k-Anonymizing Data Hosted in Smart Tokens with a Weakly-Malicious Publisher

Tristan Allard, Benjamin Nguyen, Philippe Pucheral

PRiSM Laboratory,
Univ. of Versailles, France
⟨Fname.Lname⟩@prism.uvsq.fr

INRIA Rocquencourt,
Le Chesnay, France
⟨Fname.Lname⟩@inria.fr

***Abstract*—An increasing number of surveys and articles highlight the failure of database servers to keep confidential data *really* private. Even without considering their vulnerability against external or internal attacks, mere negligences often lead to privacy disasters. The advent of powerful smart portable tokens, combining the security of smart card microcontrollers with the storage capacity of NAND Flash chips, introduces today credible alternatives to the systematic centralization of personal data. Individuals can now store their personal data (e.g., their medical folder) in their own smart tokens, kept under their control, and never disclose *in clear* their private data to the outside untrusted world. However, this new opportunity of managing and protecting personal data conflicts with the objective of implementing knowledge-based decision making tools on top of centralized data. This paper precisely addresses this issue and proposes to adapt the traditional model of Privacy-Preserving Data Publishing (PPDP) to an environment composed of a large set of tamper-resistant smart portable tokens seldom connected to a highly available but untrusted infrastructure. This combination of hypothesis makes the problem fundamentally different from any previously studied PPDP problem we are aware of.**

I. Introduction

The Smart Token: an alternative to personal data centralization. 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 [18]. 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 [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 [5]. In the UK, the personal details of 25 million citizens were lost inadvertently [4]. This growing suspicion sometimes compromises nationwide projects: for instance, the Dutch Electronic Health Record program was canceled due to privacy concerns expressed by citizens [6].

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 constitutes 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.

Reconciliation of privacy with knowledge-based decision making. 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 model where the data publisher is trustworthy, so that record owners are assumed to easily consent providing it with their personal information [14]. As pointed out above, convincing record owners about the legitimacy of this trust 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

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

their data electronically [1], pushing them to find alternatives to a systematic centralization of personal data (e.g. SPTs). In addition, the legislation in several countries authorizes statistical treatments of 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 the individuals' privacy better, 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 to a SPT (because she distrusts central servers) 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.

Motivating scenario. Imagine a scenario where SPTs embed a Personal Data Server providing facilities to store, update, delete and query data 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 into 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 with no need for 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 because she has the assurance that her data will be anonymized the moment 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 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. The vision of a full fledged Personal Data Server embedded in a smart token and managing a wide variety of personal data is drawn in [7]. More generally, 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.

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

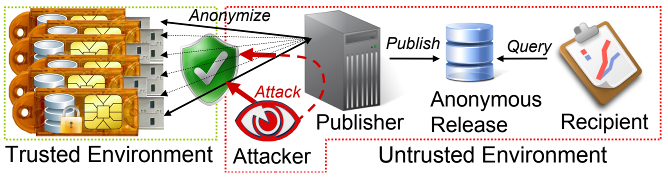


Fig. 1. Anonymous release of data stored on SPTs

Problem positioning. 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 concern has been partially addressed by a limited number of works so far, in a way which unfortunately severely limits their practical scope. Secure Multi-party Computation protocols (SMC) allows several parties to jointly compute a function without revealing their input to one another [28]. Theoretically, any problem representable as a circuit can be securely solved, but the computation cost grows exponentially with the input size [17]. This disqualifies the theoretic general solution for sanitizing widely distributed datasets. More efficient SMC constructs have been proposed to implement specific distributed PPDP protocols [30], [31], [21]. However, strong assumptions are made on the attack model (e.g., introduction of a Trusted Third Party in [21], absence of collusion between the Publisher and a Helper Third Party in [30]) and on the communication model, the underlying cryptographic protocols requiring broadcasting messages among all parties.

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 II introduces the hypothesis our study relies on and states the problem. Sections III and IV discuss respectively the two attack models the UE may endorse in our context, and propose data publishing protocols resistant to these attacks. Section V presents our experiments and demonstrates the practicability of the approach. Finally, section VI concludes by opening exciting research perspectives.

II. Problem Statement

Figure 1 illustrates the functional architecture and modus operandi 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 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.

A. 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), ROM, small static RAM, a small internal stable storage (NOR Flash or EEPROM) and security modules providing 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) [24]. In summary, a SPT can be seen as a basic but very cheap (already today only a few dollars), highly portable, highly secure computer with reasonable storage and computing capacity for personal use. For illustration purposes, Fig. 2 depicts the SPT used in the PlugDB project [8] and in our experiments.

