Safe Anonymization of Data Hosted in Smart Tokens

Tristan Allard Benjamin Nguyen
Philippe Pucheral
PRiSM Laboratory, SMIS Project,
University of Versailles, INRIA Rocquencourt,
45, Av. des Etats-Unis, Domaine de Voluceau,
78035 Versailles, France 78153 Le Chesnay, France
{First.Last}@prism.uvsq.fr {First.Last}@inria.fr

Abstract

Un nombre croissant d'études mettent en évidence l'échec des serveurs de bases de données à sauvegarder réellement le caractère privé des données confidentielles. Sans même considérer les attaques, internes ou externes, de simples négligences mènent souvent à des dévoilement massifs de données. Un nouveau type de dispositifs, appelés *Secure Portable Token* (SPT), combinant la sécurité d'une carte à puce avec les capacités de stockage FLASH, autorise et rend crédibles des alternatives à la centralisation systématique des données personnelles. Chacun peut stocker ses données personnelles (e.g., son dossier médical) dans son propre SPT sous son contrôle, et ne jamais les dévoiler *en clair* au monde extérieur. Cependant, cette nouvelle gestion des données personnelles entre en conflit avec les outils d'aide à la décision qui supposent habituellement une certaine centralisation des données. Cet article adresse précisément ce problème en proposant d'adapter le modèle traditionnel de publication de données anonymisées (Privacy-Preserving Data Publishing - PPDP) à un environnement constitué d'un grand nombre de SPTs sécurisés, se connectant rarement à une infrastructure disponible mais non digne de confiance. Cette combinaison unique d'hypothèses rend le problème fondamentalement différent de tout problème PPDP déjà traité.

Keywords: privacy, data publishing, k-anonymity, smart tokens, distributed protocol

1 Introduction

Individuals are more and more reluctant to entrust their sensitive data to any data server. This suspicion is fueled by computer security surveys pointing out the vulnerability of database servers against external and internal attacks [22]. According to these surveys, nearly half the attacks come from insiders (i.e., employees) of companies and organizations hosting the data and even the best defended servers are not spared [4, 5, 3]. There are also many examples where negligence leads to personal data leaks. To cite a few, data of thousands of Medicare and Medicaid patients in eight US states were lost in a HCA regional office and Hospitals County accidentally published medical records on the web including doctor notes, diagnoses and medical procedures [7]. In the UK, the personal details of 25 million citizens were lost inadvertently [6], as well as the details of 84.000 prisoners [8]. This growing suspicion sometimes compromises nationwide projects: for instance, the Dutch Electronic Health Record program was canceled due to privacy concerns expressed by citizens [9].

In the meantime, credible alternatives to a systematic centralization of personal data are arising. These alternatives build upon the emergence of new hardware devices called Secure Portable Tokens (SPTs for short). Whatever their form factor (SIM card, secure USB stick, wireless secure dongle), SPTs combine the tamper resistance of smart card micro-controllers with the storage capacity of NAND Flash chips. This unprecedented conjunction of portability, secure processing and

Gigabytes-sized storage constitute a real breakthrough in the secure management of personal data. Thanks to SPTs, personal records can be easily managed under the control of the record owner herself with security guarantees stronger than those provided by any central server. Today, the use of SPTs for e-governance (citizen card, driving license, passport, social security, transportation, education, etc) is actively investigated by many countries, and personal healthcare folders embedded in SPTs receive a growing interest, e.g., the Health eCard [1] in UK, the eGK card [2] in Germany, the HealthSmart Network [3] in the USA.

However, the counterpart of the privacy risks incurred by centralizing personal data is the opportunity it offers for knowledge-based decision making. *Privacy-preserving data publishing* (PPDP) is an attempt to reconcile privacy and knowledge-based decision making. A typical PPDP scenario starts by a **collection phase** where the *data publisher* (e.g., a hospital) collects data from *record owners* (e.g., patients), followed by a **construction phase** where the publisher computes the anonymization rules defining the transformations to apply to the collected data to make it anonymous, and ends with an **anonymization phase** where the publisher effectively applies the rules to the data. Anonymous data is now ready to be released to a set of *data recipients* (e.g., a drug company, a public agency, or the public) for data mining or inquiry purpose. Most research in the PPDP area considers a trusted model where the data publisher is trustworthy, so that record owners are assumed to easily consent providing it with their personal information [17]. As pointed out above, convincing record owners about the legitimacy of this trusted assumption is difficult in practice.

Hence, governments and public agencies are faced today with two conflicting objectives: (1) the need for decision making tools, usually to increase a collective benefit (e.g., to prevent a pandemic thanks to an epidemiological study), (2) the obligation to get the consent of individuals to process their data electronically [1], pushing them to find alternatives to a systematic centralization of personal data (e.g., thanks to SPTs). In addition, the legislation in several countries authorizes statistical treatments over individuals' personal data without their explicit consent (assuming this consent has been given for the initial purpose of the data collection), provided that the data is adequately anonymized [1, 2]. While the spirit of the law is to protect better the individuals' privacy, the side effect is a new incentive for individuals to refuse their consent for the initial data collection if they distrust the way their data will be anonymized. Indeed, it does not make sense for an individual to consent to the management of her healthcare data thanks to a SPT (because she distrusts any central server) while accepting that this same data will end up in a central server for anonymization purposes. The objective of this paper is precisely to address this issue, that is to safely (i.e., without privacy breaches) anonymize personal data hosted in SPTs while considering an untrusted PPDP model.

Imagine a scenario where SPTs embed a Personal Data Server providing facilities to store, update, delete, and retrieve data (potentially through queries) and to enforce access control rules. Alice carries her electronic healthcare folder on such an SPT. When Alice visits a practitioner, she is free to provide her SPT or not, depending on her willingness to let the practitioner physically access it. In the positive case, the practitioner plugs Alice's SPT to his terminal and authenticates to Alice's SPT server. According to his access rights - enforced by the embedded Personal Data Server - the practitioner queries and updates Alice's folder through his local Web browser. When Alice receives care at home, the practitioner interacts the same way with Alice's SPT thanks to his netbook or tablet PC without need of an Internet connection. Alice's data never appears on any central server and no trace of interaction is ever stored in any terminal. If Alice loses her SPT, the SPT's tamper-resistance renders potential attacks harmless. She can recover her folder from an encrypted archive stored by a trusted third party or managed by herself. If the health agency of Alice's country decides to collect sensitive data to perform an epidemiological study, Alice has no reason to be anxious

[1] `http://www.healthecard.co.uk`
[2] `http://www.gematik.de`
[3] `http://www.healthsmartnetwork.com/`

because she has the assurance that her data will be anonymized at the time it leaves her SPT. Hence, no identifying data will be exposed with sensitive data on any central server. So, Alice can enjoy her healthcare folder with full confidence without compromising a collective healthcare benefit.

The above scenario is actually not futuristic. Medical-social folders embedded on SPTs are currently experimented in the Yvelines, a district of France, to provide care and social services at home to elderly people (PlugDB[4]). The folders mix medical and social data (income, dependent's allowance, marital status, entourage, food habits, etc) to the highest benefit of statistical studies. Being able to publish anonymized data from these folders is therefore a very important challenge. This challenge is however not restricted to the healthcare domain. Similar scenarios can be envisioned each time the legislation recognizes the right of the record owner to control under which conditions her personal data is stored and accessed.

This paper focuses on this challenge, that is how to organize the data collection and the anonymization phases at the data source (i.e., at each SPT) while compromising neither privacy nor data utility. The problem is difficult due to three assumptions: (1) the data publisher and the data recipients are untrusted, (2) the SPTs are trusted but there is no direct communication between them and (3) there is no certainty about the connection frequency and duration of each SPT connection. Hence, the goal is to design a protocol which produces an anonymized version of a database horizontally split among a population of trusted SPTs, such that the untrusted environment (UE) can never learn more than the final result.

This work has clear connections with cryptographic techniques for privacy preserving data collection, secure multiparty computation and distributed privacy preserving data mining. However, to the best of our knowledge, no previous work has ever considered the conjunction of hypothesis made in this paper, that is the tamper-resistance of the SPTs, their low availability, the untrustworthiness of the publisher and the fact that each SPT contains the data of a single record owner.

The paper is organized as follows. Section 2 introduces the hypothesis our study relies on and states the problem. Sections 3, 4 and 5 discuss respectively the three models of attacks the UE may endorse in our context, and propose data publishing protocols resistant to these attacks. Section 6 presents our experiments and demonstrates the practicability of the approach. Section 7 surveys the related research areas. Finally, section 8 concludes by opening exciting research perspectives.

