Practical Fully-Decentralized Secure Aggregation for Personal Data Management Systems

Julien Mirval, Luc Bouganim, Iulian Sandu Popa

▶ **To cite this version:**

Julien Mirval, Luc Bouganim, Iulian Sandu Popa. Practical Fully-Decentralized Secure Aggregation for Personal Data Management Systems. 33rd International Conference on Scientific and Statistical Database Management, SSDBM 2021, Jul 2021, Tampla, FL, United States. pp.259-264, 10.1145/3468791.3468821 . hal-03329878

HAL Id: hal-03329878
https://hal.science/hal-03329878v1

Submitted on 8 Oct 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Copyright

Practical Fully-Decentralized Secure Aggregation for Personal Data Management Systems

Julien Mirval
julien.mirval@cozycloud.cc
Cozy Cloud
Inria-Saclay
UVSQ, Université Paris-Saclay
France

Luc Bouganim
luc.bouganim@inria.fr
Inria-Saclay
UVSQ, Université Paris-Saclay
France

Iulian Sandu-Popa
iulian.sandu-popa@uvsq.fr
UVSQ, Université Paris-Saclay
Inria-Saclay
France

ABSTRACT

Personal Data Management Systems (PDMS) are flourishing, boosted by legal and technical means like smart disclosure, data portability and data altruism. A PDMS allows its owner to easily collect, store and manage data, directly generated by her devices, or resulting from her interactions with companies or administrations. PDMSs unlock innovative usages by crossing multiple data sources from one or many users, thus requiring aggregation primitives. Indeed, aggregation primitives are essential to compute statistics on user data, but are also a fundamental building block for machine learning algorithms. This paper proposes a protocol allowing for secure aggregation in a massively distributed PDMS environment, which adapts to selective participation and PDMSs characteristics, and is reliable with respect to failures, with no compromise on accuracy. Preliminary experiments show the effectiveness of our protocol which can adapt to several contexts with varying PDMSs characteristics in terms of communication speed or CPU resources and can adjust the aggregation strategy to the estimated selective participation.

CCS CONCEPTS

• **Computer systems organization** → **Architectures**; • **Information systems** → *Data management systems.*

KEYWORDS

Privacy, secure aggregation, decentralized, machine learning.

ACM Reference Format:
Julien Mirval, Luc Bouganim, and Iulian Sandu-Popa. 2021. Practical Fully-Decentralized Secure Aggregation for Personal Data Management Systems. In *33rd International Conference on Scientific and Statistical Database Management (SSDBM 2021), July 6–7, 2021, Tampa, FL, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3468791.3468821

1 INTRODUCTION

The new privacy-protection regulations (e.g., GDPR) and smart disclosure initiatives in the last decade have boosted the development and adoption of Personal Data Management Systems (PDMSs) [2]. A PDMS (e.g., Cozy Cloud, Nextcloud, Solid) is a data platform allowing users to easily collect, store and manage into a single place data directly generated by user devices (e.g., quantified-self data, smart home data, photos) and data resulting from user interactions (e.g., social interaction data, health, bank, telecom). Users can then leverage the power of their PDMS to benefit from their personal data for their own good and in the interest of the community [7].

Consequently, the PDMS paradigm leads to an important shift in the personal data ecosystem since data becomes massively distributed, at the user-side. It also holds the promise of unlocking innovative usages. An individual can now cross her data from different data silos, e.g., health records and physical activity data. Moreover, individuals can cross data within large communities of users, e.g., to compute statistics for epidemiological studies or to train a machine learning model (ML) for recommender systems or automatic classification of user data. However, these exciting perspectives should not eclipse the security issues –user data must be kept private– and the right for any PDMS user to consent, or not, in participating in each computation.

Aggregation primitives (e.g., sum or average) are obviously essential to compute basic statistics on user data but are also a fundamental building block for machine learning algorithms. Thus, to enable such new usages, we need scalable, privacy-preserving protocols implementing data aggregation primitives with selective (i.e., consenting) participants. Ideally, the proposed protocol should provide an accurate result that fully takes advantage of high-quality data available in PDMSs. Efficiency (i.e., protocol latency and total load of the system) is of prime importance and the protocol should adapt to several contexts: the PDMSs could be limited by their communication speed or by their computation power. Finally, given the scale of such decentralized aggregation, such protocols must also be robust to node failures. To summarize, our goal is to propose an aggregation protocol for basic aggregate functions that fulfills the following properties:

- *fully decentralized and highly scalable*, with the number of participants.
- *privacy-preserving*, i.e., it protects the confidentiality of user data.
- *accurate*, i.e., it does not require a trade-off between accuracy and privacy.
- *adaptable*, i.e, it can adapt to a large spectrum of computation selectivity values (reflecting the subset of contributor nodes) and system configurations (network and cryptographic latency).
- *reliable*, i.e., it handles node failures or voluntary disconnections.

