Consent-driven Data Reuse in Multi-tasking Crowdsensing Systems: A Privacy-by-Design Solution

Mariem Brahem, Nicolas Anciaux, Valérie Issarny, Guillaume Scerri

Pervasive and Mobile Computing 83 (2022) 101614

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

Consent-driven Data Reuse in Multi-tasking Crowdsensing Systems: A Privacy-by-Design Solution

Mariem Brahem [a,b,*], Guillaume Scerri [a,b], Nicolas Anciaux [a,b], Valerie Issarny [c]

[a] Petrus Team, Inria SIF, 1 rue Honoré d'Estienne d'Orves, Palaiseau, 91120, France
[b] University of Versailles Saint-Quentin-en-Yvelines, 55 Avenue de Paris, Versailles, 78000, France
[c] MiMove Team, Inria Paris, 2 rue Simone Iff, Paris, 75012, France

ARTICLE INFO

Article history:
Received 16 August 2021
Received in revised form 7 February 2022
Accepted 10 May 2022
Available online 16 May 2022

Keywords:
Mobile crowdsensing
Privacy
Consent
Security
Trusted execution environment
SGX

ABSTRACT

Mobile crowdsensing allows gathering massive data across time and space to feed our environmental knowledge, and to link such knowledge to user behavior. However, a major challenge facing mobile crowdsensing is to guarantee privacy preservation to the contributing users. Privacy preservation in crowdsensing systems has led to two main approaches, sometimes combined, which are, respectively, to trade privacy for rewards, and to take advantage of privacy-enhancing technologies "anonymizing" the collected data. Although relevant, we claim that these approaches do not sufficiently take into account the users' own tolerance to the use of the data provided, so that the crowdsensing system guarantees users the expected level of confidentiality as well as fosters the use of crowdsensing data for different tasks. To this end, we leverage the ℓ-*Completeness* property, which ensures that the data provided can be used for all the tasks to which their owners consent as long as they are analyzed with $\ell - 1$ other sources, and that no privacy violations can occur due to the related contribution of users with less stringent privacy requirements. The challenge, therefore, is to ensure ℓ-*Completeness* when analyzing the data while allowing the data to be used for as many tasks as possible, and promoting the accuracy of the resulting knowledge. This is achieved through a clustering algorithm sensitive to the data distribution, which optimizes data reuse and utility. Nevertheless, it is critical to allow the deployment of such a solution even in the presence of a malicious adversary able to act on the server side, for which we introduce a privacy-by-design architecture leveraging Trusted Execution Environments. The implementation of a prototype using SGX enclaves further allows running experiments that show that our system incurs a reasonable performance overhead, while providing strong security properties against a malicious adversary.

1. Introduction

Mobile crowdsensing is an essential element of the Internet Of Things (IoT[1]) as it allows gathering tremendous data across time and space at low cost [1]. Indeed, thanks to the democratization of smartphones that embed, and/or connect to, increasingly rich sensing capabilities, we are able to sense a large portion of the physical environment and further relate the observed phenomena with human behavior. Various applications illustrate the benefit of mobile crowdsensing toward

* Corresponding author at: Petrus Team, Inria SIF, 1 rue Honoré d'Estienne d'Orves, Palaiseau, 91120, France.
E-mail address: mariem.brahem@uvsq.fr (M. Brahem).

[1] See Appendix A for the list of acronyms and abbreviations.

https://doi.org/10.1016/j.pmcj.2022.101614
1574-1192/

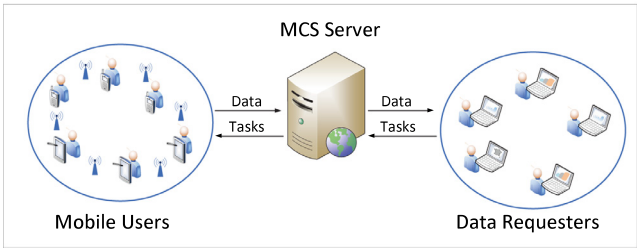


Fig. 1. Multi-tasking crowdsensing system architecture.

better informing and enhancing, e.g., environmental monitoring and awareness [2], public health monitoring and policy [3] or traffic management [4]. Still, mobile crowdsensing comes with tremendous challenges for it to be widely adopted and to effectively feed today's AI-powered systems. Challenges span: the ability to embark a sufficiently large crowd to gather the required spatio-temporal knowledge [5]; ensuring the quality of data through supporting trust management [6], context-awareness and filtering [7]; the resource-efficiency of the overall process from the data collection up to its overall aggregation [8]; and, last but not least, enforcing privacy [9].

This paper specifically focuses on the challenge of privacy-preserving mobile crowdsensing since it is the most critical one for crowdsensing to be a powerful technology that both brings valuable knowledge and serves the public good. The various dimensions of the challenge together with supporting solutions have been the focus of several surveys among which: [9–12]. Relevant studies include leveraging state of the art Privacy-Enhancing Technologies (PET) to enforce related privacy metrics [13,14]. For instance, the early work in [15] leverages decentralization together with spatial *k*-anonymity for privacy-aware task assignment. More recently, the work in [16] introduces a mechanism based on differential-privacy and distortion-privacy to guarantee that the gathering of location-based measurements is not at the expense of location privacy for the contributing users, while reducing the resulting loss of data quality. Still, leveraging PET results in the obfuscation of the crowdsensed data and thus impacts the significance of the knowledge that may be analyzed. To overcome the loss of knowledge accuracy, a significant body of research concentrates on dealing with the tension between accuracy and privacy [17]. Proposed solutions include privacy-aware auction-based approaches so that the contributors get rewarded for the loss of privacy [18–20]. In a nutshell, the more users accept to provide close-to-actual observations, the more the crowdsensing system gathers accurate knowledge and contributors get rewarded. Other approaches leverage decentralization for managing the information about the contributing users [21]. However, they focus on privacy-preservation at the time of task assignment, and do not address the complementary issue of privacy-preserving data collection at the server.

Overall, the state of the art of privacy-preserving mobile crowdsensing provides a number of advanced protocols that may be combined toward enforcing some level of privacy. And, whatever is the crowdsensing protocol implemented, it comes with a necessary trade-offs between knowledge accuracy and privacy guarantees. The challenge is then for the crowdsensing system to get the best out of the contributed data. Part of the solution lies in the elicitation of application-specific data analyses to reduce the loss of accuracy [22]. However, we argue that it is as important to foster the re-use of data across tasks, as also advocated by the IoT data marketplace trend [23]. Indeed, this results in the enhanced resource-efficiency of mobile crowdsensing. This is known as multi-tasking in crowdsensing sensing, for which recent studies focus on optimizing the allocation of tasks from the perspective of the task organizer [24], resp. task participant [25]. Our work distinguishes itself from, and complements, related research by concentrating on *fostering privacy-preserving data reuse* in multi-tasking, mobile crowdsensing systems. We further focus on participatory sensing where users explicitly register for possible participation to tasks although our approach could easily be adapted to opportunistic sensing. Fig. 1 illustrates the *Multi-tasking Crowdsensing System* (MCS) architecture we consider in our work: *data requesters* submit tasks to the *MCS server* so that the server gathers and analyzes the specified data using the provided tasks; *mobile users* register for the advertised tasks to the server to contribute relevant data. In this context, our goal is to ensure secure privacy-preserving collection and analysis of crowdsensed data at the MCS server. We refer the readers to complementary work for what concerns privacy-preserving interactions (e.g., [26,27]) and dealing with the trustworthiness of mobile users and data requesters (e.g., [26,28–30]).

The research question that the paper addresses is: "*How to foster the reuse of crowdsensed data across various eligible tasks while still guaranteeing the right level of privacy and security to the contributing users at the MCS server*?". A first design choice that we make is to address*privacy preservation according to the user's consent to the use of their data*. Consent is one of the legal frameworks set in many places (e.g., GDPR [31], CCPA [32]) as a precondition for any processing of personal data. In this context, *informed and specific* consent requires that users be informed of the type of data collected, the identity of the recipient of the results, and the precise nature and purpose of the task. In particular, these legal terms are intended to act as a safeguard against function creep and *data reuse*.[2] Thus, personal data collected for one function on the

[2] According to European Commission [33]: "*If your company/organisation has collected the data on the basis of consent (...) no further processing beyond what is covered by the original consent or the provisions of the law is possible. Further processing would require obtaining new consent*".

basis of consent cannot be extended or reused for another function without obtaining consent for that function. Defects in consent are considered a cause of nullity in many countries (e.g., see [34] for European countries such as France, Germany or UK). As a result, digital consent has become an integral part of privacy management in computing platforms [35]. This leads us to build upon the ℓ-*Completeness* property for mobile crowdsensing platforms, which we initially introduced in [36]. ℓ-*Completeness* defines the extent to which mobile users consent to the reuse of the contributed observations, that is, the property enforces that: (1) The user's data are analyzed with at least $\ell - 1$ other data sources in all the tasks the data contribute to, and (2) No individual knowledge may be inferred due to the participation to many tasks that may not involve the same contributors. The proposed consent-driven property directly derives from the properties associated with sample size determination [37] in relation with privacy [38]. One challenge facing the enforcement of ℓ-*Completeness* is to analyze the users' contributed data in as many permissible and relevant tasks as possible while guaranteeing ℓ-*Completeness*. This challenge is overcome with a distribution-sensitive clustering algorithm to optimize both data reuse and utility (*aka* knowledge accuracy) [36]. Still, to avoid defect in consent under ℓ-*Completeness*, and in accordance with the principles of *privacy-by-design* enacted in global privacy laws (e.g., Article 5 of the GDPR [31] in the EU) and recommended privacy practices (e.g., by the Federal Trade Commission [39] in the US), the server should: (i) Never expose personal information, i.e., raw contribution values or information derived from these values, and (ii) Enforce ℓ-Completeness even in the presence of attacks. A second challenge is therefore to enforce these guarantees by default on the server side.

After an overview of the consent-driven ℓ-*Completeness* approach, from formal definition to supporting algorithms [36] (Section 2), the paper makes the following contributions:

- Privacy-by-design architecture of the server implementing the proposed consent-driven, privacy-preserving data use in MCS; the architecture specifically leverages trusted execution environments so as to ensure that the task assignment as well as the collection and multi-task analyses of the crowdsensed data are both secure and privacy-preserving (Section 3).
- Prototype implementation of the MCS server that builds upon SGX enclaves [40] and the SCONE secure container environment [41] (Section 4).
- Experiment-based evaluation showing that our solution incurs a reasonable performance overhead when adding security to the consent-driven approach to privacy preservation in MCS (Section 4).

Finally, we position our contribution with respect to related work (Section 5) and offer conclusion (Section 6).

2. Consent-driven ℓ-completeness for privacy-preserving data reuse

2.1. Consent-driven privacy preservation in multi-tasking crowdsensing

Focusing on multi-tasking crowdsensing systems, consent-driven participation allows users to specify the tasks to which they consent to contribute with mobile observations according to their privacy requirements. Without loss of generality, we consider that a task submitted to the server for execution is defined by[3]: the function f (code) applied to the collected data of the specified type $\mathcal{S}$, the time period Δ during which each participant contributes observations, a minimal number ℓ of participants required to provide contributions to execute f.[4] The value of ℓ is deemed critical to both: (i) obtain a useful result (e.g., the evaluation of the noise level in a street requires the analysis of several contributions [42]), and (ii) protect the privacy of the participants as their individual contributions get aggregated with the ones of the $\ell - 1$ others. Typically, national agencies and data research centers impose a minimum number of individuals to be taken into account [38] when producing any aggregate – table, graph or map based on aggregate values – for research purposes. For instance, the INSEE confidentiality guide [43] and CASD rules [44] impose a minimum of 11 individuals for any computation based on tax data or 5 individuals for social data.

