Towards a Safe Realization of Privacy-Preserving Data Publishing Mechanisms

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*—This article addresses the issue of adapting the traditional model of Privacy-Preserving Data Publishing (PPDP) to an environment composed of a large number of tamper-resistant Secure Portable Tokens (SPTs) containing private personal data. Our model assumes that the SPTs seldom connect to a highly available but untrusted infrastructure. We illustrate the problem by studying the feasability of the simple generalization privacy mechanism to enforce k-anonymity.**

I. Introduction

Motivation. Individuals are reluctant to entrust their sensitive data to any data server. This suspicion is fueled by security surveys pointing out the vulnerability of database servers against external and internal attacks [10], or the press [1], [3], [2]. Credible alternatives to a systematic centralization of personal data are arising, built upon the emergence of new hardware devices called Secure Portable Tokens (SPTs for short) used in many projects[1] (see [4] for a vision of their use). The counterpart of the privacy risks incurred by centralizing personal data is the opportunity it offers for knowledge-based decision making and typically *Privacy-preserving data publishing* (PPDP). A 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 in order to anonymize it, and ends with an **anonymization phase** where the publisher effectively applies these rules to the data. The sanitized data thus produced can be released to a set of *data recipients* (e.g., a drug company, a public agency, or the public) for inquiry purposes. Our approach aims at reconciling the best of the two worlds, namely the high-level security provided by SPTs and the social benefits of privacy-preserving data publishing techniques.

Related Work. Most PPDP research considers a trusted data publisher [8]. The untrusted case has been investigated in the context of Secure Multi-party Computation protocols (SMC) which allow several parties to jointly compute a function without revealing their input to one another. [15] presents a generic SMC approach, but unfortunately whose cost is exponential in the input size [9]. It is unusable for large datasets. Specific PPDP SMC protocols have been proposed [17], [18], [12]. However, they make strong assumptions on the attack model (e.g., introduction of a Trusted Third Party in [12], absence of collusion between the Publisher and a Helper Third Party in [17]) and on the communication model, 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: 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.

[1]The Health card in the UK, http://www.healthecard.co.uk, the eGK card in Germany, http://www.gematik.de, the HealthSmart Network, http://www.healthsmartnetwork.com/ in the USA.

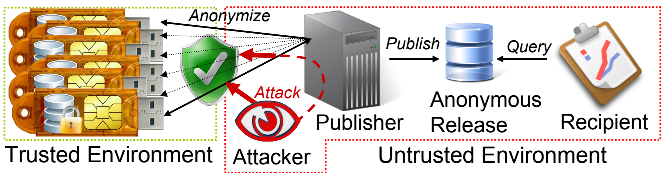


Fig. 1. Anonymous release of data stored on SPTs

Outline. We advocate getting rid of any central point of vulnerability while capitalizing on the large body of work done in the PPDP domain. This paper presents a first step towards this direction, i.e., how to implement the mechanism called *generalization* in order to enforce the simple and popular k-anonymity model. Note that we do not discuss the specific vulnerabilities due to the chosen anonymization model, but simply aim at implementing it without adding extra vulnerabilities. Section II states the problem. Section III describes the core of the solution and section IV gives the main experimental results. Finally, section V concludes by showing how this first step contributes to translating other privacy mechanisms to the SPT context.

II. Problem Statement

This paper focuses on the organization of the collection and anonymization phases at the data source (i.e., at each SPT) while compromising neither privacy nor data utility compared to a trusted central server approach. 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.

Figure 1 illustrates the functional architecture and modus operandi. The *Trusted Environment (TE)* is constituted by the (possibly very large) set of SPTs. Each SPT hosts the personal data of a single record owner. It can take part in a distributed computation involving data issued from multiple record owners since all the SPTs trust each other. The *Untrusted Environment (UE)* encompasses the rest of the computing infrastructure, in particular the data publisher and the data recipients.

Hypothesis on TE. A SPT can be seen as a basic but very cheap (today only a few dollars), highly portable, highly secure computer with reasonable storage and computing capacity for personal use. We refer to [4] for a full justification of the SPT's trustworthiness.

