Privacy-Preserving Updates to Anonymous and Confidential Databases

Alberto Trombetta, Wei Jiang, Elisa Bertino and Lorenzo Bossi

Abstract—Suppose Alice owns a k-anonymous database and needs to determine whether her database, when inserted with a tuple owned by Bob, is still k-anonymous. Also, suppose that access to the database is strictly controlled, because for example data are used for certain experiments that need to be maintained confidential. Clearly, allowing Alice to directly read the contents of the tuple breaks the privacy of Bob (e.g., a patient's medical record); on the other hand, the confidentiality of the database managed by Alice is violated once Bob has access to the contents of the database. Thus, the problem is to check whether the database inserted with the tuple is still k-anonymous, without letting Alice and Bob know the contents of the tuple and the database respectively. In this paper, we propose two protocols solving this problem on suppression-based and generalization-based k-anonymous and confidential databases. The protocols rely on well-known cryptographic assumptions, and we provide theoretical analyses to proof their soundness and experimental results to illustrate their efficiency.

Index Terms—Privacy, anonymity, data management, secure computation.

1 Introduction

It is today well understood that databases represent an important asset for many applications and thus their security is crucial. Data confidentiality is particularly relevant because of the value, often not only monetary, that data have. For example, medical data collected by following the history of patients over several years may represent an invaluable asset that needs to be adequately protected. Such a requirement has motivated a large variety of approaches aiming at better protecting data confidentiality and data ownership. Relevant approaches include query processing techniques for encrypted data and data watermarking techniques. Data confidentiality is not however the only requirement that needs to be addressed.

Today there is an increased concern for privacy. The availability of huge numbers of databases recording a large variety of information about individuals makes it possible to discover information about specific individuals by simply correlating all the available databases. Although confidentiality and privacy are often used as synonyms, they are different concepts: data confidentiality is about the difficulty (or impossibility) by an unauthorized user to learn anything about data stored in the database. Usually, confidentiality is achieved by enforcing an access policy, or possibly by using some cryptographic tools. Privacy relates to what data can be safely disclosed without leaking sensitive information regarding the legitimate owner [5]. *Thus, if one asks whether confidentiality is still required once data have been anonymized, the reply is yes if the anonymous data have a business value for the party owning them or the unauthorized disclosure of such anonymous data may damage the party owning the data or other parties.* (Note that under the context of this paper, the term anonymized or anonymization means identifying information is removed from the original data to protect personal or private information. There are many ways to perform data anonymization. We only focus on the k-anonymization approach [28], [32].)

To better understand the difference between confidentiality and anonymity, consider the case of a medical facility connected with a research institution. Suppose that all patients treated at the facility are asked before leaving the facility to *donate* their personal health care records and medical histories (under the condition that each patient's privacy is protected) to the research institution, which collects the records in a research database. To guarantee the maximum privacy to each patient, the medical facility only sends to the research database an anonymized version of the patient record. Once this anonymized record is stored in the research database, the non-anonymized version of the record is removed from the system of the medical facility. Thus the research database used by the researchers is anonymous. Suppose that certain data concerning patients are related to the use of a drug over a period of four years and certain side-effects have been observed and recorded by the researchers in the research database. It is clear that these data (even if anonymized) need to be kept confidential and accessible only to the few researchers of the institution working on this project, until further evidence is

- *A. Trombetta is with the Department of Computer Science and Communication, University of Insubria, Italy.
  E-mail: alberto.trombetta@uninsubria.it*
- *W. Jiang is with the Department of Computer Science, Missouri University of Science and Technology, Rolla, MO, USA.
  E-mail: wjiang@mst.edu*
- *E. Bertino is with the Department of Computer Science, Purdue University, West Lafayette, IN, USA.
  E-mail: bertino@cs.purdue.edu*
- *L. Bossi is with the Department of Computer Science and Communication, University of Insubria, Italy.*

found about the drug. If these anonymous data were to be disclosed, privacy of the patients would not be at risk; however the company manufacturing the drug may be aversely affected.

Recently, techniques addressing the problem of privacy via data anonymization have been developed, thus making it more difficult to link sensitive information to specific individuals. One well-known technique is k-anonymization [28], [32]. Such technique protects privacy by modifying the data so that the probability of linking a given data value, for example a given disease, to a specific individual is very small. So far, the problems of data confidentiality and anonymization have been considered separately. However, a relevant problem arises when data stored in a confidential, anonymity-preserving database need to be updated. The operation of updating such a database, e.g., by inserting a tuple containing information about a given individual, introduces two problems concerning both the anonymity and confidentiality of the data stored in the database and the privacy of the individual to whom the data to be inserted are related: *(i) Is the updated database still privacy-preserving*? and *(ii) Does the database owner need to know the data to be inserted*? Clearly, the two problems are related in the sense that they can be combined into the following problem: can the database owner decide if the updated database still preserves privacy of individuals without directly knowing the new data to be inserted? The answer we give in this work is affirmative.

It is important to note that assuring that a database maintains the privacy of individuals to whom data are referred is often of interest not only to these individuals, but also to the organization owning the database. Because of current regulations, like HIPAA [19], organizations collecting data about individuals are under the obligation of assuring individual privacy. It is thus in their interest to check that data that are entered in their databases do not violate privacy, and to perform such a verification without seeing any sensitive data of an individual.