2 Problem Statement

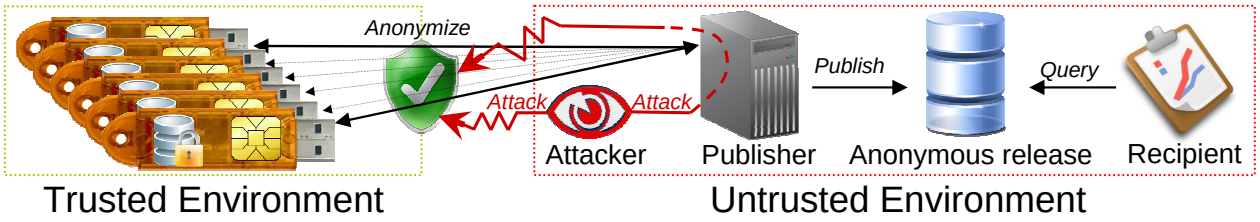


Figure 1: Anonymous release of data stored on SPTs

Figure 1 illustrates the functional architecture and mode of operation considered in the paper. The architecture is composed of two parts. The *Trusted Environment (TE)* is constituted by the set of SPTs participating in the infrastructure. Each SPT hosts the personal data of a single record owner. However, it can take part in a distributed computation involving data issued from multiple record owners since all the SPTs trust each other. The number of participating SPTs is application dependent and may vary from tens of thousands in a small environment (e.g., a specific clinical study over

[4] http://www-smis.inria.fr/~DMSP/home.php

a selected cohort) to millions in a region-wide or nation-wide initiative (e.g., an epidemiological study for a nation-wide health research program). The *Untrusted Environment (UE)* encompasses the rest of the computing infrastructure, in particular the data publisher and the data recipients.

2.1 Hypothesis on TE

Regardless of their form factor, SPTs share several hardware commonalities. Their microcontroller is typically equipped today with a 32 bit RISC processor (clocked at about 50 MHz), a ROM, a small static RAM, a small internal stable storage (NOR Flash or EEPROM) and security modules providing the tamper-resistance. The microcontroller is connected by a bus to a large external mass storage (Gigabytes of NAND Flash). Contrary to the microcontroller, this external mass storage is not hardware protected; hence data stored there must be encrypted, but the cryptographic keys and the encryption process remain confined within the microcontroller. SPTs can communicate with the outside world through various standards (e.g., USB2.0, bluetooth, 802.11) [31]. In summary, a SPT can be seen as a basic but very cheap (a few dollars), highly portable, highly secure computer with reasonable storage and computing capacity for a personal usage. For illustration purpose, Fig. 2 depicts the SPT used in the PlugDB project.

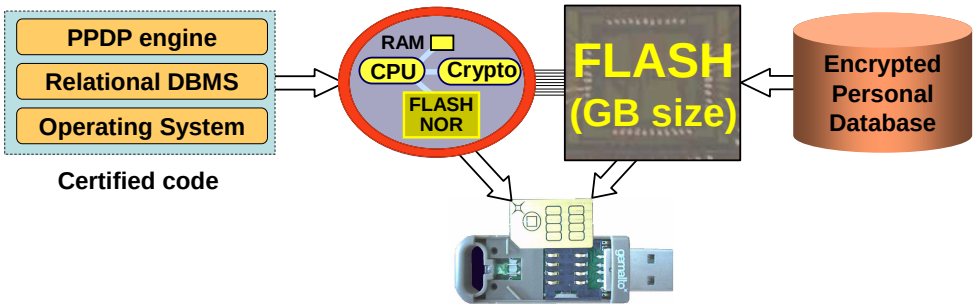


Figure 2: SPT's internal architecture

The trustworthiness of SPTs lies in the following factors: (1) the SPT's embedded software inherits the tamper resistance of the microcontroller making hardware and side-channel attacks highly difficult, (2) this software is certified according to the Common Criteria[5], making software attacks also highly difficult, (3) this software can be made auto-administered thanks to its simplicity, contrary to its traditional multi-user server counterpart, thereby precluding DBA attacks, (4) compared to a traditional server, the $\frac{Cost}{Benefit}$ ratio of an attack is increased by observations 1 and 2 and by the fact that a successful attack compromises only the data of a single individual and (5) even the SPT owner cannot directly access the data stored locally. She must authenticate, thanks to a PIN code or a certificate, and only gets data according to her privileges.

Consequently, the hypothesis of interest for TE are the following:

- *High_TE_Trust:* the multi-factors security provided by SPTs (see above) makes them highly trusted;
- *Low_TE_availability:* the SPTs are assumed seldom connected and the duration and frequency of connections are unpredictable. Although several SPTs are likely to be connected simultaneously at each point of time, no global data availability can be guaranteed;
- *Suitable_TE_Power:* the SPT computing, storage and communication capabilities are not considered as bottlenecks. Although roughly four orders of magnitude less powerful than a traditional server, each SPT has suitable CPU resource with respect to the computation task it must handle locally, cryptographic functions are implemented in hardware, and the communication throughput is high. Section 6 shows numbers confirming this statement.

[5] http://www.commoncriteriaportal.org/

2.2 Hypothesis on UE

The hypothesis on UE are the following:

- *High_UE_Availability*: UE can be modeled as a traditional server accessible through the Internet 24/7;
- *High_UE_power*: UE provides unlimited computing power and storage capacity.

Attack model. Even though the data publisher itself is not suspected to become attacker, the UE may have deviant behavior and attacks can be conducted on the data publisher's behalf. We model the attacker as follows, according to both its intents and abilities, with no assumption on which part of UE conducts them:

- $Honest-but-Curious$: the attacker obeys the protocol it is participating in but tries to infer confidential data by exploiting in any possible way the results of each step of this protocol;
- $Weakly-Malicious_{Soft}$: the attacker has *weakly-malicious* intent [41] in that it cheats the protocol to disclose confidential data only if (1) the trusted participating parties (i.e., the TE) do not detect it and (2) the final result is correct. The abilities of the attacker are said *Soft* in that it is unable to breach the hardware security of any SPT.
- $Weakly-Malicious_{Hard}$: the attacker still has *weakly-malicious* intent, but its abilities are said *Hard* because it is able to break *at least one* SPT to disclose its internal cryptographic material, leading to a collusion between UE and a member of TE. Though highly improbable, every study considering secure hardware must evaluate the impact of breaking one hardware element on the complete solution.

2.3 Hypothesis on the anonymization algorithm

In this paper, the dataset to be anonymized is classically modeled as a single table $T(ID, QID, SD)$ where each tuple represents the information related to an individual hosted by a given SPT. ID is a set of attributes identifying an individual uniquely (e.g., a social security number). QID is a set of attributes, called *quasi-identifiers*, that could potentially identify an individual depending on the data distribution (e.g., a combination of Birthdate, Sex and Zipcode). The SD attributes contain sensitive data, such as an illness in the case of medical records. The table schema, and more precisely the composition of QID and SD, is application dependent. It is assumed to be defined before the collection phase starts, and is shared by UE and all SPTs participating in the same application (e.g., the same healthcare network).

In most statistical studies, it is possible to drop ID attributes without causing any loss of precision while dropping QID attributes degrades the results. However, QID attributes can be used to join different data sources in order to link back an individual to its specific sensitive data with high probability. This type of disclosure, called *record linkage* [17], has received much consideration not only by academics, but also by legislators [2] and industrials [6]. k-anonymity [33] is both the basic building block of more sophisticated models fighting against record linkages, and the most popular of these models. k-anonymity proposes to make the record linkages ambiguous by hiding individuals into a crowd: its basic idea is to make each tuple indistinguishable from at least $(k-1)$ others. k-anonymity is often achieved by generalizing the QIDs to form equivalence classes (see [17] for a good overview), where each class contains (at least) k tuples sharing the same degraded QID.

This paper aims at restoring the ability to prevent record linkages through k-anonymity in our specific context. In others words, our goal is to publish $T'(QID', SD)$, a k-anonymized version of $T(ID, QID, SD)$ in which the ID attributes have been dropped and the QID attributes have

[6] For example, Privacy Analytics Inc (`http://www.privacyanalytics.ca/`) sells a product that de-identifies datasets based on the k-anonymity principle, e.g., to make them meet HIPAA privacy rules.

been, e.g., generalized, to prevent record linkages. Note that we do not consider here *attribute linkages* [17] which can help infer the value of some SD if their values are poorly distributed among the QIDs. Although we are aware of the existing debate on pure record linkage prevention [30], the usual models preventing attribute linkages [30, 29] are also debated [14]. This paper does not participate in this debate, by adding arguments in favour of one model or the other. We simply propose a practical solution, based on our specific constraints, to the PPDP approach having reached the most achieved practical consensus between Law, Industry, and Academy, namely the record linkage approach through k-anonymity. Considering other models is left for future work.