Publication rights licensed to ACM. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of a national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
SSDBM 2021, July 6–7, 2021, Tampa, FL, USA
© 2021 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-8413-1/21/07…$15.00
https://doi.org/10.1145/3468791.3468821

The rest of this paper is organized as following. After discussing the related works w.r.t. the required properties in Section 2, we introduce the considered architecture and threat model in Section 3. Sections 4 and 5 focus on the proposed protocol and preliminary results. Section 6 concludes with future issues.

2 RELATED WORKS

Secure aggregation is an intense research area since many years and many approaches were proposed: secure multi-party computation (SMC) and (fully) homomorphic threshold encryption (HTE), (local) differential privacy and gossip-based protocols. However, the existing solutions are not adapted to the PDMS context and fail to cover all the required properties listed above.

HTE and SMC-based solutions [4, 6, 9, 10] generally target applications in which central servers orchestrate and coordinate the participating nodes (e.g. federated learning). Such solutions are not scalable with a large number of participants and a fully decentralized setting such as in the PDMS context (e.g., the server(s) load is linear [9] or quadratic [6] with the number of participants).

Local differential privacy (LDP) has gained significant momentum in the recent years addressing problems such as machine learning [14] or basic statistics based on range queries [8]. However, LDP requires more noise than classical DP [1], either affecting accuracy or requiring a large number of participants to reduce the impact of noise, contradicting adaptability to selective participation.

Gossip-based protocols are scalable, fully decentralized, reliable and have an adjustable accuracy. Unfortunately, classical gossip-based protocols do not protect the user privacy. In [5], participants collectively learn a machine learning model in a privacy preserving way by gossiping differentially private models, impacting accuracy. In [12], participants introduce noise in the first iterations and gradually remove it in subsequent iterations. This approach makes such solutions unreliable w.r.t. node failures. Finally, we are not aware of gossip protocols tolerating selective participation and trivial adaptations produce inaccurate results.

3 SYSTEM OVERVIEW AND THREAT MODEL

In this section, we introduce first the system architecture and the related concepts. We then present the considered threat model for the proposed secure protocol.

3.1 System Architecture

P2P network. We envision a fully distributed Peer-to-Peer (P2P) system relying only on PDMSs, thus requiring an efficient communication overlay. *Distributed Hash Tables* (DHT) are structured overlays which enable a logarithmic scalability with the number of nodes. Our protocol is currently built on top of the Chord DHT [13]. Each node has an *Id* obtained by hashing a static property of the node and stores a *fingertable* (FT) to route Chord messages. FT is a table with a number of entries equal to the size of the *Id* space in bits. If X is a node *Id*, the i^{th} entry of the FT contains the IP address of the node whose *Id* is closest but lower than $X + 2^i$. Routing is done by searching in the FT the closest entry to the target address and transmitting recursively the message until it reaches its target, with a worse case of $O(log(N))$ message complexity, where N is the number of DHT nodes.

Computation model. An aggregate computation can be triggered by any node, called *querier*. The querier broadcasts the computation and each node consents or not to contribute, and in the positive case is called *contributor*. The ratio between the number of contributors and total number of nodes defines the *selectivity* σ, $0 < \sigma \leq 1$. Finally, each node (contributor or not) is a potential data processor and is then called *aggregator*.

3.2 Threat Model

We consider the *honest-but-curious* threat model, in which, an attacker can access, without altering, the data manipulated by the attacked nodes, then called *leaking nodes*. The rationale behind the honest-but-curious model is that a PDMS can hold the entire digital life of her owner and therefore needs to be highly protected against privacy threats. Recent works [3] indicate that Trusted Execution Environments (TEEs) are prime candidates to offer this protection since they guarantee that executed code and manipulated data cannot be observed. In our context, this property allows sharing data between PDMS nodes without breaking data confidentiality.

