Secure distributed queries over large sets of personal home boxes

Riad Ladjel, Nicolas Anciaux, Philippe Pucheral, Guillaume Scerri

Secure distributed queries over large sets of personal home boxes

Riad Ladjel[2,1], Nicolas Anciaux[1,2], Philippe Pucheral[2,1] and Guillaume Scerri[2,1]

[1] Inria Saclay, 91120 Palaiseau, France
fname.lname@inria.fr
[2] University of Versailles. 78000 Versailles, France
fname.lname@uvsq.fr

Abstract. Smart disclosure initiatives and new regulations such as GDPR allow individuals to get the control back on their data by gathering their entire digital life in a Personal Data Management Systems (PDMS). Multiple PDMS architectures exist and differ on their ability to preserve data privacy and to perform collective computations crossing data of multiple individuals (e.g., epidemiological or social studies) but none of them satisfy both objectives. The emergence of Trusted Execution Environments (TEE) changes the game. We propose a solution called Trusted PDMS, combining the TEE and PDMS properties to manage the data of each individual, and a complete framework to execute collective computation on top of them, with strong privacy and fault tolerance guarantees. We demonstrate the practicality of the solution through a real case-study being conducted over 10.000 patients in the healthcare field.

Keywords: Trusted Execution Environment, Secure Distributed Computing, Data Privacy.

1 Introduction

As Tim Berners Lee advocates, “time has come to restore the power of individuals on the web” [38]. Smart disclosure initiatives (e.g., Blue and Green Button in the US, MiData in UK, MesInfos in France) and new privacy-protection regulations (e.g., GDPR in Europe [18]) are a first step in this direction, allowing individuals to retrieve their personal data from the companies and administrations hosting them. Hence, they can gather their complete digital environment in a single place, called Personal Cloud or Personal Data Management Systems (PDMS) [4] and manage it under their control.

Several companies are now riding this wave with highly diverse architectural solutions, ranging from *centralized web hosting PDMSs* (e.g., CozyCloud or Meeco and governmental programs like MyData.org in Finland or MesInfos.fing.org in France), to *Zero-knowledge personal clouds* (e.g., SpiderOak or Sync), up to fully *decentralized PDMS hosted at home* (e.g., CloudLocker or MyCloudHome).

The architectural dimension of the PDMS concept raises two important and potentially conflicting challenges: (1) gathering personal data previously scattered across distinct data silos generates new opportunities but incurs new privacy threats depending on where and how these data are hosted and (2) giving the power back to each individual on his data could impede the development of new collective services of high societal interest (e.g., computing statistics or clustering data for an epidemiological study).

Decentralized PDMS architectures have recognized privacy protection virtues wrt. challenge (1) by decreasing the Benefit/Cost ratio of an attack compared to their centralized web hosting counterparts. However, they make challenge (2) harder to tackle. How can we convince individuals who selected a decentralized PMDS setting to engage their personal data in a distributed process they do not control? Conversely, how could a service provider trust a processing performed by a myriad of untrusted participants? No existing work, including Multi-Party Computation (MPC) [9,13], gossip-based [2], homomorphic encryption-based [11,19] or differential privacy-based [14] protocols fully answer this dual question in a practical way. Existing solutions are able either to compute a limited set of operations (e.g., count, sum) in a scalable way or to compute arbitrary functions on a limited number of participants.

In this paper, we argue that the emergence of Trusted Execution Environments (TEE) [32] at the edge of the network – Intel SGX, ARM's TrustZone or TPM components are becoming omnipresent on every PC, tablets, smartphones and even smart objects – drastically changes the game. TEEs are able to efficiently compute arbitrary functions over sensitive data while providing data confidentiality and code integrity. However, TEEs have been designed to protect individual device/server rather than large scale distributed edge computing processes. Moreover, while TEE tamper-resistance makes attacks difficult and costly, side-channel attacks have been shown feasible [37]. Without appropriate counter-measures, a minority of corrupted participants in a distributed processing could endanger data from the majority.

In a previous paper [25], we have introduced a generic Manifest-based framework which answers the preceding requirements by establishing a mutual trust between participants and service provider that a data processing task distributed among a large set of TEEs will deliver a correct result in a privacy-preserving way, even in the presence of corrupted participants. In a second paper [24], we have instantiated this generic framework in the medical context and demonstrated the practicality of the approach through an ongoing deployment of the technology over 10.000 patients. Capitalizing on the manifest-based framework background [25], the current journal paper extends the work initiated in [24] in two main directions:

- First, we make the manifest-based framework fault-tolerant to handle any form of participants' failures (i.e., unexpected participant disconnections, shuts down, too slow communication throughput, etc) without aborting the complete protocol. Fault-tolerance is paramount in a fully decentralized context as the one considered in [24].
- Second, we improve the communication protocol considered in [24] by obfuscating the communication patterns that could leak critical personal data. Anonymizing the communications is mandatory in the medical field where highly sensitive data can be easily inferred by observing the information flow between computation nodes.

The rest of the paper is organized as follows. Section 2 recalls background material from [25] mandatory to understand the philosophy and principles of the manifest-based framework. Section 3 details the concrete instantiation of this framework in the medical field. Section 4 presents our new fault-tolerance protocol while section 5 is devoted to our new anonymized communication protocol. Section 6 evaluates the solution, both in terms of lessons learned and performance. Finally, section 7 summarizes relevant state of the art solutions and section 8 concludes.

2 Background material

2.1 TEE as Game-changer

The emergence of *Trusted Execution Environments* (TEE) [32] drastically changes the game regarding management of personal data. A TEE combines tamper-resistant hardware and software components to guarantee: (1) *data confidentiality*, meaning that private data residing in a TEE cannot be observed from the outside and (2) *code integrity*, meaning that an attacker controlling a corrupted user environment (or OS) cannot influence the behavior of a program executing within a TEE. TEE are now omnipresent in end-user devices like PCs (e.g., Intel's Software Guard eXtention (SGX) in Intel CPUs since Skylake version in 2015), mobile devices (e.g., ARM's TrustZone in ARM processors equipping smartphones and set-top boxes) and dedicated platforms (e.g., TPM combined with CPU or MCU). All these solutions provide the two properties mentioned above with different levels of performance and resilience to side-channel attacks which could compromise data confidentiality [37]. Anyway, side-channel attacks remain complex to perform and require physically instrumenting the TEE, which prevents large scale attacks. Code integrity is more difficult to compromise and not challenged today in most environments [35].

In this paper, and as in [24], we assume that each individual is equipped with a trusted PDMS (TPDMS), that is the combination of a TEE and a PDMS in a same dedicated hardware device. This device embeds a Trusted Computing Base, i.e. a certified software composed of: (1) a PDMS managing and protecting the individual's data (storing, updating, querying data and enforcing access control rules) and (2) a code loader ensuring the confidentiality and integrity of the code (in the TEE sense) executed in the box. Thus, only the trusted PDMS, the code loader, and additional external code certified and verified by the code loader (through a signature of that code) can run in the box. Persistent personal data are stored outside the security sphere, in a stable memory attached to the box (e.g., a SD card or a disk), but encrypted by the TPDMS to protect them in confidentiality and integrity. Considering the omnipresence of TEE in most end-user devices today, various concrete instantiations of TPDMS can be devised (e.g., combination of a TPM - Trusted Platform Module - with a microcontroller, a Raspberry-Pi with ARM Trustzone or a personal cloud server with Intel SGX).

2.2 Related Trust Model

We derive the following trust model from the previous section:

Large set of trusted TPDMS, small set of corrupted TPDMS. We assume that each individual is equipped with a TPDMS managing his personal data. As mentioned above, despite the TEE tamper-resistance and the cost of such attacks, side-channel attacks have been shown feasible. Hence, in a large scale setting, we cannot totally exclude the fact that a small subset of TPDMS could have been instrumented by malicious participants opening the door to side-channel attacks compromising the confidentiality property.

Trusted computation code. We consider that the code distributed to the participants to contribute to a distributed computation has been carefully reviewed and approved beforehand by a regulatory body (e.g., an association or national privacy regulatory agency) which signed this code. But the fact that the downloaded code is trusted does not imply that a whole distributed execution is trusted.

