DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems

Julien Loudet [1,2,3]
[1] Cozy Cloud, France
julien@cozycloud.cc

Iulian Sandu-Popa [3,2]
[2] INRIA Saclay, France
<fname.lname>@inria.fr

Luc Bouganim [2,3]
[3] University of Versailles, France
<fname.lname>@uvsq.fr

ABSTRACT

Personal Data Management Systems (PDMS) advance at a rapid pace allowing us to integrate all our personal data in a single place and use it for our benefit and for the benefit of the community. This leads to a significant paradigm shift since personal data become massively distributed and opens an important question: how to query this massively distributed data in an efficient, pertinent and privacy preserving way? This demonstration proposes a fully-distributed PDMS called DISPERS, built on top of SEP2P, allowing users to securely and efficiently share and query their personal data. The demonstration platform graphically illustrates the query execution in details, showing that DISPERS leads to maximal system security with low and scalable overhead. Attendees are welcome to challenge the security provided by DISPERS using the proposed hacking tools.

PVLDB Reference Format:
Julien Loudet, Iulian Sandu-Popa, and Luc Bouganim. DISPERS: Securing Highly Distributed Queries on Personal Data Management Systems. *PVLDB*, 12(12): 1886-1889, 2019.
DOI: https://doi.org/10.14778/3352063.3352091

1. INTRODUCTION

Thanks to smart disclosure initiatives[5] and new regulations[8], we can access our personal data from the companies or government agencies that collected them. Concurrently, Personal Data Management System (PDMS) solutions arise both in academia[1] and industry[6]. The goal is to offer a data platform allowing users to easily store into a single place any personal data: (i) directly generated by user devices (e.g., quantified-self data) and (ii) user interactions (e.g., health or banking data). Users can then leverage their PDMS to use the data for their own good and in the benefit of the community. Thus, the PDMS paradigm holds the promise of unlocking new innovative usages developed around personal data. A prominent example of novel usages is related to the computations between a large number of PDMSs (e.g., participative studies).

Proceedings of the VLDB Endowment, Vol. 12, No. 12
ISSN 2150-8097.
DOI: https://doi.org/10.14778/3352063.3352091

Yet, these exciting perspectives should not eclipse the security issues raised by this novel paradigm: storing an entire digital life proportionally increases the impact of a leak. Hence, it is risky to centralize all user data on powerful servers as these servers become highly desirable targets [9]. Besides, centralization makes little sense in the PDMS context where data is naturally distributed at the users' side.

Fortunately, Trusted Execution Environments (TEE)[2, 12] are also rising and their use in the PDMS context leads to trustworthy computational data platforms. However, since no security measure (software or hardware) can be considered as unbreakable, we cannot exclude having some colluding corrupted nodes in the system and, worse, even undistinguishable (covert adversaries[3]). In this work, we assume that a PDMS is secured thanks to a TEE, but might act as a covert adversary; it offers high connectivity and availability and can establish peer-to-peer (P2P) connections with other PDMSs. Hence, we envision a fully distributed architecture of PDMSs in which participants can create large communities, contribute with their personal data and issue queries over the globally contributed data. In this context, an important issue needs to be addressed: how to query this massively distributed data in a pertinent, efficient and privacy-preserving way?

DISPERS leverages three design principles to query the users' data securely: *knowledge dispersion* to protect data-at-rest, *task compartmentalization* to protect data-in-use, and *imposed randomness* to assign data processing tasks to PDMS nodes in a verifiable random way, based on SEP2P[11]. This previous work focuses on the imposed randomness principle proposing a generic solution, analyzing its security and showing its efficiency and scalability even with a large number of colluding nodes.

To the best of our knowledge, DISPERS is the first protocol dealing with pertinent, efficient, secure and fully distributed query processing over personal data. DISPERS goes beyond classical solutions (see [11]) that have been proposed in areas such as secure DHT[17] (which mainly focus on routing attacks), secure multi-party computation[13] (which lack genericity and scalability) or distributed data aggregation using secure hardware[16] (which do not consider a large number of corrupted nodes). Note also that consensus protocols and Byzantine fault tolerant systems address a different problem and generally improve availability and integrity using replication which hurts data confidentiality[18].

Sections 2 and 3 give an overview of the of the data, query and threat models, and of the proposed solution respectively

(see [11] for additional details). Section 4 present the demonstration platform and scenario, also explained in a video [10].

