cloud be transferred to the server for it to recover S_{cloud}, the sum of the data for the N_{node} users belonging to the examined cloud:

$$S_{cloud} = \sum_{i=0}^{N_{node}-1} d_i \, , \qquad (2)$$

where, as for the previous scheme, last equality stems from the application of the homomorphic property to the partial sum provided by the single cloud; as desired, the sum of the users' data is recovered, without disclosing any of the single contributions to the data miner. Taking into account the presence of more clouds is rather straightforward: the

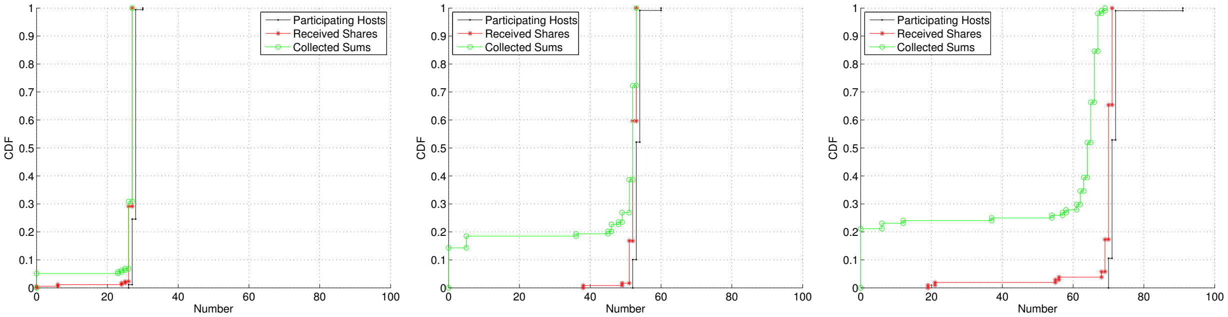

Fig. 4. Performance of the base scheme with 30, 60 and 90 nodes respectively.

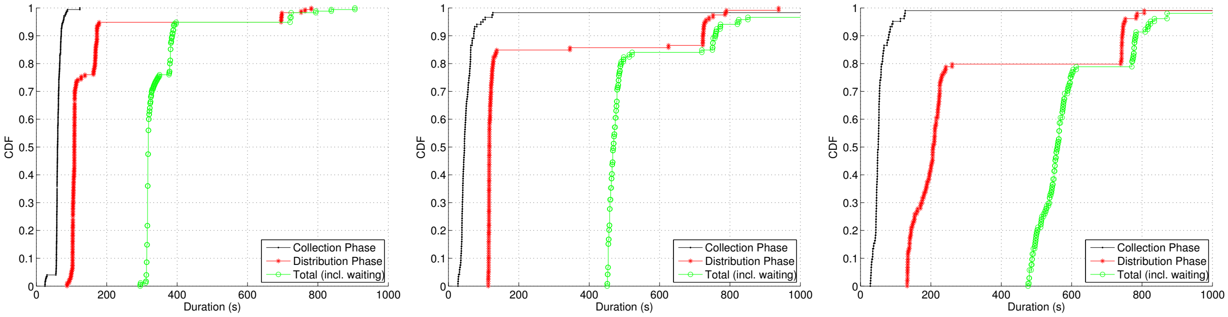

Fig. 5. CDF of the duration of the DP and CP periods for the base scheme with $N_{node} = 30$, 60 and 90 nodes.

contributions of all clouds have to be gathered, where each of them is in the form of (2), so that the sum S of all users' data is finally available to the server.

III. Experimental Results

Since secret sharing protocols are sensitive to node failures and data losses, it is important to validate the applicability of the proposed schemes in real-world scenarios. For this reason, we developed both architectures in Java and run over 5000 experiments on a PlanetLab testbed. The schemes have been implemented as described in Section II, activating two server sockets in each node, responsible of managing the distribution phase DP and the collection phase CP. Connections towards the peer nodes are established and closed run-time by each node, for disseminating the shares in the cloud, and for collecting the sum of the shares aggregated by each participating node.