Untrusted infrastructure. Besides the presence of TPDMS, no security assumptions can be made about the user's environment or the communication infrastructure.

2.3 Expected properties

The problem to be solved can be formulated as follows: how to translate the trust put in the computation code declaration, as certified by the regulatory body, into a mutual trust from all parties in the concrete distributed execution of that code under the trust model above? Solving this problem leads to satisfying the following properties:

- *Mutual trust:* assuming that the declared code is executed within TPDMSs, mutual trust guarantees that: (1) only the data strictly specified for the computation is collected at each participants' PDMS, (2) only the final result of the computation can be disclosed, i.e., none of the collected raw data of any participant is leaked, (3) this final result is honestly computed as declared and (4) the computation code has the ability to check that any collected data is genuine.
- *Deterrence of side-channel attacks:* assuming a small fraction of malicious participants are involved in the computation with instrumented TPDMS, the deterrence property must (1) guarantee that the leakage remains circumscribed to the data manipulated by the sole corrupted TPDMS and (2) prevent the attackers from targeting a specific intermediate result (e.g., sensitive data or some targeted participants).

To have a practical interest, the solution must finally: (1) be generic enough to support any distributed computations (e.g., from simple aggregate queries to advanced machine learning computations) and (2) scale to a large population (e.g., tens of thousands) of individuals.

2.4 Manifest-based Framework for Trusted PDMS

To ensure a collective computation that scrupulously respects the properties described above, we propose a framework based on a Manifest describing the computation on which all the actors agree and a distributed protocol based on TPDMSs performing the computation in compliance with that Manifest. The solution is described in Figure 1. It is conducted in three main steps:

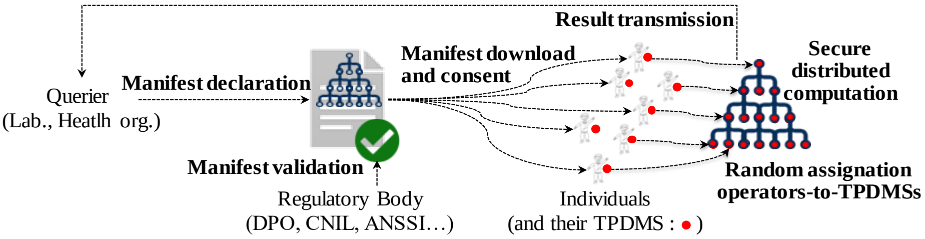

Result transmission
Manifest download and consent
Secure distributed computation
Querier (Lab., Heatlh org.)
Manifest declaration
Manifest validation
Regulatory Body (DPO, CNIL, ANSSI…)
Individuals (and their TPDMS : ●)
Random assignation operators-to-TPDMSs

Fig. 1. Manifest-based distributed computation framework.

Manifest declaration. The entity wishing to execute a distributed computation over personal data (e.g., a statistic agency, association of patients), called the *Querier*, acts as a data controller in the GDPR sense and produces a Manifest describing the computation. Individual contributors give their consent on the basis of the purpose of that manifest, and rely on regulatory bodies (e.g., WP29 members, CNIL) which validate the entire Manifest with regard to good confidentiality practices. To this end, the manifest indicates the identity of the *Querier*, which must be authorized for the purpose. It also provides the collection queries expressed in any easily interpretable declarative language (e.g., SQL), so that the regulatory body can verify that they reflect the principle of limited collection established by the legislation for the intended use. The code of the implemented operators and the organization of the data flow between them are also provided, and must correspond to the declared purpose. The number of participants plays a dual role: it represents both a threshold to be achieved for a sufficiently relevant result for the stated purpose and a privacy threshold preventing the risk of re-identification of individual data in the final result, which the regulator must also check. Once certified, the Manifest is published in a Public store where it can be downloaded by individuals wishing to participate. Example 1 shows the manifest of a distributed *group-by* query in the social-health context.

Random assignment of operators to participants. Participants download the manifest, and when a sufficient number consent to contribute with their data, each participant is assigned an operator of the Manifest. Ensuring a random assignment is critical to deter side-channel attacks on participants, by prohibiting corrupted participants from selecting specific operators in the execution process for malicious purpose (operators manipulating a large amount of data or receiving outputs from participants targeted by the attacker).

Secure distributed evaluation. Each participant's TPDMS downloads the code of the operator assigned to it and checks its signature, authenticates the TPDMS of participants supposed to exchange data with it (as specified in the random assignment) and establishes communication channels with them. The participant then executes his operator, potentially contributes personal data, and allows the computation to proceed by sending its output to its successor. Once all participants have executed their operator, the end-result is published on the public store encrypted with the public key of the *Querier*.

```
Purpose:
  Compute the avg number days of hospitalization prescribed
  group by patient’s age and dependency-level (Iso-Resource
Group, GIR)
Operators code:
  mapper source code
  reducer source code
Dataflow between the operators:
  Number of mappers: 10000
  Number of reducers: 10
Collection queries:
  SELECT GIR, to_year(sysdate-birthdate) FROM Patient;
  SELECT avg(qty)FROM Prescription WHERE prescType = ‘hospi-
talization’;
Number of participants: 10000
Querier: ARS-Health-Agency, Public key: Rex2%ÃźHj6k7âĂę
```

Example 1. ‘Group-by’ Manifest expressed by health organization.

2.5 Random Assignment Protocol

Obtaining a random assignment of operators to participants is key to prevent any potential attackers (Querier or any participants) colluding with corrupted TPDMS from being assigned a targeted operator or position in the dataflow at execution. While existing solutions have been proposed to ensure that a random number is chosen and attested in distributed settings, e.g., [7], none can be applied to reach this specific goal as they assume the list of participants is known in advance, as opposed to our case where the participant list is chosen based on collected users’ consents. We propose a solution to produce a provably random assignment, detailed in Fig. 2. As we consider TPDMS as trusted, the random assignment can be delegated to any TPDMS. However, the challenge is avoiding any malicious Participant or Querier aborting and replaying the assignment process a large number of times, picking the best one for a potential attack.

To avoid such attacks, we make sure, in a first step of our protocol, that the Querier commits to an assigning participant among the consenting participants. More precisely, each consenting participant first declares itself by publishing its identity and the hash of a random number used later to prove reception of the list of participants. Second, the protocol ensures that once the list of participants has been fixed, the assignment is actually performed randomly, and that this randomness can be checked by every participant. Hence, once the Querier has gathered enough participants willing to participate, it broadcasts the full list of participants together with the designated assigning participant, which is acknowledged by each participant by disclosing the random number chosen in the initial step. Following this, the designated assigning participant is sent the full list of participants together with all the acknowledgements. He then checks that all acknowledgements are valid, and performs a random assignment of operators to participants. Finally, he signs this assignment and sends it back to the Querier.

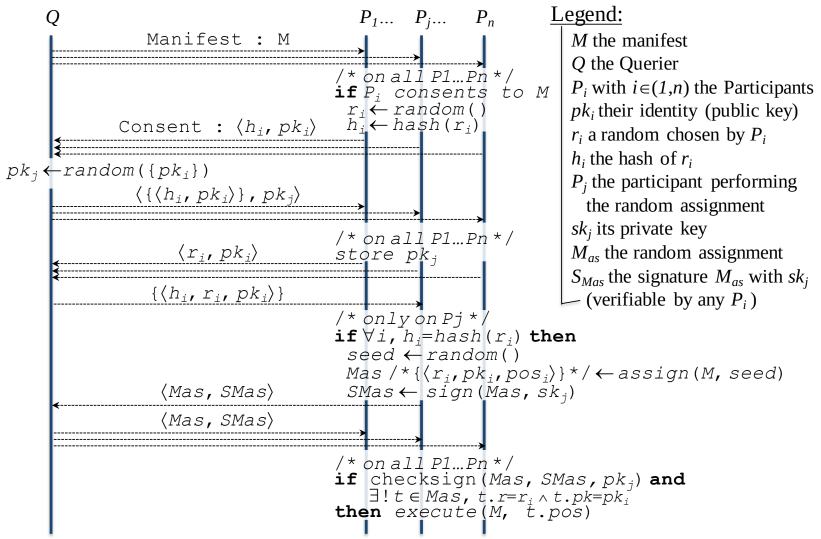


Fig. 2. Random assignment of operators to participants.