The value of ℓ (resp. Δ) is task-dependent; hence, for the sake of simplicity but without loss of generality, ℓ (resp. Δ) is aggregated as the maximum of the ℓs (resp. Δs) of all the tasks. Following, each task is associated with a *Manifest* that summarizes how the contributed data is consumed and thereby allows users to provide an informed and specific consent for the task. The Manifest is specified using a dedicated language such as, e.g., the AnonyTL language [45] introduced by the AnonySense privacy-aware system for opportunistic sensing [46]. We specifically assume the following specification for a task and its Manifest:

Definition 1 (*Task and Associated Manifest*)**.** A task T is defined as a tuple $\langle a, f, \mathcal{S}, \Delta, \ell \rangle$ such that a is the sensing function (code) to be executed on the mobile client,[5] f is a function (code) executed on an input set with non empty contributions of type $\mathcal{S}$ produced by at least ℓ consenting participants over a time period Δ. We denote by O_T the result of T. The manifest $M(T)$ is a declarative and intelligible representation of T regarding the consumption of the gathered data so that users may provide an informed and specific consent for it.

[3] See Appendix B for the list of symbols.

[4] Additional elements may characterize the data consumption by the task — e.g., the frequency of execution or the retention period of the data. Considering such parameters is area for future work.

[5] We recall that we focus primarily on the secure privacy-preserving treatment of contributed observations on the MCS server and thus do not elaborate on a in what follows.

Table 1
Consent table $\mathcal{C}$.

User	b_1	b_2	b_3
u_1	1	0	0
u_2	1	1	0
u_3	1	1	1
u_4	1	1	1
u_5	1	1	0
u_6	1	1	1
u_7	1	1	1
u_8	1	1	0
u_9	1	1	1
u_{10}	1	1	1

Given a set of m tasks $\mathcal{T} = \{T_i\}_{0<i\leq m}$ and associated manifests, managed and advertised by the multi-tasking crowdsensing system, any user can consent to a desired subset of $\mathcal{T}$. We denote the consents of a user u with the tuple $C_u = \langle b_1, \ldots, b_m\rangle$ where $b_i = 1$ if the user consents to T_i and $b_i = 0$ otherwise. By giving consents, a user accepts contributing to (only) the set of results $\{O_{T_i} = T_i.f(s_{U_i})\}_{T_i\in\mathcal{T}|C_u.b_i=1}$, with s_{U_i} being the related contributions of type $T_i.\mathcal{S}$ from a set U_i of users consenting to T_i that includes u.

Combining the contributions of users who tolerate different disclosure policies – as defined from their consents to multiple tasks – creates the risk of unintended secondary uses and may result in *defects in consent*. Consider the knowledge $\mathcal{K}$ that the crowdsensing system may disclose from the set of tasks $\mathcal{T}$, provided the table $\mathcal{C} = \{\langle u, C_u\rangle\}_{u\in\mathcal{U}}$ of consents of the set of registered users $\mathcal{U}$ to the tasks (e.g., see Table 1), i.e., $\mathcal{K} = \{O_{T_i}\}_{T_i\in\mathcal{T}|\mathcal{C}.b_i}$. The system must guarantee that it conforms to $\mathcal{C}$ while computing and delivering $\mathcal{K}$. However, we claim that meeting this constraint only is not sufficient to guarantee that there is no defect in consent. For example, consider the consents of 10 users to 3 tasks represented in Table 1. All 10 users consent to T_1 (e.g., a task computing the average noise at a given location to request the city government to take appropriate measures to reduce the nuisance). As for T_2, all the users but User u_1 consent to a more privacy-invasive task (e.g., displaying the noise measurements across the users' journeys). Assigning tasks to users based only on their respective task consents (i.e., the system assigns T_1, resp. T_2, to all 10, resp. 9, users) results in a defect in consent. Indeed, although User u_1 does not consent to reveal the detailed noise observations they contribute to, the specific observations may be inferred from composing O_{T_1} and O_{T_2}.

The consent-driven task assignment must account for the disclosure policies of the users across all the tasks they each contribute to, to avoid a defect in consent. In other words, the multi-task assignment must be achieved in such a way that the data gathered by a task T_1 cannot be analyzed together with data also gathered by a task T_2 with a weaker disclosure policy. A simple solution to the above issue would consist in assigning tasks according to either the least or the greatest, common disclosure policy of the eligible users. Going back to our example, this means assigning task T_1 to the participants consenting to either T_1 only, or both T_1 and T_2. However, this would lead to sub-optimal data reuse, with a contribution loss for certain tasks and a resulting reduced utility. A research question that arises is then: *How to avoid by design the defects in consent in a multi-tasking crowdsensing system, while ensuring efficient data reuse in as many eligible tasks as possible?*

2.2. Consent-driven ℓ-completeness

Still consider the set of tasks, $\mathcal{T} = \{T_i\}_{0<i\leq m}$ with any $T_i = \langle a_i, f_i, \mathcal{S}_i, \Delta, \ell\rangle$. We recall that the values of ℓ and Δ are identical in all tasks for simplicity, but without loss of generality (see Section 2.1). It is direct to infer that a multi-task assignment does not create any defect in consent if for any pair of distinct tasks T_i and T_j, their respective outputs $O_{T_i} = f_i(S_{U_i})$ and $O_{T_j} = f_j(S_{U_j})$ are processed over the respective input sets S_{U_i} ($|S_{U_i}| \geq \ell$) and S_{U_j} ($|S_{U_j}| \geq \ell$) such that either $S_{U_i} = S_{U_j}$ (i.e., they analyze the very same set of contributions from the same consenting users) or $S_{U_i} \cap S_{U_j} = \emptyset$ (i.e., they analyze contributions from distinct sets of users). However, such a task assignment is too restrictive. For instance, consider a set of users U_1 (resp. U_2) consenting (only) to a task T_1 (resp. T_2), and a (disjoint) set of users U_{12} consenting to tasks T_1 and T_2 (see Fig. 2). If T_1 uses all usable contributions (i.e., the result $O_{T_1} = f_1(S_{U_1} \cup S_{U_{12}})$ is produced) then T_2 cannot use any contribution in $S_{U_{12}}$ produced by any user in U_{12} (i.e., only result $O_{T_2} = f_2(S_{U_2})$ can be produced, but not $O_{T_2} = f_2(S_{U_{12}})$ neither $O_{T_2} = f_2(S_{U_2} \cup S_{U_{12}})$).

To enable a better data reuse in practice, we introduce the following definition of ℓ-Completeness.

Definition 2 (*Practical ℓ-Completeness Over Δ*)**.** Let a set of m tasks $\mathcal{T} = \{T_1, T_2, \ldots, T_m\}$ and a set of users $\mathcal{U}$ who consent to (a desired subset of) these tasks and contribute input data for these tasks over a period of time Δ.

Let a family of computation input sets $(S_{U_i})_{i\leq m}$, where U_i is the set of users consenting to $T_i \in \mathcal{T}$ whose contribution is used in $(S_{U_i})_{i\leq m}$ to evaluate tasks $T_i \in \mathcal{T}$.

Let $F(\mathcal{X}) = \{X \cap X' | X, X' \in \mathcal{X}\} \bigcup \{X \setminus X' | X, X' \in \mathcal{X}\} \bigcup \mathcal{X}$.

Let $\mathcal{F} = \bigcup_{k\in\mathbb{N}} F^k(\{U_i\}_{i\leq m})$ be the least fixed point of F containing $\{U_i\}_{i\leq m}$.

The family of computation input sets $(S_{U_i})_{i\leq m}$ is ℓ-Complete over Δ iff for all $X \in \mathcal{F}$ either $X = \emptyset$ or $|X| \geq \ell$.

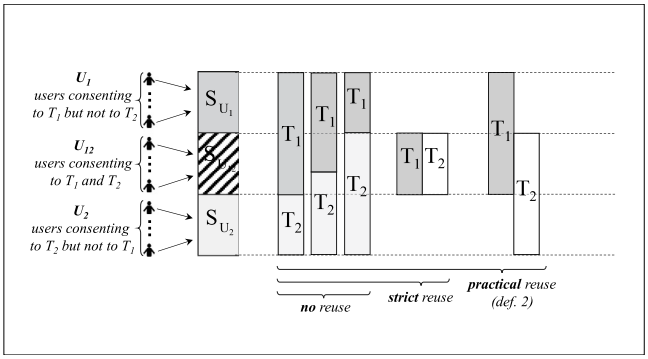


Fig. 2. Strict vs. practical ℓ-Complete data reuse schemes.

In addition to the *no reuse* and *strict reuse* schemes, Definition 2 allows for further data reuse, while still prohibiting any defect in consent (see *practical reuse* in Fig. 2). Reuse is enabled as long as applying any combination of the intersection and difference over the respective users' contributed data consumed by different tasks does not produce information on a set of less than ℓ users' contributions. In particular, Definition 2 allows evaluating both $O_{T1} = f_1(S_{U_1} \cup S_{U_{12}})$ and $O_{T_2} = f_2(S_{U_2} \cup S_{U_{12}})$ when the conditions of practical ℓ-Completeness are met. The following sections introduce concrete techniques for data reuse, which conform by design to Definition 2.

2.3. ℓ-Completeness as a clustering problem

The key idea underlying the elicitation of groups of ℓ users that foster consent-driven data reuse across tasks according to Definition 2 is to consider the ℓ-Completeness problem as a clustering problem. The objective is then to create a set of k clusters $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ of the consent table $\mathcal{C}$ (see Table 1) in such a way that the tuples (i.e., the consent policies of the embedded users) in the same cluster are as similar to each other as possible, thus resulting in a minimum contribution loss. That is, we want to minimize the sum of all the intra-cluster distances (maximum distance between any two data points), which is defined as:

$$d = \sum_{h=1\ldots k} \max_{i,j=1,\ldots,|P_h|} distance(x_{h,i}, x_{h,j}) \tag{1}$$

where: $x_{h,i}$ denotes the ith data point (in our case, a consent policy of the form $\langle b_1, \ldots, b_m \rangle$) in the cluster P_h of the consent table $\mathcal{C}$, and *distance(x,y)* is the Euclidean distance between two data points x and y, and serves characterizing the dissimilarity (in terms of common consents) between users. It follows that we aim at eliciting k subsets of users, P_i ($1 \leq i \leq k$), such that:

- $\forall i \neq j \in \{1\ldots k\} : P_i \cap P_j = \emptyset$
- $\forall i \in \{1\ldots k\} : |P_i| \geq \ell$
- The distance d (defined in Eq. (1)) is minimized.

Following, we group users based on the similarity of their consent policies, and assign them the tasks that match the consent policies of all, while discarding the others. Hence, this allows users with singular policies to contribute. Nevertheless, we also need to ensure that the assignment of data to tasks conforms to the profile of data distribution for each task. The consent-driven multi-task assignment is then implemented in three phases [36], as depicted in Fig. 3 and detailed next.

2.4. The learning and optimization phases

The *learning* phase allows evaluating the profile of data distribution for each task. It runs over a time period $\Delta' < \Delta$ – which is considered representative – to generate, for each task T, the *reference profile* φ_0^T of the data distribution associated with the task. For example, for tasks related to urban monitoring, the typical value for Δ' would be a week as we observe weekly repetitive patterns. We do not detail the computation of the reference profiles, which are task dependent. We simply highlight that the learning phase does comply with the users' consent policies [36].

The *optimization* phase then leverages the set of reference profiles $\{\varphi_0^T\}_{T \in \mathcal{T}}$ to optimize the users contributions to be considered for each task for time period Δ. For each task T, we compute the profile φ_Δ^T as the union of the contributions of all the users consenting to T over Δ. We then compare φ_Δ^T with T's reference profile φ_0^T; if the two profiles diverge, we mark the contributions of some users as *'ignored'* for this task (i.e., the corresponding value b_i in the consent table is updated from 1 to 0) until φ_Δ^T gets close enough to φ_0^T. The way users' contributions are selected is task/profile dependent

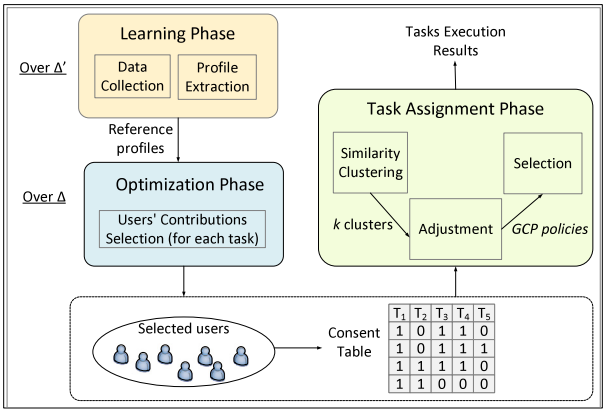


Fig. 3. Three-phase consent-driven multi-task assignment.