We thus consider that each PDMS is protected by a secure hardware solution, such as Intel SGX or ARM TrustZone, providing a TEE. Such a hardware protection makes attacks difficult to produce, but since no security measure is unbreakable, we consider that some PDMS owners have succeeded in tampering their PDMS. Since attackers may collude and thus, de facto, control more than one PDMS, the worst-case attack is represented by the maximum number of colluding nodes controlled by a single "attacker", i.e., C leaking nodes.

Additionally, the TEE of each PDMS is equipped with a trustworthy certificate. Thus, any node can verify the authenticity of other participants by checking their certificate. This prevents Sybil attacks (i.e., forging nodes to master a large portion of the system).

Finally, attackers can also observe the communications between the nodes, thus requiring secure communication channels (e.g., TLS) to protect sensitive data exchanges.

Our objective is to provide a protocol that fully protects the confidentiality of the contributors' data and all the intermediary results, with high and tunable probability, the final result not being confidential. Also, we consider that being a contributor for a given computation is not a sensitive information.

4 PROPOSED PROTOCOL

In the protocol overview, we analyze the properties listed in Section 1 and present the main ideas and techniques behind each property and its impact on the protocol. Due to space constraints, we cannot describe in detail the proposed protocol and thus discuss some identified key elements of the protocol under the form of questions/answers, considering first the privacy aspects and then the efficiency perspective.

4.1 Protocol Overview

Scalability: The DHT achieves de facto a fully decentralized and efficient architecture for inter-nodes communication. Achieving a scalable aggregation process requires multiple aggregators, arranged in a tree structure. Building and broadcasting this aggregation tree can be very costly since the tree itself can be large. We

thus employ a divide-and-conquer approach to parallelize the tree construction and diffusion and use the finger table structure to minimize communications. Finally, we reduce the knowledge (and thus the diffusion) of the tree to the minimal part strictly necessary to perform the aggregation: basically, each node of the tree only knows its parent(s) and its children.

Privacy and accuracy: We use a secret sharing scheme (without threshold) in which each contributor splits its data into s shares, making them unreadable unless someone collects all s shares. s is computed such that the probability to obtain s shares for an attacker, controlling C nodes, is inferior to a security threshold α (e.g., $\alpha = 10^{-6}$). Each i^{th} share has the value $x_i = x + \epsilon_i$ such that $\sum_{i=1}^{n} \epsilon_i = 0$, where x is the private value. This way, shares from different contributors can be aggregated separately and if no share is missing (the reliability is discussed below), the final result will be equal to the exact sum of all private data. Hence, our protocol provides also, by construction, accurate results. Note that the protocol works for complex values of x, such as an array or a matrix, which is useful for advanced aggregations, e.g., training a naive Bayes ML model.

Adaptability: The number of aggregators (i.e., the tree fan-out and its height) is tuned as a function of the number of contributors, the communication costs (i.e., the latency to send a message between two nodes) and the processing costs (i.e., the asymmetric cryptographic costs to secure a communication or to sign or verify a signature, which is, by far, the most important processing costs). This allows the protocol to always offer near-optimal performance (i.e., aggregation latency) and achieve adaptability w.r.t. the computation selectivity and PDMSs characteristics. Furthermore, our protocol can also be conveniently configured to offer the desired trade-off between the latency and the total cost of the aggregation, which are conflicting optimization objectives as discussed in Section 5.

Reliability: Handling failures and disconnections is mainly implemented at two levels. First, the aggregators in the last level of the tree (just above the leaves) execute a synchronization protocol to make sure that contributors have sent all the s shares before disconnection and remove the shares for the contributors that have sent less than s shares. This ensures that the aggregation result stays accurate despite contributors failure. Second, a list of backup aggregators is created before the tree creation. Its size depends on the observed node failure/disconnection ratio. In case an aggregator fails, it is automatically replaced with a backup node during the aggregation process (the parents monitor their children). This allows the protocol to be robust to node failures and avoids losing aggregation subtree results.

4.2 Privacy Issues

What is the impact of the secret sharing on the aggregation tree? Considering s shares for each contributor and partial results leads to build s separate aggregation trees, with exactly the same structure, to avoid inferences from an attacker on any of the intermediate results. The final sum of the shares is done by the querier (tree root). A simple means to construct such trees is to consider that each node of the tree is a group of s nodes (see Figure 1 with $s = 2$). The protocol to build the tree is described in Section 4.3 considering that, at each step, s nodes are selected instead of one. To make this selection efficient, each node in the DHT maintains a cache with the addresses of the $s - 1$ successor nodes that will form the aggregator group.