Thus, the protocol ensures that when an individual consents to a manifest, the assignment can only be made once, at random (any attempt to replay the assignment would be visible to the participants and a restart require to obtain their consents again).

2.6 Global Assessment of the Manifest-based Framework

We sum up by showing how the framework satisfies the properties identified in Section 2.3.

Deterrence of attacks. This property first states that *the data leakage due to an attack must be circumscribed to the sole data manipulated by the corrupted TPDMS.* This is intrinsically achieved by never sharing any cryptographic information among different TPDMS. Hence, any persistent data residing in a TPDMS is encrypted with secret keys known only by that TPDMS and intermediate results in transit between a predecessor and a successor TPDMS is encrypted with a session key (see below) and managed in clear-text in that successor (inheriting the confidentiality property from the TEE processing it). The second requirement is to prevent *any attacker from targeting specific personal data,* which is the precise goal of the *Random Assignment Protocol* introduced in Section 2.5.

These requirements satisfy deterrence of attacks by drastically increasing the Cost/Benefit ratio of an attack. Indeed, even if a fraction of TPDMSs is instrumented with side-channel attacks compromising data confidentiality, such attack incurs a high cost of tampering with secure hardware (with physical intervention of the PDMS owner) with a benefit limited to obtaining the data manipulated by the sole corrupted TPDMSs, and negligible probability for gaining any personal data of interest. Indeed, in a computation manifest, we distinguish between participants assigned to a collection operator (which only extracts personal raw data from the participant) and participants assigned to a computation operator (which process personal data collected from others). Then, attacking any TPDMS running a collection operator is of no interest since the attacker only gains access to his own data. Moreover, the probability of a corrupted node being assigned a computation operator is negligible in practice (see security analysis in Section 6). Thus, although more elaborate strategies could be adopted to further maximize the Cost/Benefit ratio (e.g., blurring data), they are considered unnecessary in our context.

Mutual trust. The *mutual trust* property is guaranteed if two hypotheses hold: (H1) all data exchanged between the participants' TPDMSs are encrypted with session keys and (H2) each TPDMS involved in the computation authenticates its neighboring participants as legitimate TPDMSs complying with the random assignment for that manifest. The first condition for *mutual trust* (see Section 2.3) stems from the fact that (1) the collection queries are part of the manifest certified by the Regulatory body, (2) its authenticity is cryptographically checked by each TPDMS and (3) the TPDMS evaluating these queries is part of the Trusted Computing Base thereby guaranteeing the integrity of this collection phase. The second condition is satisfied by construction since each TPDMS guarantees the confidentiality of local data and H1 guarantees the confidentiality of intermediate data in transit. The final result is itself encrypted with the public key of the Querier so that no other data is ever leaked outside the TPDMS ecosystem. The third condition is again satisfied by construction by H2 guaranteeing that only genuine operators are computed and conform to the dataflow specified in the manifest. The last condition stems from the fact that (1) local data can be manipulated in clear-text inside each TPDMS, allowing any form of verification (e.g., check signature of data produced by a smart meter or quantified-self device, or issued by an employer or a bank) and (2) H2 guarantees the integrity of the data collection operator at each participant. Note that this guarantee holds even in the presence of corrupted TPDMSs which could compromise the *confidentiality* property.

In conclusion, the proposed solution is generic enough to capture any distributed execution plan where any node can be an operator of any complexity and edges are secure communication channels between the TPDMS of participants executing the operators. Compared to the state of the art, our manifest-based approach has the ability to reconcile security with genericity and scalability. First, the TEE *confidentiality* property can be leveraged to execute each operation over clear-text genuine data. Second, the number of messages exchanged among participants only results from the distributed computation to be performed, but not from the underlying security mechanism. Hence, unlike secure Multiparty computations (MPC), homomorphic encryption, Gossip or Differential privacy approaches, no computational constraint hurting genericity nor scalability need to be introduced in the processing for security reasons.

3 A Trusted PDMS in the Medical-Social Field

This section presents an on-going deployment of a TPDMS in the medical-social field and assesses the practicality of the Manifest framework.

Overview. End of 2017, the Yvelines district in France launched a public call for tender to deploy an Electronic Healthcare Record (EHR) infrastructure facilitating the medical and social care at home for elderly people. 10.000 patients are primarily targeted by this initiative, with the objective to use it as a testbed for a larger medium-term national/international deployment. The question raised by this initiative is threefold:

- How to make patients, caregivers and professionals trust the EHR security despite the recent and massive privacy leakages due to piracy, scrutinization and opaque business practices inherent to any data centralization process?
- How to combine privacy expectations with the collective benefits of data analysis tools to rationalize care, improve business practices and predict disease evolution?
- How to make patient's healthcare folder available even in a disconnected mode considering the low adoption of internet by elderly people?

The Hippocad company, in partnership with the Inria research institute and the University of Versailles (UVSQ), won this call for tender with a solution called hereafter THPC (Trusted Health Personal Cloud). THPC is based on a home box, pictured in Figure 3, combining 3 usages: (1) effectiveness control and vigilance, (2) home care coordination and (3) supervision (forthcoming). The hardware incorporates a number of sensors and communication modules (in particular SigFox) managed by a first microcontroller (called MCU1) devoted to the communication and sensing tasks. The data delivered by the box are used by the Yvelines district to cover usage (1), that is adjusting the care payment to their duration and performing a remote vigilance of the patient home. A second microcontroller (MCU2: STM32F417, 168 MHz, 192 KB RAM, 1 MB of NOR storage) is devoted to the PDMS managing the patient folder, a µ-SD card hosting the raw patient data (encrypted by the PDMS) and a tamper-resistant TPM (Trusted Platform Module) securing the cryptographic keys and the boot phase of the PDMS in MCU2. As detailed next, the combination of a TPM with MCU2 forms a TPDMS. Care professionals interact with the PDMS (i.e., query and update the patient's folder) through Bluetooth connected smartphone apps, covering usage (2). Finally, volunteer patients accepting to contribute to distributed computations (usage (3)), will be equipped with a box variant where the SigFox module is replaced by a GPRS module.

The PDMS engine itself has been specifically designed by Inria/UVSQ to accommodate the constraints of MCU2. This embedded PDMS is a full-fledged personal database server capable of storing data in tables, querying them in SQL, and provides access control policies. Hence, care professionals can each interact with the patient's folder according to the privileges associated to their role (e.g., a physician and a nurse will not get access to the same data). Finally, the patient's data is replicated in an encrypted archive on a remote server to be able to recover it in case of crash. A specific master key recovery process (based on Shamir's secret sharing) has been designed to guarantee that no one but the patient can recover this master key.

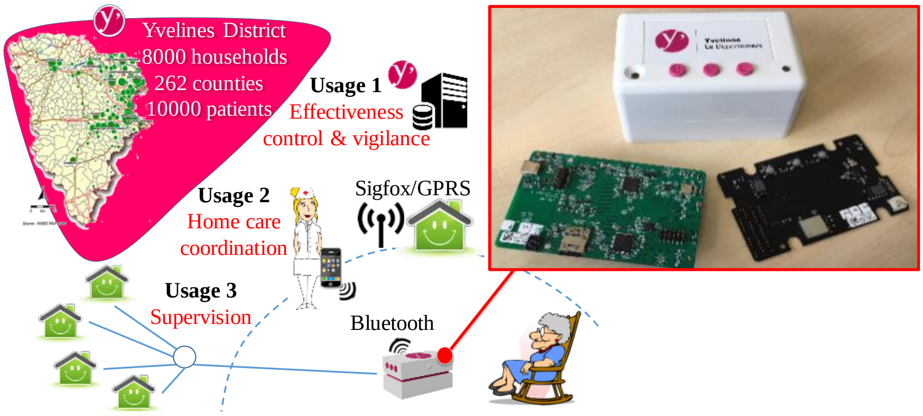


Fig. 3. Architecture of the THPC solution.