(e.g., if the profile is a distribution or an histogram, the contributions of users with over represented values may be 'ignored') and is not further discussed. Note that the contributions of users that get 'ignored' for one task can still be considered for any other – consented – task. The optimization phase allows "ignoring" subsets of users contributions that would otherwise negatively impact the data utility, while conforming to the users' privacy and consent requirements through the computation of ℓ-Complete clusters.

The *optimization* phase allows to construct reference profiles representing the global distribution of data. Such approach is efficient for several types of tasks where the distribution of the data can be represented by a reference profile. However, for some where the collected data have a different distribution than the reference profiles, this approach could result in a loss of information, for example, tasks towards anomaly/drift detection, where information related to anomalies or rare events are not represented in the reference profiles.

2.5. The task assignment phase

The *task assignment* phase takes as input the consent table updated by the optimization phase, and proceeds in 3 stages (see Fig. 3): (1) *Clustering* that assigns users to their respective clusters while maximizing the number of tasks to be executed by each user; (2) *Adjustment* to deal with clusters that contain less than ℓ users and further enforce the single greatest common consent policy, and (3) *Selection* so that the task assignment to clusters maximizes data reuse.

2.5.1. Clustering stage

The ℓ-Completeness problem does not have a constraint on the number of clusters; however, it requires that each cluster contains at least ℓ users. Thus, we pose the ℓ-Completeness problem as a clustering problem that derives from the traditional k-means algorithm. We specifically choose k-means because it is one of the most widely used algorithms for clustering due to its simplicity and efficiency with a low computational overhead [47]. Still, to overcome the limits of the standard k-means algorithm for choosing the k initial random centers, we use the combined k-means^{++} that allows achieving better accuracy by choosing the starting centers based on the weights of the data points according to their squared distance from the closest center already chosen [48]. That is, we use k-means^{++} to seed the initial centers for k-means in such a way that they are as far apart from each other as possible.

This results in Algorithm 1 that proceeds as follows: Let $\mathcal{C}$ be the consent table and $k = \left\lfloor \frac{n}{\ell} \right\rfloor$ such that n is the number of users in table $\mathcal{C}$ and ℓ is the value set for the ℓ-Completeness. The clustering stage starts by selecting k tuples to build k clusters using k-means^{++}. The idea of k-means^{++} is to choose a random consent c_1 from $\mathcal{C}$, calculate the distance from each data point to the closest center we have already chosen, sample a point with a probability proportional to the square of the distance already calculated and repeat the previous two steps until k centroids are selected. Once the initial k centers are chosen, the k-means algorithm is applied: for each tuple c in the consent table $\mathcal{C}$, the algorithm finds the cluster P_i with the closest centroid to c. Then, we add c to its closest cluster and subsequently update the centroid of the clusters.

2.5.2. Adjustment stage

The above clustering algorithm returns a set of clusters out of which some may contain less than ℓ users. Furthermore, it is unlikely that the consent policies of all the users in a given cluster are identical, which requires defining the *Greatest Common Policy* (*GCP*) for the cluster. The GCP of the cluster characterizes the allowable set of tasks so as to avoid any defect in consent (see Section 2.1). Note that GCP does not depend on the task semantics, but only on the consent table.

Algorithm 1 Similarity clustering

Input: A consent table $\mathcal{C}$, the value ℓ for ℓ-Completeness
Output: A set of clusters $\{P_1, ...P_k\}$
Let $k \leftarrow \left\lfloor \frac{n}{\ell} \right\rfloor$
Let $\{P_1, ..., P_k\}$ a set of k empty clusters
Select k distinct tuples $c_1...c_k \in \mathcal{C}$ with k-means^{++}
$\mathcal{X} \leftarrow \{c_1, ..., c_k\}$ ▷ $\mathcal{X}$ contains the initial k centroids
repeat
  for each tuple c $\in \mathcal{C}$ **do**
    Find $c_i \in \mathcal{X}$ closest to c using Equation (1)
    $P_i \leftarrow P_i \cup \{c\}$
  end for
  $\mathcal{X} \leftarrow \{centroid(P_i)\}_{i \leq k}$
until convergence **return** $\{P_1, ..., P_k\}$

Definition 3 (*Greatest Common Policy − GCP*)**.** Given a cluster $P_i \subset \mathcal{C}$, we define the *Greatest Common Policy* within P_i as the tuple $GCP(P_i) = \langle b_1, b_2, \ldots, b_m \rangle$ such that $b_i = 1$ if for any tuple p in P_i we have $p.b_i = 1$, and $b_i = 0$ otherwise. Note that we must ensure that $|P_i| \geq \ell$ to guarantee ℓ-Completeness.

Algorithm 2 introduces the computation of the clustering adjustment. This includes checking the number of users in each cluster. Multiple approaches can be used in case $|P_i| < \ell$: (1) Distribute all the tuples of the cluster to the closest clusters; (2) Merge closest small clusters with P_i until $|P_i| \geq \ell$; (3) Discard Cluster P_i. For efficiency and simplicity, we implement Option 1.

Algorithm 2 Clusters adjustment

Input: $\mathcal{P} = \{P_i\}_{i \leq k}$ a set of k clusters
Output: $\{P_i'\}_{i \leq k'}$ a set of $k' \leq k$ clusters with $|P_i'| \geq \ell$
for each cluster $P_i \in \mathcal{P}$ with $|P_i| < \ell$ **do**
  for each tuple $x \in P_i$ **do**
    Find P_j with $centroid(P_j)$ closest to x and $|P_j| \geq \ell$
    $P_j \leftarrow P_j \cup \{x\}$
    $P_i \leftarrow P_i \setminus \{x\}$
  end for
  $\mathcal{P} \leftarrow \mathcal{P} \setminus \{P_i\}$
end for
  return $\mathcal{P}$

2.5.3. Selection stage

While the optimization phase (Section 2.4) filters out users contributions so as to respect the task profile φ_Δ^T of any task T, Algorithms 1 &2 may still introduce some bias since it leads to use only a subset of the eligible data for the task. Algorithm 3 overcomes such an impact. Given a set of clusters $\mathcal{P}$ created after proportional clustering/adjustment and a set of task profiles $\{\varphi_\Delta^T\}_{T \in \mathcal{T}}$, for each task T, Algorithm 3 starts by merging the set of clusters $P \in \mathcal{P}$ with $T \in GCP(P)$ in a cluster E_T. Then, it computes the profile $\varphi_\Delta^T(E_T)$ on E_T for that task. If the resulting profile is close enough to the profile φ_Δ^T, E_T is considered as the input set for T that may be used to compute task result O_T. Otherwise, we compute the profile of each cluster in E_T, sort them according their distance to φ_Δ^T and we remove those with the highest Earth Mover's Distance (*EMD*) [49] until finding a qualified cluster. More precisely, EMD serves quantifying the difference between data distributions profile in the clusters and data distribution profile φ_Δ^T, where the *EMD* between two distributions R and S is defined as:

$$distance(R, S) = \frac{1}{m-1} \sum_{i=1}^{m-1} |\sum_{i=j}^{i} (s_j - r_i)|$$

As a result, Algorithm 3 selects, for each task, an input set that maximizes data reuse and conforms to the task profile. More details and examples about our algorithms are presented in our recent work [36].

2.6. Compliance with practical ℓ-completeness

The Clustering Algorithm 1 and associated Adjustment Algorithm 2 build a partition of the users' contributions such that each partition is a cluster of size $\geq \ell$ and the intersection of any two partitions is the empty set. The Selection

Algorithm 3 Selection

Input: $\mathcal{P}$: the set of clusters created after clustering/adjustment, $\{\phi_{\Delta}^{T}\}$: the set of task profiles
Output: The set of input sets $\{E_T\}$ qualified for each task

```
for each T ∈ 𝒯 do
    E_T ← ∪{P ∈ 𝒫, T ∈ GCP(P)}
    qualified ← false
    while ! qualified do
        φ_{E_T} ← compute_profile_T(E_T)
        if φ_{E_T} ≃ φ^T_Δ then
            qualified ← true
        else                                   ▷ else remove the worst cluster from E_T
            Worst ← P ∈ E_T s.t. P with the highest
              EMD(compute_profile_T(P), φ^T_Δ)
            E_T = E_T \ Worst
        end if
    end while
end forreturn {E_T}
```

Algorithm 3 potentially rejects certain clusters – as a whole – initially assigned to a task, in case they lead to a too high deviation from the reference data distribution profile for the task. Since each task (re-)uses the union of a given set of clusters, any composition of union, intersection and difference over the sets of users whose contribution is effectively used in the different tasks is obviously either the empty set or a set containing one or more partitions (corresponding to more than ℓ users by construction). Hence, the result complies with practical ℓ-Completeness (see Definition 2) by design.

3. Privacy-by-design architecture

3.1. Security model

As detailed previously, the central server of our MCS architecture (see Fig. 1): Collects the consents of the participants to the recorded tasks as well as their contributions; Executes the tasks using the contributions – in accordance with consents –; and Delivers the results to the corresponding requesters. To avoid defects in consents and thereby guarantee consent-driven privacy-preservation, the server should: (i) Never expose personal information, i.e., raw contribution values or information derived from these values, and (ii) Enforce the ℓ-Completeness property, even in the presence of attacks. We first introduce the security assumption we make for the implementation of an MCS server that complies to these two requirements, hence adhering to privacy-by-design principles.

In our previous work [36], the server is assumed *fully trusted*. This means that the entire server software stack – including the OS, hypervisors and applications – is: fully protected from the outside, bug free, without security vulnerability, and maintained by fully honest administrators and employees never causing any negligence nor error. Under this assumption, the only information accessible to attackers is that remaining outside of this fully trusted environment: tasks results exposed to the requesters, and data exchanged between the server, users and requesters. Task results never reveal information the users did not consent to, provided the ℓ-Completeness algorithms and task processing are properly implemented on the server side. Furthermore, data exchanges can be protected using existing anonymous communication and cryptographic techniques (e.g., mix networks [50]). However, in practice, although appropriate legal and contractual clauses may be adopted to protect the stakeholders in the event of non-compliant data disclosure or data use, it is impossible to fully prevent data breaches due to negligence or malicious actors as shown by numerous recent examples (e.g., Equifax[6]). Hence, we need to alleviate the strong trust assumption we place in the server in [36]. Two main security models can then be considered:

1. *Honest-but-curious (HBC) adversary* (see e.g., [51]). Under honest-but-curious (*aka* semi-honest or passive) security assumptions, attackers have potential – read – access to all data stored or manipulated on the server side, but cannot actively corrupt code execution or running protocols. This is the weakest assumption apart from complete trust in the server, since the server is assumed to honestly perform all computations. To avoid HBC attacks, all data must be protected against "snooping" and thus never be stored or used in clear on the server side; furthermore, it must be impossible to infer information about the processed data from the computation itself (typically through execution flow).

[6] https://en.wikipedia.org/wiki/2017_Equifax_data_breach.

2. *Malicious adversary*. In addition to observing server-side execution, malicious intruders may also attempt to interfere with execution, resulting in disclosure of data – and insiders may also unintentionally introduce bugs or make mistakes that allow such interference. Such attacks cannot be ignored in the interest of both users (whose consent must be respected) and requesters (which are also expected to avoid manipulating data without users consent, for regulatory compliance reasons). Indeed, any interference with, e.g., the ℓ-Completeness algorithms would lead to non consent-compliant results.

We thus consider the latter worst-case security model. We note that this does not imply that the server is malicious or would deliberately attempt to violate the consent, and data confidentiality, of participants for secondary use of the data. However, we clarify that because the server is honest, its purpose is to demonstrate to the participants (mobile users and requesters) that any intruder attacker acting on the server side will not be able to: violate the confidentiality or integrity of the data, or corrupt the user-consent compliance process. This also prevents leakage due to server vulnerabilities that could be exploited by external attackers, as the worst case scenario is that an attacker takes control of the server.