How is the number of shares computed? An attacker could cleverly locate her controlled nodes in the DHT to obtain the s shares of a group (typically controlling a node and its $s - 1$ successors). We avoid this attack by reusing the concept of imposed location that we proposed in [11]: the node *Id* in the DHT is computed by hashing the public key from the PDMS certificate (see Section 3.2). The nodes are then uniformly distributed in the DHT space and the PDMS owner (here the attacker) cannot influence this placement: the uniform distribution also applies to leaking nodes. As a consequence, s can be easily computed and is minimal when $s = \lceil \log(\alpha)/\log(C/N) \rceil$.

Do contributors/aggregators have to check the correctness of the received query? Basically, the answer is yes. Indeed, a trivial attack would be to impersonate s aggregators (at the bottom of the tree) and ask a set of contributors for their shares, with the same protocol. If no control is done, the contributor cannot distinguish a real query from a fake query. To avoid such an attack, every aggregator must check the signature of the incoming query using the public key of the sender, having previously checked the validity of the sender's certificate. Since all the nodes are honest but curious, they must follow the protocol and thus cannot create a specific query that would lead to the disclosure of certain data.

4.3 Efficiency Issues

What is the divide-and-conquer approach to build the aggregation tree? Assuming the querier knows the height h and the fan-out f of the aggregation tree, it starts creating a tree assigning the whole DHT to its successor(s). Recursively, each aggregator in the tree (i.e., a parent node) is assigned to a DHT region that it will subdivide and delegate to other aggregators in that region. When an aggregator oversees a DHT region, it looks for f nodes that are (almost) evenly spaced across the region. The node responsible for finding peers is a parent aggregator, while the selected nodes are child aggregators. Each child then becomes the parent of the region between itself and the next sibling. This process goes on until the height h is reached. At the last tree level, the tree leaves (i.e., the contributors) are found by using a localized DHT broadcast in the respective region. Contributors willing to participate reply with their private data, after establishing a secure channel with their aggregator parent. The aggregators at level d aggregate the data they receive before sending them to the previous level of the tree down to the root (i.e., the querier) which performs the final aggregation to obtain the result. Figure 1 illustrates this process with two nodes per group (blue and red) by using letters to represent a group. The fan-out is 4 and the height is 3 (excluding the querier Q). Q selects his successor, A, who is responsible of the whole DHT and is the root of the tree. A uses its finger table to contact C, E and B. C recursively contacts D. The second level of the tree is built. Then B, E, C and D contact recursively the nodes for the second level (the figure only shows what happens with E for readability). When leaves are contacted, they send one share to each aggregators of the group (i.e., blue and red) which are summed-up separately in each aggregation tree and finally summed-up by Q.