The trustworthiness of SPTs lies in the following factors:

- the SPT's embedded software inherits the tamper resistance of the microcontroller making hardware and side-channel attacks highly difficult,
- this software is certified according to the Common Criteria[5], making software attacks also highly difficult,
- this software can be made auto-administered thanks to its simplicity, contrary to its traditional multi-user server counterpart, thereby precluding DBA attacks,
- 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.

B. Hypothesis on UE

The Untrusted Environment (UE) encompasses the rest of the computing infrastructure, in particular the data publisher. The UE has unlimited computing power and storage capacity, and is available 24/7. The UE may have deviant behavior.

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

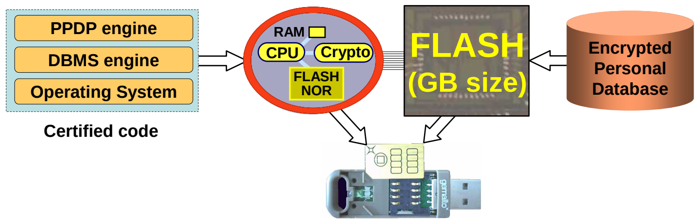


Fig. 2. SPT's internal architecture

Whatever the trust put on the data publisher, the possibility of attacks (e.g., conducted by employees or subcontractors) cannot be disregarded. We model the attackers as follows, according to their intents, 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 the protocol;
- $Weakly-Malicious$: the attacker has *weakly-malicious* intent [29] 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.

Honest-but-Curious is an appropriate attack model for a well established data publisher (e.g., a government agency) which should try to keep sensitive data away from the prying eyes of unscrupulous employees. The Weakly-Malicious model is better adapted to situations where, e.g., the anonymization process is delegated to a third-party providing less guarantees. We do not consider Strongly Malicious attackers because the context of the study is such that the publisher is always liable of its actions and taking the risk of being detected seems too high to be considered. For simplicity, we assume in this paper that the SPT security cannot be broken. We show in [10] that, though highly improbable, hardware attacks can also be defeated by a mechanism guaranteeing a detection probability of attack incompatible with *weakly-malicious* intent [29].

C. Hypothesis on the anonymization algorithm

Data model. We model the dataset to be anonymized 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 uniquely identifying an individual (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).

k-anonymity, a popular privacy criterion. The first anonymization action is to drop ID attributes. 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* [14], has received much consideration not only by academics, but also by legislators [2], [1], and industrials. For example, Privacy Analytics Inc [6] develops a tool that de-identifies datasets based on the k*-anonymity* privacy model, e.g., to make them meet HIPAA privacy rules [2]. k-anonymity [27] 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: it is often achieved by generalizing the QIDs to form equivalence classes (see [14] for a good overview), where each class contains (at least) k tuples sharing the same generalized QID'.

This paper aims at publishing $T'(QID', SD)$, a k-anonymized version of $T(ID, QID, SD)$ in the SPT context. Note that we do not consider here *attribute linkages* [14] 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 [23], the other models preventing attribute linkage [23], [22] are also debated [13]. This paper does not participate in this debate. 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 k-anonymity. This work is expected to pave the way for supporting other anonymization models in the near future.

k-anonymity by generalization. The approach proposed in this paper can work with any algorithm that 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., [26], [19]). 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 $op1 \succeq op2$ (resp. $op1 \preceq op2$) to mean that $op1$ generalizes (resp. specializes) $op2$.

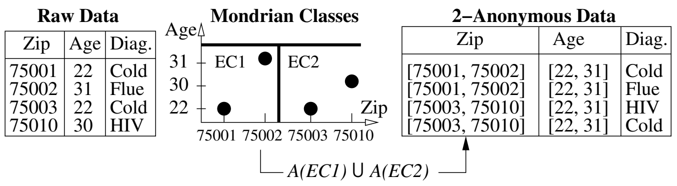
Without loss of generality, we illustrate this paper by using the well known Mondrian algorithm [20] whose basic principles can be intuitively stated as the following:

- 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 raw data, their corresponding Mondrian subspaces, and the 2-anonymous dataset eventually delivered. Anonymizing the tuples whose QIDs are in the equivalence class EC_1 simply means replacing their QIDs by EC_1's generalization nodes. Note that the larger the dataset to be

[6]http://www.privacyanalytics.ca/



Raw Data

Zip	Age	Diag.
75001	22	Cold
75002	31	Flue
75003	22	Cold
75010	30	HIV

2–Anonymous Data

Zip	Age	Diag.
[75001, 75002]	[22, 31]	Cold
[75001, 75002]	[22, 31]	Flue
[75003, 75010]	[22, 31]	HIV
[75003, 75010]	[22, 31]	Cold

Fig. 3. 2-anonymous Equivalence Classes

anonymized, the more precise the generalization nodes will be, while maintaining the same k-anonymity privacy level.