To address the malicious adversary model, we leverage the technology of *Trusted Execution Environments* (TEE) [52] (e.g., Intel Software Guard Extensions (SGX) [40], AMD SEV [53] or even ARM TrustZone [54]). This allows specific and well-identified pieces of code to be executed inside secure (hardware) *enclaves* of the TEE, which provide hardware-based data and code confidentiality, integrity and freshness guarantees against a malicious server (adversary with full control on the server, including OS, hypervisor and software stack — see for example [55] in the case of Intel SGX). One of the main issues in securing local processing in TEEs is to cope with different types of side-channel attacks [56]. Some side channels, typically cache attacks [57], page level memory observation [58] or timing can be mitigated through either careful coding of oblivious operators [59] or modification of the execution environment [60]. Other attacks [61–63] are on lower level components of the TEE and are thus outside of the control of the user and should be dealt with by the TEE provider. A precise study of such side channels is highly dependent of the particular TEE used, and is thus outside of the scope of this paper. However, depending on the specific TEE, any mitigation should be readily applied to our architecture. In our work, this is done by fixing the size of data exchanges between enclaves (using padding), and by resorting to only sequential data access patterns in our algorithms so that it is relatively easy to code them in a way that is not affected by cache attacks.

In order to achieve our stated goals, we make the following – classical – assumptions:

(i) We do not consider attacks on client-side devices (the scope of our study is the trust provided on the server side).
(ii) Users' consents are provided faithfully, which implies a strong notion of identity for participants, provided through a PKI infrastructure. This is needed to avoid cases where adversary participants provide a large number of fake data/consents to single out a targeted user contribution among $\ell - 1$ fake identities (our ℓ-Completeness definition assumes data provided from different actual users).
(iii) Data exchanges between users and server-side enclaves are performed over secure channels, and users are guaranteed to communicate only with the correct enclaves, which is typically achieved through a combination of attestation and secure channel creation [64].
(iv) The code implementing ℓ-Completeness can be certified beforehand (e.g., it is open source for community checking, or was approved by a regulatory body, an association or a national privacy regulatory agency).
(v) The TEE enclaves available on the server side provide a proof that a specific piece of code has produced a specific result (this is typically achieved through attestation [64]), and the data manipulated within an enclave is private, even in the presence of an adversary controlling the server (e.g., the OS).

We then assume that the adversary has full control of what happens outside of the enclaves, and that data outside enclaves can be maliciously manipulated (even though encrypted) by any server-side intruder seeking to disclose personal information (i.e., raw contribution values or information derived from these values) to which participants have not explicitly consented.

3.2. Design challenges

As discussed in the previous section, a TEE provides confidentiality and integrity guarantees against a powerful adversary, through an isolated secure memory area (enclave) where the code and sensitive data can be safely processed. Then, a straightforward design for a secure architecture supporting consent-driven multi-task assignment based on ℓ-Completeness (see Fig. 3) is to have a single enclave that:

collects all inputs (users' consents and data contributions), builds ℓ-complete clusters for tasks (cf. Algorithms 1–3), executes tasks, and provides task results. While promising at first glance, this architecture suffers several shortcomings:

- *Enclave limitations (memory, fault tolerance, stateless)*: Considering Intel SGX as a TEE example, it supports only a limited memory space ($\sim$94 MB) for applications running inside enclaves. Meanwhile, task assignment/execution involves large data related to multiple users.
- *Limited scalability*: We need to extend the trust of a single TEE to multiple ones, and thus support a distributed architecture with multiple enclaves to overcome the limits posed by a single-node architecture (single CPU). The architecture should be easily extensible to support multiple enclaves for task execution in case of massive datasets.

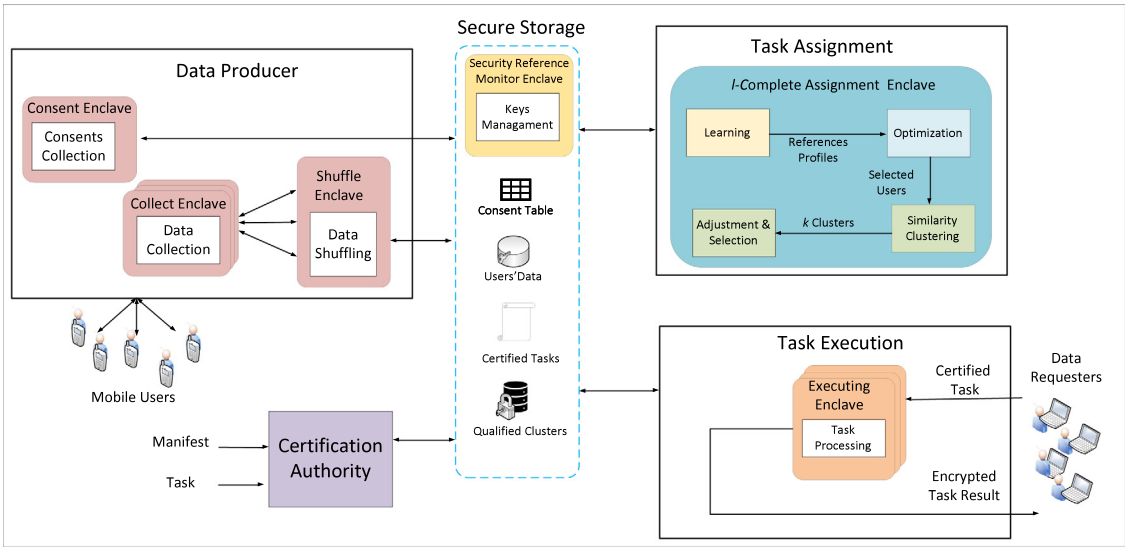


Fig. 4. Privacy-by-design MCS server architecture using enclaves.

- *Limited size of the Trusted Computing Base (TCB)*: Our architecture should minimize the TCB inside enclaves to reduce the size of the task code that run inside enclaves and protect application memory from unauthorized OS accesses.
- *Meeting our security objectives*: The deployment of a data processing module is not trivial; the secure execution of each task is challenging since a given task has only access to ℓ-complete clusters. Keys to decrypt clusters need to be protected on the system as well as securely distributed across enclaves. Thus, the execution must be split in several units running in different enclaves. This introduces an additional security issue: How to protect the system, by-design, in the presence of an attacker on the server side, able to observe or corrupt data and execution in order to infer data and reuse contributions beyond users' consents?

We overcome the above limitations by introducing a secure architecture with multiple enclaves; this ensures that the code and data running inside enclaves are correct and not modified by an attacker. We believe that any vulnerability in the TCB may allow an attacker to compromise the integrity of the system. Thus, the objective is to keep as small as possible the size of the TCB within the enclaves. We also add a mechanism to protect the access to users' data for task execution by securely transferring and distributing keys to enclaves based on their rights. Each task-processing enclave should have access to the corresponding ℓ-complete clusters.

3.3. Secure architecture design

The security of the proposed MCS server architecture is based on two pillars: (1) User consents and contributions are handled in the clear only within secure enclaves, and (2) Control and data flows between components are controlled by the enclaves themselves so that user contributions can only be used in consented tasks in the form of ℓ-Complete datasets. Below we describe the components and their associated enclaves, and then how the flow of data between components is secured.

3.3.1. Architecture components

Fig. 4 illustrates the secure architecture of the MCS server, which consists in the following core components, with enclaves embedded to face malicious attackers (from left-to-right, top-to-bottom in the figure):

- *Data Producer*. It collects data contributed by users as part of active tasks. The component also manages the user registry and associated consent policies represented in the consent table. Due to scalability issues and the sensitive information processed, the component embeds a number of enclaves:
  - *Consent Enclave*: Collects user consents to tasks and their revocation according to the task manifests. The *Consent Enclave* updates the consent table accordingly, which is stored securely.
  - *Collect Enclaves*: Collects user contributions to tasks. The enclave is responsible for authenticating users who log-in (using the chosen method — see assumption (ii) in Section 3.1), collecting their contribution – the data sensed by the user – and inserting it into the *Secure Storage*. In practice, allocating a single *Collect Enclave* for a large number of users would not be scalable, as the enclave would potentially have to simultaneously open

and maintain one secure communication channel per logged-in user. Therefore, we consider allocating multiple *Collect Enclaves*.

- *Shuffle Enclave*: The introduction of multiple *Collect Enclaves* potentially leads to the disclosure of more knowledge about the users when their contributed data is transferred to the *Secure Storage*. The *Shuffle Enclave* prevents such disclosure by buffering and shuffling the contributions from the different *Collect Enclaves* until they reach a sufficient size, before inserting them in the *Secure Storage*. The *Shuffle Enclave* is used to break the correlation between the encrypted stored data and the collected data from users for a curious observer. However, if time/order of events in the data are important, such timestamps would be part of the data and shuffling them would not remove these timestamps.

- *Certification Authority*. It is an – external (e.g., trusted authority such as CNIL[7] in France) – component that acts as a task controller that manages and certifies tasks. In particular, it validates that the implementation of any submitted task conforms to its Manifest (see Definition 1).
- *Secure Storage*. The component is responsible for storing the consent table, user data and certified tasks in encrypted form. It also maintains a *Reference Monitor*, which is embedded in a dedicated enclave, used to interact with the *Task Execution* component:
  - *Reference Monitor Enclave* (*Key Management Enclave*): The role of the *Reference Monitor Enclave* is: (i) To encrypt each cluster produced by the ℓ-*complete Assignment Enclave* using a different key, and (ii) To regulate the dissemination of cluster keys to tasks (i.e., the *Executing enclaves*) according to the GCP (see Definition 3) of each cluster (i.e., a cluster key κ is authorized to the *Executing Enclave* of task T_i iff T_i is in the GCP of that cluster). Therefore, we guarantee that any task has access only to authorized ℓ-complete datasets during execution.
- *Task Assignment*. The component implements the three-phase multi-task assignment presented in Section 2 through the following enclave:
  - ℓ-*Complete Assignment Enclave*: At the end of the Δ time period, the enclave takes as input all consents and data contributions collected over that period and produces a set of clusters that are qualified for each task.
- *Task Execution*. The component creates one dedicated enclave per task:
  - *Executing Enclaves*: Each of them evaluates a given task over a ℓ-complete dataset collected during a Δ period – retrieved from the *Reference Monitor Enclave* – and returns the encrypted results to the relevant data requesters.

3.3.2. Secure data flow enforcement

The flow of data between the different components must be controlled end-to-end – leveraging the attestation capabilities of TEEs – so that user contributions can only feed consented tasks in the form of ℓ-complete datasets, with the results of tasks only accessible to their respective requestors. This is primarily established by requiring that: (1) At login time, users verify that the *Collect Enclave* they are connecting to is authentic, and submit their contributions with the guarantee that they are indeed being processed by a legitimate entry point in our architecture, or drop out; (2) At runtime, the enclaves check their predecessors and successors against the expected data flow, and open secure channels to exchange data along that data flow – the *Reference Monitor Enclave* additionally enforces consent policies (see previous section) –, or abort; and (3) After the task has been executed, the *Executing Enclave* provides the results encrypted with a session key that is itself encrypted with the public key of the target requester, so that only that requester can access the result.

More precisely, the proposed server architecture enforces the above data flow through the following treatment of any task (see Fig. 5):

The process starts with task certification. A Manifest together with the task are submitted to the *Certification Authority*, which assesses that the submitted task conforms to its Manifest. Once certified, the task is added to the list of certified tasks in the *Secure Storage*, with the public key of the corresponding data (task) requester. The private key is used to decrypt the task execution result. This ensures that the tasks consented to by participants are indeed the tasks executed on the server, even if a malicious adversary controls the server. The list of certified tasks may be downloaded by users willing to contribute to tasks. Users then connect ① to the *Consent Enclave* to give their consents based on the details of the Manifest. Participants download and install the sensing task on their smartphones to contribute their individual observations, by connecting to a *Collect Enclave* ②. Both enclaves insert the contributed data (consents and uploaded sensed data) ③ into consent and users' data tables. The *Task Assignment* enclave ④ takes as input the content – inserted over period Δ – of these two tables, and computes the set of qualified clusters that maximizes data reuse, conforms to the task profiles and ℓ-Completeness definition (cf. Algorithms 1–3). The resulting clusters are inserted into the secure storage. The *Task Execution* component creates the set of *Executing Enclaves* (from the list of certified tasks), which authenticate to the *Reference Monitor Enclave* to retrieve the clusters keys they are granted ⑤, decrypt the corresponding clusters

[7] https://www.cnil.fr/.