The approach proposed in this paper can work with any algorithm that (1) builds non-overlapping equivalence classes (EC) from QIDs and (2) keeps close semantics between an equivalence class and the values of the QIDs it contains. Most generalization-based algorithms fall in this category, e.g., [32, 36, 27, 37]. Such algorithms are based on generalization taxonomies, each taxonomy defining the generalization hierarchy of a QID attribute. Basically, an equivalence class is defined by a set of *generalization nodes* (one per taxonomy) that partitions tuples based on their QID values. In the following, we use $op1 \preceq op2$ to mean that $op1$ specializes $op2$.

Without loss of generality, we illustrate this paper by using the well known Mondrian algorithm [28] whose basic principles can be intuitively stated as the following:

- Plot the collected data into a QID dimensional space;
- Divide recursively the space into subspaces (or equivalence classes) that contain at least k points;
- Replace the QIDs by the boundaries of the subspaces.

Each resulting subspace is an equivalence class whose generalization nodes are the subspace's boundaries.

Figure 3 shows the two 2-anonymous equivalence classes computed by the Mondrian algorithm taking as inputs the plotted QIDs (the dots in the figure represent the distinct QIDs, for clarity, we do not plot the cardinality of each *QID*). Anonymizing the tuples whose QIDs are in the equivalence class EC_1 simply means replacing their QIDs by EC_1's boundaries (see the right of the figure). Note that the larger the dataset to be anonymized, the more precise the boundaries will be, while maintaining the same k-anonymity privacy level.

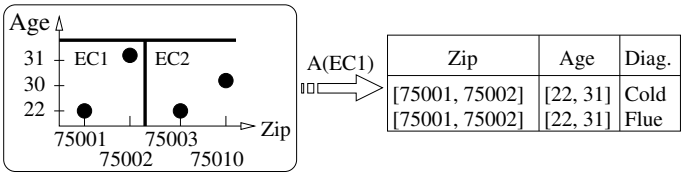


Figure 3: 2-anonymous Equivalence Classes

Such an algorithm requires all QID values to compute the ECs and all the SD to produce the final result. In its traditional form, it is run by the central trusted publisher, based on the collected raw data. It is worth noting the impossibility to directly transpose this computation in our context, because the publisher is part of the Untrusted Environment, and therefore cannot access any raw data.

2.4 Problem Statement

We address in this paper the problem of producing a k-anonymous version of a dataset defined by the union of the data hosted by a collection of SPTs (a subset of interest in TE) such that:

1. UE gets the final anonymized result but cannot learn anything else about individual's data ($Honest-but-Curious$ and $Weakly-Malicious_{Soft}$ UE), or, if the attacker has cracked an SPT, then the TE can detect this incursion and limit its scope ($Weakly-Malicious_{Hard}$);
2. The anonymized result is as useful as if it had been computed by the same anonymization algorithm run by a traditional publisher on a central server.

The next sections propose a solution to this problem for each identified attack model, namely $Honest-but-Curious$, $Weakly-Malicious_{Soft}$ and $Weakly-Malicious_{Hard}$.

3 Honest-but-Curious UE

Let us first consider the simplest attack model, namely $Honest - but - Curious$, where UE is assumed to fully respect the protocols defined but can make any inferences or offline calculations it wants to disclose the association between QIDs and SDs.

3.1 Naïve algorithm

Algorithm 1 is self-explanatory. During the **Collection phase**, the SPTs that connect communicate to the UE their QIDs. When the UE decides that the sample of QIDs is big enough, it stops the Collection phase and launches the **Construction phase**, during which it computes the equivalence classes and their generalization nodes based on the QIDs collected previously. Note that the Collection and Construction phases are flexible in the sense that the UE can take into account the quality of the classes built to decide when it has collected enough data. For the sake of simplicity, we consider the sample size fixed to a value P from the start. When classes are ready, UE finally launches the **Anonymization phase** during which each SPT that participated in the Collection phase determines the equivalence class it belongs to and sends its sensitive data in the form of an *anonymized tuple* (see Alg. 1).

This algorithm requires that the authentication of SPTs to the UE does not disclose the identity of each SPT. This precludes UE from keeping track of all messages sent by the same SPT during the protocol, hence linking the QID of a participating SPT to its SD. Standard cryptographic tools [23] can help building such anonymous channels (for instance, the Tor network [13], based on the Onion-routing protocol [21]).

Algorithm 1 Naïve Algorithm

Require: An anonymous communication channel between the SPTs and UE, the k-anonymity level, the number P of QIDs required by the class construction phase.

1: **Collection phase:** For $i = 1,\ldots,P$, each SPT_i that connects send its quasi-identifier QID_i to UE.
2: **Construction phase:** UE computes the equivalence classes EC_j and their corresponding generalization nodes $EC_j.\eta$, respecting the k-anonymity principle, and publishes the result.
3: **Anonymization phase:** Each input SPT SPT_i looks up the equivalence class corresponding to its QID and sends an *anonymized tuple* in the form of : $(EC_j.\eta, SD_i)$ where $QID_i \preceq EC_j.\eta$.

The correctness of the algorithm can be trivially stated. After the first phase of the algorithm, UE knows all the QIDs but no SD. Since UE is $Honest - but - Curious$, it correctly calculates and publishes q different k-anonymous equivalence classes, with $\cap EC_{j=1..q} = \emptyset$. There are at least k different SPTs whose QID belongs to each equivalence class. During the Anonymization phase, when a SPT sends an anonymized tuple $(EC_j.\eta, SD_i)$ it is impossible to distinguish it from at least $(k-1)$ other tuples that will eventually be sent, and therefore it is impossible for UE to construct a link between a given SD and a QID with more precision than k.

The weakness of the Naïve algorithm lies in parameter P. P denotes the number of QIDs required by the Construction phase, which is a fixed parameter of the Collection phase being run. Since each SPT hosts a single QID, P is equal to the number of distinct SPTs that connect during the Collection phase. In practical situations, P is likely to be smaller than the total number of existing SPTs, denoted hereafter by N. In the algorithm presented above, the same set of SPTs is assumed to participate to *both* the first and the third phases of the protocol. This assumption is very strong since no hypothesis is made on the frequency of the SPT connections and the latter is presumably very low for some SPTs (e.g., a healthy patient). The latency of the algorithm is thus potentially unbounded.

3.2 Robust algorithm

The objective of the Robust algorithm is to avoid the unbounded latency of the Naïve algorithm to make it applicable to real scenarios. To this end, Algorithm 2 collects the quasi-identifiers and the sensitive data during a single collection phase: it is no longer mandatory for the same SPT to connect twice during the protocol.

The Robust algorithm must however guarantee that the association (QID, SD) remains hidden to UE. Consequently, its phases are slightly different from the Naïve's phases. During the **Collection phase**, the SPTs send to UE tuples of the form $(QID, E_{\kappa 1}(SD))$, where E denotes a symmetric encryption scheme (e.g., based on the AES encryption function) taking a secret key $\kappa 1$ as parameter shared by all SPTs (key management is discussed next). The QIDs are still sent in the clear to allow the UE to construct the equivalence classes during the **Construction phase** (similarly to Naïve). During the **Anonymization phase**, any SPT that connects downloads a class (or more if its connection duration allows it) and is able to decrypt the SDs it contains, producing anonymized tuples of the form $(EC_j.\eta,(E_{\kappa 1}^{-1}(E_{\kappa 1}(SD))))$. Note that in practice, k remains low (in the order of 10^2); it follows that downloading and decrypting between k and $(2k-1)$ tuples does not present any bottleneck (*Suitable_TE_Power* hypothesis). Algorithm 2 summarizes the sequence of phases performed by an SPT running the Robust algorithm.

However, SPTs primarily serve other purposes than PPDP - as illustrated by the motivating scenario. Consequently, despite the affordable decryption cost of a class, SPTs may disconnect during this task (*Low_TE_Availability* hypothesis). As a result, SPTs should run the decryption task in background and send each anonymized tuple on the fly, instead of sending all of them at the end of the task and run the risk of losing the results of the job performed. Performing this process in background leads to a new problem: the SPTs and the UE must be able to distinguish, in a set of encrypted tuples, which ones have already been anonymized (1) to avoid doing the same job twice for the SPTs, and (2) to remove possible duplicates generated during parallel anonymizations for the UE. In addition, the background process must be organized such that there is no way for UE to infer the association between QID and SD by simply spying the SPT input and output flows. The rationale is to add a signature to each equivalence class, indicating which tuples have already been processed. This signature must be collision-resistant to allow several SPTs to contribute in parallel to the same equivalence class and must not disclose any information to UE which could help inferring the association between QID and SD. The solution is as follows: (1) The UE associates an ID to each encrypted tuple collected and (2) the SPT sends to UE a message authentication code (MAC) [19] of the ID of each tuple it has processed. As the encryption scheme, the message authentication scheme is parameterized by a key $\kappa 2$ shared by all SPTs. Thanks to their MACs, duplicate tuples can be identified by the SPTs and UE without revealing to UE which collected tuples have actually been processed so far.