2. QUERY AND THREAT MODELS

User Profile. To achieve pertinence, each query only targets the subset of PDMSs exposing a given *user profile*: a structured description indicating the user's attributes (e.g. location, age, interests). Besides pertinence, a second benefit of user profiles is to increase efficiency by not flooding the entire network with each query. A *profile* p_i extracted by PDMS_i is a set of concepts $p_i = \{co_i^1, co_i^2, \ldots, co_i^k\}$. Each concept *co* is the concatenation of one or more metadata terms describing its semantics and a value, e.g., *location — Versailles*, *age — 37*, *occupation — researcher*.

Query Model. We consider classical aggregate queries (e.g., top-k) over the data supplied by the nodes targeted by the query. Such queries allow users to compute generic statistics (e.g., recommendations). In DISPERS, a query is a triplet: (i) *target profile* – a logical expression of concepts indicating which nodes, called targets, qualify to answer; (ii) *local query* – the expression of the query to be computed locally by each target; and (iii) *aggregate query* – an aggregative expression applied over the local query results.

Let us consider three examples of DISPERS query types: (i) *closed item list*: given a set of movies find their average grade as given by researchers living in Lisbon; (ii) *open item list*: get the top-10 ranked movies by researchers living in New York; (iii) *statistics*: what is the average number of sick leave days in 2017 of researchers living in Paris.

While the data and query model definitions and expressivity are significant issues, the system security is paramount for gaining users' trust and encouraging them to contribute with their data, justifying our focus on privacy preservation.

Threat Model. We make the following assumptions:
(1) *Each PDMS device is supplied with a trustworthy certificate attesting that it is genuine.* Otherwise, an attacker can emulate nodes (Sybil attack) and master a large proportion of nodes, thus defeating any countermeasure. With the TEE protection, we already offer a certain level of security at the node and system levels. Yet, no hardware security can be described as unbreakable.
(2) *Some nodes may be corrupted by a lab attack.* A lab attack is the most advanced, comprehensive and invasive hardware attack where the attacker has access to laboratory equipment and performs reverse engineering of a device.
(3) *Corrupted nodes are covert adversaries.* They only derive from the protocol if they cannot be detected[3], as detected malicious behaviors lead to exclusion.

The main objective of DISPERS is to offer protection beyond the TEE security. Therefore, our threat model considers colluding attackers, conducting lab attacks on their PDMS, thus mastering a set of corrupted nodes, called *colluding nodes*. We assume that the maximum number of colluding nodes mastered by a single or colluding attackers can be (over)estimated and will be used to calibrate DISPERS.

3. PROPOSED SOLUTION

Relying on a fully-distributed system requires a communication overlay allowing for efficient node discovery, data indexing and search. A *distributed hash table* (DHT)[15] offers an optimized means to locate the node(s) storing a specific data item. It provides an interface allowing any node to store or search for an item. Hence, DISPERS leverages the classical DHT techniques as a basis for communication efficiency and scalability.

To query our system in a secure and efficient manner, we build a distributed protocol on top of this P2P overlay relying exclusively on PDMS nodes. This implies some unavoidable data disclosure risk whenever a colluding node (covert adversary) is selected as a query actor. Therefore, to maximize the system security, we need to minimize the benefit of corrupting a node. This translates into two requirements:
R1: Minimize the private information any node could have access to whenever it is assigned with a data related task.
R2: Ensure that an attacker controlling several colluding nodes cannot influence the selection of the processing nodes.

Our query protocol relies on three design principles described below, which, combined, answer both requirements.

Knowledge dispersion. *No single (or few) node(s) should store a significant amount of sensitive data, unless it owns that data.*

To efficiently evaluate a query in DISPERS, we first need to obtain the list of node addresses that match the target profile. This requires the maintenance of a profile index that associates profile concepts to node addresses. The *knowledge dispersion* design principle aims at protecting the data-at-rest which is thus composed of: (i) the profile index; and (ii) the personal data of each PDMS owner (already secured by the TEE). We distributively store the index in the DHT: each node is responsible for a set of concepts and indexes all the node addresses matching one of them. A node that stores a concept index is called a *Concept Indexer* (CI). Even though the DHT uniformly distributes the knowledge among the nodes, if one were to be corrupted, it could access the entire list of IP addresses it indexes. We reinforce the knowledge dispersion by splitting each IP address into *s* shares using *Shamir's Secret Sharing scheme*[14]. Then, at least p ($p \leq s$) shares are required to reconstruct the secret. Disclosing a single concept index (i.e., the list of IP addresses sharing this concept) now requires p colluding nodes which are randomly selected (using the DHT). Based on this distributed profile indexing, a naive query protocol can be easily designed:

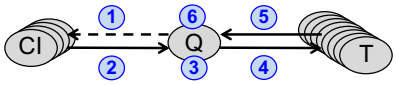

Figure 1: Naive protocol

Naive protocol (see Figure 1):

1. The querier (Q) searches the *CIs* using the DHT;
2. The *CIs* reply with the shares of IP address lists;
3. Q selects the final Targets (*Ts*) using the target profile;
4. Q sends the local query to the *Ts*;
5. The *Ts* answer with their local results;
6. Q computes the aggregate query result.

However, this protocol exposes two major shortcomings both resulting from the querier's central position. First, a corrupted querier may have access to the list of targets, their local query results and the association between targets and local results, i.e., all the sensitive data! Second, this protocol is not efficient as the querier acts as a bottleneck. Thus, thanks to the first design principle, the *data-at-rest* is protected, but the *data-in-use* – used to compute the query result – is still open to attacks.

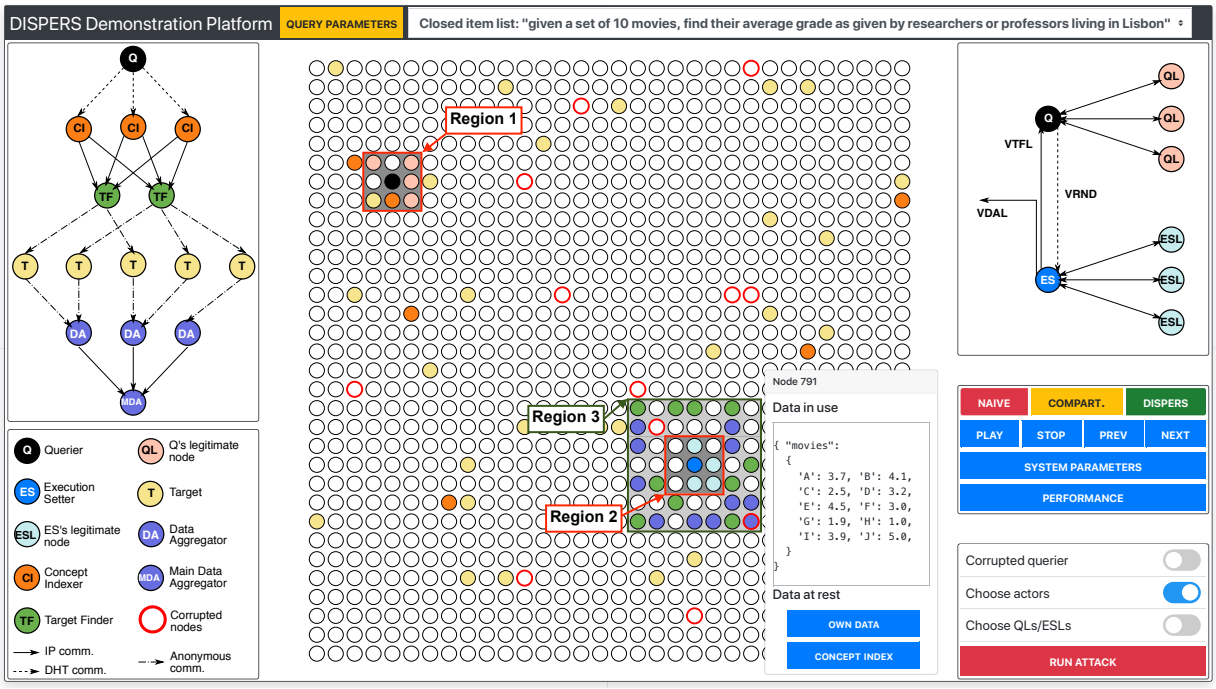

Figure 2: Demonstration platform screenshot

Task compartmentalization. *The execution must be split into atomic tasks assigned to distinct actors who must have access to the minimum information to perform their task.*

Analyzing the naive protocol leads to identifying three different roles cumulated by the querier: (1) contacting the *CIs* (step 1), (2) finding the targets (steps 3 and 4), and (3) computing the aggregate query (step 6).