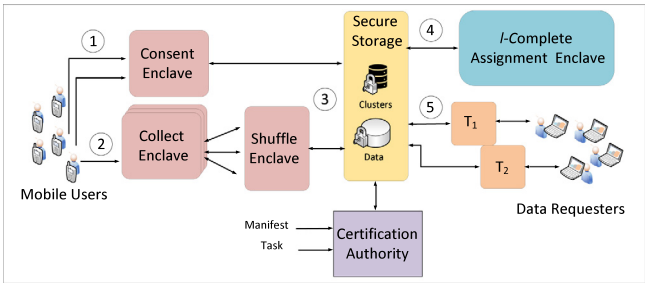


Fig. 5. Privacy-by-design MCS secure data flow.

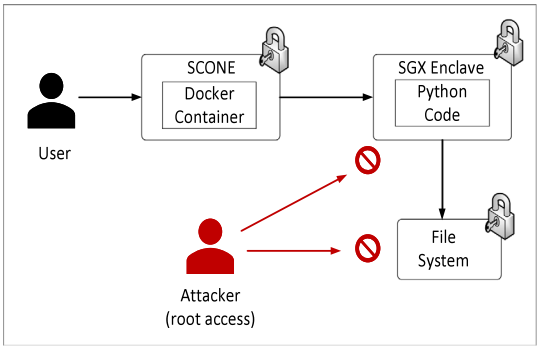


Fig. 6. Implementation with SGX.

and evaluate the task to produce the final results encrypted for the data requesters. The *Task Assignment* and *Executing* Enclaves can be freed (or killed) once the tasks are no longer relevant.

The next section introduces a prototype implementation of the proposed architecture, with an evaluation of the performance assessing the impact of the main parameters.

4. Experiment-based evaluation

4.1. Prototype implementation

We use SGX enclaves [40] for our prototype implementation. Other TEEs are eligible but they have several limitations, e.g.: ARM TrustZone [54] does not integrate any remote attestation mechanism, AMD's TEE [53] is limited in terms of security guarantees since integrity protection cannot be ensured. SGX provides strong confidentiality and integrity guarantees against malicious adversaries with privileged root access. It guarantees that the code and data embedded in the enclave are protected from untrusted applications outside the enclave. It offers a remote attestation mechanism that allows an enclave to attest its identity to another entity outside the platform. The system components are implemented in Python, and we leverage the SCONE platform [65] (Secure CONtainer Environment), a shielded execution framework to enable unmodified applications to run inside SGX enclaves (see Fig. 6).

Using SCONE, the input data and the computation (Python code) are encrypted before being uploaded in the system. They are decrypted inside enclaves using keys that are transparently obtained from the SCONE CAS (Configuration and Attestation Service). As shown in Fig. 6, we create a secure container image based on SCONE. The application is executed within a Docker container, the enclave code is verified by SGX when the enclave is created. We rely on SCONE to protect the File system, it encrypts specified files and creates a file system protection file that stores the message authentication codes for file chunks and the keys used for encryption.

The experiments reported next are run on a server with Intel Xeon E-2276G 6-core CPUs with 12 MB cache and SGX 1-FLC support, 64 GB RAM. The machine runs Ubuntu Linux 18.04 (kernel 4.15.0-142) with SGX DCAP 1.41.

4.2. Performance evaluation of task assignment

The following assesses the overhead of security (i.e., using SGX hardware) on the *Task Assignment* component, which runs the most complex algorithms (cf. Algorithms 1–3, not considering the application-specific tasks). For users' contributions, we used a dataset made available to us by the authors of [66], which provides the users' activities, positions

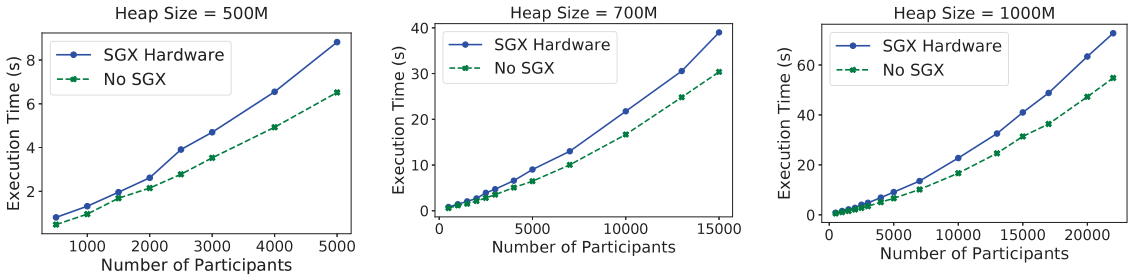


Fig. 7. Execution time *wrt* Number of participants.

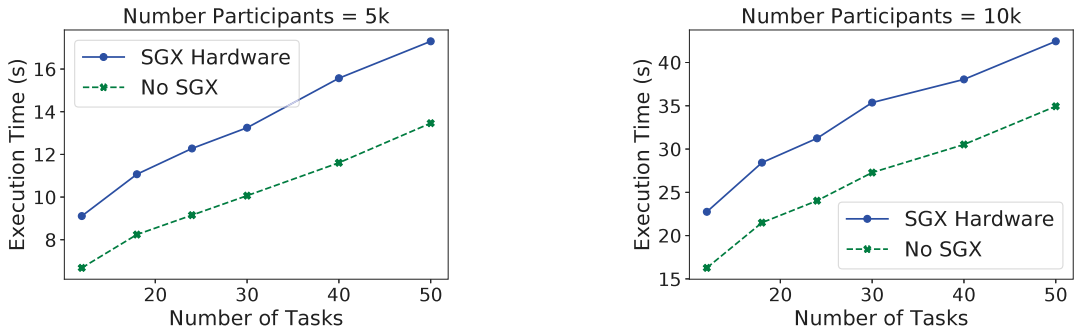


Fig. 8. Execution time *wrt* Number of tasks.

and environmental noise measurements collected by their crowdsensing app. For the consent table, we used a Bernoulli distribution (with a probability $\alpha = 0.6$) to generate the consents of the participants (note that this is a worst case for our algorithm, with higher average distance between policies). For more precision, each execution is repeated 10 times with a set of random users, the result obtained for each evaluation represents the average of the different executions.

4.2.1. Impact of the number of participants

Fig. 7 shows the performance overhead of executing our task assignment module with hardware and native (i.e., no-SGX) modes while varying the number of participants (ℓ= 10 and number of tasks = 12). The total cost increases with the number of participants for both executions. The SGX hardware mode incurs an overhead of $\backsim$ 30% because it requires handling encryption and paging operations. Indeed, the performance cost of the reads and writes increases as the application allocates memory that is larger than the EPC size ($\backsim$94 MB), which may be reduced by reducing as much as possible the required heap size.

As a result, the amount of the heap size (i.e., the value of the parameter that controls how much heap memory is allowed for the application) does not significantly affect performance (see Fig. 7). In general, applications may gain performance when higher memory is allocated. However, this is not the case for applications using SGX enclaves because of the limited EPC size where the use of more memory leads to more EPC swapping operations.

4.2.2. Impact of the number of tasks

As Fig. 8 illustrates, we observe a linear increase of the execution time when increasing the number of tasks (ℓ =10). The SGX hardware mode is slower compared to the native execution due to the fact that the clustering algorithm requires more memory and the EPC size is limited. However, we believe that Intel may release in the future new generation of its hardware with larger EPC sizes. This could help us to reduce the performance overhead and improve the cost of our task assignment process.

4.2.3. Impact of ℓ

Fig. 9 reports the total execution time while increasing the value of ℓ (Number of tasks = 12). The execution using the SGX hardware mode is higher than the native execution. However, we can see that the difference diminishes when we increase the value of ℓ. Note that for $\ell = 50$, the cost of execution using SGX is very close to the cost using a native execution. The reason is that the execution time decreases when we decrease the number of clusters. Increasing the ℓ value leads to decreasing the number of clusters generated by the *k*-means algorithm. Thus, the ℓ value must be adapted to the number of participants to avoid having a large number of clusters. Recent studies have demonstrated that it is possible to obtain the optimal *k* value (i.e. the optimal ℓ value) for the *k*-means algorithm. For example, Franti et al. [67]

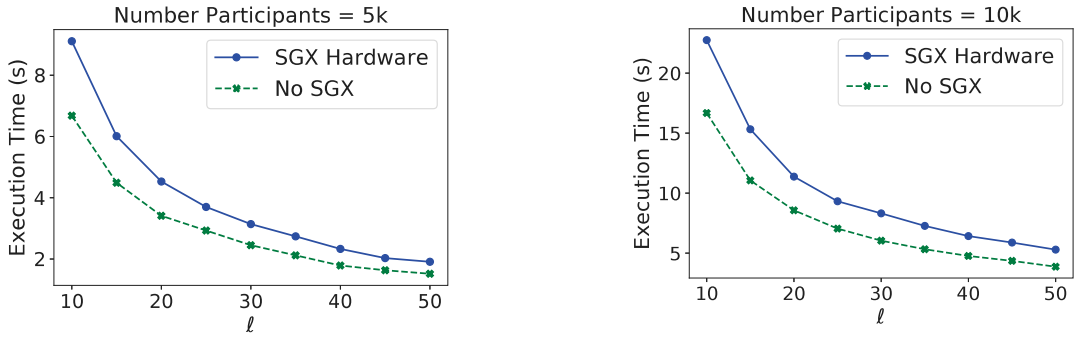


Fig. 9. Execution time *wrt* ℓ.

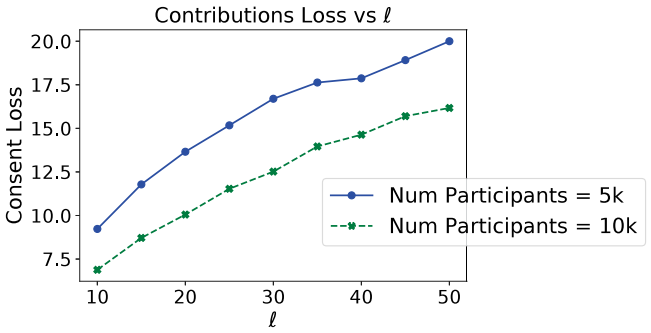


Fig. 10. Contributions loss *wrt* ℓ.

demonstrate that the CI-value of k-means increases linearly with k, where the CI-value defines the cluster centroids that are wrongly located. They also show that the performance of k-means degrades as the CI-value increases.

Following, estimating an optimal ℓ_{opt} value of ℓ for our clustering algorithm first requires to provide an estimated range of ℓ_{opt} as $[\ell_{min}, \ell_{max}]$. The values of ℓ_{min} and ℓ_{max} are estimated based on the acceptable range of the contributions loss, where Eq. (2) defines the ratio of contributions loss within a purely consent-driven system[8]:

$$C_{loss}(\{T_i\}_{i\leq m}) = \frac{|\{c.b_i \in \mathcal{C}, i \leq m, c.b_i = 1, GCP(c).b_i = 0\}|}{|\{c.b_i \in \mathcal{C}, c.b_i = 1\}|} \tag{2}$$

The range of the contributions loss is then defined by the system, it represents the number of contributions that can be discarded by the adjustment stage divided by the total number of contributions. Fig. 10 reports the total contributions loss according to the value of ℓ with 12 tasks. For example, for the dataset composed of 5000 participants, if we consider a range of contributions loss as [10%, 15%], an estimated range of ℓ_{opt} is [10, 20].

The second step to estimate ℓ_{opt} is to consider the performance of the k-means algorithm in determining the optimal number of clusters. For this, we may leverage approaches that allow determining the optimal k value of k-means such as, e.g., the Elbow Method [68]. The idea is to compute the within-cluster-sum of squared errors (WSS) for different values of $k \in [k_{min}, k_{max}]$, and choose the k for which the WSS starts to diminish. The use of such type of methods allows computing the k_{opt} (i.e., ℓ_{opt}) in order to improve the performance of our clustering algorithm in terms of both contributions loss and execution time.

Summarizing, the reported experiments show that the use of SCONE and the current Intel SGX capacity, performing securely clustering/task assignment inside Intel SGX, is practical. It incurs an overhead of ∽30%, but we believe that this is acceptable since the algorithms are executed only once over a single time period Δ, while the execution of tasks is repeated at each Δ. The overhead may also lower with future version of Intel SGX hardware with larger EPC sizes.