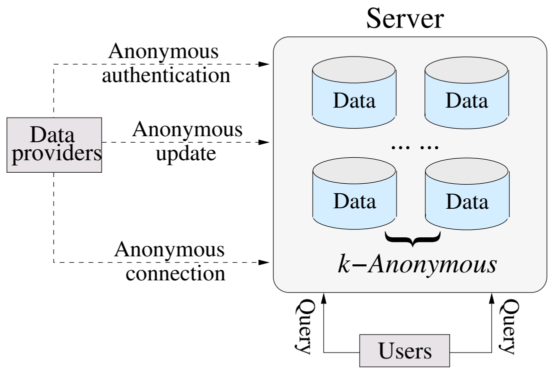


Fig. 1. Anonymous Database System

1.1 Problem Statement

Figure 1 captures the main participating parties in our application domain. We assume that the information concerning a single patient (or data provider) is stored in a single tuple, and DB is kept confidentially at the server. The users in Figure 1 can be treated as medical researchers who have the access to DB. Since DB is anonymous, the data provider's privacy is protected from these researchers. (Note that to follow the traditional convention, in Section 4 and later sections, we use Bob and Alice to represent the data provider and the server respectively.)

As mentioned before, since DB contains privacy-sensitive data, one main concern is to protect the privacy of patients. Such task is guaranteed through the use of anonymization. Intuitively, if the database DB is anonymous, it is not possible to infer the patients' identities from the information contained in DB. This is achieved by blending information about patients. See Section 3 for a precise definition. Suppose now that a new patient has to be treated. Obviously, this means that the database has to be updated in order to store the tuple t containing the medical data of this patient.

The modification of the anonymous database DB can be naively performed as follows: the party who is managing the database or the server simply checks whether the updated database $DB \cup \{t\}$ is still anonymous. Under this approach, the *entire* tuple t has to be revealed to the party managing the database server, thus violating the privacy of the patient. Another possibility would be to make available the entire database to the patient so that the patient can verify by himself/herself if the insertion of his/her data violates his/her own privacy. This approach however requires making available the entire database to the patient thus violating data confidentiality.

In order to devise a suitable solution, several problems need to be addressed: Problem 1: *without revealing the contents of t and DB, how to preserve data integrity by establishing the anonymity of $DB \cup \{t\}$?* Problem 2: *once such anonymity is established, how to perform this update?* Problem 3: *what can be done if database anonymity is not preserved*? Finally, problem 4: *what is the initial content of the database, when no data about users has been inserted yet*? In this paper, we propose two protocols solving Problem 1, which is the central problem addressed by our paper. However, because the other problems are crucial from a more practical point of view, we discuss them as well in Section 7.

Note that to assure a higher level of anonymity to the party inserting the data, we require that the communication between this party and the database occurs through an *anonymous connection*, as provided by protocols like Crowds [27] or Onion routing [26]. This is necessary since traffic analysis can potentially reveal sensitive information based on users' IP addresses. In addition, sensitive information about the party inserting

Requirement	*Objective*	*Protocol*
Anonymous connection	Protect IP address and sensitive info	Crowds [27], Onion Routing [26]
Anonymous authentication	Protect sensitive authentication info	Policy-hiding access control [20]
Anonymous update	Protect non-anonymous data	**Proposed in this paper**

TABLE 1
Anonymous Database System Requirements

the data may be leaked from the access control policies adopted by the anonymous database system, in that an important requirement is that only authorized parties, for example patients, should be able to enter data in the database. Therefore, the question is how to enforce authorization without requiring the parties inserting the data to disclose their identities. An approach that can be used is based on techniques for *user anonymous authentication* and credential verification [20]. The above discussion illustrates that the problem of anonymous updates to confidential databases is complex and requires the combination of several techniques, some of which are proposed for the first time in this paper. Figure 1 summarizes the various phases of a comprehensive approach to the problem of anonymous updates to confidential databases, while Table 1 summarizes the required techniques and identifies the role of our techniques in such approach.

1.2 Proposed Solutions

All protocols we propose to solve Problem 1 rely on the fact that the anonymity of DB is not affected by inserting t if the information contained in t, properly anonymized, is already contained in DB. Then, Problem 1 is equivalent to privately checking whether there is a match between (a properly anonymized version of) t and (at least) one tuple contained in DB. The first protocol is aimed at suppression-based anonymous databases, and it allows the owner of DB to properly anonymize the tuple t, without gaining any useful knowledge on its contents and without having to send to t's owner newly generated data. To achieve such goal, the parties secure their messages by encrypting them. In order to perform the privacy-preserving verification of the database anonymity upon the insertion, the parties use a commutative and homomorphic encryption scheme. The second protocol (see Section 5) is aimed at generalization-based anonymous databases, and it relies on a secure set intersection protocol, such as the one found in [3], to support privacy-preserving updates on a generalization-based k-anonymous DB.

The paper is organized as follows: Section 2 reviews related work on anonymity and privacy in data management. Section 3 introduces notions about anonymity and privacy that we need in order to define our protocols and prove their correctness and security. The protocols are defined respectively in Section 4 and Section 5 with proofs of correctness and security. Section 6 analyzes the complexity of the proposed protocol and presents experimental complexity results we obtained by running such protocols on real-life data. Section 7 concludes the paper and outlines future work.

2 Related Work

A preliminary approach to this problem was investigated in [33]. However these protocols have some serious limitations, in that they do not support generalization-based updates, which is the main strategy adopted for data anonymization. Therefore, if the database is not anonymous with respect to a tuple to be inserted, the insertion cannot be performed. In addition one of the protocols is extremely inefficient. In the current paper, we present two efficient protocols, one of which also support the private update of a generalization-based anonymous database. We also provide security proofs and experimental results for both protocols. So far no experimental results had been reported concerning such type of protocols; our results show that both protocols perform very efficiently. In what follows, we briefly address other research directions relevant for our work.