The security of the Robust algorithm relies on the use of two secret keys ($\kappa 1$ for the encryption and $\kappa 2$ for the MAC) shared by all SPTs. We do the simplifying assumption that these keys are pre-installed by the SPT provider, though more dynamic protocols could be easily devised. Let us stress that even the SPT's owner cannot spy the hidden content and the computation made by his own SPT (in the same way as a banking card owner cannot gain access to the encryption keys pre-installed in his smart card microcontroller). Sharing secrets among all SPTs make sense in the model of attack considered in this section, where the SPT hardware security is assumed unbreakable. This assumption will be relaxed when the $Weakly-Malicious_{Hard}$ model of attack will be considered.

As in the Naïve algorithm, k-anonymity is guaranteed by the fact that UE never gets access to a (QID_i, SD_i) tuple. The only tuples it has at its disposal are in the form $(QID_i, E_{\kappa 1}(SD_i))$, with no way to decrypt SD_i. During the anonymization phase, the partial states observed by UE give no information allowing to infer the association between a given SD and a QID with more precision than k since UE cannot reverse the MAC of each D_j.

Algorithm 2 Robust Algorithm

Require: The k-anonymity level, the number P of QIDs required by the class construction phase, $E_{\kappa 1}$ and $M_{\kappa 2}$ the encryption and MAC functions parameterized by secret keys $\kappa 1$ and $\kappa 2$ shared among the SPTs.

1: **Collection phase:** For $i = 1,\ldots,P$, each SPT_i that connects sends its encrypted tuple $(QID_i, E_{\kappa 1}(SD_i))$ to UE.
2: UE assigns a unique (arbitrary) identifier TID_i to each tuple
3: **Construction phase:** UE computes the equivalence classes EC_j and their corresponding generalization nodes $EC_j.\eta$.
4: Let $T_j = \{(TID_i, E_{\kappa 1}(SD_i))|QID_i \in EC_j\}$ represent the set of tuples of EC_j where the QID attribute of each tuple has been replaced by the corresponding TID.
5: $\forall j$,let $D_j = \emptyset$ represent the received MACs of the tuples anonymized in EC_j.
6: **Anonymization phase:**
7: **repeat**
8: UE sends (T_j, D_j) to a connecting SPT_m
9: **for all** $(TID_i, E_{\kappa 1}(SD_i)) \in T_j$ **do**
10: **if** $M_{\kappa 2}(TID_i) \notin D_j$ **then**
11: SPT_m sends $(M_{\kappa 2}(TID_i), E_{\kappa 1}^{-1}(E_{\kappa 1}(SD)))$ to UE
12: UE computes $D_j \leftarrow D_j \cup M_{\kappa 2}(TID_i)$
13: **end if**
14: **end for**
15: **until** $\forall j, D_j$ has same cardinality as T_j

4 Weakly-Malicious$_{Soft}$ UE

This section deals with the attacks that a *Weakly-Malicious*$_{Soft}$ UE can launch, and upgrades the Robust algorithm accordingly. In this attack model, SPTs are still presumed unbreakable. Hence, the possible attacks lie in modifying the input sent to the SPTs during the Anonymization phase in order to infer the links between QIDs and clear text decrypted SD values.

4.1 Differential Attacks

The UE can launch attacks that reduce the privacy of a set of individuals, called the *target set*, by computing the difference between the anonymized results of two equivalence classes whose contents overlap. The idea behind this class of attacks, called *differential attacks* is that the sensitive data corresponding to QIDs in both classes appears in both results while the sensitive data corresponding to QIDs in a single class may appear in a single result. Fig. 4 depicts such an attack. In this example, two different sets of equivalence classes have been defined by UE over the same dataset. By computing the differences between the two versions of EC_1, the attacker infers that (1) $QID = (75001, 22) \rightarrow SD = cold$, (2) $QID = (75002, 31) \rightarrow SD = flue$, and (3) $QID = (75003, 22) \rightarrow SD = HIV$.

More generally, let ED be the universe of encrypted data, SD be the universe of sensitive data, $EC_1, EC_2 \in EC$ two equivalence classes, $\star$ the wildcard symbol, and $A : \{ED\} \rightarrow \{SD\}$ the anonymization function mapping a set of encrypted data to a set of clear text sensitive data. A differential attack consists in computing (1) $A(EC_1.\{(\star, ED)\}) \cap A(EC_2.\ \{(\star, ED)\})$ to yield the sensitive data corresponding to $EC_1.\{(QID, \star)\} \cap EC_2.\{(QID, \star)\}$, (2) $A(EC_1.\{(\star, ED)\}) - A(EC_2.\{(\star, ED)\})$ to yield the sensitive data corresponding to $EC_1.\{(QID, \star)\} - EC_2.\{(QID, \star)\}$, and (3) $A(EC_2.\{(\star, ED)\}) - A(EC_1.\{(\star, ED)\})$ to yield the sensitive data corresponding to $EC_2.\{(QID,\star)\} - EC_1.\{(QID, \star)\}$. If one of these differences contains less than k sensitive data, k-

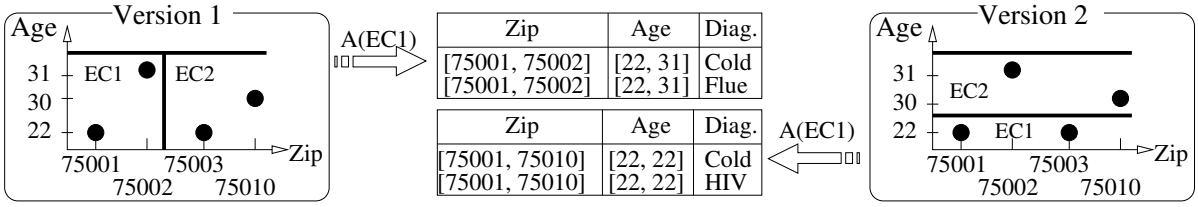


Figure 4: An *overlapping* Differential Attack

anonymity is broken. Hence, a weakly-malicious UE can launch a differential attack by: (1) producing equivalence classes such that the elements of the target set appear in their differences, then (2) anonymizing them thanks to the connected SPTs, and (3) cross-analyzing the results.

Note that previous works have faced similar disclosures in the context of delivering subsequent k-anonymous releases of an evolving dataset [24, 18]. Indeed, insertions and deletions make a dataset *naturally* prone to such disclosures. However, in our context, the source of the problem is different: it is the UE, publisher included, that introduces these breaches into classes. Whereas the solutions proposed in previous work lie in the definition of models and techniques performed by the publisher to avoid unsafe publishing (see Section 7 for more details), we focus here on reinforcing the protocol to ensure that equivalence classes are free of attacks.

4.2 Safety properties of equivalence classes

To preclude differential attacks, the equivalence classes must verify the following properties[7].

Local properties. Local properties are related to the content of each equivalence class, independently from the others:

- *Cardinality*: The given equivalence class contains at least k tuples: $\forall EC_i \in EC, |T_i| \geq k$.
- *Origin*: All tuples in the given equivalence class originate from a SPT. The $Origin$ property prevents *forging attacks*. A forging attack consists in inserting l forged tuples into a class to make the sensitive data of the other tuples $(k-l)$-anonymous.
- *Distinguishability*: All tuples in a given equivalence class are distinct. The Distinguishability property prevents *cloning attacks*. A cloning attack consists in building a class with a target set of m tuples (with $m \leq k$), and replicating these tuples to yield at least k tuples. The extreme case of a cloning attack occurs when m equals to 1, and the remaining $(k-1)$ tuples are clones of the single target tuple.
- *Specialization*: The QID of each tuple specializes its *class's* $node$. In other words, each tuple must belong to the proper class. Let $EC_i \in EC, \forall t \in T_i, t.QID \preceq EC_i.\eta$. The Specialization property prevents *copying attacks*. A copying attack consists in copying a tuple from an equivalence class EC_i into another class EC_j such that $EC_j.\eta$ does not generalize the tuple's QID.

Global properties. Global properties are related to the relation of a given equivalence class with the whole set of classes already sent to a given SPT:

- *Invariance*: The Invariance property requires a given class to be always associated with the same content: $\forall EC_i, EC_j \in EC \times EC, EC_i.\eta = EC_j.\eta \Rightarrow T_i = T_j$. The Invariance property prevents *masking attacks*. A masking attack consists in anonymizing an equivalence class into which l tuples are masked, i.e., not sent to a SPT, then re-anonymizing the same class with its complete content to isolate these l tuples from the result.

[7]For the sake of clarity, we consider in the following that the generalization taxonomy used to build the equivalence classes is defined over a single attribute. The extension to the multi-attribute case is straightforward.