Applying task compartmentalization leads to new actors definition: the *Target Finders* (*TF*) determining the relevant nodes based on the target profile, and the *Data Aggregators* (*DA*) aggregating the individual results to obtain the query result. To further reduce the potential data disclosure (and avoid bottlenecks), we consider having several *TFs* and *DAs*. Considering several *DAs* requires defining the *Main Data Aggregator* (*MDA*) which performs the final aggregation of the partially aggregated results. We can propose a second, more robust, protocol:

Compartmentalized protocol (see Figure 2, top-left corner)

1. *Q* selects the actors and searches (DHT) for the *CIs*;
2. The *CIs* distribute their shares of IP addresses to the *TFs*;
3. The *TFs* apply the target profile and contact the targets (*Ts*) using anonymous communications (e.g., TOR like[7])[1];
4. The *Ts* send their local results to a randomly chosen *DA*, still using anonymous communications;
5. The *DAs* compute and send partial aggregates to the *MDA*;
6. The *MDA* computes the final result.

Distributing the query processing on distinct actors increases parallelism and thus efficiency. More importantly, it offers a maximum degree of task compartmentalization:, i.e., inter-task, involving distinct dedicated actors for both target computation and data aggregation tasks; and intra-task, since several actors are chosen for each query task.

[1]The goal of the anonymous communications (steps 2 and 3) is to prevent an attacker spying the communications of the *TFs* or *DAs* from learning the targets' IP addresses w.r.t. a given query.

Moreover, the metadata is also compartmentalized as only the required information is given to the query actors and, when possible, even pseudonymized. Nevertheless, an attacker controlling several nodes may influence the choice of query actors to control some *TFs* and *DAs* and obtain confidential data. We thus need to enforce a random selection.

Imposed randomness. *Actors must be randomly selected, and that selection cannot be influenced by an attacker*[11].

This principle is applied by (1) imposing the location of nodes in the DHT (based on their public key); (2) selecting k nodes in a small region "around" the querier (Region 1 in Figure 2); k and the region size are computed based on the maximum number of colluding nodes in the system such that the region "never" includes k or more colluding nodes (probabilistic guarantees), i.e., at least one is honest – these nodes are called *legitimate nodes*; (3) using these nodes to generate a verifiable random number in a distributed fashion following an algorithm based on CSAR[4]; (4) using this random number to designate a location in the DHT overlay, around which the actors will be selected randomly, by k *legitimate actors list builders* located in Region 2 of Figure 2. Actors must be located in a region (Region 3 in Figure 2) which size is computed based on the number of needed actors, still using probabilistic guarantees.

Thus, our protocol answers the requirement R1 by applying the *knowledge dispersion* and *task compartmentalization* principles to protect, respectively, the data-at-rest and the data-in-use during the query evaluation process. Requirement R2 is addressed by applying the *imposed randomness* principle in the selection of the nodes processing a query.

This guarantees that an attacker *cannot obtain more private information than she can get by observing the data randomly reaching her corrupted nodes.* Thus, the impact of an attack remains *proportional to the number of colluding nodes*, which is the best situation in our context. While, in theory, our protocol is agnostic to the proportion of colluding nodes, a large proportion of colluding nodes mastered

by a single individual would inexorably lead to large disclosure. Given the covert adversaries assumption, *this is true whatever the protocol* with a reasonable overhead, but unrealistic as TEEs provide strong defenses against attackers, preventing large-scale corruption.

4. DEMONSTRATION

The purpose of this demonstration is to illustrate the DISPERS system thanks to a simulator and a graphical front-end; and to demonstrate the rationale of our three design principles. Attendees may optionally try to hack the system, their goal being to disclose some confidential data.

4.1 Demonstration Platform and Scenario

We introduce our approach using the graphical interface as depicted in Figure 2. Attendees can select or configure a query (top) and use the command panel (middle-right) to execute one of the query protocols, change the system parameters (e.g., number of colluding nodes), or run the protocol step by step. The last button allows exhibiting figures on the security and scalability of the DISPERS protocol.