Hypothesis on UE. UE has unlimited computing power and storage capacity, and is available 24/7. The UE may have deviant behavior of two types:

- $Honest-but-Curious$: the attacker obeys the protocol it is participating in but tries to infer confidential data;
- $Weakly-Malicious$: the attacker has *weakly-malicious* intent [16] in that it cheats the protocol to disclose confidential data only if (1) the TE does 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). The Weakly-Malicious model is better adapted to situations where 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. We assume in this paper that the SPT is not attacked. We show in [6] that, though highly improbable, hardware attacks can also be defeated by a mechanism guaranteeing a detection probability defeating *weakly-malicious* intent.

Hypothesis on the anonymization algorithm. 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. QID is a set of attributes, called *quasi-identifiers*, that could potentially identify an individual. The SD attributes contain sensitive data. In this article, for simplicity's sake, we consider the well studied generalization mechanism that drops ID and coarsens QID. Generalization is used to enforce the k-anonymity [14] privacy model that consists in building *equivalence classes* of at least k tuples indistinguishable wrt their (generalized) QID.

III. Preventing Dishonest Actions

A. Honest-but-Curious UE

A *Honest-but-Curious* adversary attempts to draw inferences that increase his knowledge about the link between one or more QIDs and SDs. A Honest-but-Curious adversary does not deviate from the protocol that he is supposed to execute. Keeping the link hidden from inferences throughout the Collection/Construction/Anonymization phases led us to the following adaptation.

During the **Collection phase**, each connecting SPT (that agrees to participate in the study) sends the publisher its QID in clear and its SD encrypted by a symmetric encryption scheme (e.g., based on the AES encryption function). The encryption scheme takes as parameter a secret key shared by all SPTs (key management is discussed in the next paragraph). The publisher decides to stop collecting tuples and to launch the **Construction phase** once it has received enough QIDs to build equivalence classes precise enough for its application-dependent requirements. During the **Anonymization phase**, any SPT that connects downloads a class (or more if its connection duration allows), and returns to UE the decrypted SDs it contains (in random order). The returned SDs are k-anonymous with certainty because the partial states observed by UE give no information allowing it to infer the association between a given SD and QID in clear with more precision than k. The part of the protocol allowing an SPT to treat partially an equivalence class (e.g., if it has been disconected during the protocol) does not affect the core of the protocol, we do not detail it here but refer the interested reader to [5]. Algorithm 1, called *Robust*, summarizes the adapted phases.

Algorithm 1 Robust

```
1: Collection phase: Each SPT that connects to UE sends its
   tuple $\langle QID, E_\kappa(SD)\rangle$, where $E_\kappa$ is a symetric encryption
   function parameterized by the secret key $\kappa$.
2: Construction phase: UE computes the equivalence
   classes through a generalization-based mechanism.
3: Anonymization phase: Each SPT that connects to UE
   downloads an equivalence class (or more if its connection
   allows), and uploads the corresponding $k$ $SD$ decrypted
   and shuffled.
```

Sharing cryptographic material among all SPTs does not hurt the Weakly-Malicious assumptions since SPTs are considered unbreakable (see [6] when this assumption does not hold). We do the simplifying assumption that the SPT provider pre-installs the cryptographic materials inside the SPT's secured chip, though more dynamic protocols could easily be devised. Let us stress that even the SPT's owner cannot spy the hidden content or 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 her smart card microcontroller).

B. Weakly-Malicious UE

A Weakly-Malicious UE has the additional ability to cheat the protocol only if the cheat remains undetected by the trusted environment. The weakly-malicious protocol (WM) differs from Robust only in that SPTs check the safety of equivalence classes before sanitizing them. We focus below on explaining the safety properties of classes without detailing the whole protocol (similar to Robust).

1) Weakly-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 dataset.

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.

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*.

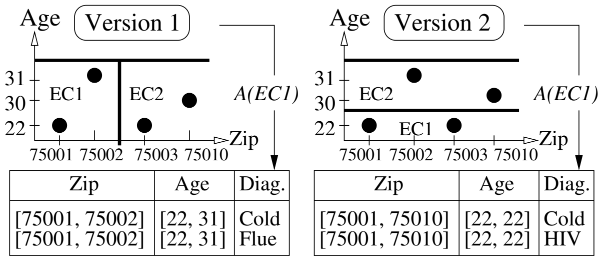
Fig. 2 depicts a differential attack. For instance, the version 2 of EC_1 contains one tuple copied from its first version and one new tuple. By computing the differences between the two versions, the attacker infers that (1) $QID = (75001, 22) \rightarrow SD = cold$, (2) $QID = (75002, 31) \rightarrow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

2) 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



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. 2. An *overlapping* Differential Attack

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.