Configuration. For each experiment, we build clouds using N_{node} PlanetLab nodes, which have been organized in geographically close sets (grouping them by country) to optimize the performance of the communication rings. For the base scheme, we consider different values of N_{node} (namely, $N_{node} = 30, 60, 90$), while for the enhanced scheme we choose $N_{node} = 90$ and vary the number of sets $z = 3, 5, 10$ (resulting in $|I_r| = 30, 18, 9$ nodes per set). For each scenario, we repeat the experiments at least 100 times, monitoring the duration of the share distribution phase and collection phase, as well as the failure rate of the nodes, that is, the probability that it is not possible to recover the data of the nodes belonging to the same cloud *during* the experiment. Indeed, we constantly monitor the state of the nodes in order to exclude nodes which are down at the beginning of the experiment, but once the experiment starts, failures cannot be recovered. It is also interesting to monitor the number of partial sums eventually recovered by the first node in the ring (which in our implementation is responsible for collecting all the sums of shares), from which an optimal tuning of k can be obtained for achieving a desired success probability.

Base Scheme. Figure 4 summarizes the results obtained for the base scheme, when k is equal to the number of nodes in the ring, i.e. the number of shares generated by each node is equal to N_{node}. The three distinct figures report the CDF (Cumulative Distribution Function) of the number of hosts participating in the experiments[1] (black line), the number of shares received by node 1 during the DP (red line) and the number of sums of shares collected during the CP (green line), for different N_{node} values. These figures reveal that the base scheme exhibits a good performance in those setups where $N_{node} = 30$ or 60: here the number of collected sums is typically coincident with the number of participating hosts minus 1 (the collecting node is not counted). However, as the number of nodes increases, the probability of collecting no data (because node 1 fails or the CP experiences a timeout) increases: it is equal to 0.2 when $N_{node} = 90$.

Figure 5 depicts the CDF of the duration for the DP and CP phase measured at node 1, which is also the node responsible for the ignition of the CP. It clearly emerges that large rings result in long running times (especially during the DP), which

[1] Note that some PlanetLab hosts were temporarily down or unreachable already before the beginning of the experiments so the number of hosts participating to the scheme was slightly below the corresponding N_{node} value.

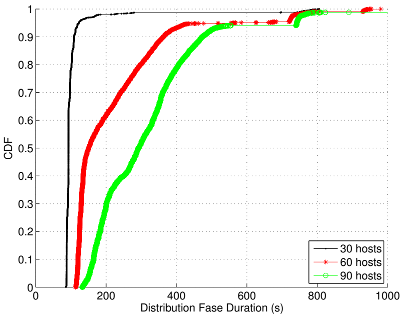

Fig. 6. CDF of the DP period duration for the base scheme with 30, 60 and 90 nodes.

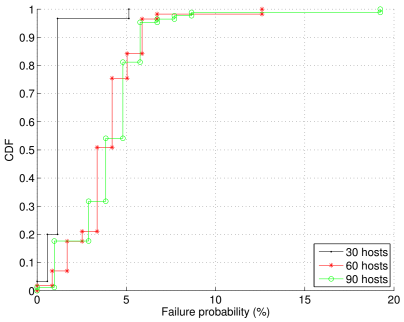

Fig. 7. CDF of the node failure probability for the base scheme with 30, 60 and 90 nodes.

in turn increases the probability of experiencing node failures during the scheme execution. Since the CP is meaningful only when all the nodes complete the dissemination of their shares, it is necessary to consider a waiting time before starting the CP (which we set to $300s$). Moreover, to avoid nodes being blocked into the distribution phase while some nodes of the rings are not available, we also set-up a timeout equal to $600s$ to interrupt the distribution towards the inactive nodes. The effects of this timer are clearly visible from the figure: as the ring size increases, an increasing percentage of nodes experiences a timeout during the DP (from about 5% when $N_{node} = 30$ to about 20% when $N_{node} = 90$), which is reflected into the total time required for completing the DP and CP (green curves).

Analyzing the root cause of these delays in greater detail, figure 6 shows the duration of the DP for each node across all experiments (note that the CP duration can be measured only by node 1 and it is shown in figure 5). While for the $N_{node} = 30$ case the delay is almost constant for all the nodes and lower than $100s$, as the ring size gets larger, a significant fraction of nodes has a DP with random distribution in the $[100s - 500s]$ range, while about 5% of the nodes terminate the DP because of the timeout expiration. As we will explain shortly, these high and variable running times are not due to failures but to overloaded PlanetLab hosts which react very slowly to communications. However, since the aggregated data can be recovered when k secret sums are available, by opportunistically limiting the k value to the fraction of nodes which complete the DP in a reasonable time, it is possible to limit the latency of the privacy-preserving aggregation process. For example, for $N_{node} = 60$, when $k = N_{node}/2$ a sufficient number of sums can be available within $200s$. In the basic scheme, the reason for such high delays is due to the fact the the DP involves *all* the nodes in the cloud and requires the exchange of $N_{node}(N_{node} - 1)$ messages. Still, depending on the use case, such running times are acceptable for the well functioning of the application, and guarantees privacy without relying on third parties.

