Privacy-Preserving Queries on Highly Distributed Personal Data Management Systems

Julien Loudet, Luc Bouganim, Iulian Sandu Popa

Privacy-Preserving Queries on Highly Distributed Personal Data Management Systems

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

Luc Bouganim[2,3]
[2]INRIA Saclay
France
<fname.lname>@inria.fr

Iulian Sandu-Popa[2,3]
[3]University of Versailles
France
<fname.lname>@uvsq.fr

1 INTRODUCTION

The time of individualized management and control over one's personal data is upon us. Thanks to smart disclosure initiatives [4] 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 are flourishing in academia [1] and industry [3]. Their goal is to offer a data platform allowing users to easily store into a single place any personal data: data directly generated by user devices (e.g., quantified-self data, smart home data, photos, etc.) and user interaction data (e.g., user preferences, social interaction data, health, bank, telecom, etc.). Users can then leverage the power of their PDMS to use their personal data for their own good and in the benefit of the community. Thus, the PDMS paradigm holds the promise of unlocking new innovative usages while preserving the current ones developed around personal data. A prominent example of novel usages is related to the computations between a large number of PDMSs (e.g., recommendations, participative studies, collective decisions).

However, these exciting perspectives should not eclipse the security issues raised by the PDMS paradigm. Indeed, each PDMS can store potentially the entire digital life of its owner, thereby proportionally increasing the impact of a leakage. Hence, centralizing all users' data into few powerful servers is risky since the data servers become genuine honeypots: huge amounts of personal data belonging to millions of individuals could be leaked or lost as illustrated by recent massive attacks [5]. Besides, such a centralized solution makes little sense in the PDMS context in which data is naturally distributed at the users' side.

Fortunately, Trusted Execution Environments (TEE) [6, 7] are also rising and employing them in PDMS context leads to trustworthy computational data platforms. Hence, in this work, we assume that all PDMSs are secured thanks to a TEE. A PDMS can be considered to offer high connectivity and availability and can establish peer-to-peer (P2P) connections with other PDMSs. As such, we envision a fully distributed architecture of PDMS devices 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?

2 DESCRIPTION OF THE SOLUTION

To achieve a high degree of pertinence, in our system 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 query processing efficiency by avoiding flooding the entire network with each query. The queries can be, for instance, aggregate queries allowing users to compute generic statistics (e.g., recommendations of films). While the data and query model definitions and expressivity are significant issues, the global system security is paramount for a large system adoption, i.e., gaining users' trust and encouraging them to contribute with their data. We thus focus on privacy preservation during query execution in the abovementioned system.

As previously stated, TEEs provide a high level of data confidentiality against malicious PDMS owners. However, since no security measure can be considered as unbreakable, we cannot exclude having some corrupted nodes in the system and, even worse, those corrupted nodes might very well be undistinguishable from honest nodes. As we consider a fully distributed system, the query processing relies exclusively on PDMS nodes. This implies some data disclosure risk whenever a corrupted node 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 corrupted nodes cannot influence the selection of the nodes processing a query.

To efficiently index and retrieve node profiles, we leverage the classical Distributed Hash Table (DHT) P2P overlay [9, 11]: each node is responsible for a set of use profile attributes and indexes all the node IP addresses that match one of them. To query our system in a secure and efficient manner, we build a distributed protocol on top of this P2P overlay.

Our protocol relies on three design principles, namely *knowledge dispersion*, *task compartmentalization* and *imposed randomness*, which, once combined, answer both requirements.

Knowledge dispersion. This principle aims at protecting the *data-at-rest* (i.e. the distributed index used to retrieve the nodes corresponding to a given user profile) and can be summed up by: no single node (or few nodes) should store a significant amount of sensitive data, unless it owns that data.

In practice, this design principle is realized through the use of *Shamir's Secret Sharing technique* [10]. Even though the DHT uniformly distributes the knowledge among the nodes, if one node were to be corrupted it could access the entire list of IP addresses corresponding to the set of user attributes it indexes. By applying this technique the entries in the distributed index become "shards" that, taken alone, cannot be decrypted, thus protecting the data.

Task compartmentalization. The second principle concerns the query execution. Thanks to the first design principle the *data-at-rest* is protected, but the *data-in-use* — the data that is used to compute the final result — is still open to attacks. To protect it we perform task compartmentalization: we split the query execution in elementary tasks assigned to distinct nodes, so that each actor has only access to the minimal information it needs to know to perform its task.

In practice, this principle is enforced through the use of different actors having different roles. Also, whenever possible, several nodes are assigned to the same role to further split the access to sensitive data. The roles are: the *Concept Indexors* indexing the attributes, the *Target Finders* determining the relevant nodes based on a *target profile*, and the *Data Aggregators* aggregating the individual results to obtain the answer to the query.