The first research direction deals with algorithms for database anonymization. The idea of protecting databases through data suppression or data perturbation has been extensively investigated in the area of statistical databases [1]. Relevant work has been carried out by Sweeney [32], who initially proposed the notion of k-anonymity for databases in the context of medical data, and by Aggarwal et al. [2], who have developed complexity results concerning algorithms for k-anonymization. The problem of computing a k-anonymization of a set of tuples while maintaining the confidentiality of their content is addressed by Zhong et al. [35]. However, these proposals do not deal with the problem of private updates to k-anonymous databases. The problem of protecting the privacy of time-varying data has recently spurred an intense research activity which can be roughly divided into two broad groups depending on whether data is continuously released in a stream and anonymized in an on-line fashion, or data is produced in different releases and subsequently anonymized in order to prevent correlations among different releases. Relevant work in this directions include [9], [14], [18], [21] and [34]. Again, none of these works address the problem of checking that

The second research direction is related to Secure Multi-party Computation (SMC) techniques. SMC represents an important class of techniques widely investigated in the area of cryptography. General techniques for performing secure computations are today available [16]. However, these techniques generally are not efficient. Such shortcomings has motivated further research in order to devise more efficient protocols for particular problems. Of particular relevance for data management are the techniques presented in [3], [13], in which the authors address the problems of efficiently and privately computing set intersection and (in case of [3]) database oriented operations, such as joins.

The third research direction is related to the area of private information retrieval, which can be seen as an application of the secure multi-party computation techniques to the area of data management. Here, the focus is to devise efficient techniques for posing expressive queries over a database without letting the database know the actual queries [10], [22]. Again, the problem of privately updating a database has not been addressed in that these techniques only deal with data retrieval.

Finally, the fourth research direction is related to query processing techniques for encrypted data [7], [17], [30]. These approaches do not address the k-anonymity problem since their goal is to encrypt data, so that their management can be outsourced to external entities. Thus, the goal is to protect data confidentiality from the external entities managing the data; however, data are fully available to the clients, which is not the case under our approach.

3 Basic Definitions and Primitives

3.1 Anonymity Definitions

We consider a table $T = \{t_1, \ldots, t_n\}$ over the attribute set A. The idea is to form subsets of indistinguishable tuples by masking the values of some well-chosen attributes. In particular, when using a *suppression-based* anonymization method, we mask with the special value $*$, the value deployed by Alice for the anonymization. When using a *generalization-based* anonymization method, original values are replaced by more general ones, according to apriori established *value generalization hierarchies* (VGHs) [32]. We adopt the following notations thereafter:

- Quasi-Identifier (QI): a set of attributes that can be used with certain external information to identify a specific individual.
- $T[QI]$: $T[QI]$ is the projection of T to the set of attributes contained in QI.

Definition 3.1: $T[QI]$ satisfies k-anonymity if and only if each record in it appears at least k times [32].

With respect to suppression-based anonymization [23], [32] QI can be classified into two subsets: suppressed attributes QI^s and non-suppressed attributes $QI^{\overline{s}}$. When T is k-anonymous, then for every tuple $t \in T$, there exists a subset of tuples $\{t_1, \ldots, t_z\} \subseteq T$ ($z \geq k-1$) such that for every attribute in QI^s, the corresponding value is replaced by $*$ (indicating suppressions of the original values), and for every attribute in $QI^{\overline{s}}$, the condition $t[QI^{\overline{s}}] = t_i[QI^{\overline{s}}]$ holds. Suppose $QI = \{$AREA, POSITION, SALARY$\}$, Table 3 shows a suppression based k-anonymization with $k = 2$. Choosing the suppressed attributes for every tuple of T is referred as the *anonymization* problem, and finding the anonymization that minimizes the number of masked values is an NP-hard problem [2], [23].

TABLE 2
Original Dataset

AREA	POSITION	SALARY
Data Mining	Associate Professor	\$90,000
Intrusion Detection	Assistant Professor	\$78,000
Handheld Systems	Research Assistant	\$17,000
Handheld Systems	Research Assistant	\$15,500
Query Processing	Associate Professor	\$100,000
Digital Forensics	Assistant Professor	\$78,000

TABLE 3
Suppressed Data with $k = 2$

AREA	POSITION	SALARY
*	Associate Professor	*
*	Assistant Professor	*
Handheld Systems	Research Assistant	*
Handheld Systems	Research Assistant	*
*	Associate Professor	*
*	Assistant Professor	*

TABLE 4
Generalized Data with $k = 2$

AREA	POSITION	SALARY
Database Systems	Associate Professor	[61k, 120k]
Information Security	Assistant Professor	[61k, 120k]
Operating Systems	Research Assistant	[11k, 30k]
Operation Systems	Research Assistant	[11k, 30k]
Database Systems	Associate Professor	[61k, 120k]
Information Security	Assistant Professor	[61k, 120k]

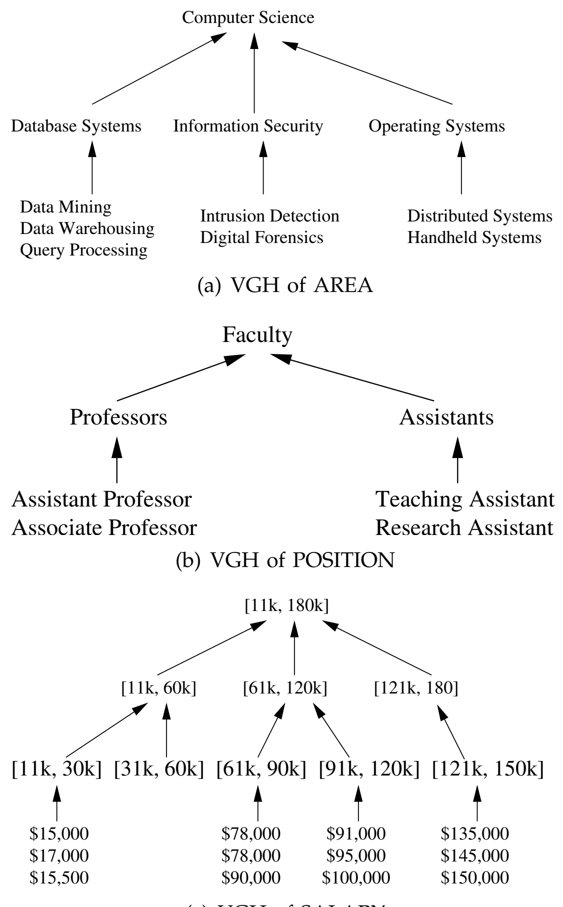
TABLE 5
The witness Set

AREA	POSITION	SALARY
Database Systems	Associate Professor	[61k, 120k]
Information Security	Assistant Professor	[61k, 120k]
Operating Systems	Research Assistant	[11k, 30k]