Figure 7 shows the distribution of the failures seen across the participating nodes. The figure shows that the probability of the single nodes failing during the experiments is overall quite low, with almost all of the nodes failing less than 10% of the times. This relatively small failure probability still can have a non negligible impact when the basic scheme is employed, especially in scenarios where the running time is quite long, i.e., when the number of involved nodes is large. In such cases, the enhanced scheme provides much higher performances, as we will show next.

Enhanced Scheme. The enhanced scheme has been designed for mitigating the impact of slow hosts in the dissemination of the shares. Indeed, nodes are organized into sets and the total number of shares to be distributed is equal to the number of sets. Figure 8 summarizes the overall results achieved by this scheme with $N_{node} = 90$ and with an increasing number of sets ($z = 3$, 5 and 10). In this experiment, for each set, shares are sent to a randomly selected nodes. The figure shows the number of hosts active during the experiments, the number of shares received by each host (random with average z), the number of partial sums received by node 1 of each set (close to $|I_r|$) and the total number of shares collected in each set (ideally equal to the number of nodes). The figure shows that the total number of shares collected by node 1 of each set closely follows the number of participating nodes and that the best performance is achieved when $z = 3$. Indeed, when z is small the number of shares to be sent is small as well, with lower delays and higher probability of success, while with large values of z the enhanced scheme tends to the basic scheme and the total number of shares collected deviates from the desired result.

Finally, figure 9 shows that the enhanced scheme dramatically reduces the duration of both DP and CP, resulting in a much shorter time needed to complete the gathering of the data. Interestingly, the running time is almost insensitive with respect to the number of sets and overall over 80% of the experiments lasted less than 200 seconds, which results into a time reduction equal to 67% compared to the equivalent basic scheme.

IV. Conclusion

In this paper we have experimentally investigated the performance of two partly distributed multi-cloud, privacy-preserving schemes suitable for data mining algorithms based on linear operations. Two different approaches have been considered, a basic and an enhanced scheme: their behavior has been experimentally assessed on PlanetLab for validating

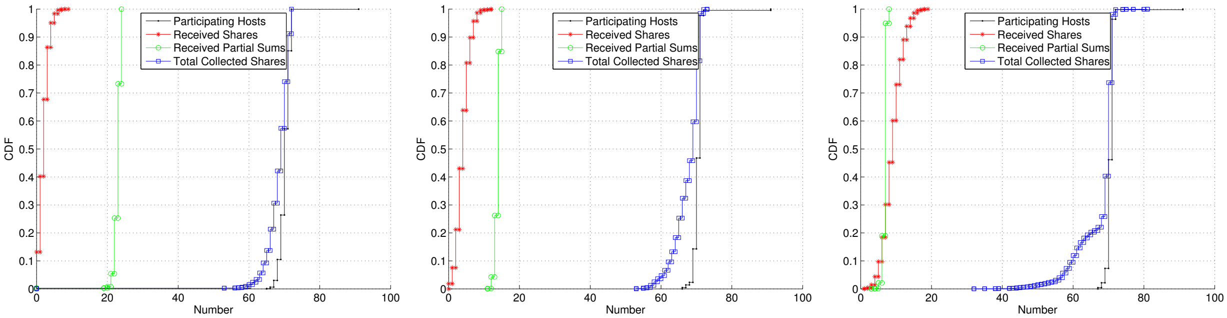

Fig. 8. Performance of the enhanced scheme with 90 nodes and $z = 3$, $z = 5$ and $z = 10$.