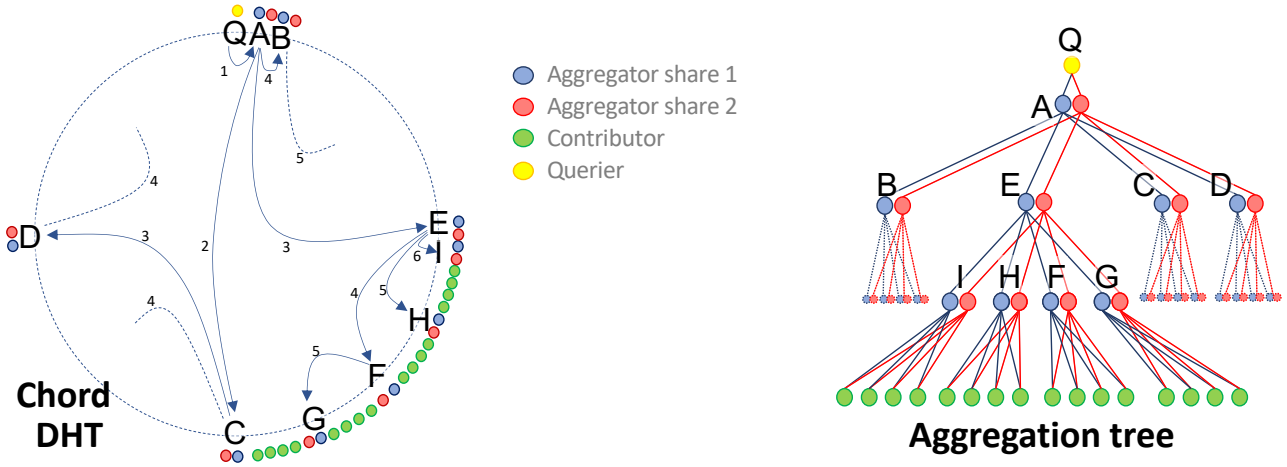

Figure 1: Building the aggregation tree based on DHT

How does an aggregator contribute? If a node selected as aggregator in the tree wishes to contribute, it can simply add its data to the partial aggregate it computes before sending it to its parent. Note that it will add it without splitting the data into shares since its parent cannot guess this addition. To compute an average, we need to count the number of contributors and thus, the aggregator will add s to the count of share contributions: each aggregator accounts the number of shares it received, and the total will be finally divided by s to obtain the number of contributors. Consequently, aggregators do not appear as leaves in the aggregation tree. Note that this is not the case for backup nodes which must have the possibility to appear as leaves of the tree in case they wish to contribute.

How are the tree fan-out f and height h computed? At one extreme, a binary tree ($f = 2$) distributes the query load on a maximum number of aggregator nodes but increases the communications costs, including the creation of many secure channels to transfer the intermediate results. At the other extreme, a tree limited to a unique aggregator ($f = \sigma \times N$) minimizes the communications and thus the number of secure channels (1 per contributor). It minimizes the total system load induced by the query but concentrates most of that load on this unique aggregator (that becomes overloaded by asymmetric crypto operations for the communication decryption). An "ideal" aggregation tree would be completely balanced, with the same fan-out all along the tree. Moreover, this fan-out (and thus the height of the tree) would be cleverly chosen to optimize the query latency without impacting too much the total load. Note that this depends on the PDMS characteristics, i.e., communication speed or computation power. Finally, the tree height is simply computed based on the number of contributors ($\sigma \times N$) and the tree fan-out. σ can be estimated, for instance by contacting all nodes within a region of the DHT, and checking the ratio of nodes willing to participate. Since nodes are uniformly distributed in the DHT thanks to the hash of their public key, choosing a sample of the population should give a good estimation of σ.

5 PRELIMINARY RESULTS

As in most evaluations of distributed systems [13], we implemented a simulator allowing varying any parameter: number of nodes N, of colluding nodes C, security threshold α, selectivity σ, and β, a ratio defined below. Our simulator captures two metrics: (i) for the network utilization, we consider the *number of exchanged messages* as the most important metric (compared to, e.g., the message size); (ii) for the PDMS resource utilization, the simulator counts the *asymmetric cryptographic operations* which are, by far, the most expensive operations. The output of the simulator is the protocol latency and total work. They depend on β, the relative cost of one asymmetric cryptographic operation denoted *crypt* and the latency when sending a message between two PDMSs denoted *com*. Specifically, $\beta = crypt/(crypt + com)$ with $0 \leq \beta \leq 1$. However, note that the two extremes values of β are not realistic, i.e., $\beta = 0$ when $crypt = 0$ or $com = +\infty$, $\beta = 1$ when $com = 0$ or $crypt = +\infty$.

Our protocol is adaptive to σ and β, thus called ***Adaptive*** in this section. To measure the impact of these two parameters on the aggregation costs, we compare the *Adaptive* protocol to two other simplified versions. First, ***Full tree*** is a classical aggregation tree that does not adapt to the query selectivity, i.e., it considers $\sigma = 1$: a tree is created recursively until all nodes are included, but only those willing to contribute will send back shares. Second, ***Single level*** considers that $\beta = 0$, i.e., the communication cost is so high that we must minimize it, thus concentrating all the computation on a single group, collecting the shares from all participants, and sending the results to the querier.

We consider a network with $N = 1,000,000$ nodes, a quite large attack level ($C = 10,000$) and a high security threshold ($\alpha = 10^{-6}$) and compare the above protocols in relative terms, i.e., dividing the latency/total work of *Full tree*/*Single Level* by the one of *Adaptive*.