For generalization-based anonymization [32], we assume that each attribute value can be mapped to a more general value. The main step in most generalization based k-anonymity protocols is to replace a specific value with a more general value. For instance, Figure 2 contains VGHs for attributes AREA, POSITION and SALARY. According to the VGH of AREA, we say that the value "Data Mining" can be generalized to "Database Systems". (Suppression can be viewed as an extreme form of generalization, in which the generalized attributes cannot be further generalized.) Let T refer to Table 4 and $QI = \{$AREA, POSITION, SALARY$\}$. Then T ($T[QI]$) satisfies 2-anonymity. According to the three VGHs, it is easy to verify that the original data represented by Table 2 can be generalized to T. When T is k-anonymous, we can delete duplicate tuples, and we call the resulting set the *witness* set of T. Table 5 represents a *witness* set of Table 4.

3.2 Cryptographic Primitives

The protocol in Section 4 uses a commutative, product-homomorphic encryption scheme E. Loosely speak-



(a) VGH of AREA

(b) VGH of POSITION

(c) VGH of SALARY

Fig. 2. Value Generalization Hierarchies

ing, a commutative, product-homomorphic encryption scheme ensures that the order in which encryptions are performed is irrelevant (commutativity) and it allows to consistently perform arithmetic operations over encrypted data (homomorphic property). Further, for the security proofs we require that the encryption scheme E satisfies the indistinguishability property. We extend the definition of commutative, indistinguishable encryption scheme presented in [3], in order to obtain an encryption scheme which also product-homomorphic. Given a finite set $\mathcal{K}$ of keys and a finite domain $\mathcal{D}$, a *commutative, product-homomorphic encryption scheme* E is a polynomial time computable function $E : \mathcal{K} \times \mathcal{D} \to \mathcal{D}$ satisfying the following properties:

1) *Commutativity*. For all key pairs $K_1, K_2 \in \mathcal{K}$ and value $d \in \mathcal{D}$, the following equality holds:

$$E_{K_1}(E_{K_2}(d)) = E_{K_2}(E_{K_1}(d)) \tag{1}$$

2) *Product-homomorphism*. For every $K \in \mathcal{K}$ and every value pairs $d_1, d_2 \in \mathcal{D}$, the following equality holds:

$$E_K(d_1) \cdot E_K(d_2) = E_K(d_1 \cdot d_2) \tag{2}$$

3) *Indistinguishability* [15]. It is infeasible to distinguish an encryption from a randomly chosen value in the same domain and having the same length. In other words, it is infeasible for an adversary, with finite computational capability, to extract information about a plaintext from the ciphertext.

We will use the indistinguishability property when proving in Section 4.1 the security of Protocols 4.1 and 5.1. As an example of commutative, product-homomorphic, indistinguishable encryption scheme, we consider the following setting. Let $\mathcal{D}$ be a subgroup of prime order q of $\mathbb{Z}_p$, with p prime, such that q is equal to $(p-1)/2$ and is prime as well. Let $d \in \mathcal{D}$ and $K \in \mathcal{K} = \{0, \ldots, q-1\}$. Assuming the Decision Diffie-Hellman assumption [6], the function

$$E_K(d) \overset{\text{def}}{=} d^K \bmod p \tag{3}$$

is a commutative, product-homomorphic, indistinguishable encryption scheme [3].

Finally, following [8], [25], we introduce a simple tuple coding scheme that we will use in Protocol 5.1: Alice and Bob agree on a set $\{g_1, g_2 \ldots, g_u\}$ of generators of $\mathcal{D}$. Let d be a tuple $\langle d_1, d_2, \ldots, d_u \rangle$ with elements taken from $\mathbb{Z}_q$, we define the *encoding* of a tuple d as

$$c(\langle d_1, d_2, \ldots, d_u \rangle) = \prod_{i=1}^{u} g_i^{d_i} \bmod q \tag{4}$$

Such coding scheme is known in literature as DLREP (Discrete Logarithm Representation). Note that such coding acts like a collision-free hash function, as explained in [24, Section 4.3].

4 Private Update for Suppression-based Anonymous and Confidential Databases

In this section, we assume that the database is anonymized using a suppression-based method. Note that our protocols are not required to further improve the privacy of users other than that provided by the fact that the updated database is still k-anonymous. Suppose that Alice owns a k-anonymous table T over the QI attributes. Alice has to decide whether $T \cup t$ – where t is a tuple owned by Bob – is still k-anonymous, without directly knowing the values in t (assuming t and T have the same schema). This problem amounts to decide whether t matches any tuple in T on the non-suppressed QI attributes. If this is the case, then t, properly anonymized, can be inserted into T. Otherwise, the insertion of t into T is rejected.

A trivial solution requires as a first step Alice to send Bob the suppressed attributes names, for every tuple in the witness set $\{\delta_1, \ldots, \delta_w\}$ of T. In this way, Bob knows what values are to be suppressed from his tuple. After Bob computes the anonymized or suppressed versions $\bar{t}_i$ of tuple t, $1 \leq i \leq w$, he and Alice can start a protocol (e.g., the Intersection Size Protocol in [3]) for privately testing the equality of $\bar{t}_i$ and δ_i. As a drawback, Bob gains knowledge about the suppressed attributes of Alice.