Such an algorithm requires all the QID values to compute the ECs and all the SD values 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 UE, and therefore cannot access any raw data.

D. 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;
2) The anonymized result is as useful (i.e., has the same quality) 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 of the *Honest-but-Curious* and *Weakly-Malicious* attack models.

III. 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 inference or offline calculation it wants to disclose the association between QIDs and SDs.

A. Naïve algorithm

Naïve's phases. 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 s 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, and their communication, 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 can help building such anonymous channels [25].

Algorithm 1 Naïve Algorithm

```
Require: An anonymous communication channel between the SPTs and $UE$, the $k$-anonymity level, the number $s$ of $QID$s required by the class construction phase.
1: Collection phase: For $i = 1,\dots,s$, each $SPT_i$ that connects sends its quasi-identifier $QID_i$ to $UE$.
2: Construction phase: $UE$ computes $EC$, the set of equivalence classes respecting the $k$-anonymity privacy criterion, and publishes their corresponding generalization nodes (denoted $EC_j.\eta$ for the class $EC_j \in EC$).
3: Anonymization phase: Each input $SPT_i$ looks up the equivalence class corresponding to its $QID$ and sends an anonymized tuple in the form of : $\langle j, SD_i \rangle$ where $QID_i \preceq EC_j.\eta$.
```

Correctness. 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 $|EC|$ different k-anonymous equivalence classes, with $\cap EC_{j=1\dots|EC|} = \varnothing$. 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 $\langle j, SD_i \rangle$, where j is the class's ID, it is impossible to distinguish it from at least $(k-1)$ other tuples that will eventually be sent into the j^{th} class, and therefore it is impossible for UE to construct a link between a given SD and a QID with more precision than k.

Unbounded latency. The weakness of the Naïve algorithm lies in parameter s, 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, s is equal to the number of distinct SPTs that connect during the Collection phase. In practical situations, s is likely to be smaller than the total number of existing SPTs. 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.

B. Robust algorithm

The objective of the Robust algorithm is to avoid the unbounded latency of the Naïve algorithm to make it applicable to real life scenarios.

Robust's phases. 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 $\langle QID, E_{\kappa_1}(SD) \rangle$, where E denotes a symmetric encryption scheme (e.g., based on the AES encryption function) taking a secret key κ_1 as a parameter shared by all SPTs (key management is discussed next). The QIDs are still sent in the clear to allow 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), shuffles the class's tuples, and returns to UE anonymized tuples of the form $\langle E_{\kappa_1}^{-1}(E_{\kappa_1}(SD)) \rangle$. The shuffling step avoids UE to link a decrypted SD to its encrypted version based on its position in the returned result. 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. Algorithm 2 summarizes the sequence of phases performed by an SPT running the Robust algorithm.

Classes partially treated. 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. 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) [16] of the ID of each tuple it has processed. As the encryption scheme, the message authentication scheme is parameterized by a key κ_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.

Key management. The security of the Robust algorithm relies on the use of two secret keys (κ_1 for the encryption and κ_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

Algorithm 2 Robust Algorithm

```
Require: The $k$-anonymity level, the number $s$ of $QID$s
    required by the class construction phase, the encryption
    and MAC functions, $E_{\kappa_1}$ and $M_{\kappa 2}$ parameterized by secret
    keys $\kappa_1$ and $\kappa_2$ shared among the SPTs.
 1: Collection phase: For $i = 1,\ldots,s$, each $SPT_i$ that
    connects sends its tuple $\langle QID_i, E_{\kappa_1}(SD_i)\rangle$ to UE.
 2: $UE$ assigns a unique (arbitrary) identifier $TID_i$ to each
    tuple.
 3: Construction phase: $UE$ computes $EC$, the set of equiv-
    alence classes respecting the $k$-anonymity level.
 4: Let $EC_j.T = \{\langle TID_i, E_{\kappa_1}(SD_i)\rangle\}$ $s.t.$ $QID_i \preceq EC_j.\eta$
    represent the set of tuples of the class $EC_j \in EC$.
 5: Let $EC_j.D = \varnothing$ represent the received distinct MACs of
    the tuples anonymized in $EC_j$.
 6: Anonymization phase:
 7: repeat
 8:    $UE$ picks $EC_j$,a class not fully anonymized yet, and
       sends $(EC_j.T, EC_j.D)$ to a connecting $SPT_m$.
 9:    $SPT_m$ shuffles the tuples of $EC_j.T$.
10:    for all $\langle TID_i, E_{\kappa_1}(SD_i)\rangle \in EC_j.T$ do
11:       if $M_{\kappa_2}(TID_i) \notin D_j$ then
12:          $SPT_m$ sends $\langle M_{\kappa_2}(TID_i), E^{-1}_{\kappa_1}(E_{\kappa_1}(SD))\rangle$ to
             $UE$.
13:          $UE$ computes $D_j \leftarrow D_j \cup M_{\kappa_2}(TID_i)$.
14:       end if
15:    end for
16: until $\forall j = 1...|EC|$, $D_j$ has same cardinality as $EC_j.T$
```

spy the hidden content and the computation made by her 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 makes sense given the provable security guarantees they provide (see Sec. II).