We first confirmed that the adaptive protocol is scalable. With increasing values of N, we obtained a logarithmic increase of the latency, thanks to the DHT and the divide-and-conquer approach. We also verified that the number of colluding nodes C has a small impact on the protocol latency, with reasonable values of C w.r.t. N

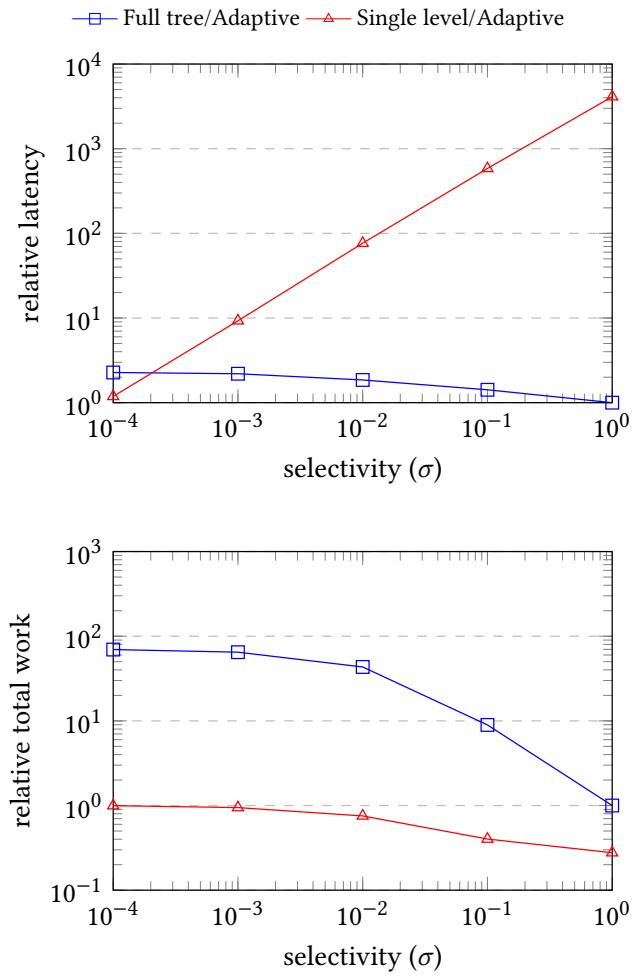

Figure 2: Latency and total work relatively to the *Adaptive strategy* varying σ

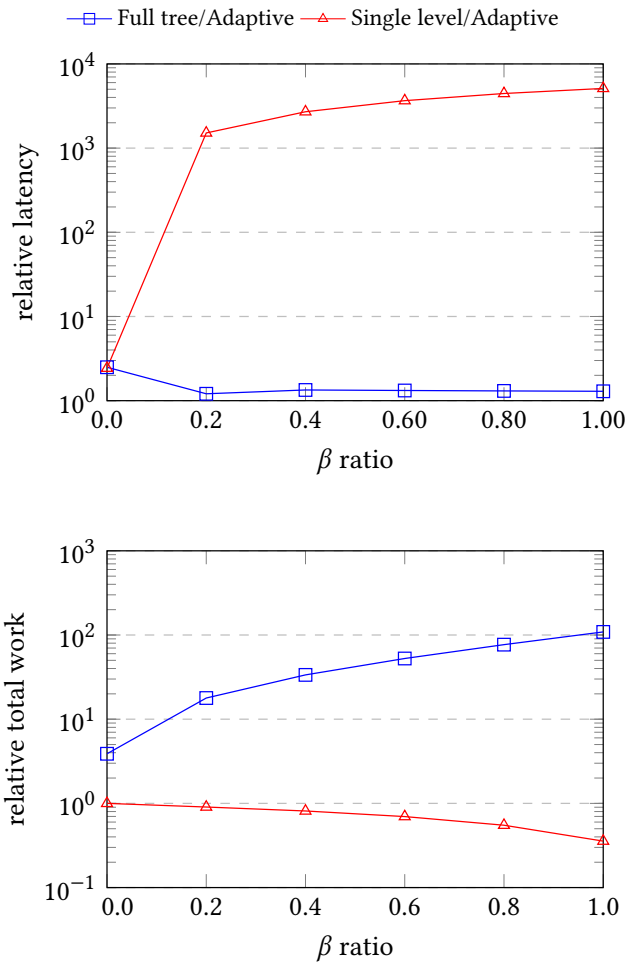

Figure 3: Latency and total work relatively to the *Adaptive strategy* varying β

in accordance with the considered threat model. Thus, in the rest of this section, we focus on the adaptability feature of our protocol and leave the evaluation of its reliability for future works. We vary the selectivity σ (keeping $\beta = 0.5$) and the PDMSs characteristics β (keeping $\sigma = 0.01$). The results are presented in Figures 2 and 3 (log scale on Y axis for all graphs and on X axis for selectivity only).