A solution that addresses such drawback is based on the following protocol. Assume Alice and Bob agree on a commutative and product-homomorphic encryption scheme E and $QI = \{A_1, \ldots, A_u\}$. Further, they agree on a coding $c(\cdot)$ (Equation 4) as well. Since other *non-QI* attributes do not play any role in our computation, without loss of generality, let $\delta_i = \langle v'_1, \ldots, v'_s \rangle$ be the tuple containing only the s non-suppressed QI attributes of witness w_i, and $t = \langle v_1, \ldots, v_u \rangle$. Protocol 4.1 allows Alice to compute an anonymized version of t without letting her know the contents of t and, at the the same time, without letting Bob know what are the suppressed attributes of the tuples in T.

The protocol works as follows: at Step 1, Alice sends Bob an encrypted version of δ_i, containing only the s non-suppressed QI attributes. At Step 2, Bob encrypts the information received from Alice and sends it to her, along with encrypted version of each value in his tuple t. At Steps 3-4, Alice examines if the non-suppressed QI attributes of δ_i is equal to those of t.

Protocol 4.1:

1) Alice codes her tuple δ_i into $c(\langle v'_1, \ldots, v'_s \rangle)$, denoted as $c(\delta_i)$. Then, she encrypts $c(\delta_i)$ with her private key and sends $E_A(c(\delta_i))$ to Bob.
2) Bob individually codes each attribute value in t to get the tuple of coded values $\langle c(v_1), \ldots, c(v_u) \rangle$, encrypts each coding and $E_A(c(\delta_i))$ with his key B and sends (i) $\langle E_B(c(v_1)), \ldots, E_B(c(v_u)) \rangle$, and (ii) $E_B(E_A(c(\delta_i)))$ to Alice.
3) Since E is a commutative encryption scheme, $E_B(E_A(c(\delta_i))) = E_A(E_B(c(\delta_i)))$, Alice decrypts $E_A(E_B(c(\delta_i)))$ to obtain $E_B(c(\delta_i))$.
4) Since the encrypted values sent by Bob are ordered according to the ordering of the attributes in T (assume this is a public information known to both Alice and Bob), Alice knows which are, among the encrypted values sent by Bob, the one corresponding to the suppressed and non-suppressed QI attributes. Thus, Alice computes
$$E_B(c(v_1)) \times \ldots \times E_B(c(v_s)) \tag{5}$$
where $v_1, \ldots, v_s$ are the values of non-suppressed attributes contained in tuple t. As already mentioned, E is a product-homomorphic encryption scheme. Based also on the definition of function $c(\cdot)$, this implies that Expression 5 is equal to
$$E_B(c(\langle v_1, \ldots, v_s \rangle)) \tag{6}$$
5) Alice checks whether
$$E_B(c(\langle v_1, \ldots, v_s \rangle)) = E_B(c(\langle v'_1, \ldots, v'_s \rangle))$$
If true, t (properly anonymized) can be inserted to table T. Otherwise, when inserted to T, t breaks k-anonymity.

4.1 Proof of Correctness

Basically, Protocol 4.1 determines if given an anonymous tuple in Alice's database, its unsuppressed attribute values match those of Bob's tuple. Following the same notations used previously, the next two propositions guarantee the correctness of Protocol 4.1.

Proposition 4.1: Given Bob's tuple t and Alice's tuple δ_i, if every non-suppressed attribute value in δ_i is equal to the corresponding attribute value in t, the condition that $E_B(c(v_1, \ldots, v_s)) = E_B(c(v'_1, \ldots, v'_s))$ always holds.
Proof. The proof is trivial. Since $v_1 = v'_1, \ldots, v_s = v'_s$, $c(\langle v_1, \ldots, v_s \rangle) = c(\langle v'_1, \ldots v'_s \rangle)$. As a result, the condition $E_B(c(\langle v_1, \ldots, v_s \rangle)) = E_B(c(\langle v'_1, \ldots, v'_s \rangle))$ holds. □

Proposition 4.2: Given Bob's tuple t and Alice's tuple δ_i, if there exists any non-suppressed attribute value in δ_i is **not** equal to the corresponding attribute value in t, then the condition $E_B(c(\langle v_1, \ldots, v_s \rangle)) = E_B(c(\langle v'_1, \ldots, v'_s \rangle))$ does hold with **negligible** probability (almost 0) provided that c is collision-free.
Proof. Let $\vec{a} = \langle v'_1, \ldots, v'_s \rangle$ and $\vec{b} = \langle v_1, \ldots, v_s \rangle$. The expression $\vec{a} \neq \vec{b}$ means there exists at least one pair (v'_i, v_i) $(1 \leq i \leq s)$ such that $v'_i \neq v_i$. Suppose the claim not to be true, then
$$Prob[E_B(c(\vec{a})) = E_B(c(\vec{b})) | \vec{a} \neq \vec{b}] \geq \frac{1}{p(\cdot)} \tag{7}$$
where $p(\cdot)$ is some positive polynomial. The above equation is equivalent to
$$Prob[c(\vec{a}) = c(\vec{b}) | \vec{a} \neq \vec{b}] \geq \frac{1}{p(\cdot)} \tag{8}$$
However, because the encoding c is collision-free [24, Section 4.3], we have:
$$Prob[c(\vec{a}) = c(\vec{b}) | \vec{a} \neq \vec{b}] < \frac{1}{p(\cdot)} \tag{9}$$
This contradicts Equation 8, and thus contradicts the initial assumptions, and the proposition follows. □