THPC as an instance of Trusted PDMS. The THPC platform described above is an illustrative example of TPDMS. As introduced in Section 2.1, a TPDMS is a combination of a TEE and a PDMS software embedded in a same dedicated hardware device, providing confidentiality and integrity guarantees for the code running in this device. The presence of two separate MCUs answers security concerns, indeed the Trusted Computing Base (TCB) is limited to the code located in MCU2 and does not include drivers and sensors (managed by MCU1) and is thus minimalistic. Additionally, the TCB is cryptographically signed. The TPM protecting the box is used at boot time (and NOR flash time) to check the genuineness of the PDMS code by checking the signature. The PDMS code in turn can download external pieces of code corresponding to the operators extracted from a Manifest, check their integrity thanks to the code signature provided by the Regulatory body, and run it. Hence, no code other than the TCB and signed operators can run in the box. The TPM also protects the cryptographic certificate that securely identifies the box and the master key protecting the personal database footprint on the µ-SD card. Note however that, while the TPM is tamper-resistant, the MCU2 is not. Hence, a motivated attacker could physically instrument his box to spy the content of the RAM at run time.

Distributed computations of interest. The next critical step of the project is to integrate usage (3) (supervision). GPRS variant of the boxes are under development to establish a communication network via a central server settled by the Hippocad company, which plays the role of a communication gateway between the THPC boxes (it relays encrypted data bunches between THPC boxes but cannot access to the underlying data). Two essential distributed computations are considered, namely the *Group-by* and *K-means* computations. *Group-by* allows computing simple statistics by evaluating aggregate functions (sum, average, min, max, etc.) on population subsets grouped by various dimensions (the level of dependence or GIR, age, gender, income, etc.). Such statistics are of general interest in their own and are also often used to calibrate more sophisticated data analysis techniques (e.g., accurately calibrate the k parameter of a *K-means* computation). *K-means* is one of the most popular clustering technique and is broadly used to analyze health data [26]. To date however, few studies was conducted on home care [23] because data management techniques for this type of care are still emerging. Yet, *K-means* techniques already delivered significant results to predict the evolution of patient dependency level after a hip fracture [16] or Alzheimer symptoms, and derive from that the required evolution of the home cares to be provided and their cost. The first two Manifest-based computations considered in the project cover these use cases as follows:

- The *Group-by* manifest is the one presented in Example 1, using the usual map-reduce implementation of a Group-by computation, where operators executed by participants are the map and reduce task respectively. It computes the sum and average duration of home visits by professionals grouped by professional category and level of dependence (GIR) of the patient. Such statistics are expected to help adjusting the duration of interventions and the level of assistance according to the patients' situation.
- The *K-means* manifest is inspired by a previous study conducted in Canada with elderly people in home care. This study analyses 37 variables, and provides 7 centroids [6] that finely characterize the people cared for. On a similar map-reduce basis, we define K-means manifests computed over distributed PDMSs in three steps: (1) k initial means representing the centroid of k clusters are randomly generated by the Querier and sent to all participants to initialize the processing, (2) each participant playing a mapper role computes its distance with these k means and sends back its data to the reducer node managing the closest cluster, (3) each reducer recomputes the new centroid of the cluster it manages based on the data received from the mappers and sends it back to all participants. Steps 2 and 3 are repeated a given number of times or until convergence.

In Section 6, we give preliminary measures obtained by a combination of real measures and simulations for these two manifests since they are not yet deployed. Running manifests in the THPC context has required an adaptation of the random assignation protocol to cope with the intrinsic communication bandwidth's limitation of GPRS.

Adaptation of the Random Assignment Protocol to the THPC context. Given the low bandwidth of the THPC boxes (GPRS communications), a critical problem is limiting the amount of data transmitted to all participants, as hundreds of KBs broadcasted to all thousands of participants would not be compatible with acceptable performance. In order to reach this goal, we optimize the two main parts of the random assignation protocol (Section 2.5) that lead to transmission of large amounts of data. The main optimization is making sure that we do not need to transmit neither the whole assignment nor the whole manifest to all participants as they only need their part of the assignment and the manifest related to their part of the computation. However, we need to make sure that the integrity of the whole manifest and assignment is ensured. In order to achieve these two seemingly antagonistic goals, we make use of Merkle hash trees [27] over the corresponding data structures. The properties of the Merkle hash tree ensures that given the root of the hash tree, it is possible to provide a small checksum proving (in the cryptographic sense) that an element belongs to the corresponding hash tree, and it is computationally infeasible to forge such a proof. Note that the checksum is a logarithmic (in the number of values in the tree) number of hashed values and thus stays manageable (small size). Additionally, we avoid broadcasting the whole list of participants as only the assigning participant needs to perform checks on this list. We only broadcast a cryptographic hash of this list, and only send it in full to the assigning participant who actually needs to check it. The assigning participant however does not need to send back the full assignment, only a Merkle hash tree signed with its private key, and the random seed used to generate the assignment (so that the Querier can reconstruct it) is sent back. Finally, in order to perform its task in the manifest, any participant only needs its position together with the corresponding operator, collection queries and data flow and proof of membership to the logical manifest. Additionally, the participant needs to receive proof that the assignment is correct.

Summing up, we reduce the communication load during assignment building phase from a few broadcasts of a few hundreds of KB (for tens of thousands of participants) to only one large download for the assigning participant (again a few hundreds of KB), and small downloads/uploads (a few tens of Bytes) for all other participants, drastically reducing the overall communication load, and making it manageable in constrained setting.

4 Fault tolerance protocol

Any distributed solution involving end-users computing resources must consider the case of participants' failures, i.e., becoming unreachable due to unexpected disconnections, shuts down, low communication throughput, etc. This statement is particularly true in our medical-social context involving battery-powered devices connected to the network by a GPRS module.

With the Manifest-based framework presented in Section 2.4, any participant failure conducts to stop the execution (fortunately without exposing any result), forcing the querier to restart processing from its beginning. The objective is thus to support a ratio of participant failures while enabling the execution to be completed. However, failures may impact the security of the solution: a malicious participant may deliberately attempt to weaken other participants' connectivity (e.g., denial of service attack) to harm the confidentiality or integrity of the computation. We consider here both the security and the performance aspects of handling failures.

Participants failures in our context may occur either during step 2 (random assignment of operators) or step 3 (secure distributed evaluation). Failures at step 2 are easily tackled by removing faulty participants from the protocol. Failures at step 3 are more difficult to address. Traditional fault-tolerant solutions rely either on redundant execution methods (e.g., perform k independent executions of a same operator and select a result) or on check-pointing mechanisms (e.g., store intermediate results of operators and restart computation from these points). Both solutions unfortunately increase data exposure, either increasing the number of participants processing the same data or introducing additional persistent copies of such data. We select the first solution anyway (redundant execution) because its negative effects are largely alleviated by the randomness measure integrated in our protocol, proscribing attackers targeting specific nodes.

We explain below how to integrate redundant execution in a manifest, for the case of a n-ary tree-based execution plan. Let assume a redundancy factor of k (with k = 2 or 3 in practice), a failure-resilient solution can be formed as follows:

1. For a manifest *M* requesting *N* participants, *k.N* participants are actually selected.
2. When assigning operators to participants, *k* participants are assigned the same operator in M. They inherit the same position in the execution plan and the same operator to run, to form a so-called bundle of redundant participants. Hence, participants p_i, p_j, …, $p_k \in$ bundle$_l$ all execute the same operator *op$_l$*.
3. The assignment function from participant to role is de facto no longer injective. However, the position of each participant in the execution plan is still determined at random. Consequently, bundles are also populated randomly.
4. Each participant in a successor bundle is connected to all participants in an antecedent bundle by edges in the execution plan (antecedent/successor refer to the position of participants/bundles in the execution plan/tree).
5. Instead of iterating for each antecedent, a participant iterates for each bundle, and the participants in this bundle are considered one after the other at random. If one does not answer after a certain delay, it is considered as 'failed' and a next participant in the same bundle is contacted. If all participants of the same bundle fails, the whole processing is abandoned.