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 *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.

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 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 aim at providing to each SPT an invariant global view over the dataset, based on a data structure called the *Summary*. The summary describes both the content and the boundaries of the classes, and allows SPTs to enforce global properties

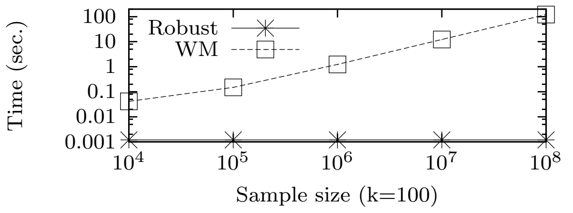


Fig. 3. SPT's Internal Time Consumption Scaling

collectively. For space reasons, we do not detail further the caveat actions, but refer the interested reader to [5] for a complete explanation.

IV. Experimental Validation

Experimental platform. The algorithms presented in this paper have been implemented and are being integrated in a larger prototype named PlugDB[2] and demonstrated at [7]. PlugDB aims at managing secure portable medical-social folders. The hardware platform is provided by Gemalto (the world leader in smart-cards). Since it is still under test, the performance measurements have been conducted on a cycle-accurate hardware emulator. The protocols considered for the experiments cover both *Honest-but-Curious* (denoted Robust) and *Weakly-Malicious UE* (denoted WM) preventions. We concentrate below on the evaluation of the time spent internally in each SPT to participate to each phase of the protocol. We also measured the latency but due to lack of space we leave this point in [5], the main result being that the latency of the collection phase dominates the latency of the anonymization phase. The detailed discussion and figures can be found in [5].

Internal time consumption. The internal time consumption's 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. Their distribution has no impact on the protocol (it was uniform in these experiments). The worst case for WM occurs when k is low. In this situation, the transfer cost of the Summary and the checking cost of *Mutual Exclusion* dominate the other costs because of the high number of equivalence classes. On the contrary, since Robust does not use any Summary, its worst case occurs for a high k value, where the tuples' transfer cost overwhelms the other costs. Even in the worst cases, the execution time amounts to couples of seconds.

Scaling. Fig. 3 shows the scaling of the protocols wrt to the number of SPTs in the sample with $k = 100$. WM scales linearly with the number of SPTs sampled (from 10^{-2} millisec for 10^4 SPTs, to 10^2 sec for a nationwide 10^8 SPTs). This linearly increasing cost reflects the linear increase of the Summary's size. Robust remains constant, around 10^{-3} sec.

V. Beyond k-Anonymity

The diversity of data recipients in terms of usage and trust preclude the election of a "one-size-fits-all" model. The enforcement of the k-anonymity model in a context where data is hosted on smart tokens is a first step towards the design of a broad framework able to adapt to various privacy criterion (e.g., l-diversity [13], PRAM [11]). For this, the key enabler lies in broadening the scope of the techniques explained in this paper, to privacy mechanisms other than generalization. We strongly believe that this is possible. First, PPDP aims at obfuscating the link between identifying and sensitive data: some mechanisms act on the identifying part of data (e.g., generalizing the quasi-identifiers), some others on the sensitive part (e.g., perturbing the sensitive data). In our context, it appears that the construction phase, run by UE, can act indistinctly on either the former or the latter. Second, differential attacks are *not* specific to any mechanism. Their threat appears as soon as the collected dataset is divided into smaller partitions, which is necessary in a context where the sanitization task is distributed to light trusted devices that are unable to treat the dataset as a whole. Hence, (variations of) the safety properties defined in this paper are necessary for securing the realization of other mechanisms. Third and last, the preliminary results demonstrate the feasibility of the approach and are a strong incentive to pursue in this direction.

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

References

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

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

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

[4] 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.

[5] T. Allard, B. Nguyen, and P. Pucheral. k-anonymizing data hosted in smart tokens with a weakly-malicious publisher. Technical Report 2011/27, PRISM Laboratory, 2011.

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

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

[8] B. C. M. Fung, K. Wang, R. Chen, and P. S. Yu. Privacy-preserving data publishing: A survey of recent developments. *ACM Comput. Surv.*, 42:14:1–14:53, 2010.

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

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

[11] J. Gouweleeuw, P. Kooiman, L. Willenborg, and P.-P. de Wolf. Post randomisation for statistical disclosure control: Theory and implementation. *Journal of Official Statistics*, 14, 1998.

[12] 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.

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

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

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

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

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

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