4.2 Proof of Security

We rely on the notion of security of a two-party protocol with semi-honest parties as in [16]. We underline that the adopted proof methodology is a standard cryptographic tool. What follows is an informal, concise presentation of what is needed in the proof of Proposition 4.3. For the formal definitions we refer to [16, Section 7.2.2]. By *view* of party P (where P is chosen among Alice and Bob) of Protocol 4.1, we mean the sequence $v_P = (in^P, m^P_1, m^P_2, \ldots, m^P_u)$, where in^P is the input provided by party P to Protocol 4.1 and m^P_1, m^P_2,…, m^P_u are the messages that party P receives during the execution of Protocol 5.1. Note that the output of Protocol 4.1 is implicit in the sequence of messages received by party P. Protocol 4.1 is *secure* (in the semi-honest model) iff the views v_{Alice} and v_{Bob} are efficiently simulated *only* knowing, respectively, the inputs in^{Alice}, in^{Bob} and the output of Protocol 4.1. This holds provided that both Alice and Bob are *semi-honest*: they follow Protocol 4.1 properly, except that they keep track of all their intermediate computations.

Proposition 4.3: Protocol 4.1 is secure.

Proof. The parties of Protocol 4.1, Alice and Bob, are alternatively replaced by their corresponding simulators S^{Alice} and S^{Bob}. By hypothesis, the simulator S^P, where $P \in \{\text{Alice}, \text{Bob}\}$, knows the input in^P and the output of Protocol 4.1. Further, it does *not* know the messages both parties exchange during the execution of Protocol 4.1. The simulator S^P acts as her/his corresponding party P, but instead of sending to the other party the messages requested by Protocol 4.1, S^P sends a random message. We now show in details how the simulators S^{Alice} and S^{Bob} behave and show that they cannot be (easily) distinguished from their respective parties.

Simulating Alice: The simulator S^{Alice} follows Protocol 4.1 as Alice does, except in Step 1. Instead of sending the value $E_A(c(\delta_i))$, S^{Alice} sends a random message r, where r is uniformly drawn from $\mathcal{D}$. There is no polynomial time algorithm that tells apart $E_A(c(\delta_i))$ from r, since E is an indistinguishable encryption scheme (Property 3 of Section 3.2). This means that the simulator S^{Alice} cannot be distinguished from Alice in polynomial time.

Simulating Bob: The simulator S^{Bob} is more complex to define. The simulator S^{Bob} behaves as Bob, following Protocol 4.1, except in Step 2. Concerning message *(a)*, the simulator S^{Bob} – instead of sending $E_B(E_A(c(\delta_i)))$ and to Alice – produces a random values r uniformly drawn from the domain $\mathcal{D}$, and sends it to Alice. Like for the Alice's simulation, there is no polynomial time algorithm able of telling apart $E_B(E_A(c(\delta_i)))$ from r. This amount to say, regarding message *(a)* in Step 2 of Protocol 4.1, that Alice and her simulator S^{Alice} are not polynomial-time distinguishable. Proving that message *(b)* can be simulated requires the following:

Fact *Let $v_1, v_2, \ldots, v_u, r_1, r_2, \ldots, r_u$ and K be uniformly chosen respectively from $\mathcal{D}$ and $\mathcal{K}$. Let $t = \langle v_1, v_2, \ldots, v_u \rangle$, $r = \langle r_1, r_2, \ldots, r_u \rangle$ and $E_K(t) = \langle E_K(v_1), E_K(v_2), \ldots, E_K(v_u) \rangle$. Then, $\langle t, E_K(t) \rangle$ is indistinguishable from $\langle t, r \rangle$.* This fact follows from the *generalized* Diffie-Hellman assumption [31] which states that the tuple $\langle g^{K_1}, g^{K_2}, \ldots, g^{K_u}, g^{K_1 \cdot \cdots \cdot K_u} \rangle$ is indistinguishable from the tuple $\langle g^{K_1}, g^{K_2}, \ldots, g^{K_u}, r \rangle$, where g is a generator of $\mathcal{D}$, $K_1, \ldots K_u, K$, r are uniformly drawn respectively from $\mathcal{K}$ and $\mathcal{D}$.

We prove the case in which $u = 2$. By DDH, the triple $\langle g^{K_1}, g^{K_2}, g^{K_1 \cdot K_2} \rangle$ is indistinguishable from $\langle g^{K_1}, g^{K_2}, r \rangle$ (we recall that g is a generator of $\mathcal{D}$, K_1, K_2, r are uniformly chosen from $\mathcal{K}$ and $\mathcal{D}$, and the exponentiation is made modulo a prime p). Let K be another key uniformly chosen from $\mathcal{K}$. Then we have that $\langle g^{K_1}, g^{K_2}, g^{K_1 \cdot K_2 \cdot K} \rangle$ is indistinguishable from $\langle g^{K_1}, g^{K_2}, r' = r^K \rangle$. Let's consider the quadruple $\langle g^{K_1}, g^{K_2}, g^{K_1 \cdot K_2 \cdot K}, g^{K_1 \cdot K_2 \cdot K} \rangle$, indistinguishable from $\langle g^{K_1}, g^{K_2}, r', r' \rangle$, from which, we derive a new quadruple $\langle g^{K_1}, g^{K_2}, g^{K_1 \cdot K_2}, g^{K_2 \cdot K} \rangle$, by deleting K_2 and K_1 in the exponents of the 3rd and 4th items. Such quadruple is indistinguishable from the quadruple $\langle g^{K_1}, g^{K_2}, r_1 = r'/r^{K_2}, r_2 = r'/r^{K_1} \rangle$. Rewriting g^{K_1} and g^{K_2} respectively as v_1 and v_2, we get that $\langle v_1, v_2, E_K(v_1), E_K(v_2) \rangle$ is indistinguishable from $\langle v_1, v_2, r_1, r_2 \rangle$.