Let's first focus on the *Single Level* protocol studied to show the impact of an extreme strategy, i.e., concentrating all the load on a single (group of) node(s). As expected, *Single Level* always provides a better total work than *Adaptive* and *Full tree*. However, the latency increases linearly with the number of participants leading rapidly to prohibitive costs. Practically, *Single Level* is competitive only if the selectivity is extremely high (i.e., tens to a few hundreds of contributors) or $\beta = 0$ (i.e., unrealistic setting).

Execution based on aggregation trees (*Full tree* or *Adaptive*) are much scalable for handling many contributors by distributing the workload. Note that for a maximal selectivity, both approaches have exactly the same latency, as their structure is identical. However, *Full tree* becomes more costly for both latency and total work as soon as the selectivity is below 1. Indeed, the adaptive fan-out and tree depth of *Adaptive* can reduce the latency up to a factor of 3 and especially the total work up to two orders of magnitude, which indicates the importance of adapting the aggregation structure to the computation and system settings.

In the last part of our experimental evaluation, we study in more details the *Adaptive* protocol. In particular, we evaluate the impact of the tree fan-out on the latency and the total work of the protocol with different values of β while keeping $\sigma = 0.01$. The results are presented in Figure 4 (log scale on the X axis for both graphs). As above and to increase the readability, we represent relative values for both the latency and the total work, i.e., the ratio between the latency value (or the total work value) and its minimum observed value.

As expected, increasing the fan-out, decreases the total work, as the aggregation tree includes less nodes. This reduces the total amount of communications (and hence reduces the number of secure channels), but concentrates the cryptographic load on a

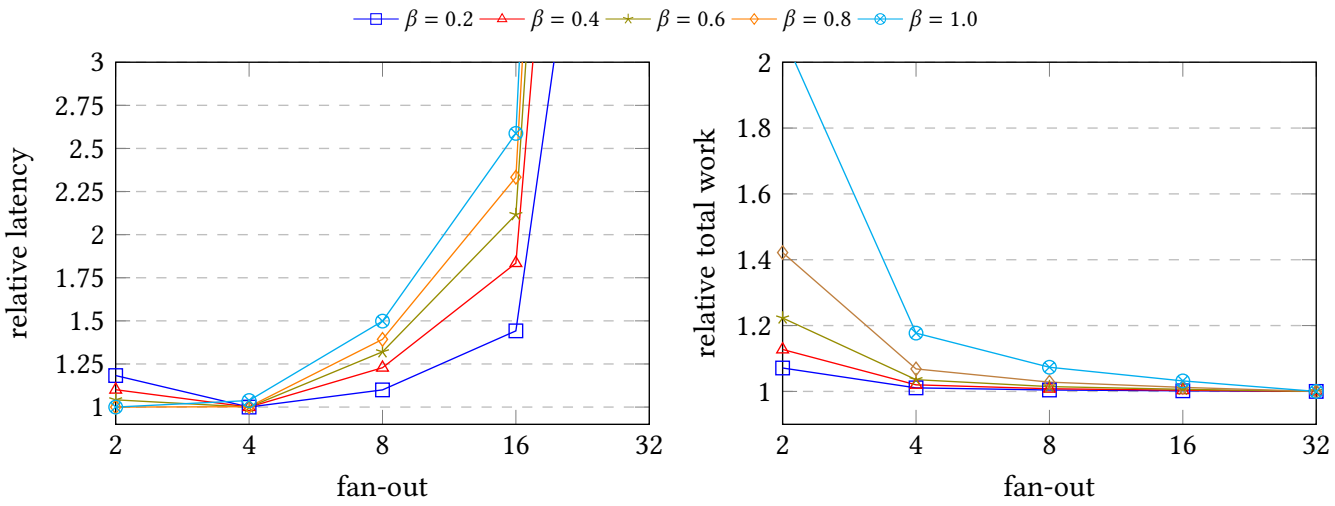

Figure 4: Relative latency and relative total work w.r.t. the minimum value varying the fan-out

few nodes, leading generally to a higher latency. However, we observed an exception to this behavior for small fan-out values. In this case, the communication overhead required to construct the tree leads to sub-optimal latency. With small values of β (i.e., the communication cost is larger than the cryptographic cost), this overhead is more prominent. On the contrary, once the fan-out increases, smaller values of β result in a decreased latency, as the cryptographic operations, which are dominant, are relatively cheaper.