Imposed randomness. The third and last design principle stipulates how the query actors should be selected. Indeed, if an attacker controls enough nodes in the network and manages to execute a query where all the actors are corrupted, then all our previous precautions would be nullified. This reason is why we force *imposed randomness*: the actors for each query must be randomly selected and the selection cannot be influenced by an attacker controlling several nodes.

In practice, this principle is applied by generating verifiable random numbers in a distributed fashion using an algorithm based on CSAR [2]. Those random numbers will designate a location on the DHT overlay, around which the actors will be selected (also randomly, using such number).

Thus, our protocol answers the requirement R1 by applying *knowledge dispersion* and *task compartmentalization* design 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* design principle in the selection of the nodes processing a query.

This guarantees that an attacker cannot obtain more private information than what she can passively get from observing the information randomly reaching its corrupted nodes. Thus, the impact of a collusion attack remains proportional with the number of corrupted nodes, which is the best situation given our context.

To assess the security offered by our protocol we have implemented a simulation in which we measured the impact of collusion attacks at different scales: where an attacker controls 1, 50, 0.01%, 1% nodes in a network that contains 10k, 100k, 1M and 10M nodes. For each level of attack we repeatidly simulated the execution of a query and measured the latency and total effort in terms of cryptographic operations as well as the disclosure rate.

We realized the same measurements with two relaxed versions of our protocol: (i) a naïve version that does not respect any design principle (the querier is at the center of the execution and handles all the information), and (ii) a basic distributed version that only respects the first two (i.e. without *imposed randomness* — the information is split between different actors but they are not chosen randomly).

We finally compared the results we obtained: the naïve version only requires one corrupted node to leak all of the exchanged information and does not scale well as the Querier acts as a bottleneck; the basic distributed version requires an inconsiderate amount of actors to achieve an acceptable disclosure (i.e. their number has to be greatly superior to the number of corrupted nodes) which, in turn, greatly increases the cost of the execution of a query; finally, our protocol offers the best features: the private information disclosure is guaranteed to increase linearly with the number of corrupted nodes, while the enforcement cost only increases logarithmically.

3 CONCLUSION

Personal Data Management Systems arrive at a rapid pace allowing users to share their personal data within large P2P communities. While the benefits are unquestionable, the important risks of private personal data leakage and misuse represent a major obstacle on the way of the massive adoption of such systems. This work is one of the first efforts to deal with this important and challenging issue. To this end, we propose a fully-distributed P2P system laying the foundation for secure, pertinent and efficient evaluation of aggregate queries based on user profiles. By considering a realistic threat model and guided by three design principles (knowledge dispersion, task compartmentalization and imposed randomness), we proposed a secure and efficient distributed protocol to protect both the data-at-rest and the data-in-use against an attacker controlling many nodes in the system. Furthermore, our simulation-based experiments show that our solution offers interesting properties: the private information leakage increases linearly with the number of corrupted nodes, while the cost of the security mechanisms increases logarithmically.

ACKNOWLEDGMENTS

This research is supported by the ANR PersSoCloud grant n°ANR-16-CE39-0014.

REFERENCES

[1] Serge Abiteboul, Benjamin André, and Daniel Kaplan. 2015. Managing your digital life. *Commun. ACM* 58, 5 (2015), 32–35.
[2] Michael Backes, Peter Druschel, Andreas Haeberlen, and Dominique Unruh. 2009. CSAR: A Practical and Provable Technique to Make Randomized Systems Accountable. In *NDSS*, Vol. 9. 341–353.
[3] Cozy Cloud. 2013. Cozy allow you to control your personal data (pictures, bank statements, bills, health reinbursements) in a secure and private space. Retrieved September 3, 2018 from https://cozy.io/en
[4] Fing. 2013. The mesinfos project explores and implements the self data concept in france. Retrieved September 3, 2018 from http://mesinfos.fing.org/english
[5] Troy Hunt. 2013. ';–Have I been pwned? Check if you have an account that has been compromised in a data breach. Retrieved September 3, 2018 from https://haveibeenpwned.org
[6] Saliha Lallali, Nicolas Anciaux, Iulian Sandu Popa, and Philippe Pucheral. 2017. Supporting secure keyword search in the personal cloud. *Information Systems* 72 (2017), 1–26.
[7] Frank McKeen, Ilya Alexandrovich, Alex Berenzon, Carlos V Rozas, Hisham Shafi, Vedvyas Shanbhogue, and Uday R Savagaonkar. 2013. Innovative instructions and software model for isolated execution. *HASP@ ISCA* 10 (2013).
[8] European Parliament. 2016. Regulation (EU) 2016/679 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data. Law. Retrieved September 3, 2018 from https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679
[9] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. 2001. *A scalable content-addressable network*. Vol. 31. ACM.
[10] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
[11] 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 Computer Communication Review* 31, 4 (2001), 149–160.