Correctness. As in the Naïve algorithm, k-anonymity is guaranteed by the fact that UE never gets access to a $\langle QID_i, SD_i\rangle$ tuple. The only tuples it has at its disposal are in the form $\langle QID_i, E_{\kappa_1}(SD_i)\rangle$, 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.

IV. Weakly-Malicious UE

This section starts with an exhaustive list of the malicious actions upon which a *Weakly-Malicious* UE can base its attacks. The possible actions lie in tampering the data sent to the SPTs during the Anonymization phase in order to infer the links between QIDs and clear text decrypted SD values. Second, it upgrades the Robust algorithm with a set of *safety properties* preventing the weakly-malicious UE from acting maliciously.

A. Malicious Actions

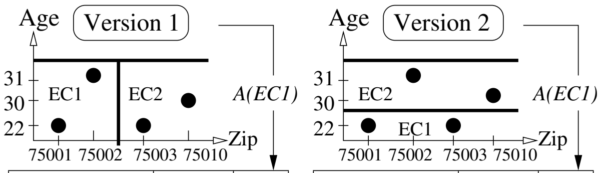
Without loss of generality, any malicious action is either a *Destroy*, *Create*, or *Copy* action. The data upon which UE can apply these actions is the collected tuples and their MAC (used for removing possible duplicates).

Destroy and Create Actions. Destroying t tuple(s) in an equivalence class that contained k tuples leads to a $(k-t)$-anonymous class. For the same reason, creating t false tuples and adding them to $(k-t)$ collected tuples results in a class containing k tuples but that is in fact $(k-t)$-anonymous. In the following, we denote these actions **A1** and **A2** respectively. On the contrary, destroying and creating tuples's MACs does not lead to any disclosure, so does not present any interest to a weakly-malicious UE.

Copy Actions. Tuples can be copied in two ways: either the UE produces a class that contains copies of the same set of tuples (intra-class copy, denoted **A3**), or it produces two classes, one containing a subset of tuples from the other (inter-class copy, denoted **A4**). Intra-class copies lead to a direct reduction of the k-anonymity level of the class, as previous actions do. Inter-class copies lead to inferences that are based on computing the differences between their respective SDs and QIDs. Indeed, (1) the SDs returned for both classes correspond to the collected QIDs belonging to both (the copied subset of tuples), and (2) the SDs returned for only one class correspond to the collected QIDs belonging to that class only. After having computed the differences, the UE is thus able to draw a correspondence between subsets of QIDs and SDs whose cardinality is less than k. These attacks are called *differential attacks*.

Fig. 4 depicts a differential attack. For instance, the version 2 of EC_1 contains one tuple copied from its first version and one new tuple. By computing the differences between the two versions, the attacker infers that (1) $QID = (75001, 22) \rightarrow SD = cold$, (2) $QID = (75002, 31) \rightarrow SD = flue$, and (3) $QID = (75003, 22) \rightarrow SD = HIV$.

Note that previous works have faced similar disclosures in the context of delivering subsequent k-anonymous releases of an evolving dataset [12], [15]. 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 for publishing multiple releases of a dataset lie in the definition of models and techniques performed by the publisher to avoid unsafe publishing, we focus here on reinforcing the protocol to ensure that equivalence classes are free of attacks.



Zip	Age	Diag.
[75001, 75002]	[22, 31]	Cold
[75001, 75002]	[22, 31]	Flue

Zip	Age	Diag.
[75001, 75010]	[22, 22]	Cold
[75001, 75010]	[22, 22]	HIV

Fig. 4. An *overlapping* Differential Attack

B. Safety properties of equivalence classes

To prevent UE from acting maliciously, the equivalence classes must verify the following properties.

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.
- *Origin*: All tuples in the given equivalence class originate from a SPT.
- *Distinguishability*: All tuples in a given equivalence class are distinct.
- *Specialization*: The QID of each tuple specializes its class's generalization node. In other words, each tuple must belong to the proper class.