Our results confirm that there is a sweet spot for the fan-out depending on the PDMSs and network characteristics. The results also indicate that, depending on the application requirements, the fan-out can be adjusted to obtain a better trade-off between latency and total work. For example, training a machine learning model on user's data may be less restricted in terms of latency than a real-time traffic analysis. For instance, when $\beta = 0.6$, choosing a fan-out of 8 leads to a total work only 3% higher than the optimal value, while the latency is 32% larger than the optimal value.

6 CONCLUSION AND FUTURE WORKS

In this short paper, we made the first steps towards the design of an aggregation protocol providing interesting properties: highly scalable, privacy preserving, adaptable to selective participation, to several system settings, with a tree-like structure enabling robustness to failure; all this without compromise on the result quality. This protocol could be a building block to compute statistics on large communities of PDMS users or even to train ML algorithms. There is still a long way to go before providing all the required properties with efficient and secure protocols. Our next steps are to focus on the reliability aspect, selectivity estimation, and performance enhancements in the case of ML algorithms manipulating large datasets and requiring many iterations on users' data. This is an exciting research agenda with innovative usages in perspective.

REFERENCES

[1] Mário S. Alvim, Konstantinos Chatzikokolakis, Catuscia Palamidessi, and Anna Pazii. 2018. Local Differential Privacy on Metric Spaces: Optimizing the Trade-Off with Utility. In *IEEE CSF*. 262–267. https://doi.org/10.1109/CSF.2018.00026
[2] Nicolas Anciaux, Philippe Bonnet, Luc Bouganim, Benjamin Nguyen, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. 2019. Personal data management systems: The security and functionality standpoint. *Information Systems* 80 (2019), 13–35.
[3] Nicolas Anciaux, Luc Bouganim, Philippe Pucheral, Iulian Sandu Popa, and Guillaume Scerri. 2019. Personal Database Security and Trusted Execution Environments: A Tutorial at the Crossroads. *Proc. VLDB Endow.* 12, 12 (2019), 1994–1997.
[4] Yoshinori Aono, Takuya Hayashi, Lihua Wang, Shiho Moriai, et al. 2017. Privacy-preserving deep learning via additively homomorphic encryption. *IEEE Transactions on Information Forensics and Security* 13, 5 (2017), 1333–1345.
[5] Aurélien Bellet, Rachid Guerraoui, Mahsa Taziki, and Marc Tommasi. 2018. Personalized and private peer-to-peer machine learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 473–481.
[6] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. 2017. Practical secure aggregation for privacy-preserving machine learning. In *ACM CCS*. 1175–1191.
[7] EU Commission. 25 October 2020. Proposal for a Regulation on European data governance (Data Governance Act), COM/2020/767. [eur-lex].
[8] Graham Cormode, Tejas Kulkarni, and Divesh Srivastava. 2019. Answering Range Queries Under Local Differential Privacy. *PVLDB* 12, 10 (2019). https://doi.org/10.14778/3339490.3339496
[9] Henry Corrigan-Gibbs and Dan Boneh. 2017. Prio: Private, robust, and scalable computation of aggregate statistics. In *NSDI*. 259–282.
[10] David Froelicher, Juan Ramón Troncoso-Pastoriza, Joao Sa Sousa, and Jean-Pierre Hubaux. 2020. Drynx: Decentralized, secure, verifiable system for statistical queries and machine learning on distributed datasets. *IEEE Transactions on Information Forensics and Security* 15 (2020), 3035–3050.
[11] Julien Loudet, Iulian Sandu Popa, and Luc Bouganim. 2019. SEP2P: Secure and Efficient P2P Personal Data Processing. In *EDBT*.
[12] Yilin Mo and Richard M Murray. 2016. Privacy preserving average consensus. *IEEE Trans. Automat. Control* 62, 2 (2016), 753–765.
[13] Ion Stoica, Robert Morris, David Karger, M Frans Kaashoek, and Hari Balakrishnan. 2001. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM* 31, 4 (2001), 149–160.
[14] Kai Zheng, Wenlong Mou, and Liwei Wang. 2017. Collect at Once, Use Effectively: Making Non-interactive Locally Private Learning Possible. In *ICML*, Vol. 70.