Correctness. Any participant consuming an input data (resp. sending an output data) checks beforehand the integrity and identity of its antecedents (resp. successor) in the execution plan, as in a standard (i.e., non-fault-tolerant) execution. If a complete bundle fails, the error is propagated along the execution plan such that the execution ends with an error and no result is published. Finally, what if participants of a bundle play the role of data collectors, each being connected to its local PDMS? The local data are not the same at each participant and the output delivered by the bundle hence depends on which active participant is finally selected in the bundle. This randomness in the result of the computation is actually not different from the one incurred by selecting N among potentially P consenting participants in the protocol and does not hurt the consistency of the execution.

In terms of performance, this strategy does not impact the response time since participants in the same bundle work in parallel (it can even be better considering that the input of an antecedent bundle may arrive faster than the one of a single antecedent). However, the overall computation cost (sum of all computation costs) is increased by a factor of k and the number of communications by a factor of k*k. While this grows extremely rapidly, note that we only need to consider very small values of k (typically 2) as we allow for one failure per bundle, and the probability of a whole bundle failing is extremely small even if bundles are small.

5 Anonymous communication protocol

Anonymizing the communications is mandatory in the medical field. In our distributed computation framework, sensitive data can be inferred by observing the information flow between computation nodes. Typically, in canonical map-reduce computations encoded with our manifest-based framework, as shown in Example 1, each participant acting as a mapper node sends its <*GIR*, [*age, #days-of-hospitalization*]> to the given reducer in charge of aggregating the information for that *GIR*. Observing the communications would reveal the recipient reducer for any participant and hence disclose its *GIR* value (i.e., its level of dependence).

Distributed execution plans often exhibit such data dependent data flows, as directing tuples to be processed together to a same computation node is necessary to evaluate certain statistics (e.g., computing a median) and/or improve performance. In the general case, two main strategies can be adopted to hide data dependent communications: (1) use anonymous communication networks (e.g., use TOR) to hide any link between data recipient nodes and source nodes, or (2) "cover" data dependent communications within fixed, data independent patterns. Resorting to the first approach requires tackling the issue of adapting onion routing protocols to our resource constrained THPC platform. This is still considered an open issue in the general case of constrained IoT devices (see, e.g. [20]), making such approach infeasible in practice in the short term.

Using the second approach would simply mean replacing sensitive communication patterns with data independent ones (e.g., broadcasts) at the price of extra communications. In the previous example, this could be achieved by enforcing (as part of the manifest) that any message sent from a mapper node to a reducer node also triggers sending one extra 'empty' messages of same size to all other reducers, thus forming a 'broadcast-like' communication pattern (and hence hiding the value of GIR). The expected performance penalty is high, especially in the context of our THPC solution, considering the limitation of GPRS in terms of communication bandwidth.

We present below a simple way to adopt data independent communication patterns in a manifest, while limiting the communication overhead to a minimum acceptable in our context.

Adopting data independent communication patterns. Let consider for simplicity a n-ary tree-based execution plan in the manifest, where at a given level l in the tree, the data exchanges issued from the child nodes to the parent nodes (at level l+1) reveal information on data values processed at the child nodes. For the sake of simplicity, we consider that each child node transmits a unique message to a given parent node, selected on the basis of a sensitive information hold at the child node. The naive solution to avoid exposing sensitive information is to cover such child-to-parent message by broadcasting an empty message of same size to all other potential parent nodes at level l+1. In terms of extra communications, with n_l nodes at level l each with a single message of size $|t|$ bytes to be transmitted to a parent node at level l+1, this causes $n_l \times (n_{l+1} - 1)$ additional messages, with extra size $|t| \times n_l \times (n_{l+1} - 1)$ bytes in total. In practice, considering, e.g. *10.000* mappers and *10* reducers as in Example 1 and a tuple size of *100* bytes, this leads to *9.000* extra communications with *900* KB data exchanged (mostly composed of 'empty' tuples), with unacceptable performance in the THPC setting.

Minimizing communication overhead. To reduce the communication overhead, we modify the distributed execution plan in the manifest as follows (see Fig. 4): for each level l in the tree where data exchanges issued from child nodes to parent nodes (at level l+1) reveal information on data values (1) we form a k-equipartition[1] of the set of child nodes, we allocate one *scrambler* node per k-partition (with ${}^{n_l}/_{k}$ scrambler nodes allocated in total) and connect each child node to the scrambler node responsible for its k-partition; and (2) we connect each scrambler node to all the parent nodes and fix at exactly $k' \leq k$ the number of messages each scrambler sends to each parent node.

Each scrambler node acts in two phases. First, it collects one tuple $< P_i, E_{K_{P_i}}(M_{C_j}) >$ per child node C_j of the k-partition it takes in charge, with P_i the parent node the message has to be transmitted and $E_{K_{P_i}}(M_{C_j})$ the message M produced by C_j encrypted with the public key[2] K of P_i. Second, the scrambler prepares k' messages packages (each of same size) destined to each parent node, it places the encrypted messages collected from the child nodes in the appropriate package for the expected parent nodes and fills in the remaining packages with '*empty*' messages (indistinguishable from others, as being same size and encrypted). In terms of extra communications, this causes $n_{l+1} \times {}^{n_l}/_{k}$ additional messages, each of size $k' \times |t|$ bytes.

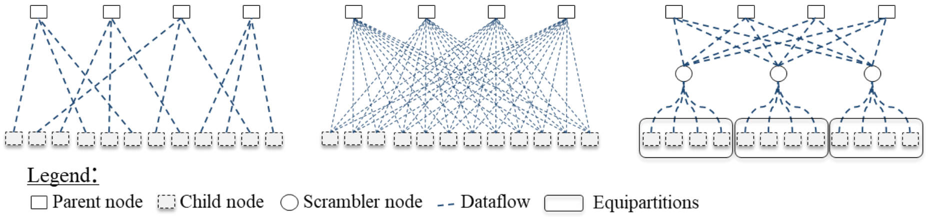


Fig. 4. Covering sensitive data exchanges with data independent communication patterns (Left: data dependent; Middle: naive, Right: scrambler-based with k=4).

Resilience to attacks. The communication pattern introduced by scrambler nodes is fully deterministic and prevents from disclosing sensitive information (the only information disclosed is the size $k' \times |t|$ of data exchanged from scramblers to parent nodes). The *deterrence of side-channel attacks* property must (1) guarantee that the leakage remains circumscribed to the data manipulated by the sole corrupted TPDMS and (2) prevent the attackers from targeting a specific intermediate result (e.g., sensitive data or data of some targeted participants). If a given scrambler is corrupted, it only reveals information regarding the data flow of the partition it has in charge, which ensures that the leakage remains circumscribed (first part of the property). Remark also that only the local communication pattern is exposed to corrupted scramblers, but not the content (payload) of the routed messages (as being encrypted with the recipient node's public key). Hence, lower k leads thus to more k-partitions and more scramblers, with a better resilience to side channel attacks. In addition, the random assignment of the (scrambler) operators to participants prevents potential attackers from targeting specific scramblers (enforcing the second part of the property). Note also that the impact on *mutual trust* is null, as the addition of scramblers does not change the deterministic nature of the query

[1] A k-equipartition of a set is the partitioning of this set in partitions of cardinality k.

[2] We assume that each node is endowed with a public/private key pair.

execution plan (nodes can check the integrity of predecessors, enforcing the global integrity of the query tree).

Performance analysis. In terms of performance, the value of k determines the size of the k-partitions and the number of scramblers $\lceil n_l/k \rceil$, but has no effect on the total volume of data exchanged. Indeed, the addition of scramblers let unchanged the number of messages issued from child nodes (in our setting, n_l messages, one per child node, transmitted to the scrambler responsible for its partition), but it introduces $\lceil n_l/k \rceil \times n_{l+1} \times k'$ (with $1 \leq k' \leq k$) additional messages from scramblers to parent nodes, each of size $|t|$ bytes. Hence, the lower k' leads to the better efficiency. The worst case in terms of communications is $k = k'$. At one extreme, with $k = k' = n_l$ a single scrambler is introduced which transmits n_{l+1} groups of $k' = n_l$ messages with in total the same overhead as that of the naïve solution in number of transmitted bytes. At the other extreme, $k = k' = 1$ leads to introduce n_l scramblers each sending n_{l+1} messages (one message per parent node) with the same global overhead. The performance with scramblers hence becomes better than the naïve solution when $k'/k < (n_{l+1} - 1)/n_{l+1}$.