- *Mutual Exclusion*: The Mutual Exclusion property requires that nodes of distinct classes generalize distinct sets of values. Let $L : Node \rightarrow Node$ the function returning the leaves that specialize a node in its generalization tree, then: $\forall EC_i, EC_j \in EC \times EC, EC_i.\eta \neq EC_j.\eta \Rightarrow L(EC_i.\eta) \cap L(EC_j.\eta) = \emptyset$. The Mutual Exclusion property prevents *overlapping attacks* as the one depicted in Figure 4.

By construction, the Cardinality, Origin, Distinguishability and Specialization properties guarantee that each sensitive data returned to UE is associated to a quasi-identifier indistinguishable from (at least) $(k-1)$ other quasi-identifiers of the proper equivalence class. The Invariance and Mutual Exclusion properties guarantee that equivalence classes generalization nodes cannot be tampered during the course of the protocol by UE to generate overlapping classes. As a result, processing equivalence classes satisfying each of these properties will generate a k-anonymous result set with certainty.

4.3 Checking local properties

Checking the local properties in an SPT is rather straightforward. To test the Cardinality property, each SPT receiving an equivalence class during the anonymization phase checks that the number of tuples in the class is higher than k. To test the Origin property, a simple solution is for each SPT participating in the Collection phase to compute a MAC of its tuple. Then the MAC is checked by the SPTs participating in the Anonymization phase. To test the Distinguishability property, each SPT is assigned a unique identifier $SPTID_i$ which now serves as tuple identifier TID_i (UE generated it in the above algorithm versions). Tuple identifiers are encrypted with the tuples during the Collection phase, then each SPT participating in the Anonymization phase checks the uniqueness of this identifier in the equivalence class. To test the Specialization property, QIDs are encrypted with the tuples during the Collection phase, then each SPT participating in the Anonymization phase checks that the QIDs of all tuples specialize their class's node.

4.4 Checking global properties

Each SPT receives a single equivalence class per session, so checking the global properties *would* require that SPTs share information among them about the received classes. Unfortunately, SPTs are not able to communicate with each other: each SPT can solely rely on its own history. In the algorithm, UE can easily select the equivalence class sent to each SPT such that all the properties are satisfied from the SPT viewpoint while they are violated from a global viewpoint. There is no ultimate solution to this problem since UE can delete any information sent by SPTs willing to build a common history or share a global viewpoint. Considering UE is presumed to have weakly malicious intentions, we propose to deter UE to launch differential attacks through *caveat actions* that increase the probability of detecting a global property violation by increasing the number of SPTs likely to observe it.

Caveat actions. The first caveat action (that was also required by the Naïve algorithm) is to force UE to use an anonymous channel to communicate with the SPTs. This precludes UE to control which SPT will receive which equivalence class. Consequently, the probability to send classes violating the global properties to the same SPT is no longer null. However, since UE still chooses the classes it sends, it can minimize the number of SPTs receiving two classes violating the global properties (e.g., by limiting the duration of the attack) to minimize the detection probability.

The second caveat action is to force UE to produce a *Summary* of the equivalence classes, which contains for every class its generalization nodes plus a digest of its content (e.g., a hash of its tuples). Each SPT participating in the anonymization phase primarily downloads this Summary. SPTs that participate for the first time check the Mutual Exclusion property by verifying that no EC node generalizes another one within the current summary. SPTs that have already participated

check the Invariance property by verifying that the current summary contains the same classes (nodes and digests) than the previous summary (persistently stored). Finally, the SPT checks that the equivalence class it downloads is consistent with regards to the Summary. Note that the detection of Invariance or Mutual Exclusion violations is probabilistic since the verification relies on the SPT's own history. Actually, every SPT that connects to UE, before and during or during and after an attack, detects the violation of global properties due to disagreeing summaries.

Detection Probability. We use the following probability model. We divide the total latency of the protocol into S steps of equal duration. We note $C = multiset\{C_i\}$ the distribution of the SPT connections, where $C_i \geq 1$ is the number of connections made by SPT SPT_i to UE. We note N the total number of SPTs, with $N \gg S$. For simplicity's sake we suppose that the connections are at random moments of the protocol, with equal probability.

Let us consider an atomic differential attack that consists in sending a modified summary at a single step S_{cheat} and to only one connected SPT denoted SPT_{cheat}. This attack will have the lowest probability of detection. The only situation in which this attack is not detected is if SPT_{cheat} never reconnects during the protocol. Hence, the lower bound for the probability $\mathbb{P}(detect)$ to detect an attack is $\mathbb{P}(detect) = 1 - \overline{\mathbb{P}(detect)}$ where $\overline{\mathbb{P}(detect)}$ is the probability that UE selects SPT_{cheat} having $C_{cheat} = 1$. We see that detection probability directly depends on the proportion of SPTs that connect only once.

In a worst case scenario, i.e. for skewed distributions favoring SPTs with $C_i = 1$, $\mathbb{P}(detect)$ can therefore be quite low. However, detection probability can be increased by forcing a minimum number L of SPTs to participate in the anonymization of each equivalence class. Let $\mathbb{P}_{mindetect}$ denote the minimal probability of detection to be guaranteed, $\mathbb{P}_{mindetect} \geq (1-\overline{\mathbb{P}(detect)}^{L})$. Hence, L can be computed by: $L > \frac{ln(1-\mathbb{P}_{mindetect})}{ln\mathbb{P}(detect)}$. For instance, even for highly unfavorable (and unrealistic) values such as $\overline{\mathbb{P}(detect)} = 0.8$, $L = 21$ guarantees a probability of detection $P_{mindetect} \geq 0.99$. It is important to note that L can be chosen as big as we want, in particular, L can be greater than k, the anonymity parameter. In such a scenario, L different SPTs will check the invariance and mutual exclusion properties, but will not necessarily decrypt tuples. High probabilistic detection comes at a cost of sending the equivalence classes to more SPTs, and waiting for at least L different SPTs to connect (in the best case scenario, they connect during the same step). However, this has barely no consequence on the global latency of the protocol, provided L is small compared to the number of SPTs connecting each step.

4.5 SPT algorithm for Weakly-Malicious$_{Soft}$ UE

Algorithm 3 details the anonymization phase of the algorithm to be executed by each SPT. If a property check is not fulfilled, the SPT stops the execution and raises an alarm (e.g., to the destination of the SPT owner or a trusted third party). For the sake of conciseness, we do not detail the Collection phase and EC Construction phase of the algorithm because their extensions compared to the Robust algorithm are straightforward in the light of sections 4.3 and 4.4. We also do not detail the mechanism used to avoid an SPT from downloading an equivalence class already fully processed (extra information needs to be exchanged between SPTs and UE to this end but does not impact the core of the protocol).

5 Weakly-Malicious$_{Hard}$ UE

Throughout this paper, we assumed that SPTs were unbreakable because of their proven high tamper-resistance [16]. Though breaking the security of a single SPT requires significant resources and is highly improbable, we evaluate the global impact of breaking one element of the architecture.

Algorithm 3 $Weakly - Malicious_{Soft}$- SPT

Require: An anonymous communication channel between the SPTs and UE, the k-anonymity level, $E_{\kappa 1}$ and $M_{\kappa 2}$ the encryption and MAC functions parametrized by secret keys $\kappa 1$ and $\kappa 2$ shared among the SPTs, and a hash function H.

Receive the current Summary S: let $|S|$ denote the number of classes in S, and $EC_i^S.\delta$ and $EC_i^S.\eta$ respectively the digest and generalization nodes of the class EC_i in S;
if $\nexists$ previous Summary S_p **then**
  for all $EC_i^S, EC_j^S \in S \times S \mid EC_i^S.\eta \neq EC_j^S.\eta$ **do**
    Check the *Mutual Exclusion* property: $L(EC_i^S.\eta) \cap L(EC_j^S.\eta) = \emptyset$;
  end for
else
  Check that $|S| = |S_p|$;
  for all $EC_i^S \in S$ **do**
    Check that $\exists EC_j^{S_p} \in S_p | EC_i^S.\eta = EC_j^{S_p}.\eta$ and $EC_i^S.\delta = EC_j^{S_p}.\delta$;
  end for
end if
Select an equivalence class EC_i^S, and download its content T_i and list of processed tuples D_i;
Check the consistency between S and the downloaded content: $EC_i^S.\delta = H(T_i)$;
Check the *Cardinality* property: $|T_i| \geq k$;
Initialize the collection of TIDs $\Theta \leftarrow \emptyset$ and the collection of safe decrypted tuples $\Delta \leftarrow \emptyset$;
for all $t \in T_i$ **do**
  $d \leftarrow E_{\kappa 1}^{-1}(t)$;
  Check the *Origin* property: $M_{\kappa 2}(d) = t.MAC$
  Check the *Specialization* property: $d.QID \preceq EC_i^S.\eta$;
  Check the *Distinguishability* property: $d.TID \notin \Theta$;
  $\Theta \leftarrow \Theta \cup d.TID$;
  if $M_{\kappa 2}(d.TID) \notin D_i$ **then**
    $\Delta \leftarrow \Delta \cup d$
  end if
end for
for all $d \in \Delta$ **do**
  Send to UE: $(M_{\kappa 2}(d.TID), d.SD)$,
end for

In the current protocol, if UE succeeds in breaking one SPT, it unveils not only the SPT's personal data (SD) but also its cryptographic materials (notably encryption and hashing keys) which could in turn be used to decrypt the content of all equivalence classes. To limit the scope of such attack, the traditional solution is to use χ different keys and organize the encryption process so that the impact of compromising one key is divided by χ. Similarly, we partition SPTs into C *clusters*, *randomly* and *evenly*, SPTs belonging to different clusters being equipped with distinct cryptographic materials. Consequently, each SPT can now contribute only to the decryption of the partial content of a subset of equivalence classes.This section concentrates on the impact of introducing clusters into the protocol over (1) the selection of equivalence classes by the SPTs and the correlated resistance to cardinality attacks , (2) the resistance to differential attacks.