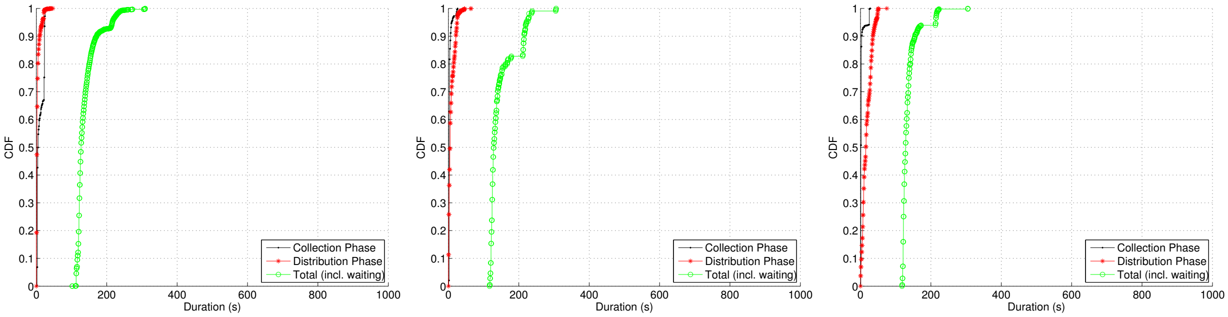

Fig. 9. CDF of the duration for the enhanced scheme with $N_{node} = 90$ nodes and $z = 3$, $z = 5$ and $z = 10$

the feasibility of the approach and for providing some insights useful for the tuning of the scheme operational parameters. Experimental results show that unexpected latencies may emerge, especially during the distribution of the shares, because of the time-varying load of PlanetLab nodes. To mitigate the effects of these latencies on the overall time required for data aggregation, it is possible to select a secret sharing scheme with a threshold value much lower than the total number of shares (e.g. $k = N_{node}/2$) for the basic scheme case or to opportunistically choose the number of sets for the enhanced scheme.

Although, overall, the enhanced scheme offers a reduced latency, it is important to observe that there are also some potential complications in its implementation. Indeed, for the basic scheme, each node can independently verify that it has collected the total number of expected shares (i.e. $N_{node} - 1$) and drop the sum of the received shares whenever this number is lower than expected. This means that each sum received by the server is a useful data and that the reception of k sums guarantees the reconstruction of the aggregated data. Conversely, for the enhanced scheme, a share of the aggregated data is not available at a single node, but only collecting the shares received by each node of a set. Therefore, it is necessary to consider a mechanism to track the total number of shares collected in each set.

REFERENCES

[1] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD Rec.*, vol. 33, no. 1, pp. 50–57, Mar. 2004.

[2] R. Agrawal and R. Srikant, "Privacy-preserving data mining," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00. New York, NY, USA: ACM, 2000, pp. 439–450. [Online]. Available: {http://doi.acm.org/10.1145/342009.335438}

[3] Z. Erkin, T. Veugen, T. Toft, and R. Lagendijk, "Privacy-preserving user clustering in a social network," in *Information Forensics and Security, 2009. WIFS 2009. First IEEE Int. Workshop on*, Dec 2009, pp. 96–100.

[4] A. C. C. Yao, "How to generate and exchange secrets," in *27th Foundations of Computer Science (FOCS)*, 1986, pp. 162–167.

[5] M. Ben-Or, S. Goldwasser, and A. Wigderson, "Completeness theorems for non-cryptographic fault-tolerant distributed computation," in *20th ACM symposium on Theory of computing (STOC)*, 1988, pp. 1–10.

[6] R. Cramer, I. Damgard, and U. Maurer, "Multiparty computations from any linear secret sharing scheme," in *Int. Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT)*, 2000.

[7] S. Bogdanov, D. Laur and J. Willemson, "Sharemind: A framework for fast privacy-preserving computations," in *13th European Symposium on Research in Computer Security (ESORICS)*, vol. 5283, Oct 2008, pp. 192–206.

[8] M. M. D. Burkhart, M. Strasser and X. Dimitropoulos, "Sepia: Privacy-preserving aggregation of multi-domain network events and statistics," in *19th USENIX Security Symposium (USENIX)*, 2010, pp. 223–240.

[9] Y. Duan, J. Canny, and J. Zhan, "P4p: practical large-scale privacy-preserving distributed computation robust against malicious users," in *USENIX Security Symposium*, Aug. 2010.

[10] A. Iacovazzi, A. D'Alconzo, F. Ricciato, and M. Burkhart. Elementary secure-multiparty computation for massive-scale collaborative network monitoring. *C*omput. Netw. vol. 57, no. 17 (Dec. 2013), pp. 3728–3742.

[11] M.L. Merani, C. Barcellona, I. Tinnirello,"Multi-Cloud Privacy Preserving Schemes for Linear Data Mining," in *I*EEE ICC 2015, June 2015.

[12] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu, "Tools for privacy preserving distributed data mining," in *SIGKDD Explorations*, 2002, pp. 28–24.

[13] A. Shamir, "How to share a secret," in *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[14] J. Dun, "A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters," in *Journal of Cybernetics*, pp. 32-57, 1973.