The simulator S^{Bob} sends a sequence $\langle r_1, r_2, \ldots, r_u \rangle$ of uniformly chosen values from $\mathcal{D}$ to Alice, instead of the sequence $\langle E_B(v_1), E_B(v_2), \ldots, E_B(v_u) \rangle$. Having previously proved that sequences of encrypted values from $\mathcal{D}$ are indistinguishable from sequences of randomly chosen values from the same domain, we have that one cannot tell apart Bob from its simulator S^{Bob} in polynomial time. □

5 Private Update for Generalization-based Anonymous and Confidential Databases

In this section, we assume that the table T is anonymized using a generalization-based method; let $\Gamma_1, \ldots, \Gamma_u$ be u disjoint value generalization hierarchies (VGHs) corresponding to $A_1, \ldots, A_u \in \mathcal{A}_t^{anon}$ known to Alice. Let $\delta \in T$, and let $GetSpec(\delta[A_1, \ldots, A_u], \Gamma_1, \ldots, \Gamma_u)$ ($GetSpec(\delta)$ for short) denote a function which returns a set γ of specific values (values at the bottom of a VGH) related to each attribute $A_i \in \mathcal{A}_t^{anon}$ such that every value in γ can be generalized to $\delta[A_i]$ for some i according to Γ_i. For example, let T refer to Table 4 and $\mathcal{A}_t^{anon} = \{$AREA, POSITION, SALARY$\}$. If $\delta =$ [Operating Systems, Research Assistant, [11k,30k]], then based on the VGHs (presented in Figure 2) $GetSpec(\delta) = \{$Distributed Systems, Handheld Systems, Research Assistant, \$15,000, \$17,000, \$15,500$\}$.

Let t be Bob's private tuple, and assume that Bob knows the set $\mathcal{A}_t^{anon}$. Bob can generate a set τ containing the corresponding values $t[A_1], \ldots, t[A_u]$; the size of τ is always u. We denote by $SSI(\gamma, \tau)$ as a secure protocol that computes the cardinality of $\gamma \cap \tau$. (Either protocol proposed in [3], [13] can be adopted as SSI. However, we used the protocol in [3] for our implementation.) Upon receiving an initial request from Bob, Alice starts the protocol by randomly choosing a tuple δ from the witness set T_w of T. After Alice computes $\gamma = GetSpec(\delta)$, she and Bob privately compute $SSI(\gamma, \tau)$. Note that Bob does not need to know any Γ_i. We claim that if $SSI(\gamma, \tau) = u$ (the size of $\mathcal{A}_t^{anon}$), $t[A_1, \ldots, A_u]$ can be generalized to δ, and hence this insertion into T can be safely performed without breaking the k-anonymity property. We will prove this claim later in the section. The protocol's details follow:

Protocol 5.1:

1) Alice randomly chooses a $\delta \in T_w$
2) Alice computes $\gamma = GetSpec(\delta)$
3) Alice and Bob collaboratively compute $s = SSI(\gamma, \tau)$

4) If $s = u$ then t's generalized form can be safely inserted to T
5) Otherwise, Alice computes $T_w \leftarrow T_w - \{\delta\}$ and repeat the above procedures until either $s = u$ or $T_w = \emptyset$

The following example illustrates the execution of the protocol. Suppose Alice has the witness set T_w (Table 5) of T (Table 4), and $u = 3$ in this example. If Bob's tuple t = [Data Mining, Teaching Assistant, \$15,000], then τ = {Data Mining, Teaching Assistant, \$15,000}. Suppose δ = [Database Systems, Associate Professor, [61k, 120k]], then γ = {Data Mining, Data Warehousing, Query Processing, Associate Professor, \$78,000, \$90,000, \$91,000, \$95,000, \$100,000}. Since $|\gamma \cap \tau| = 1 < u$ (or $\tau \not\subseteq \gamma$), $SSI(\gamma, \tau)$ returns 1, and we can conclude that t cannot be anonymized (or generalized) to δ. Using the same analysis, we can verify that t cannot be anonymized to any record in T_w. On the other hand, if t = [Distributed System, Research Assistant, \$17,000], t can be anonymized to δ = [Operating Systems, Research Assistant, [11k, 30k]].

5.1 Proof of Correctness

Proposition 5.1: Referring to Protocol 5.1, given τ (generated from t) and $\gamma = GetSpec(\delta)$, if $SSI(\gamma, \tau) = u$ ($\tau \subseteq \gamma$), $t[A_1, \ldots, A_u]$ can be generalized to δ.

Proof. We prove this claim via a contrapositive argument. Assume $SSI(\gamma, \tau) = u$ and $t[A_1, \ldots, A_u]$ cannot be generalized to δ, then $\exists \tau_i \in \tau$ such that τ_i cannot be generalized to any value in $\{\delta[A_1], \ldots, \delta[A_u]\}$ because $\Gamma_1, \ldots, \Gamma_u$ are disjoint. On the other hand, since $SSI(\gamma, \tau) = u$ implies $\tau \subseteq \gamma$, τ_i must match some value $\gamma_j \in \gamma$. Based on the definition of $GetSpec(\delta)$, we know that every value in γ can be generalized to some value in $\{\delta[A_1], \ldots, \delta[A_u]\}$. Therefore, it must be the case that τ_i can be generalized to some value in $\{\delta[A_1], \ldots, \delta[A_u]\}$. This contradicts the assumption. In addition, since we assume $\Gamma_1, \ldots, \Gamma_u$ are disjoint, for any two $\tau_i, \tau_j \in \tau$, they cannot be generalized to the same value in $\{\delta[A_1], \ldots, \delta[A_u]\}$. This guarantees that for $1 \leq i \leq u$, $t[A_i]$ (or τ_i) can be generalized to $\delta[A_i]$ as long as $SSI(\gamma, \tau) = u$ holds. □

5.2 Security Analysis

The security of Protocol 5.1 depends on that of the SSI protocol, and detailed security analyses of SSI can be found in [3], [13]. The SSI protocol presented in [3] is easy to implement and efficient to perform. Although the protocol leaks the intersection size between γ and τ to the participating parties, it does provide sufficient privacy protection in our application.. In case this linkage of intersection sizes is not acceptable, we can adopt one variation of the SSI protocol presented in [13]. We can make the protocol only return whether or not $|\gamma \cap \tau| = u$ without disclosing the intersection size. Under the context of Secure Multi-party Computation [16], this variation of SSI does not leak any information that cannot be inferred from the final result and the private input data. Thus, using SSI proposed in [13], Protocol 5.1 can achieve very high security.