Preventing Cardinality Attacks. SPTs append their cluster ID (CID) to the data sent during Collection phase to allow SPTs participating in Anonymization phase to choose the class into which their CID appears. As a side effect, the attacker can now group QIDs by CIDs in each class to form subgroups containing less than k QIDs . The disclosure occurs by linking a set of returned SDs (less than k) to the subgroup(s) of QIDs of the same cardinality. To avoid such links, SPTs

participating in the Anonymization phase of a given class EC_j must return at most GCD_{EC_j} SDs, GCD_{EC_j} being the greatest common divisor of the cardinalities of the subgroups of QIDs in EC_j. Note that UE can not cheat on tuples CIDs since it would be immediately detected by an SPT checking the Origin property.

Preventing Forging Attacks. Breaking a SPT allows UE to impersonate its cluster through the disclosed cryptographic materials. To reduce the k-anonymity of a set of target tuples to a $(k-j)$-anonymity, UE builds a class containing these tuples along with j forged tuples, encrypted by the cryptographic materials of the broken cluster. Consequently, the tampered class contains far more tuples from the broken cluster than from any other cluster. This is both the strength of the attack (it reduces the effective k) and its weakness (since SPTs are randomly and evenly partitioned into clusters, typical clusters should send about the same number of tuples in each class). We prevent this attack by detecting *over-represented clusters* through typicality tests on the number of tuples sent per cluster. When a SPT receives a class, it counts the number of tuples originating from each CID (recall that the UE cannot cheat on CIDs), identifies its distribution, and runs the corresponding typicality test for each CID.

Due to the inherent uncertainty of typicality tests, there is a margin between the "normal" and the "over-represented" categories of cluster; it represents the number of tuples that the UE can insert in a class withouth being detected. The margin depends on the false positive rate that we wish to tolerate (i.e., false alarms) which depends on the distribution of counts inside each class. In practice, this latter is (at least, close to be) Uniform merely because each cluster posts uniformly into the QID space. As a result, the margin remains very small compared to the number of tuples that should be inserted to reduce k significantly. A simple way to disable the effects of the margin is to increase the required k-anonymity level: $k_{final} = k_{wanted} + margin$.

Let detail the worst case example, i.e., the counts of the downloaded class follow a Normal distribution. Let α be the threshold under which a cluster is considered over-represented. For each cluster CID_i in the considered class, let cnt_{CID_i} be the number of tuples in this cluster, $cntAvg_{\overline{CID_i}}$ be the average number of tuples of all clusters other than CID_i in this class and $stdev_{\overline{CID_i}}$ their standard deviation. So $z^{obs}_{CID_i} = \frac{cnt_{CID_i} - cntAvg_{\overline{CID_i}}}{stdev_{\overline{CID_i}}}$. We consider CID_i as over-represented if $\mathbb{P}(z_{NormalDistrib} \geq z^{obs}_{CID_i}) < \alpha$. For each cluster C_i of the considered class, let z^{max} be the maximum z-score of a cluster before we consider it as over-represented. z^{max} is a constant for a given α, and is such that: $\mathbb{P}(z_{Normal\_Distrib} \geq z^{max}) = \alpha$. Let $cnt^{max}_{C_i}$ be the maximum representativeness of C_i: $cnt^{max}_{C_i} = z^{max} \times stdev_{\overline{C_i}} + cntAvg_{\overline{C_i}}$. As a result, the margin of C_i corresponding to a normal distribution is: $margin^{normal}_{C_i} = cnt^{max}_{C_i} - cntAvg_{\overline{C_i}}$.

6 Experimental Validation

6.1 Experimental platform

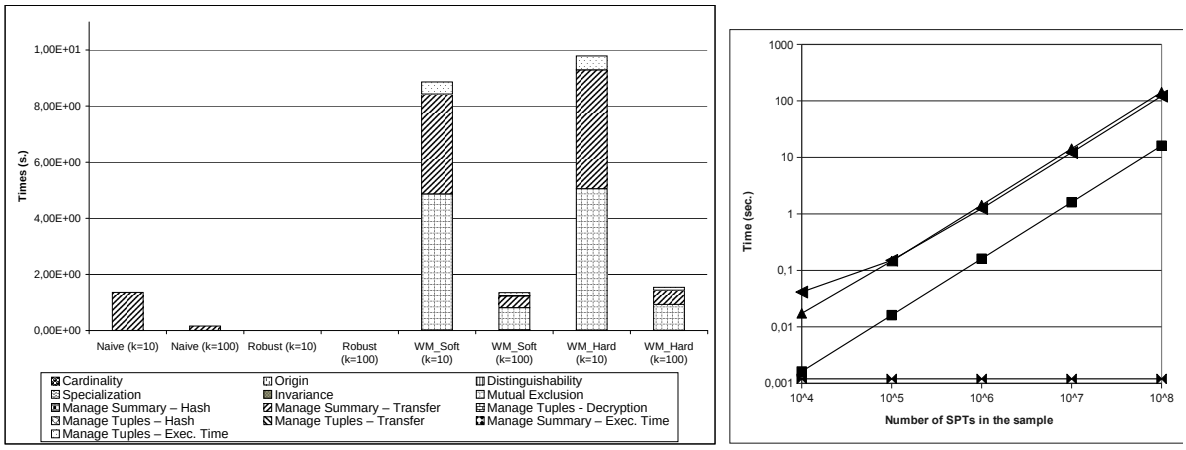
The algorithms presented in this paper have been implemented and integrated in a larger prototype named PlugDB[8]. PlugDB aims at managing secure portable medical-social folders with the objective to increase quality and coordination of care provided at home to dependent patients. A complete chain of software (web server, application and DBMS server) has been developed and is embedded in the secure USB Flash platform pictured in Figure 2. The hardware platform is provided by Gemalto (the world leader in smart-cards), industrial partner of the project. The project is founded by the Yvelines Distinct of France and by the French National Research Agency and will be experimented in the field in 2010 in a medical network handling elderly people. The hardware

[8] http://www-smis.inria.fr/~DMSP/home.php

platform is still under test so the performance measurements have been conducted on a cycle-accurate hardware emulator.

The algorithms considered for the experiments are *Naive*, *Robust*, *WM_Soft* ($Weakly-Malicious_{Soft}$) and *WM_Hard* ($Weakly-Malicious_{Hard}$). We concentrate on the evaluation (1) of the time spent internally in each SPT to participate to each phase of the protocol, and (2) of the protocol latency. We obtained the results of point (1) by performance measurements conducted on the hardware emulator, and the results of point (2) by simulations considering a synthetic dataset and two distributions of SPT connection frequency (*Uniform* and *Skewed*).