4.3. Performance evaluation of task execution

We now evaluate the performance of our *Task Execution* component using one *Executing Enclave*. We analyze the effectiveness of our algorithm using the same dataset (Section 4.2) which provides the users' activities, positions and

[8] Non consent-driven approaches are out of the scope of this paper.

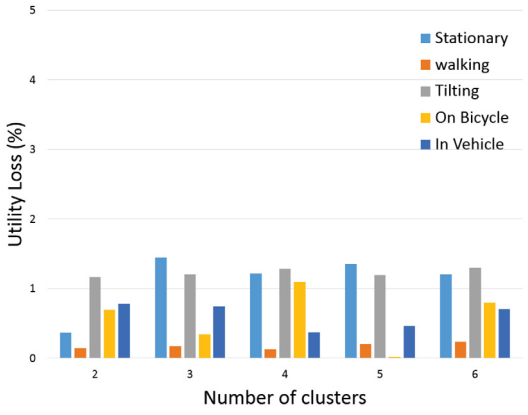


Fig. 11. Utility loss per activity *wrt* # of clusters.

environmental noise measurements. We further carry out the analysis considering an illustrative task that computes the average daily exposure to noise according to the user's activity (i.e., Stationary, Walking, In-vehicle, Tilting, On-bicycle).

4.3.1. Utility loss

For any result O_{T_i} obtained for a task T_i, we evaluate the utility loss as follows:

$$U_{loss}(T_i) = \frac{f_i(\{S_U | C_U \in P_i, GCP(P_i).b_i = 1\})}{f_i(\{S_U | C_U.b_i = 1\})} \tag{3}$$

with S_U data contributed by a set of users U. In the case of a multi-dimensional result (e.g., noise values aggregated by activity), we evaluate the value of U_{Loss} for each dimension.

Fig. 11 analyzes the impact of the clustering-based algorithm on the utility loss (see Eq. (3)) by comparing the execution of the task on the clusters (with clustering) with the execution on the original data (without clustering), according to the number of clusters $|\mathcal{P}| \leq k$ with $k = \left\lfloor \frac{n}{\ell} \right\rfloor$ (see Algorithm 1). Specifically, it shows the information loss when measuring the noise pollution level. We compare the execution on the overall data *vs* a qualified cluster (i.e., the input set produced with Algorithm 3 for that task). We see that the information loss remains negligible with the increase in the number of clusters. The reason is that the cluster follows the global distribution of data; therefore, the difference between the distribution of data in the cluster and the original data is low, which results in a low information loss. The results in particular highlight the effectiveness of the selection stage to achieve higher utility and less information loss compared to base clustering (see [36] for more detail).

4.3.2. Execution time

Fig. 12 shows that the execution time (in milliseconds) increases almost linearly with the increase in the number of participants. The hardware mode is slightly slower compared to the native version. Our architecture allows protecting task processing against attackers with privileged access by securely executing tasks inside SGX enclaves. All data (consent data, user's data) and Python code (task query) are only decrypted inside enclaves.

Based on the experiments, we can conclude that our system combines strong security guarantees (integrity and confidentiality) using SGX, and offers acceptable performance overhead compared to a native execution.

We performed additional experiments to evaluate the clustering algorithm by varying the value of ℓ, the number of tasks as well as the number of participants in our recent work [36]. The objective of these experiments is to evaluate the efficiency of our clustering algorithm compared to other approaches in terms of contributions loss.

5. Related work

5.1. Multi-task allocation in participatory sensing systems

The goal of multi-task allocation is to minimize a total cost function while guaranteeing data quality for multiple tasks. In [69], the objective is to select the minimum subset of participants that satisfies quality-of-information metrics (i.e., granularity and quantity) under a total budget constraint. Zhang et al. [70] introduce a strategy to predict the mobility of participants so as to select the minimum number of participants while ensuring the best spatio-temporal coverage. Li et al. [71] also aim at minimizing the number of participants while meeting a predefined level of coverage. *TaskMe* [72], a framework for multi-task allocation, deals with (1) maximizing the number of accomplished tasks when few

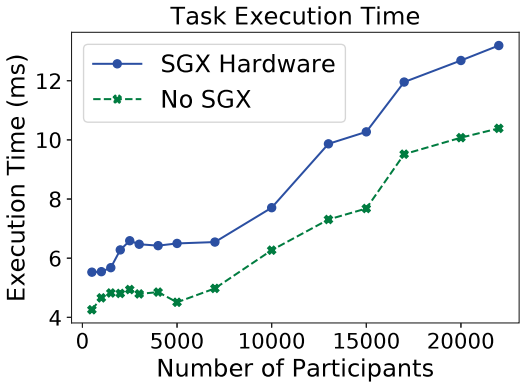


Fig. 12. Task execution time (ms) *wrt* Number of participants.

participants are available; and (2) the opposite problem with many participants and few tasks. *PSAllocator* [24] addresses the multi-task allocation problem from another perspective, that is, maximizing the overall system utility under sensing capability constraints. This includes the work in [25] that aims at maximizing data utility according to participant-side factors (e.g., participant bandwidth, participant availability) when assigning tasks. Zhu et al. [73] introduce a greedy-based approach to maximize the number of accomplished tasks under the sensing capacity and time constraints. In general, existing solutions to the multi-task assignment mainly focus on optimizing the allocation of tasks without taking into account the consents of users. Our work thus distinguishes itself by aiming at maximizing data reuse while taking into account users' consents, which is essential when dealing with the collection of users' data as with crowdsensing systems. In addition, by considering the distribution of data, the overall system utility is optimized.

5.2. Privacy-preserving crowdsensing sensing

Due to the collection of data from individuals, often including sensitive information like location, privacy preservation in crowdsensing systems is the focus of various research. Part of the contributions focuses on leveraging Privacy Enhancing Technology (PET) to implement privacy-preserving communication channels [26,27]. In a complementary way, PET is exploited for the obfuscation of the crowdsensed data [15,16], which leads to alter the accuracy of the data analyses. Such a drawback is addressed by the conjunct optimization of data utility and privacy preservation [17], which primarily result in trading privacy for rewards through supporting auction-based mechanisms [18–20]. Other work investigates the design of privacy-preserving architectures for crowdsensing systems. The SPPEAR architecture, in particular, introduces a cryptography-based protocol for managing interactions between users and the data requester – the latter being additionally responsible for the execution of the task – to ensure accountability of the system entities [74]. The PEPSI infrastructure also introduces cryptography-based protocols to protect the privacy of both mobile nodes and data requesters during data report and query execution [28]. The work of [29] also addresses the trustworthiness of the data providers (mobile users) for which it introduces trustworthy "mobile security agents" that allow assessing the contribution of participants. The trustworthiness of the data requester is investigated in [30], which aims at overcoming an attacker masquerading a legitimate data requester (task provider), using cryptography protocols. Our work is complementary to the above by focusing on privacy-preserving task assignment in the context of multi-tasking crowdsensing, further overcoming the potential malicious behavior of the crowdsensing server.

5.3. Server side privacy-preserving data management

Privacy preserving data management is a widely studied topic. We focus here on server side privacy preserving data management using TEEs. There are three main lines of work to be considered . The first pertains to securely storing and addressing data using TEEs. This range from academic work like EnclaveDB [75] proposing full fledged data management systems within enclaves, to work tackling more specific problems like indexing and storing data [76]. Commercial solutions are also emerging for such problems, e.g., edgelessDB.[9] These offer a number of potential implementations of our secure storage component (see Section 3.3). The second line of work focuses on performing specific data oriented computations using enclaves. These are not directly related to our architectural design, but provide a number of potential implementation avenues for tasks. We can further distinguish two main subtopics. First, implementation of specific data operators within an enclave, typically with the aim of closing side channels (e.g., [59]), provides a number of oblivious operators; [77] introduces oblivious operators for computing on encrypted data. Second, a number of work focuses on

9 www.edgeless.systems.

distributing data computations across enclaves, mostly using existing frameworks like map-reduce [78,79] or SPARK [80]. These could be used to implement tasks that process large amounts of data in a distributed manner. Finally, the third line of work to be considered, which is the most closely related to our architecture solution, are studies that concentrate on chaining different enclaves performing different processing. These provide precise mechanisms for chaining attestations that can be used in our architecture. In particular, [81] provides a way of executing a sequence of tasks on private data, and [82] provides a way of executing a number of tasks within enclaves, chaining attestations and providing overall guarantees to the end user. Both of these approaches provide ways of ensuring integrity of the computation in our architecture.

5.4. Consent in computing platforms

To the best of our knowledge and despite the growing awareness about privacy and consent in multiple fields, the concept of consent has never been considered in the existing literature about mobile crowdsensing systems. Yet, Luger et al. [83] alert multidisciplinary experts about *"a crisis of consent for ubiquitous computing"*. They call the designers to balance their design objectives against a series of consent considerations. Jones [35] discusses the importance of consent in computing, explaining the moral magic of consent that renders permissible an otherwise impermissible action. In [84,85], the authors study the role of consent in privacy policies for social media users. In the case of Facebook, they consider that consent is flawed and claim the need for improvement to create more transparency about users' personal data. Recently, Okoyomon et al. [86] compare the privacy policies of Google Play Store apps with their behaviors and highlight the level of the defect in consent and lack of prioritizing user privacy. In light of this, our work introduces a consent-driven approach to multi-task allocation in participatory sensing systems. Our solution specifically maximizes the number of tasks assigned to participants according to their consents while minimizing the utility loss.

6. Conclusion

We have introduced a solution to privacy-preserving multi-task allocation in mobile crowdsensing systems that fosters data reuse while adhering to privacy-by-design principles.

We leverage the ℓ-completeness consent-driven property for multi-task allocation so as to allow the reuse of crowdsensed contributions across eligible tasks while strictly adhering to the consents of users. Indeed, respecting *consent* is an essential property of crowdsensing systems to guarantee privacy to their users. This leads us to claim that consent-based properties, such as ℓ-completeness, should be an essential part of standards oriented towards privacy in crowdsensing systems.

We further extend the security properties of consent-based privacy preserving multi-task allocation by considering the existence of a malicious adversary for the mobile crowdsensing server. We specifically introduce a server architecture that ensures the integrity and confidentiality of the proposed solution through the use of trusted execution environments (e.g., Intel SGX enclaves). That is, the server architecture ensures that no user's contribution can ever be exposed in clear text outside secure enclaves and apart from the tasks results produced. Evaluation using a prototype implementation and a dataset from a deployed crowdsensing app shows the relevance of the supporting algorithms. Experiments also show that the use of enclaves incurs a reasonable overhead.