After explaining the demonstration platform, we focus on a given query and run protocols of increasing complexity and resistance to attacks, thus explaining the rationale of each design principle. We can consider different queries such as: "*given a set of movies find their average grade as given by researchers or professors living in Lisbon*", which can be decomposed in: (1) a target profile: (city = Lisbon) $\wedge$ ((job = researcher) $\vee$ (job = professor)); (2) a local query: select grades for movies in {item list}; and (3) an aggregate query computing the grades average. Then, we present the query execution with the three frames detailing the DISPERS protocol, which is executed in two steps - the query setup phase and the query processing phase. The video in [10] explains the demonstration platform and scenario in detail.

4.2 Hacking Game

The goal of this game is to achieve a deeper understanding of DISPERS by trying to defeat its security, e.g., exhibiting some confidential data of a given node (let's call it Bob). Players will use a laptop, equipped with the same environment and will be assisted by one of the demonstration authors. We expect them to defeat easily the naive protocol. Playing with the queries, the parameters and inspecting the content of colluding nodes, the attendee may obtain Bob's data with the compartmentalized protocol but, based on [11], we are confident that this will be unfeasible with DISPERS. Attendees proposing some interesting means to try defeating DISPERS will win some goodies.

4.3 Lessons Learned

Privacy protection. The main expected outcome is to convince the audience that building a secure fully distributed PDMS system is within reach. DISPERS protects the private data of its participants even in the presence of colluding nodes, pushing the security level offered by TEEs much further thanks to (1) *imposed randomness* ensured by the SEP2P protocol[11] – guaranteeing that an attacker cannot influence the choice of processing nodes and get more private information than by passively observing the data manipulated by corrupted nodes; and (2) *knowledge dispersion* and *task compartmentalization* which reduce the data leakage to a minimum when covert adversaries are (randomly) involved in the query processing.

Efficiency and scalability. The cost of the security mechanisms remains very low even with a large number of colluding nodes. To this end, our platform provides the detailed query execution cost (i.e., both cryptographic and communication latency) with different system parameter settings.

5. REFERENCES

[1] S. Abiteboul, B. André, and D. Kaplan. Managing your digital life. *Comm. of the ACM*, 58(5), 2015.

[2] N. Anciaux, P. Bonnet, L. Bouganim, B. Nguyen, P. Pucheral, I. S. Popa, and G. Scerri. Personal data management systems: The security and functionality standpoint. *Information Systems*, 80, 2019.

[3] Y. Aumann and Y. Lindell. Security against covert adversaries: Efficient protocols for realistic adversaries. In *Theory of Cryptography*, 2007.

[4] M. Backes, P. Druschel, A. Haeberlen, and D. Unruh. Csar: A practical and provable technique to make randomized systems accountable. In *Network and Distributed System Security Symp.*, volume 9, 2009.

[5] Blue Button. Find your health data `https://www.healthit.gov/topic/health-it-initiatives/blue-button`, 2017.

[6] Cozy Cloud. Your digital home. `https://cozy.io/en`, 2018.

[7] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, 2004.

[8] European Parliament. General data protection regulation. Law, 2016.

[9] T. Hunt. ';–have i been pwned? largest and recent breaches. `https://haveibeenpwned.com/`, 2018.

[10] J. Loudet, I. Sandu-Popa, and L. Bouganim. DISPERS demonstration video : `http://petrus.inria.fr/~bouganim/DISPERS.mp4`.

[11] J. Loudet, I. Sandu-Popa, and L. Bouganim. SEP2P: Secure and efficient P2P personal data processing. In *EDBT*, 2019.

[12] C. Priebe, K. Vaswani, and M. Costa. Enclavedb: A secure database using SGX. In *IEEE Symposium on Security and Privacy*, 2018.

[13] E. Saleh, A. Alsa'deh, A. Kayed, and C. Meinel. Processing over encrypted data: between theory and practice. *ACM SIGMOD Record*, 45(3):5–16, 2016.

[14] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11), 1979.

[15] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *ACM SIGCOMM Computer Comm. Review*, 31(4), 2001.

[16] Q.-C. To, B. Nguyen, and P. Pucheral. Private and scalable execution of sql aggregates on a secure decentralized architecture. *ACM Transactions on Database Systems (TODS)*, 41(3):16, 2016.

[17] Q. Wang and N. Borisov. Octopus: A secure and anonymous dht lookup. In *Distributed Computing Systems (ICDCS)*, 2012.

[18] J. Yin, J.-P. Martin, A. Venkataramani, L. Alvisi, and M. Dahlin. Separating agreement from execution for byzantine fault tolerant services. In *ACM Symposium on Operating Systems Principles (SOSP)*, 2003.