6.2 SPT computing time



(a) Detailed times (10^6 SPTs in the sample)

(b) Scaling (k=100)

Figure 5: SPT's Time Consumption

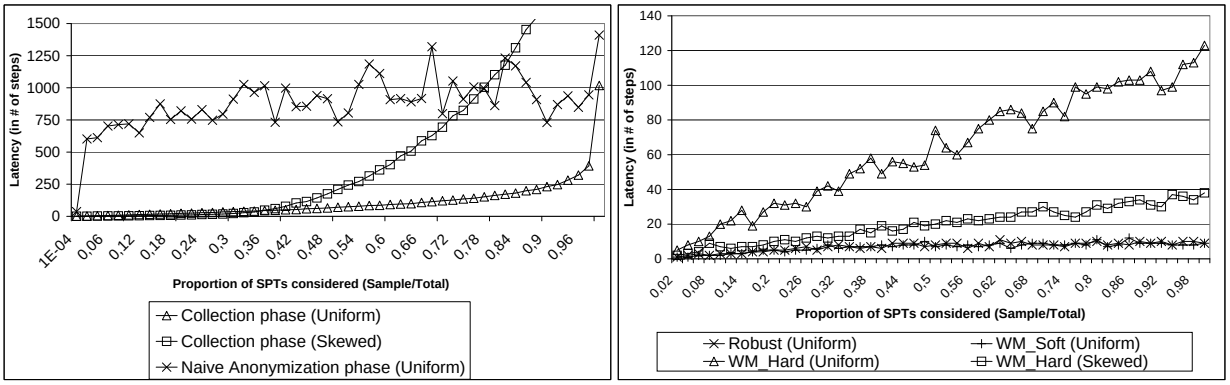
Figure 5(a) details the time consumed by a SPT for each basic operation performed during the anonymization protocol. Note that for convenience we plot together the times taken by the Mutual Exclusion and Invariance properties; actually, only one of them is checked during the same session of an SPT (see Alg. 3). The measure has been performed with a sample of 10^6 SPTs, partitioned into 10^2 clusters for WM_Hard, k varying from 10 to 100. Depending on the algorithm, the worst case occurs either when k is minimal (i.e., the more numerous the classes, the higher the cost of managing the summary) or when k is maximal (i.e., the more numerous the tuples in each class, the higher the cost of managing them). For each algorithm, we plot these two cases to assess whether performance bottlenecks could compromise the feasibility of the approach. The cycle-accurate hardware simulator we used for this experiment is clocked at 50Mhz, corresponding to the CPU clock of the target platform. All basic cryptographic operations (i.e., encryption and hashing) are implemented in hardware with good performances (e.g., encrypting a block of 128bits with AES costs 150 cycles). Although Hi-Speed USB2 (480 Mbps theoretical bandwidth) is announced for the near future, today's implementation of the communication channel is far less efficient. The measured throughput is 12Mbps (i.e., Full-Speed USB2), which amounts to 8Mbps of useful bandwidth when we exclude the overhead of the USB protocol itself.

The worst case for Naive, WM_Soft, and WM_Hard occurs when k is low. In this situation, the transfer cost of the Summary and the checking cost of Global Properties dominate the other costs because of the high number of equivalence classes. Operations related to tuples (ie, transfer, hashing, and decryption) are cheap since Naïve solely uploads its sensitive data while WM_Hard downloads k tuples and uploads GCD sensitive data. On the contrary, the worst case for Robust

occurs for a high k value: for Robust, the tuples' transfer cost overwhelms the other costs. The main conclusion is that, even in the worst cases, the execution time amounts to about ten seconds, confirming the *Suitable_TE_Power* hypothesis.

Figure 5(b) shows the scaling of all the protocols wrt to the number of SPTs in the sample - chosen to be on a *nation-wide scale* - with $k = 100$. Apparently, Naïve, WM_Soft, and WM_Hard scale linearly with the number of SPTs sampled. This is due to the linear increase in size of the Summary (cost of transferring it and checking the global properties). Robust remains constant, around 10^{-1} sec.; indeed, it does not use the Summary so its consumption only depends on k.

6.3 Protocols' latencies



(a) Collection Phase and Naive Anonymization Phase (b) Robust, WM_Soft, WM_Hard Anonymization Phases

Figure 6: Latencies

Figures 6(a) and 6(b) plot respectively the latency of the Collection phase and of the Anonymization phase of the protocol. The latency is measured in terms of connection steps of equal duration, this duration being application dependent. At each given step, each SPT SPT_i has a probability P_i of connecting to UE and executing the algorithm. We have studied two sorts of distributions concerning the connectivity of the SPTs : a uniform distribution, where $\forall SPT_i, P_i = 0.01$, and a skewed distribution where $P_i = 0.001$ for 60% of the SPTs, $P_i = 0.1$ for 30% of the SPTs, and $P_i = 0.01$ for 10% of the SPTs.

The latency of the collection phase is the same, regardless of the protocol studied. This latency depends on the connectivity distribution and on the proportion of SPTs in the sample. Figure 6(a) shows that the latency is about 160 steps when considering a sample of 80% of the total 10^6 SPTs for a uniform distribution. For a skewed distribution, it takes approximately 1000 steps to achieve the same results. Note that this latency does not vary much with the total number of SPTs, and depends on P_i, since the protocol will take longer, the less often SPTs connect.

On the same figure, since the times involved are of the same magnitude, we have also plotted the latency of the anonymization phase of the Naïve algorithm. This latency is about 1000 steps, regardless of the proportion of SPTs reconnecting for a uniform distribution and over 8000 steps for a skewed distribution (not plotted). These high numbers are explained by the fact that the same set of SPTs must connect at each phase of the protocol (see Section 3).

Figure 6(b) shows the latency of the anonymization phase of the other algorithms. As expected, the most efficient algorithms are WM_Soft and Robust. Nevertheless, the latency of the WM_Hard algorithm remains acceptable; it is around 100 steps for the uniform distribution and only 30 steps for the skewed distribution. Indeed, in a skewed distribution, the SPTs connecting frequently during the anonymization phase are able to contribute many times. As a conclusion, the latency of the

$Robust$, WM_{Soft} and WM_{Hard} protocols is determined by the latency of their collection phase, itself being related to the size of the sample of interest in the complete population of SPTs.

6.4 Detection of Attacks

Figure 6.4 compares the theoretical lower bound of the detection probability.

We fix $S = 100$, $N = 1.000.000$ and $N_1 = \overline{\mathbb{P}(detect)} \times N$, the number of SPTs which connect only once to the system during the protocol (i.e., with $C_i = 1$). We have chosen $N_1 = 800.000$ (resp. $\overline{\mathbb{P}(detect)} = 0.8$) meaning that 80% of the SPTs will only connect once to the system. In order to provide a worst case scenario, the remaining 20% of the SPTs are assumed to connect only twice during the protocol. Although we feel this is not a realistic distribution, we have chosen it to illustrate the attack detection in the worst case, which is a strongly skewed distribution in favor of N_1. We evaluated $\mathbb{P}(detect)$ in function of $L \in [1 : 30]$ by executing 10.000 runs, with the UE cheating at a given step, and we see if an SPT detects it. We convert this into a detection probability. We also plot the theoretical lower bound as a comparison, and as expected, experimental detection is higher. We see that convergence to a value such as 99% detection can be achieved in practice with a very small number of SPTs (Fig 6.4). Note that the total number of SPTs does not influence the result as long as $N \gg S$.

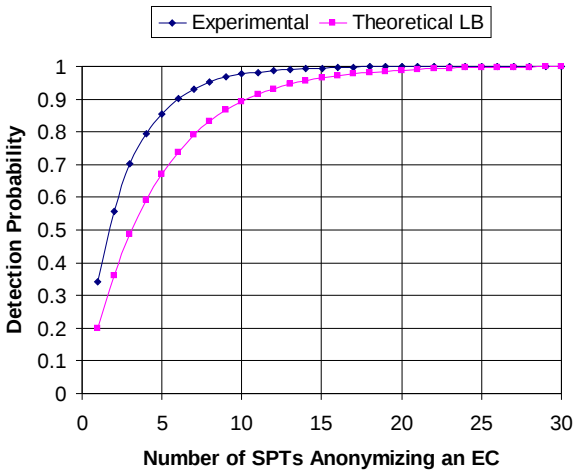


Figure 7: Probability of Attack Detection

7 Related Research Areas

[17] distinguishes two models of PPDP depending on whether the data publisher is trusted or not. In the trusted model, a large body of work has been conducted in the definition of privacy models and on algorithms implementing these models. Our contribution considers the untrusted model and we briefly review below four closely related research areas. Considering the richness of these areas, only a few representative papers are cited in each.

Privacy Preserving Data Collection. A first category of work considers the problem of collecting data so that the data publisher is unable to link any record owner to her response. This can be achieved by shuffling the responses so that the data publisher receives them in a random order. Different shuffling protocols relying on cryptographic techniques have been proposed with various message complexity and collusion resistance between participants [38, 12]. These protocols usually consider that all record owners remain online and assume that responses and record owners are not linkable by content (in other words, responses do not contain quasi-identifiers). An alternative to enforce privacy preserving data collection is for record owners to randomly perturb their sensitive data at submission time. The applicability of random perturbation to data mining problems has been investigated in [11, 10]. However, the more perturbed the data are, the less useful the collected data. In addition, data mining algorithms must be adapted to cope with perturbed data [15]. Hence, both alternatives do not match our problem statement.