Global properties. Global properties stem from the relation of a given equivalence class with the whole set of classes already sent to an SPT:

- *Mutual Exclusion*: The Mutual Exclusion property requires that nodes of distinct classes generalize distinct sets of values.
- *Invariance*: The Invariance property requires that each class always be associated with the same content.

Coverage of Malicious Actions. The *Cardinality* property prevents the **A1** actions to endanger the k-anonymity of any equivalence class. The *Origin* property guarantees that any attempt of performing **A2** actions will be detected, and the *Distinguishability* property provides the same guarantee for **A3** actions. Together, the *Specialization*, *Mutual Exclusion*, and *Invariance* properties make **A4** actions inoperative.

As a result, the safety properties defeat all malicious actions defined above, thereby precluding UE to launch any attack based on these actions. Processing equivalence classes satisfying all these properties will generate a k-anonymous result set with certainty.

C. 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 *Distinguishability* property, each SPT is assigned a unique identifier $SPTID_i$ which serves as tuple identifier TID_i. 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 *Origin* property, a simple solution is for each SPT participating in the Collection phase to compute a MAC of its tuple. Finally, the MAC is checked by the SPTs participating in the Anonymization phase. To test the *Specialization* property, QIDs are encrypted within 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.

D. Checking global properties

No global history. Each SPT receives a single equivalence class per session, so checking the global properties *would* require that SPTs share information among them about the classes received. Unfortunately, SPTs are not able to communicate directly 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's 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 trying to build a common history or share a global viewpoint. However, considering that UE has weakly-malicious intentions, we propose to deter it from violating global properties by making any violation visible to SPTs through *caveat actions*.

Caveat actions. The first caveat action (that was also required by the Naïve algorithm) is to use anonymous communication channels between UE and SPTs. This precludes UE to control which SPT receives which equivalence class. Consequently, the probability to send classes violating the global properties to the same SPT is no longer null. 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 the Summary $\mathcal{S}$, asserts the global properties based on $\mathcal{S}$, downloads a class, and checks the consistency between $\mathcal{S}$ and the downloaded class. If UE corrupts the content of a class, it will have to cascade the corruption to $\mathcal{S}$ in order to ensure its consistency with the class, and send the corrupted $\mathcal{S}$ to all the connecting SPTs. Any SPT receiving two disagreeing summaries will detect the attack. As a result, the detection is probabilistic, and its probability depends on the number of SPTs receiving each version of the summary. We have shown that the detection probability can be brought to highly deterring values (e.g., over 0.99) by tuning the minimal number of SPTs receiving each class. We refer the reader to [9] for a detailed discussion on this point.

Minimizing the cost of *Mutual Exclusion*. Although smart implementations of the *Mutual Exclusion* property can be designed in order to avoid a nested-loop style comparison of classes (e.g., in a sort-merge fashion), *Mutual Exclusion* remains one of the most costly checks. However, by slightly extending the *Invariance* property to encompass the classes's nodes in addition to their content, we can avoid the cost of checking *Mutual Exclusion* between summaries received at different moments. Indeed, if during its first connection, a SPT checks that the summary asserts *Mutual Exclusion*, it has only to check that the summary never changes during its following connections to guarantee that classes never overlap.

E. SPT algorithm for Weakly-Malicious UE

Algorithm 3 details the anonymization phase of the algorithm to be executed by each SPT. If a property check is

Algorithm 3 Weakly-Malicious - SPT's Side

```
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, a hash function $H$, and a function returning the leaves that specialize a given node in its generalization hierarchy $L$.
 1: Receive the current Summary $\mathcal{S}$: let $\mathcal{S}.EC$ denote the classes of $\mathcal{S}$, and $EC_i.\delta$ and $EC_i.\eta$ respectively the digest and generalization nodes of the class $EC_i$;
 2: if $\nexists$ previous summary $\mathcal{S}_p$ then
 3:     for all $EC_iEC_j \in \mathcal{S}.EC^2$ $s.t.$ $EC_i.\eta \neq EC_j.\eta$ do
 4:         Check the Mutual Exclusion property: $L(EC_i.\eta) \cap L(EC_j.\eta) = \varnothing$;
 5:     end for
 6: else
 7:     Check the Invariance of the number of classes: $|\mathcal{S}.EC| = |\mathcal{S}_p.EC|$;
 8:     for all $EC_i \in \mathcal{S}$ do
 9:         Check the Invariance of the classes's nodes and contents: $\exists EC_j \in \mathcal{S}_p$ $s.t.$ $EC_i.\eta = EC_j.\eta$ and $EC_i.\delta = EC_j.\delta$;
10:     end for
11: end if
12: Download a class $EC_i$, i.e., its content $EC_i.T$ and its list of processed tuples $EC_i.D$;
13: Check the consistency between $\mathcal{S}$ and the class's content: $EC_i.\delta = H(EC_i.T)$;
14: Shuffle $EC_i.T$;
15: Check the Cardinality property: $|EC_i.T| \geq k$;
16: Init. the TIDs and decrypted tuples sets: $\Theta \leftarrow \varnothing$, $\Delta \leftarrow \varnothing$;
17: for all $t \in EC_i.T$ do
18:     $d \leftarrow E_{\kappa_1}^{-1}(t)$;
19:     Check the Origin property: $M_{\kappa_2}(d) = t.MAC$
20:     Check the Specialization property: $d.QID \preceq EC_i.\eta$;
21:     Check the Distinguishability property: $d.TID \notin \Theta$;
22:     $\Theta \leftarrow \Theta \cup d.TID$;
23:     if $M_{\kappa_2}(d.TID) \notin D_i$ then
24:         $\Delta \leftarrow \Delta \cup d$
25:     end if
26: end for
27: for all $d \in \Delta$ do
28:     Send to UE: $\langle M_{\kappa_2}(d.TID), d.SD)\rangle$,
29: end for
```