Our future work concentrates on addressing a decentralized architecture involving processing on the client side. With the generalization of TEEs in edge/users devices (e.g., Intel SGX is becoming omnipresent on every PC and tablets, ARM's TrustZone on smartphones...), we can expect protecting code and data on the user side. TEEs hence becomes a game changer for the organization of secure and decentralized computations among data scattered on multiple users devices, with a better balance of responsibilities in the processing between central authorities and the participants. A next step in our work is to decentralize part of the processing on the user side, ensuring that each participant is equipped with an enclave. Computation could indeed be optimized by organizing the distribution of the processing among groups of participants [8,87], with similar consent policies.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. List of acronyms and abbreviations

AMD SEV	AMD Secure Encrypted Virtualization
CAS	Configuration & Attestation Service
CASD	Secure Access Data Center

CCPA	California Consumer Privacy Act
CNIL	*Commission Nationale de l'Informatique et des Libertés* — National Commission for Information Technology and Civil Liberties
EMD	Earth Mover's Distance
EPC	Enclave Page Cache
GCP	Greatest Common Policy
GDPR	General Data Protection Regulation Privacy Act
HBC	Honest-but-curious
INSEE	French National Institute of Statistics and Economic Studies
IoT	Internet Of Things
MCS	Multi-tasking Crowdsensing System
PET	Privacy-Enhancing Technologies
PKI	Public Key Infrastructure
SCONE	Secure CONtainer Environment
SGX	(Intel) Software Guard eXtensions
TCB	Trusted Computing Base
TEE	Trusted Execution Environments

Appendix B. List of symbols

$\mathcal{T}$	A set of tasks
T	A sensing & analysis task
$M(T)$	Manifest of T
a	Sensing function of T executed on the mobile client/data provider
f	Function (code) of T executed on the MCS server
$\mathcal{S}$	Type of contributions to T
Δ	A time period
O_T	The result of T
ℓ	The ℓ-completeness value — The minimum number of participants to a task
u	A user in the MCS
U_i	A set of users consenting to task T_i
S_U	Data contributed by the set of users U
$\mathcal{U}$	Set of registered users
$\mathcal{K}$	Knowledge that may be disclosed from a set of tasks
b_i	The consent of a user to task T_i
C_u	Tuple of consents b_i
$\mathcal{C}$	Table of consents
k	The number of clusters for k-means
P_i	A cluster of $\mathcal{C}$

$\mathcal{P}$	Set of clusters
φ_0^T	Reference profile of T
φ_Δ^T	Profile of T during Δ
$GCP(P_i)$	Greatest Common Policy within a cluster P_i
E_T	Qualified cluster for T
$C_{loss}(T)$	Contributions loss of T
$U_{loss}(T)$	Utility loss of T

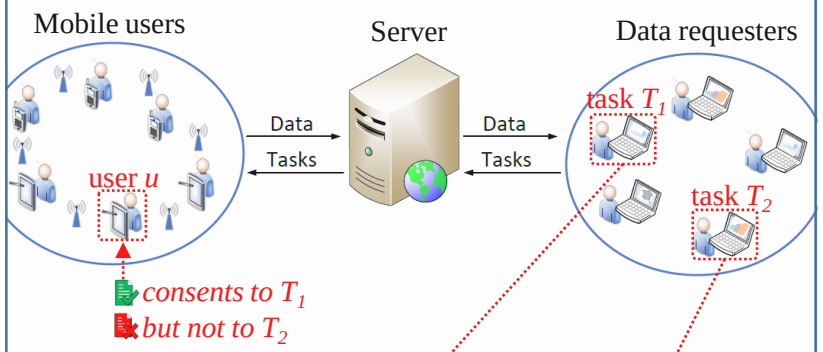
References

[1] J. Liu, H. Shen, H.S. Narman, W. Chung, Z. Lin, A Survey of Mobile Crowdsensing Techniques: A Critical Component for The Internet of Things, vol. 2, (3) Association for Computing Machinery, New York, NY, USA, 2018.
[2] A. Longo, M.A. Bochicchio, M. Zappatore, Apollon project: A massive online open lab for citizen science driven environmental monitoring, in: 2020 IEEE Global Engineering Education Conference, EDUCON, 2020, pp. 1703–1712.
[3] L.A. Kalogiros, K. Lagouvardos, S. Nikoletseas, N. Papadopoulos, P. Tzamalis, Allergymap: A hybrid mHealth mobile crowdsensing system for allergic diseases epidemiology: a multidisciplinary case study, in: 2018 IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops, 2018, pp. 597–602.
[4] J. Wan, J. Liu, Z. Shao, A. Vasilakos, M. Imran, K. Zhou, Mobile crowd sensing for traffic prediction in internet of vehicles, Sensors 16 (1) (2016).
[5] X. Zhang, Z. Yang, W. Sun, Y. Liu, S. Tang, K. Xing, X. Mao, Incentives for mobile crowd sensing: A survey, IEEE Commun. Surv. Tutor. 18 (1) (2016) 54–67.
[6] D. Wu, S. Si, S. Wu, R. Wang, Dynamic trust relationships aware data privacy protection in mobile crowd-sensing, IEEE Internet Things J. 5 (4) (2018) 2958–2970.
[7] X. Hu, X. Li, E.C. Ngai, V.C.M. Leung, P. Kruchten, Multidimensional context-aware social network architecture for mobile crowdsensing, IEEE Commun. Mag. 52 (6) (2014) 78–87.
[8] Y. Du, F. Sailhan, V. Issarny, Let opportunistic crowdsensors work together for resource-efficient, quality-aware observations, in: PerCom 2020: IEEE International Conference on Pervasive Computing and Communications, 2020.
[9] D. Christin, Privacy in mobile participatory sensing: Current trends and future challenges, J. Syst. Softw. 116 (2016) 57–68, http://dx.doi.org/10.1016/j.jss.2015.03.067, URL http://www.sciencedirect.com/science/article/pii/S0164121215000692.
[10] I. Krontiris, M. Langheinrich, K. Shilton, Trust and privacy in mobile experience sharing: future challenges and avenues for research, IEEE Commun. Mag. 52 (8) (2014) 50–55.
[11] L. Pournajaf, D.A. Garcia-Ulloa, L. Xiong, V. Sunderam, Participant privacy in mobile crowd sensing task management: A survey of methods and challenges, SIGMOD Rec. 44 (4) (2016) 23–34.
[12] M.A. Alsheikh, Y. Jiao, D. Niyato, P. Wang, D. Leong, Z. Han, The accuracy-privacy trade-off of mobile crowdsensing, IEEE Commun. Mag. 55 (6) (2017) 132–139.
[13] L. Sweeney, K-Anonymity: A Model for Protecting Privacy, vol. 10, (5) World Scientific Publishing Co., Inc., USA, 2002.
[14] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam, L-diversity: privacy beyond k-anonymity, in: 22nd International Conference on Data Engineering, ICDE'06, 2006, pp. 24–24.
[15] L. Kazemi, C. Shahabi, A privacy-aware framework for participatory sensing, SIGKDD Explor. Newsl. 13 (1) (2011) 43–51.
[16] L. Wang, D. Zhang, D. Yang, B.Y. Lim, X. Han, X. Ma, Sparse mobile crowdsensing with differential and distortion location privacy, IEEE Trans. Inf. Forensics Secur. 15 (2020) 2735–2749.
[17] C. Luo, X. Liu, W. Xue, Y. Shen, J. Li, W. Hu, A.X. Liu, Predictable privacy-preserving mobile crowd sensing: A tale of two roles, IEEE/ACM Trans. Netw. 27 (1) (2019) 361–374.
[18] J. Lin, D. Yang, M. Li, J. Xu, G. Xue, Frameworks for privacy-preserving mobile crowdsensing incentive mechanisms, IEEE Trans. Mob. Comput. 17 (8) (2018) 1851–1864.
[19] H. Jin, L. Su, H. Xiao, K. Nahrstedt, Incentive mechanism for privacy-aware data aggregation in mobile crowd sensing systems, IEEE/ACM Trans. Netw. 26 (5) (2018) 2019–2032.
[20] Z. Wang, J. Hu, R. Lv, J. Wei, Q. Wang, D. Yang, H. Qi, Personalized privacy-preserving task allocation for mobile crowdsensing, IEEE Trans. Mob. Comput. 18 (6) (2019) 1330–1341.
[21] J. Ni, K. Zhang, Q. Xia, X. Lin, X.S. Shen, Enabling strong privacy preservation and accurate task allocation for mobile crowdsensing, IEEE Trans. Mob. Comput. 19 (6) (2020) 1317–1331.
[22] C. Zhao, S. Yang, J.A. McCann, On the data quality in privacy-preserving mobile crowdsensing systems with untruthful reporting, IEEE Trans. Mob. Comput. (2019) 1.
[23] K. Mišura, M. Žagar, Data marketplace for Internet of Things, in: 2016 International Conference on Smart Systems and Technologies, SST, 2016, pp. 255–260.
[24] J. Wang, Y. Wang, D. Zhang, F. Wang, Y. He, L. Ma, PSAllocator: Multi-task allocation for participatory sensing with sensing capability constraints, in: Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing, 2017, pp. 1139–1151.
[25] J. Wang, F. Wang, Y. Wang, D. Zhang, B. Lim, L. Wang, Allocating heterogeneous tasks in participatory sensing with diverse participant-side factors, IEEE Trans. Mob. Comput. 18 (9) (2018) 1979–1991.
[26] Q. Chen, S. Zheng, Z. Weng, Data collection with privacy preserving in participatory sensing, in: 2017 IEEE 23rd International Conference on Parallel and Distributed Systems, ICPADS, 2017, pp. 49–56, http://dx.doi.org/10.1109/ICPADS.2017.00018.
[27] M. Connolly, I. Dusparic, M. Bouroche, An identity privacy preserving incentivization scheme for participatory sensing, in: 2018 Eleventh International Conference on Mobile Computing and Ubiquitous Network, ICMU, 2018, pp. 1–6, http://dx.doi.org/10.23919/ICMU.2018.8653614.
[28] E. De Cristofaro, C. Soriente, Extended capabilities for a privacy-enhanced participatory sensing infrastructure (PEPSI), IEEE Trans. Inf. Forensics Secur. 8 (12) (2013) 2021–2033, http://dx.doi.org/10.1109/TIFS.2013.2287092.
[29] F. Restuccia, S.K. Das, FIDES: A trust-based framework for secure user incentivization in participatory sensing, in: Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014, 2014, pp. 1–10, http://dx.doi.org/10.1109/WoWMoM.2014.6918972.
[30] L. Xue, J. Ni, C. Huang, X. Lin, X. Shen, Forward secure and fine-grained data sharing for mobile crowdsensing, in: 2019 17th International Conference on Privacy, Security and Trust, PST, 2019, pp. 1–9, http://dx.doi.org/10.1109/PST47121.2019.8949066.