Calibration of the parameters k and k'. Reducing the value of k increases the resilience to attacks (with lower k, more scramblers, and a better resilience to side channel attacks). It also improves the degree of parallelism (more scramblers run in parallel) and plays on the overall resource/energy consumption (due to less messages processed at each scrambler, with less memory consumed and less energy). This is of importance in the THPC context where the memory, processing and lifetime of each node is limited. Assuming k has been reduced appropriately to match (privacy and) resource constraints, the second step is to minimize the value of k' to reduce the communication overhead. At the same time, a too small value k' increases the risk of execution failure, if more than k' messages have to be transmitted at execution from a given scrambler to a given parent node. To enable fine tuning the value of k' at runtime, the strategy we adopt is to ask each scrambler, once all input tuples have been collected from the k-partition they have in charge, to first transmit to all parent nodes the maximum number of tuples each has to transmit to this parent nodes, and ask the parent nodes to send back to scramblers the maximal received value, such that k' is fixed in all as the maximal value received from all parent nodes[3]. Note that during this phase, the only additional data transmitted from scrambler to parent is the number of intended messages for this specific parent node. As this data is known to the parent node regardless of the protocol used to fix k, it does not negatively impact security. In practice, well calibrated query execution tree lead to process in all parent nodes a roughly similar amount of tuples (for good load balancing and efficiency), leading to select k' bigger but close to $\lceil n_l/k \rceil$ to minimize the number of empty tuples to be send. Typically, considering Example 1, where $n_{l+1} = 10$ and $n_l = 10.000$, with 10 scramblers (i.e., $k = 1000$), most executions end up with $k' \leq 200$, which means 5 times less data transmitted than using the naïve strategy (equivalent to $k' = 900$).

[3] Note that this formally makes the communication flow data dependent as the chosen k' depends on the data sent to each scrambler. This, however, only leaks information on the distribution of data, not on any individual data. We do not view this as a significant threat.

In conclusion, the principle described here can be implemented to protect sensitive (data dependent) communication patterns with acceptable overhead in many practical examples of distributed PDMS calculations, ranging from simple statistical queries to big-data (map-reduce style) processing, as illustrated in the section on validation. The process of adding scramblers can be performed automatically by a precompiler taking as input a logical manifest and producing a transformed logical manifest covering the communications identified as sensitive. The appropriate value of the *k'* parameter does not need to be established at pre-compilation, but can be adjusted at runtime (as described above). The selection of the value of *k* to form the *k*-equipartitions is dictated by resource constraints in our context and must be provided for at pre-compilation. Tuning of the value of k and study of optimal strategies, as well as their integration in a precompiler are left for future work.

6 Validation

While the THPC platform is still under deployment over the 10.000 targeted patients, we can already draw interesting lessons learned and present preliminary performance and security results of the Manifest framework applied to the *Group-by* and *K-means* cases.

6.1 Lessons learned for the Deployment of THPC Solution

An important criterion for the Yvelines District when selecting the THPC solution was its compliance with the new GDPR regulations and its ability to foster its adoption by patients and professionals.

Adoption by patients. From the patients' perspective, a crucial point was to provide differentiated views of their medical-social folder (e.g., a nurse is not supposed to see the income of the elderly person). To this end, a RBAC matrix (role-based access control) has been defined so that a professional owning a badge with a certificate attesting role R can play this role with the appropriate privileges on all patients' boxes. Each patient can explicitly - and physically - express his consent (or not) to the access of a given professional by allowing access to his box during the consultation, as he would do with a paper folder. The patient can also express his consent, with the help of his referent, for each manifest. A notable effect of our proposal is to consent to a specific use of the data and to disclose only the computed result rather than all raw data as usual (e.g., consenting to an Android application manifest provides an unconditional access to the raw data).

Adoption by professionals. Professionals are reluctant to use an EHR solution which could disclose their contact details, planning and statistical information that may reveal their professional practice (e.g., quantity of drugs prescribed or duration and frequency of home visits). A decentralized and secured solution is a great vector of adoption compared to any central server solution. Similarly, professionals are usually reluctant to see the data related to their practice involved in statistical studies unless strict anonymization guarantees can be provided. While the consent of the professionals is not requested for distributed computations, a desirable effect of our proposal is to never disclose individual data referring to a given professional, and submit all computation to regulatory approval.

6.2 Performance and Security Evaluation of the Manifest-based Framework

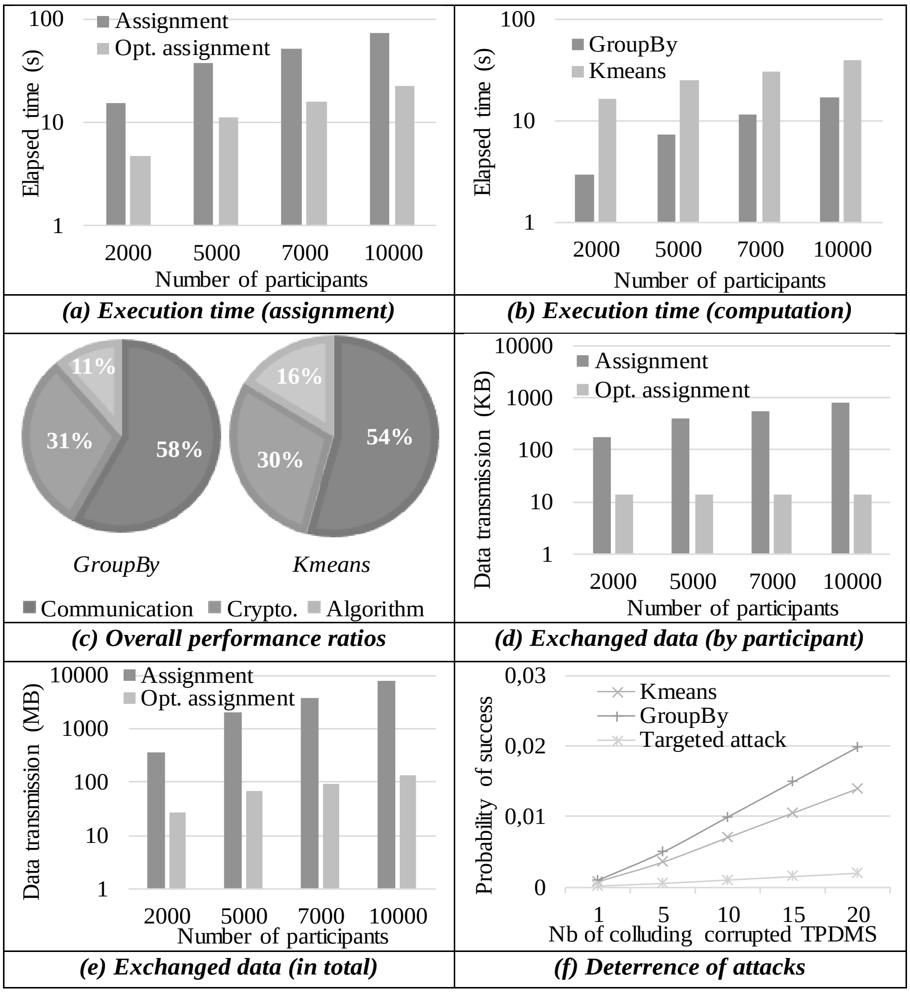
We validate the effectiveness of our approach on the Group-by and K-means use-cases.