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). Due to lack of space, we do not detail the mechanisms (1) used to avoid an SPT from downloading an equivalence class already fully processed, and (2) used to assert that each class has been sent to enough SPTs to guarantee the desired detection probability .

V. Experimental Validation

A. Experimental platform

The algorithms presented in this paper have been implemented and are being integrated in a larger prototype named PlugDB [7]. 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. This prototype has been demonstrated at [11]. 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 District of France and by the French National Research Agency and will soon be experimented in the field in a medical network handling elderly people. The hardware 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 *Naïve*, *Robust*, and *WM* (weakly-malicious). 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 simulation.

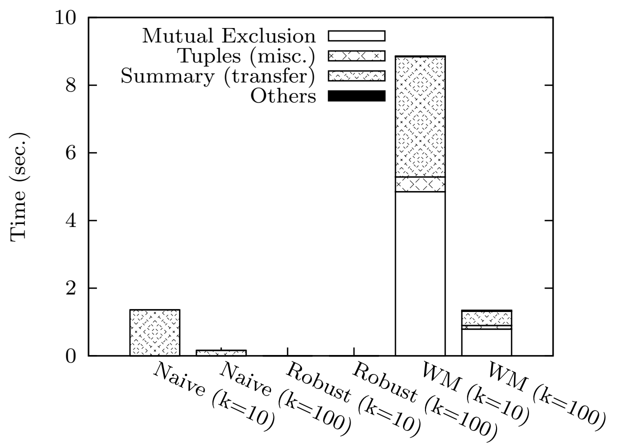
B. Internal Time Consumption

Settings. The cycle-accurate hardware simulator we used for this experiment is clocked at 50Mhz, corresponding to the CPU clock of the target platform. Cryptographic operations 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.

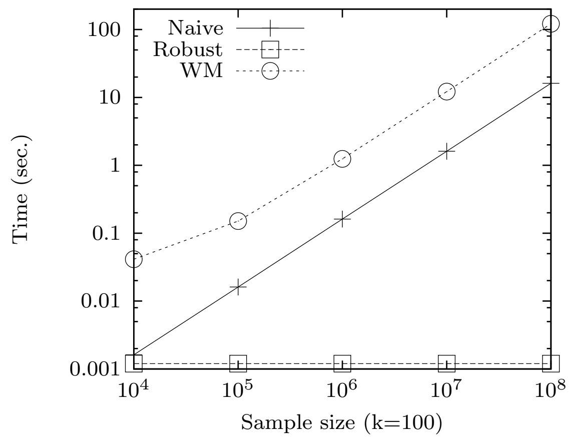
Internal time consumption. Figure 5(a) details the time consumed by a SPT for each basic operation performed during the anonymization protocol. The measure has been performed with a sample of 10^6 SPTs, k varying from 10 to 100. The dataset was synthetically generated; two numerical attributes formed the QID and one string attribute formed the SD.