Secure Multiparty Computation for Privacy-Preserving Data Mining. The objective of Secure Multiparty Computation (SMC) is for several parties to jointly compute a function without

revealing their input to one another [39]. [20] proved that any problem representable as a circuit can be securely solved, but the computation cost depends on the input's size [35] limiting the applicability of SMC results. By making a trade-off between generality and efficiency, various solutions have however been proposed to solve a wide variety of information sharing problems . [40] discusses the difficulties involved in constructing highly efficient SMC protocols when they are applied to privacy-preserving data mining. SMC usually assumes a connection among partners (e.g., point-to-point or broadcast channel) to support a varying amount of corrupted partners. Hence, SMC hypothesis are different from ours since, in our context, record owners are trustworthy, they are not connected to each other and they agree to reveal their inputs to the data publisher provided their privacy is preserved.

Distributed privacy-preserving data mining algorithms. These works focus on data anonymization of vertically or horizontally partitioned databases. [34] presents a two-party framework that generates k-anonymous data from two vertically partitioned sources without disclosing data from one site to the other. Horizontal partitioned databases have been considered notably in [26, 25]. In [25], the sites engage in a distributed anonymization protocol where each site produces a local anonymized dataset, the union of which forms a virtual database that is guaranteed to be k-anonymous. Sites are mapped to a ring topology randomly. Each site executes its part of the protocol independently and passes the computation result along the ring. The computation invokes a set of SMC protocols to realize a distributed version of the Mondrian algorithm [28]. While our method falls into the horizontally partitioned databases class, our working hypotheses are very different. In our context, each distributed site contains only its own data, making local anonymization senseless, and no direct communication between participating sites is considered.

Multi-Release Privacy-Preserving Data Publishing. Disclosures similar to those caused by Differential Attacks arise in the context of delivering subsequent k-anonymous releases of an evolving dataset [24, 18]. Indeed, comparing the states of a given class before and after the insertion/deletion may reveal the data of the individuals inserted, deleted, or still. Early works [24] considered delaying the dataset modifications that lead to disclosures (e.g., for k-anonymity, waiting that there is a sufficient number of inserts/deletes such that the differences between before/after states of classes yield at least k tuples). However, first, delays are unbounded, and second, the computational complexity may be high due to checking the states of all classes previously released before releasing the new ones. This has led to defining specific models [18] that do not lead to such disclosures. They mainly consist in ensuring a certain invariance between the states of classes. In our context, the problem is different because since we consider the untrusted PPDP model, the UE may introduce into classes breaches leading to a disclosure. Whereas previous solutions lie in the definition of models and techniques performed by the publisher, our techniques aim at armouring the protocol to ensure that classes are free of attacks.

8 Concluding remarks

The increasing suspicion on the ability of DB servers to protect data against attacks and negligence urge the DB community to design credible alternatives to the centralization of personal data. This paper considers a new environment, where private data is stored by individuals into tamper-resistant smart tokens under their control, and enables a privacy safe data exploitation. Unfortunately, this individual-centric environment conflicts with the collective requirement for knowledge-based decision making. This paper tries to reconcile the best of the two worlds. To this end, we propose new distributed PPDP algorithms coping with the smart token's limited availability and the outside world's untrustiness. We prevent attacks launched from the outside world by detecting them with a configurable probability allowing to trade efficiency for security.

This work opens important research perspectives. So far, we have not considered a multi-release mechanism (e.g., a longitudinal study over a given cohort). A naïve way for preventing

multi-release disclosures can consist in first generating high-quality equivalence classes by gathering data from a representative sample of the cohort, and second keeping these classes fixed during the length of the longitudinal study. For each release (except the first), SPTs send only their sensitive data, in the class that generalizes their QID. Although being privacy safe, this ad hoc multi-release mechanism may lead to an uneven publication of the SD among the classes and then to a lower quality of the anonymized data. Because they build classes based on both the identifying and the sensitive data, more sophisticated multi-release models can not be straightforwardly ported to our current context; they require to mitigate the role of UE wrt the computation of classes. To tackle this issue, new ways of sharing data between SPTs must be envisioned, for example based on another architectural layer made of the more available SPTs (e.g., the doctors SPTs) that can communicate directly with one another, e.g., in a peer-to-peer fashion. More generally, our expectation is that this work can pave the way for the definition of other privacy preserving algorithms which consider a large number of highly secure personal tokens seldom connected to an untrusted but highly available infrastructure.

Acknowledgements

The authors wish to thank Indrajit Ray and Indrakshi Ray from Colorado State University and Luc Bouganim from INRIA for fruitful comments on this paper. This work is partially supported by the French National Agency for Research (ANR) under RNTL grant PlugDB and by the French Yvelines District under grant DMSP.

References

[1] European Parliament and Council: Directive 95/46/EC on the protection of individuals with regard to the processing of personal data and on the free movement of such data, 1995.
[2] US Dept. of HHS: Standards for privacy of individually identifiable health information; final rule, 2002.
[3] Computer World. NASA sites hacked, December 2003.
[4] The financial times. chinese military hacked into pentagon, September 2007.
[5] The Washington Post. Consultant Breached FBI's Computers, July 2007.
[6] Times Online. UK government loses personal data on 25 million citizens, November 2007.
[7] FierceHealthIT news. GA hospital health data breach due to outsourcing error, September 2008.
[8] The Telegraph. Prisoner data blunder firm PA Consulting loses multi-million Government contract, September 2008.
[9] ICMCC. Dutch nationwide EHR postponed. Are they in good company?, 2009.
[10] D. Agrawal and C. C. Aggarwal. On the design and quantification of privacy preserving data mining algorithms. In *PODS*, 2001.
[11] R. Agrawal and R. Srikant. Privacy-preserving data mining. *SIGMOD Rec.*, 29(2), 2000.
[12] J. Brickell and V. Shmatikov. Efficient anonymity-preserving data collection. In *KDD*, 2006.
[13] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *USENIX Security Symp.*, 2004.
[14] J. Domingo-Ferrer and V. Torra. A critique of k-anonymity and some of its enhancements. In *Proc. of the Int. Conf. on Availability, Reliability and Security*, 2008.
[15] W. Du and Z. Zhan. Using randomized response techniques for privacy-preserving data mining. In *KDD*, 2003.
[16] Eurosmart. Smart usb token. White paper, Eurosmart, 2008.
[17] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey on recent developments. *ACM Comp. Surveys*, 2010. To appear.

[18] B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Anonymity for continuous data publishing. In *EDBT*, 2008.
[19] O. Goldreich. *Foundations of Cryptography: Vol. 2, Basic Applications*. 2004.
[20] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the ACM Symp. on Theory of Computing*, 1987.
[21] D. Goldschlag, M. Reed, and P. Syverson. Onion routing for anonymous and private internet connections. *Communications of the ACM*, 42, 1999.
[22] L. A. Gordon, M. P. Loeb, W. Lucyshin, and R. Richardson. CSI/FBI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2006.
[23] L. C. Guillou, M. Ugon, and J.-J. Quisquater. Cryptographic authentication protocols for smart cards. *Computer Networks*, 36(4), 2001.
[24] B. Ji-Won, S. Yonglak, B. Elisa, and L. Ninghui. Secure anonymization for incremental datasets. In *VLDB Workshop on Secure Data Management*, 2006.
[25] P. Jurczyk and L. Xiong. Privacy-preserving data publishing for horizontally partitioned databases. Technical report, Emory University, Math&CS Dept., 2008.
[26] M. Kantarcioglu. Privacy-preserving distributed mining of association rules on horizontally partitioned data. *IEEE TKDE*, 16(9), 2004.
[27] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, 2005.
[28] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.
[29] N. Li and T. Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.
[30] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.
[31] D. Praca. Next Generation Smart Card: New Features, New Architecture and System Integration. Deliverable of the Inspired IST project, 2005.
[32] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6), 2001.
[33] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002.
[34] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, 2002.
[35] J. Vaidya and C. Clifton. Privacy-preserving data mining: Why, how, and when. *IEEE Security and Privacy*, 2(6):19–27, 2004.
[36] K. Wang, P. S. Yu, and S. Chakraborty. Bottom-up generalization: A data mining solution to privacy protection. In *ICDM*, 2004.
[37] J. Xu, W. Wang, J. Pei, X. Wang, B. Shi, and A. W.-C. Fu. Utility-based anonymization using local recoding. In *KDD*, 2006.
[38] Z. Yang, S. Zhong, and R. N. Wright. Anonymity-preserving data collection. In *KDD*, 2005.
[39] A. C. Yao. Protocols for secure computations. In *Proc. of the Symp. on Foundations of Computer Science*, 1982.
[40] L. Yehuda and P. Benny. Secure multiparty computation for privacy-preserving data mining. *J. of Privacy and Confidentiality*, 1(1), 2009.
[41] N. Zhang and W. Zhao. Distributed privacy preserving information sharing. In *VLDB*, 2005.