Experimental setting. We implemented the corresponding mappers and reducers code in the THPC box with a server used to route (encrypted) messages between participants, as described in Section 4. We computed the execution time while considering different numbers of participants and amount of data transferred during the computations. We used a simulation to derive execution times with large numbers of participants. The results are shown in Fig. 4 (the curves are in log. scale). For the Group-by case we consider an implementation with 10 reducers and 50 different group keys, while for the K-means we consider 7 different clusters with 1 cluster per reducer as in [5] using a traditional distance metric [21]. We used synthetic datasets, as the objective is not to choose the most efficient implementation of a given computation, but rather to assess the efficiency of the manifest-based protocol on real use-cases. As cryptographic tools we used ECC 256 bits for asymmetric encryption, ECDSA signature scheme, AES 128 bits for symmetric encryption and SHA-2 as a hash function, leveraging the hardware acceleration provided by MCU2.

Performance evaluation. Figures 4.a-b-d-e plot the various costs associated with our protocol. First, the optimization of our random assignment protocol has a strong impact on its execution time, from 75 seconds without optimization down to 22 seconds with 10000 participants (this cost depends on the number of participants, but not on the query performed), as well as on the volume of data exchanges, from 800 KB per participant without optimization down to 13 KB (with in total, 7 GB exchanged data down to 130 MB). Once the assignation is performed, the query computation time remains reasonable, with 18 seconds (resp. 40 seconds) for a Group-by (resp. K-means) over 10000 participants. Finally, the overhead incurred by the random assignation is limited (e.g., between 20 and 30% of overall time), and the main part of the cost is due to communications (see Fig. 4.c).

Fig. 4.f shows the tremendous impact of the random assignment protocol in terms of security. It plots the probability for a set of colluding malicious participants, acting with corrupted TPDMSs in Group-by and K-means computations, to be assigned a reducer operator (hence gaining access to data produced by other participants) or to the data of a given participant of interest (targeted attack). This probability remains very low (few percent) even if several participants successfully corrupt their TPDMS and collude.

The main lessons drawn from these experiments are: First, even with the hardware limitations of the box in terms of computing power and communication throughput, the global time is rather low and acceptable for this kind of study (less than a minute for 10000 participants in comparison with manual epidemiological surveys which may last weeks). Second, the optimization of the assignment protocol has a decisive impact on both execution times and data volumes exchanged, with a significant financial benefit in the context of pay-per-use communication services (such as GPRS network).



(a) Execution time (assignment)

(b) Execution time (computation)

(c) Overall performance ratios

(d) Exchanged data (by participant)

(e) Exchanged data (in total)

(f) Deterrence of attacks

Fig. 4. Performance and security evaluation.

7 Related works

The first part of this section analyses the pros and cons of existing alternatives in terms of PDMS architectures and positions the TPDMS solution in this landscape. The second part of this state of the art is devoted to the solutions proposed to perform secure database computations and it positions our manifest-based contribution relatively to these works.

7.1 Analysis of PDMS Architecture Alternatives

The *Personal Data Management Systems* (PDMS) [4] concept, also called Personal Cloud, PIMS [1], Personal Data Server [3] or Personal Data Store [28], attracts significant attention from the research and industrial communities. We briefly review the main families of solutions and compare their ability to tackle the two challenges identified in the introduction, namely privacy preservation and distributed collective computations.

Centralized web hosting solutions. CozyCloud, Digi.me, Meeco, or Perkeep and governmental programs like MyData.org (Finland), MesInfos.fing.org (France) or MyDex.org (UK) are representative of this family. Individuals centralize their personal data in a server managed by the PDMS provider and can access them through the internet. These approaches rely on strong security hypotheses: (i) the PDMS provider and its employees are assumed to be fully-honest, and (ii) the PDMS code as well as all applications running on top of it must be trusted. This is critical in a centralized context exacerbating the Benefit/Cost ratio of an attack. On the other hand, collective computations are simplified by the centralization but the security of such processing remains an issue.

Zero-knowledge personal clouds such as SpiderOak or Sync and to a certain extent Digi.me mentioned above, propose a variation of the centralized web hosting solutions where data is stored encrypted in the cloud and the user inherits the responsibility to store and manage the encryption keys elsewhere. The price to pay for this increase of security is the difficulty to develop advanced (local or distributed) services on top of zero-knowledge personal clouds, reducing their use to a robust personal data safe.

Home cloud software solutions (e.g., OpenPDS [28], DataBox [12]) manage personal data at the extremities of the network (e.g., within the user's equipment) to circumvent the security risks of data centralization. Hence, queries on user's data can be computed locally and only the result is sent back to the querier. However, these solutions implicitly assume that the computing platform at the user side is trusted and cannot be tampered with.

Home cloud box (e.g., CloudLocker, MyCloudHome and many personal NAS solutions) go further in this direction by offering a dedicated box that can store TBs of data and run a server plugged on an individual's home internet gateway. This solution alleviates the burden of administrating a server on the individual's device and logically isolates the user's computing environment from the box, they, however do not focus on security. Home cloud software nor home cloud box consider secure distributed processing as a primary target.

The first conclusion that can be drawn from this analysis is that online personal cloud solutions have the technical ability to perform distributed computations but suffer from very strong hypotheses in terms of security. Conversely, decentralized approaches are more likely to gain acceptance from the individuals but do not provide any – privacy preserving – solution to perform distributed computation (exporting their data on a central server to perform the computation would obviously hurt the decentralization principle).

Decentralizing the processing implies to temporarily transfer personal data among participants, transforming each into a vulnerability point. Two guarantees must then be provided: (i) data confidentiality, i.e., any PDMS owner cannot access the data in transit of other participants, and (ii) distributed computation integrity, i.e., any participant's PDMS can attest that any result it supplies corresponds to the assigned computation.

Our proposed TPDMS solution falls in the Home cloud box family, with the salient difference that the box now provides tamper-resistant defenses against attacks. Indeed, compared to a regular Home cloud box, a TPDMS provides means to securely execute external code in the box, opening the door to the design of secure distributed computation protocols, in the line of the Manifest-based framework.

7.2 Secure Database Computations Alternatives

A large majority of works on secure database computations address the case of outsourced databases where honest-but-curious cloud services manage large sets of sensitive data. We position our contribution relatively to these works, and then analyze the main approaches to address secure distributed databases computations, including works relying on TEEs.

Secure database outsourcing. Several works focus on protecting outsourced databases with encryption, either by using onion encryption [30] or by exploiting homomorphic encryption [11,19]. However, most of the existing encryption schemes applied to databases have been shown vulnerable to inference attacks [10]. Going further induce fully homomorphic encryption with intractable performance issues. In any case, these solutions hurt the decentralized assumption of our work. Data can also be protected thanks to differentially private principles [14], again usually performed by an honest-but-curious central server. These principles apply only to specific problems and deliver imprecise results, hurting our genericity objective.

Multi-Party Computations (MPC). MPC allows n users to perform computations involving their inputs without learning anything more than the final result. However, MPC assumes honest-but-curious users while the participants in our context can be active attackers (e.g., may attempt corrupting processing, replay messages, execute alternative code, etc.). Second, the cryptographic techniques involved in MPC either do not scale in the number of participants for performance reasons or can solve only specific problems. Typically, MPC adaptations to distributed databases contexts, like SMCQL [9,13], either support only few tens of participants or are limited to specific database operations.

Secure distributed database computations schemes. Several works suggest distributed computation schemes providing anonymous data exchanges and confidential processing mixing gossip-style protocols, encryption and differential privacy. Gossip-based protocols, such as [2], allow to work on fragmented clear-text data exchanged among nodes and, when communication may reveal data content, noise is added to provide differentially private communication patterns. Gossip protocols scale well but are not generic in terms of possible computations. Moreover, they consider an honest-but-curious threat model.

Hardware-based database computing. Some works [5,8] deploy secure hardware at database server side. They basically split the query processing in one part executed directly on the encrypted data and the other executed inside the secure hardware on clear-text data and make the processing oblivious to prevent adversary learning anything from the data access pattern. These solutions are centralized by nature and do not match our context. Decentralized processing solutions based on secure hardware have also been proposed for aggregate queries [36] or participatory sensing [34] but do not match our genericity objective.

SGX-based data computing. To the best of our knowledge, all works regarding executing data oriented task using SGX have a unique controller (e.g, [15, 29, 33]), as opposed to our setting where no unique individual is supposed to be in control of the computation. Additionally, most of the time this controller also provides the data to be computed on. This greatly simplifies the problem as a same controller verifies all enclaves and organizes the computation. Additionally, various works [17, 31] focus on performing data operations, from indexing to a whole DBMS, within SGX. These works are not directly related to ours as they are all in a centralized setting. The work closest to ours is Ryoan [22] but the techniques for organizing the computation are fundamentally distinct from ours.