Depending on the algorithm, the worst case occurs either when k is minimal or maximal. For each algorithm, we plot these two cases to assess whether performance bottlenecks could compromise the feasibility of the approach. The worst case for Naïve and WM occurs when k is low. In this situation, the transfer cost of the Summary for both, and the checking cost of *Mutual Exclusion* for WM only, dominate the other costs because of the high number of equivalence classes. Note that a SPT only checks *Mutual Exclusion* once, i.e., at its first connection. It then checks *Invariance* during its subsequent

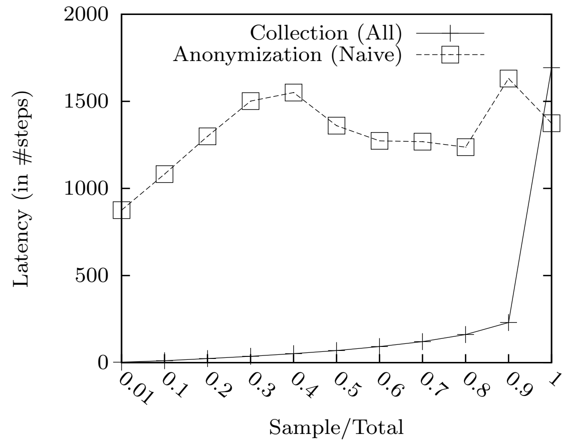
[7]http://www-smis.inria.fr/~DMSP/home.php



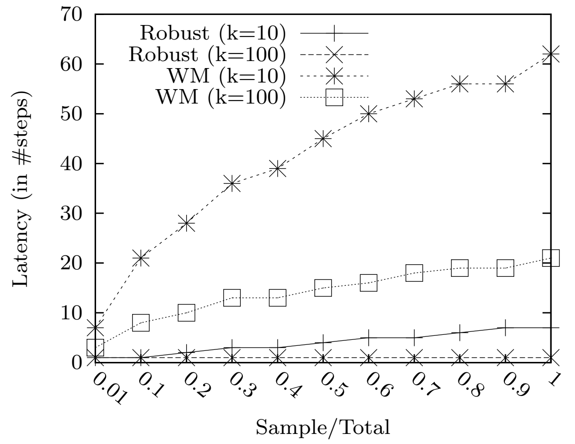
(a) Internal Time Consumption (details)



(b) Internal Time Consumption (scaling)



(c) Latencies: Collection Phase of all and Anonymization Phase of Naïve



(d) Latencies: Anonymization Phases of Robust and WM

Fig. 5. SPT's Time Consumption

connections (see Section IV). Operations related to tuples (ie, transfer, hashing, and decryption) are cheap since Naïve solely uploads its sensitive data, and WM downloads and uploads between k and $2k - 1$ tuples. On the contrary, the worst case for Robust occurs for a high k value, where the tuples' transfer cost overwhelms the other costs. Indeed, since Robust does not make use of any summary, its cost does not depend on the number of classes but on the cardinality of each class. As a conclusion, this figure confirms the feasibility of the approach by showing that, even in the worst cases, the execution time amounts to couples of seconds.

Scaling. 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 and WM 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^{-3} sec.; indeed, it does not use any summary so the time it consumes only depends on k.

C. Latencies

Figures 5(c) and 5(d) plot respectively the latency of the Collection phase and of the Anonymization phase of the protocols, considering a population of 10^6 SPTs. 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 $\mathbb{P}_i$ of connecting to UE and executing the algorithm. We plot and compare below the latencies corresponding to a uniform connection distribution, where $\forall\ SPT_i, \mathbb{P}_i = 0.01$.

Collection phase (all) / Anonymization phase (Naïve). 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 5(c) shows that the latency is about 160 steps when considering a sample of 80% of the total 10^6 SPTs. Note that this latency does not vary much with the total number of SPTs, and depends on $\mathbb{P}_i$ because the less often SPTs connect, the longer the protocol will be. 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 (and would be even bigger for a skewed distribution). These high numbers are explained by the fact that the same set of SPTs must connect at each phase of the protocol (see Section III).

Anonymization phase (Robust and WM). Figure 5(d) shows the latency of the anonymization phase of the Robust and WM algorithms for $k = 10$ and $k = 100$. For Robust, we assumed that a connecting SPT anonymizes exactly one class during its session. The latency is linear and depends on the total number of classes to anonymize divided by the number of connected SPTs per step. The Robust's latency is constant and equal to 1 for $k = 100$ because there are more SPTs that connect during one step than the total number of classes. The WM's latency behaves also linearly. It differs from the Robust's one in that its increased protection incurs the supplementary cost of sending each class to several SPTs in order to guarantee the desired detection probability (in the measures, the minimal detection probability was set to 0.99).

As a conclusion, it appears from these figures that the latency of the Robust and Weakly-Malicious 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. Note that we do not plot the latencies of a skewed distribution merely because it presents a limited interest ((1) the latency of the Collection phase depends on the number and connection probabilities of SPTs that connect few because UE may have to wait for their connection to reach the desired sample size, and (2) the latency of the Anonymization phase depends on the number of SPTs that connect at each step).

VI. Concluding remarks

The increasing suspicion on the ability of DB servers to protect data against attacks and negligences 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 portable tokens under their control. 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 secure distributed PPDP algorithms coping with the smart token's limited availability and the outside world's untrustworthiness.