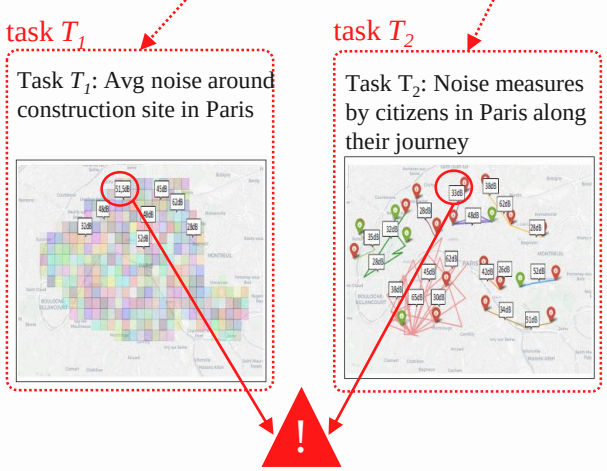
[31] European Council, Regulation EU 2016/679 of the European Parliament and of the Council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46, Off. J. Eur. Union (OJ) 59 (1–88) (2016) 294.
[32] S.L. Pardau, The California consumer privacy act: Towards a European-style privacy regime in the United States, J. Tech. L. Pol'Y 23 (2018) 68.
[33] European Commission explanations, https://ec.europa.eu/info/law/law-topic/data-protection/reform/rules-business-and-organisations/principles-gdpr/purpose-data-processing/can-we-use-data-another-purpose_en.
[34] M. Fabre-Magnan, Defects on consent in contract law, in: European Contract Code, second Ed., The Hague, 2004, 1998.
[35] M.L. Jones, The development of consent to computing, IEEE Ann. Hist. Comput. 41 (4) (2019) 34–47.
[36] M. Brahem, G. Scerri, N. Anciaux, V. Issarny, Consent-driven data use in crowdsensing platforms: When data reuse meets privacy-preservation, in: 2021 IEEE International Conference on Pervasive Computing and Communications, PerCom, 2021, pp. 1–10, http://dx.doi.org/10.1109/PERCOM50583.2021.9439125.
[37] J.E. Bartlett, J.W. Kotrlik, C.C. Higgins, Organizational research: Determining appropriate sample size in survey research, Inf. Technol. Learn. Perform. J. 19 (2001).
[38] C.M. O'Keefe, Privacy and confidentiality in service science and big data analytics, in: IFIP International Summer School on Privacy and Identity Management, Springer, 2014, pp. 54–70.
[39] E. Ramirez, Privacy by design and the new privacy framework of the US federal trade commission, in: Privacy By Design Conference, Hong Kong June 13th, 2012.
[40] Intel SGX, https://software.intel.com/en-us/sgx.
[41] SCONE, SCONE — A secure container environment, 2021, https://scontain.com/.
[42] R. Ventura, V. Mallet, V. Issarny, Assimilation of mobile phone measurements for noise mapping of a neighborhood, J. Acoust. Soc. Am. 144 (3) (2018) 1279–1292, http://dx.doi.org/10.1121/1.5052173.
[43] French National Institute of Statistics and Economic Studies (INSEE), Guide to statistical confidentiality, Tech. Rep., INSEE, 2020, URL https://www.insee.fr/en/statistiques/fichier/2388575/guide-secret-en.pdf.
[44] Secure Access Data Center (CASD), Confidentiality rules, Tech. Rep., CASD, 2020, URL https://www.casd.eu/wp/wp-content/uploads/Output_Confidentiality_Rules.pdf.
[45] AnonyTL specification.
[46] M. Shin, C. Cornelius, D. Peebles, A. Kapadia, D. Kotz, N. Triandopoulos, AnonySense: A System for anonymous opportunistic sensing, Pervasive Mob. Comput. 7 (1) (2011) 16–30.
[47] A.K. Jain, Data clustering: 50 years beyond K-means, Pattern Recognit. Lett. 31 (8) (2010) 651–666.
[48] D. Arthur, S. Vassilvitskii, k-means++: The advantages of careful seeding, Tech. Rep., Stanford, 2006.
[49] Y. Rubner, C. Tomasi, L.J. Guibas, The earth mover's distance as a metric for image retrieval, Int. J. Comput. Vis. 40 (2) (2000) 99–121.
[50] K. Sampigethaya, R. Poovendran, A survey on mix networks and their secure applications, Proc. IEEE 94 (12) (2006) 2142–2181, http://dx.doi.org/10.1109/JPROC.2006.889687.
[51] A. Paverd, A. Martin, I. Brown, Modelling and automatically analysing privacy properties for honest-but-curious adversaries, Tech. Rep., 2014.
[52] M. Sabt, M. Achemlal, A. Bouabdallah, Trusted execution environment: What it is, and what it is not, in: TrustCom/BigDataSE/ISPA (1), 2015.
[53] AMD Secure Technology, https://www.amd.com/en/technologies/security.
[54] ARM. Building a secure system using TrustZone technology, http://infocenter.arm.com/help/topiom.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf.
[55] A. Baumann, M. Peinado, G. Hunt, Shielding applications from an untrusted cloud with haven, ACM Trans. Comput. Syst. 33 (3) (2015) 1–26.
[56] W. Wang, G. Chen, X. Pan, Y. Zhang, X. Wang, V. Bindschaedler, H. Tang, C.A. Gunter, Leaky Cauldron on the dark land: Understanding memory side-channel hazards in SGX, in: CCS, ACM, 2017, pp. 2421–2434.
[57] J. Götzfried, M. Eckert, S. Schinzel, T. Müller, Cache attacks on Intel SGX, in: Proceedings of the 10th European Workshop on Systems Security, 2017, pp. 1–6.
[58] N. Weichbrodt, A. Kurmus, P.R. Pietzuch, R. Kapitza, AsyncShock: Exploiting synchronisation bugs in intel SGX enclaves, in: ESORICS (1), in: Lecture Notes in Computer Science, vol.9878, Springer, 2016, pp. 440–457.
[59] W. Zheng, A. Dave, J.G. Beekman, R.A. Popa, J.E. Gonzalez, I. Stoica, Opaque: An oblivious and encrypted distributed analytics platform, in: NSDI, USENIX Association, 2017, pp. 283–298.
[60] O. Oleksenko, B. Trach, R. Krahn, M. Silberstein, C. Fetzer, Varys: Protecting SGX enclaves from practical side-channel attacks, in: USENIX Annual Technical Conference, USENIX Association, 2018, pp. 227–240.
[61] Z. Chen, G. Vasilakis, K. Murdock, E. Dean, D. Oswald, F.D. Garcia, VoltPillager: Hardware-based fault injection attacks against Intel SGX enclaves using the SVID voltage scaling interface, in: USENIX Security Symposium, USENIX Association, 2021, pp. 699–716.
[62] K. Murdock, D.F. Oswald, F.D. Garcia, J.V. Bulck, F. Piessens, D. Gruss, Plundervolt: How a little bit of undervolting can create a lot of trouble, IEEE Secur. Priv. 18 (5) (2020) 28–37.
[63] J.V. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T.F. Wenisch, Y. Yarom, R. Strackx, Foreshadow: Extracting the keys to the Intel SGX kingdom with transient out-of-order execution, in: USENIX Security Symposium, USENIX Association, 2018, pp. 991–1008.
[64] R. Bahmani, M. Barbosa, F. Brasser, B. Portela, A. Sadeghi, G. Scerri, B. Warinschi, Secure multiparty computation from SGX, in: Financial Cryptography, in: Lecture Notes in Computer Science, vol.10322, Springer, 2017, pp. 477–497.
[65] S. Arnautov, B. Trach, F. Gregor, T. Knauth, A. Martin, C. Priebe, J. Lind, D. Muthukumaran, D. O'keeffe, M.L. Stillwell, et al., {SCONE} Secure linux containers with intel {SGX}, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI}16), 2016, pp. 689–703.
[66] V. Issarny, V. Mallet, K. Nguyen, P.-G. Raverdy, F. Rebhi, R. Ventura, Dos and don'ts in mobile phone sensing middleware: Learning from a large-scale experiment, in: ACM/IFIP/USENIX Middleware 2016, 2016.
[67] P. Fränti, S. Sieranoja, How much can k-means be improved by using better initialization and repeats? Pattern Recognit. 93 (2019) 95–112.
[68] M. Syakur, B. Khotimah, E. Rochman, B.D. Satoto, Integration k-means clustering method and elbow method for identification of the best customer profile cluster, in: IOP Conference Series: Materials Science and Engineering, vol. 336, (1) IOP Publishing, 2018, 012017.
[69] Z. Song, C.H. Liu, J. Wu, J. Ma, W. Wang, Qoi-aware multitask-oriented dynamic participant selection with budget constraints, IEEE Trans. Veh. Technol. 63 (9) (2014) 4618–4632.
[70] D. Zhang, H. Xiong, L. Wang, G. Chen, CrowdRecruiter: Selecting participants for piggyback crowdsensing under probabilistic coverage constraint, in: Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2014, pp. 703–714.
[71] H. Li, T. Li, Y. Wang, Dynamic participant recruitment of mobile crowd sensing for heterogeneous sensing tasks, in: 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, IEEE, 2015, pp. 136–144.
[72] Y. Liu, B. Guo, Y. Wang, W. Wu, Z. Yu, D. Zhang, TaskMe: Multi-task allocation in mobile crowd sensing, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2016, pp. 403–414.
[73] W. Zhu, W. Guo, Z. Yu, H. Xiong, Multitask allocation to heterogeneous participants in mobile crowd sensing, Wirel. Commun. Mobile Comput. 2018 (2018).

[74] S. Gisdakis, T. Giannetsos, P. Papadimitratos, SPPEAR: Security and privacy-preserving architecture for participatory-sensing applications, in: Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless and Mobile Networks, in: WiSec '14, Association for Computing Machinery, New York, NY, USA, 2014, pp. 39–50, http://dx.doi.org/10.1145/2627393.2627402.
[75] C. Priebe, K. Vaswani, M. Costa, EnclaveDB: A secure database using SGX, in: IEEE Symposium on Security and Privacy, IEEE Computer Society, 2018, pp. 264–278.
[76] B. Fuhry, R. Bahmani, F. Brasser, F. Hahn, F. Kerschbaum, A. Sadeghi, HardIDX: Practical and secure index with SGX in a malicious environment, J. Comput. Secur. 26 (5) (2018) 677–706.
[77] S. Eskandarian, M. Zaharia, ObliDB: Oblivious query processing for secure databases, Proc. VLDB Endow. 13 (2) (2019) 169–183.
[78] F. Schuster, M. Costa, C. Fournet, C. Gkantsidis, M. Peinado, G. Mainar-Ruiz, M. Russinovich, VC3: trustworthy data analytics in the cloud using SGX, in: IEEE Symposium on Security and Privacy, IEEE Computer Society, 2015, pp. 38–54.
[79] T.T.A. Dinh, P. Saxena, E. Chang, B.C. Ooi, C. Zhang, M2R: enabling stronger privacy in MapReduce computation, in: USENIX Security Symposium, USENIX Association, 2015, pp. 447–462.
[80] D.L. Quoc, F. Gregor, J. Singh, C. Fetzer, SGX-PySpark: Secure distributed data analytics, in: WWW, ACM, 2019, pp. 3563–3564.
[81] T. Hunt, Z. Zhu, Y. Xu, S. Peter, E. Witchel, Ryoan: A distributed sandbox for untrusted computation on secret data, ACM Trans. Comput. Syst. 35 (4) (2018) 13:1–13:32.
[82] R. Ladjel, N. Anciaux, P. Pucheral, G. Scerri, A manifest-based framework for organizing the management of personal data at the edge of the network, in: ISD, ISEN Yncréa Méditerranée / Association for Information Systems, 2019.
[83] E. Luger, T. Rodden, An informed view on consent for UbiComp, in: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, in: UbiComp '13, 2013, pp. 529–538.
[84] A. Bechmann, Non-informed consent cultures: Privacy policies and app contracts on facebook, J. Media Bus. Stud. 11 (2015) 21–38, http://dx.doi.org/10.1080/16522354.2014.11073574.
[85] B. Custers, S. Van der Hof, B. Schermer, Privacy expectations of social media users: The role of informed consent in privacy policies, Policy Internet 6 (2014) http://dx.doi.org/10.1002/1944-2866.POI366.
[86] E. Okoyomon, N. Samarin, P. Wijesekera, A.E.B. On, N. Vallina-Rodriguez, I. Reyes, A. Feal, S. Egelman, On the ridiculousness of notice and consent: Contradictions in app privacy policies, in: Proc. Workshop on Technology and Consumer Protection, ConPro '19, 2019.
[87] Y. Du, F. Sailhan, V. Issarny, IAM — INterpolation and aggregation on the move: Collaborative crowdsensing for spatio-temporal phenomena, in: MobiQuitous 2020 — International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, 2020.

Multi-tasking Crowdensing System



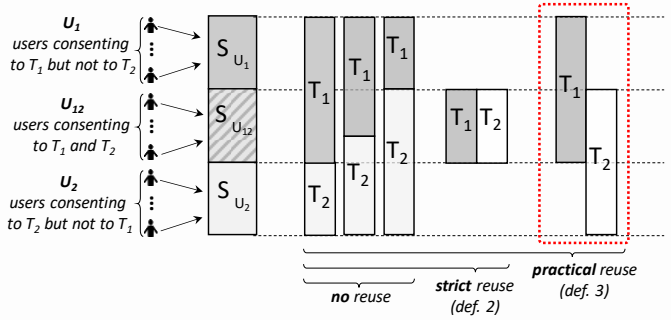
Privacy vs. data reuse: the problem of defects in consent



Defect in consent for a user ***u*** !
u consents to T_1 but not to T_2
but noise measure of *u* is revealed

New method to avoid defects in consents and favor data reuse: *l*-completeness

***l*-complete users-to-tasks assignment:**
all the intersections and differences between sets of users assigned to tasks contain at least n users or none



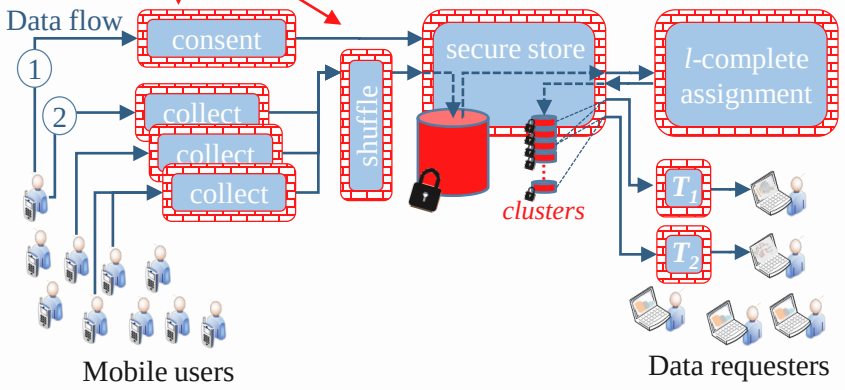
3-stages clustering algorithms for users-to-tasks assignments

① Clustering stage:
↳ cluster users to maximize the number of common consented tasks per *cluster*

② Adjustement stage:
↳ adjust clusters with less than *l* users based on clusters *Greatest Common Policy*

③ Selection stage
↳ select clusters for tasks to maximize data reuse

Privacy-by design implementation on Trusted Execution Environment



Validation using SCONE/Intel SGX: Efficient using large datasets