8 Conclusion

The concept of TPDMS combined with a Manifest-based framework leverages the security properties provided by TEE to build a comprehensive personal data management solution. This solution reconciles the privacy preservation expected by individuals with the ability to perform collective computations of prime societal interest. We expect that such solution could pave the way to new research works and industrial initiatives tackling the privacy-preserving distributed computing challenge with a new vision.

References

1. Abiteboul, S., André, B., Kaplan, D.: Managing your digital life. CACM,58(5) (2015).
2. Allard, T., Hébrail, G., Pacitti, E., Masseglia, F.: Chiaroscuro: Transparency and privacy for massive personal time-series clustering. In: ACM SIGMOD, (2015).
3. Allard, T., Anciaux, N., Bouganim, L., Yanli, G., Le Folgoc, L., Nguyen, B., Pucheral, P., Ray, I., Yin, S.: Secure Personal Data Servers: a vision paper. In: VLDB, (2010).
4. Anciaux, N., Bonnet, P., Bouganim, L., Nguyen, et al.: Personal Data Management Systems: The Security and Functionality Standpoint. Information Systems, 80, (2019).
5. Arasu, A., Kaushik, R.. Oblivious query processing. *arXiv:1312.4012* (2013).
6. Armstrong, J., Zhu, M., Hirdes, J., Stolee, P.: K-means cluster analysis of rehabilitation service users in the home health care system of Ontario: Examining the heterogeneity of a complex geriatric population. Arch. of physical medicine and rehab., 93(12) (2012).
7. Backes, M., Druschel, P., Haeberlen, A., Unruh, D.: A Practical and Provable Technique to Make Randomized Systems Accountable. In: NDSS, 9 (2009).
8. Bajaj, S., Sion, R.. Trusteddb: A trusted hardware-based database with privacy and data confidentiality. *IEEE Trans Knowl Data Eng, 26*(3), 752-765 (2013).
9. Bater, J., Elliott, G., Eggen, C., Rogers, J.: SMCQL: secure query processing for private data networks. PVLDB, 10(6) (2017).

10. Bindschaedler, V., Grubbs, P., Cash, D., Ristenpart, T., Shmatikov, V.. The tao of inference in privacy-protected databases. *Proceedings of the VLDB Endowment*, *11*(11), 1715-1728 (2018).
11. Boneh, D., Gentry, C., Halevi, S., Wang, F., Wu, D. J.. Private database queries using somewhat homomorphic encryption. In *International Conference on Applied Cryptography and Network Security* (pp. 102-118). Springer, Berlin, Heidelberg (2013).
12. Chaudhry, A., Crowcroft, J., Howard, H., Haddadi, H., Howard, H., Madhavapeddy, A., Mortier, R.: Personal data: thinking inside the box. In: Critical Alternatives (2015).
13. Damgård, I., Keller, M., Larraia, E., Pastro, et al..: Practical covertly secure MPC for dishonest majority - or: Breaking the SPDZ limits. In: ESORICS (2013).
14. Dwork, C.: Differential privacy. In: ICALP (2006).
15. Dinh, T. T. A., Saxena, P., Chang, E. C., Ooi, B. C., Zhang, C.. M2R: Enabling stronger privacy in MapReduce computation. In *24th {USENIX} Security Symposium ({USENIX} Security 15)* (pp. 447-462) (2015).
16. Elbattah, M., Molloy, O.: Clustering-Aided Approach for Predicting Patient Outcomes with Application to Elderly Healthcare in Ireland. In: Workshops at AAAI (2017).
17. Fuhry, B., Bahmani, R., Brasser, F., Hahn, F., Kerschbaum, F., Sadeghi, A. R.. Hardidx: Practical and secure index with SGX in a malicious environment. *Journal of Computer Security*, *26*(5), 677-706 (2018).
18. General Data Protection Regulation. (2016). https://gdpr-info.eu/. Accessed May, 2020
19. Ge, T., Zdonik, S.: Answering Aggregation Queries in a Secure Model. In: VLDB (2007).
20. Hiller, J., Pennekamp, J., Dahlmanns, M., Henze, M., Panchenko, A., Wehrle, K.: Tailoring onion routing to the Internet of Things: Security and privacy in untrusted environments. In: IEEE 27th International Conference on Network Protocols ICNP (2019).
21. Huang, Z.: Extensions to the k-means Algorithm for Clustering Large Data Sets with Categorical Values, pp 283–304. Data Mining and Knowledge Discovery (1998).
22. Hunt, T., Zhu, Z., Xu, Y., Peter, S., Witchel, E. Ryoan: A distributed sandbox for untrusted computation on secret data. *TOCS*, *35*(4), 1-32 (2018).
23. Johnson, S., Bacsu, T., Jeffery, B., Novik, N.: No Place Like Home: A Systematic Review of Home Care for Older Adults. Canadian Journal on Aging, 37(4) (2018).
24. Ladjel, R., Anciaux, N, Pucheral, P., Scerri, G.: A manifest-based framework for organizing the management of personal data at the edge of the network. In: ISD (2019).
25. Ladjel, R., Anciaux, N, Pucheral, P., Scerri, G.: Trustworthy Distributed Computations on Personal Data Using Trusted Execution Environments. In: TrustCom (2019).
26. Liao, M., Li, Y, Kianifard, F, Obi, Z, Arcona, S.: Cluster analysis and its application to healthcare claims data: a study of end-stage renal disease patients who initiated hemodialysis. BMC Nephrology (2016).
27. Merkle, C.: Protocols for public key cryptosystems. In: S&P (1980).
28. De Montjoye, Y., Shmueli, E., Wang, S., Pentland, A.: OpenPDS: Protecting the privacy of metadata through SafeAnswers. PloS one, 9(7) (2014).
29. Pires, R., Gavril, D., Felber, P., Onica, E., Pasin, M. A lightweight MapReduce framework for secure processing with SGX. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)* (pp. 1100-1107). IEEE (2017).
30. Popa, R. A., Redfield, C. M., Zeldovich, N., Balakrishnan, H. CryptDB: processing queries on an encrypted database. *Communications of the ACM*, *55*(9), 103-111 (2012).
31. Priebe, C., Vaswani, K., Costa, M. Enclavedb: A secure database using SGX. In *2018 IEEE Symposium on Security and Privacy (SP)* (pp. 264-278). IEEE (2018).
32. Sabt, M., Achemlal, M., Bouabdallah, A.: Trusted Execution Environment: What it is, and what it is not. In: TrustCom/BigDataSE/ISPA (1) (2015).

33. Schuster, F., Costa, M., Fournet, C., Gkantsidis, C., Peinado, M., Mainar-Ruiz, G., Russinovich, M. VC3: Trustworthy data analytics in the cloud using SGX. In *2015 IEEE Symposium on Security and Privacy* (pp. 38-54). IEEE (2015).
34. That, D. H. T., Popa, I. S., Zeitouni, K., Borcea, C. PAMPAS: privacy-aware mobile participatory sensing using secure probes. In *Proceedings of the 28th International Conference on Scientific and Statistical Database Management* (pp. 1-12) (2016).
35. Tramèr, F., Zhang, F., Lin, H., Hubaux J., Juels, A., Shi, E.: Sealed-glass proofs: Using transparent enclaves to prove and sell knowledge. In: EuroS&P (2017).
36. To, Q. C., Nguyen, B., & Pucheral, P.. Privacy-Preserving Query Execution using a Decentralized Architecture and Tamper Resistant Hardware. In *EDBT* (pp. 487-498) (2014).
37. Wang, W., Chen, G., Pan, X., Zhang, Y., Wang, X., Tang, H., Gunter, A.: Leaky cauldron on the dark land: Understanding memory side-channel hazards in SGX. In: CCS (2017)
38. www.inrupt.com/blog/one-small-step-for-the-web, Sept. (2018). Accessed May, 2020.