This work paves the way for new practical privacy-preserving distributed protocols exploiting the emergence of more and more powerful smart tokens. It deserves to be pursued in different directions. First, we have considered in this paper that SPTs were unbreakable. Indeed, breaking the security of a single SPT requires significant resources and is highly improbable due to the high ratio Cost/Benefit of an attack. However, it is important to guarantee that one single attacked SPT cannot compromize the complete system. A preliminary idea to tackle this issue is to partition SPTs into a set of clusters, such that SPTs belonging to different clusters are equipped with distinct cryptographic materials. The impact of clustering on our protocols needs to be more deeply investigated. Another important issue is to generalize the approach to a wider variety of anonymization algorithms. The first results obtained in this paper are a strong incentive to go in this direction.

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] The financial times. chinese military hacked into pentagon, September 2007.

[4] Times Online. UK government loses personal data on 25 million citizens, November 2007.

[5] FierceHealthIT news. GA hospital health data breach due to outsourcing error, September 2008.

[6] ICMCC. Dutch nationwide EHR postponed. Are they in good company?, 2009.

[7] T. Allard, N. Anciaux, L. Bouganim, Y. Guo, L. Le Folgoc, B. Nguyen, P. Pucheral, I. Ray, I. Ray, and S. Yin. Secure personal data servers: a vision paper. In *VLDB*, 2010.

[8] T. Allard, N. Anciaux, L. Bouganim, P. Pucheral, and R. Thion. Seamless access to healthcare folders with strong privacy guarantees. *Journal of Healthcare Delivery Reform Initiatives*, 1, 2009.

[9] T. Allard, B. Nguyen, and P. Pucheral. Safe anonymization of data hosted in smart tokens. Technical Report 2010/25, PRISM Laboratory, 2010.

[10] T. Allard, B. Nguyen, and P. Pucheral. Sanitizing microdata without leak: Combining preventive and curative actions. In *ISPEC*, 2011.

[11] N. Anciaux, L. Bouganim, Y. Guo, P. Pucheral, J.-J. Vandewalle, and S. Yin. Pluggable personal data servers. In *SIGMOD*, 2010.

[12] J.-W. Byun, Y. Sohn, E. Bertino, and N. Li. Secure anonymization for incremental datasets. In *VLDB Workshop on Secure Data Management*, 2006.

[13] 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.

[14] 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.

[15] B. C. M. Fung, K. Wang, A. W.-C. Fu, and J. Pei. Anonymity for continuous data publishing. In *EDBT*, 2008.

[16] O. Goldreich. *Foundations of Cryptography: Vol. 2, Basic Applications*. 2004.

[17] O. Goldreich, S. Micali, and A. Wigderson. How to play any mental game. In *Proc. of the ACM Symp. on Theory of Computing*, 1987.

[18] L. A. Gordon, M. P. Loeb, W. Lucyshin, and R. Richardson. CSI/FBI Computer Crime and Security Survey. Technical report, Computer Security Institute, 2006.

[19] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Incognito: efficient full-domain k-anonymity. In *SIGMOD*, 2005.

[20] K. LeFevre, D. J. DeWitt, and R. Ramakrishnan. Mondrian multidimensional k-anonymity. In *ICDE*, 2006.

[21] F. Li, J. Ma, and J.-h. Li. Distributed anonymous data perturbation method for privacy-preserving data mining. *J. of Zhejiang University*, 10(7), 2009.

[22] N. Li and T. Li. t-closeness: Privacy beyond k-anonymity and l-diversity. In *ICDE*, 2007.

[23] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. In *ICDE*, 2006.

[24] D. Praca. Next Generation Smart Card: New Features, New Architecture and System Integration. Deliverable of the Inspired IST project, 2005.

[25] J. Ren and J. Wu. Survey on anonymous communications in computer networks. *Comput. Commun.*, 33:420–431, 2010.

[26] P. Samarati. Protecting respondents' identities in microdata release. *IEEE TKDE*, 13(6), 2001.

[27] L. Sweeney. k-anonymity: a model for protecting privacy. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.*, 10(5), 2002.

[28] A. C. Yao. Protocols for secure computations. In *Proc. of the Symp. on Foundations of Computer Science*, 1982.

[29] N. Zhang and W. Zhao. Distributed privacy preserving information sharing. In *VLDB*, 2005.

[30] S. Zhong, Z. Yang, and T. Chen. k-anonymous data collection. *Inf. Sci.*, 179(17), 2009.

[31] S. Zhong, Z. Yang, and R. N. Wright. Privacy-enhancing k-anonymization of customer data. In *PODS*, 2005.