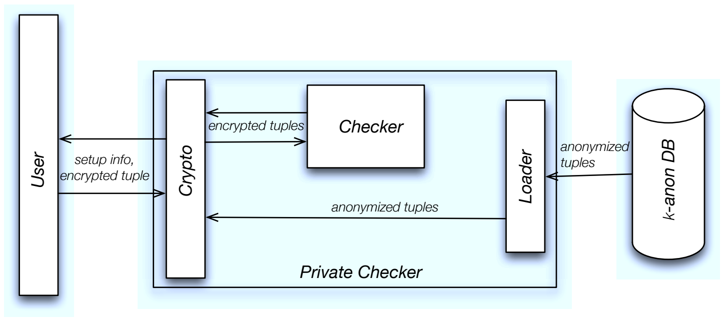


Fig. 3. Prototype architecture overview

6 Architecture and Experimental Results

Our prototype of a *Private Checker* (that is, Alice) is composed by the following modules: a *crypto* module that is in charge of encrypting all the tuples exchanged between an *user* (that is, Bob) and the Private Updater, using the techniques exposed in Sections 4 and 5; a *checker* module that performs all the controls, as prescribed by Protocols 4.1 and 5.1; a *loader* module that reads chunks of anonymized tuples from the k-anonymous DB. The chunk size is fixed in order to minimize the network overload. In Figure 3 such modules are represented along with labeled arrows denoting what information are exchanged among them. Note that the functionality provided by the Private Checker prototype regards the check on whether the tuple insertion into the k-anonymous DB is possible. We do not address the issue of actually inserting a properly anonymized version of the tuple.

The information flow across the above mentioned modules is as follows: after an initial *setup phase* in which the user and the Private Checker prototype exchange public values for correctly performing the subsequent cryptographic operations, the user sends the encryption $E(c(\delta_i))$ of of her/his tuple to the Private Checker; the *loader* module reads from the k-anonymous DB the first chunk of tuples to be checked with $E(c(\delta_i))$. Such tuples are then encrypted by the *crypto* module. The *checker* module performs the above mentioned check one tuple at time in collaboration with the user, according to either Protocol 4.1 (in the case of suppression-based anonymization) or Protocol 5.1 (in the case of generalization-based anonymization). If none of the tuples in the chunk matches the User tuple, then the loader reads another chunk of tuples from the k-anonymous DB. Note the communication between the prototype and User is mediated by an anonymizer (like Crowds, not shown in figure) and that all the tuples are encrypted.

We briefly discuss the complexity of our protocols in terms of the number of messages exchanged and their size. It turns out that the number of messages exchanged during executions of Protocol 4.1 and Protocol 5.1 is bounded by a linear function of the number of witnesses of the anonymous database. Protocol 4.1 requires that Alice sends Bob the encrypted version of tuple δ_i. Bob encrypts it with his own private key and sends it back to Alice. Further, Bob sends Alice the encrypted version of tuple t. Then, Bob sends Alice the encrypted values contained in t, in order to let Alice compute the actual, encrypted version of anonymized tuple t. Finally, Alice and Bob exchange the encrypted version of tuple δ_i for checking whether such tuple and the encrypted, anonymized version of t match. Assuming the worst-case scenario, this has to be executed w times. Thus, the number of messages is $6 \cdot w$. The complexity of Protocol 5.1 relies on the size of T_w ($|T_w|$) and the complexity of the SSI protocol. The number of calls to the SSI protocol is bounded by T_w, and detailed complexity analyses of SSI can be found in [3], [13].

We implemented both Protocols 4.1 and 5.1 using mySQL 5.1 and C++ using the NTL libraries version 5.5 for the numerical computations. We tested our implementation on the Income database from the UC Irvine Machine Learning Repository [4]. The database has size equal to 50.7 MB and contains about 286k tuples. Such database has been anonymized using both suppression and generalization-based approaches, for values of parameter k equal to 2, 5, 10, 20 and 50. The resulting anonymized databases have been imported into MySQL 5.0. We then tested several times the insertion of a tuple in such anonymized databases. All the experiments were run on an Intel(R) Core(TM)2 1.8 GHz CPU with 1 GB of physical memory running Linux Debian.

We report the average execution times (expressed in milliseconds) of Protocol 4.1 and Protocol 5.1 respectively in Figures 4 and 5. The experiments confirm the fact that the time spent by both protocols in testing whether the tuple can be safely inserted in the anonymized database decreases as the value of k increases. Intuitively, this is due to the fact that the larger the k is, the smaller the witness set. fewer are the partitions in which table T is divided Consequently, fewer protocol runs are needed to check whether the update can be made. Further, we report that the experiments confirm the fact that the execution times of of Protocols 4.1 and 5.1 grow as $(\text{dataset size})/k$. That is, each protocol has to check the anonymized tuple to be inserted against every witness in the worst case, and the larger the parameter k is, the fewer the witnesses are.

We report in Figures 6 and 7 the cpu and network latency times for Protocols 4.1 and 5.1, as the parameter k increases. As it is shown, latency time accounts for a very large portion of the elapsed time for the executions of Protocols 4.1 and 5.1.

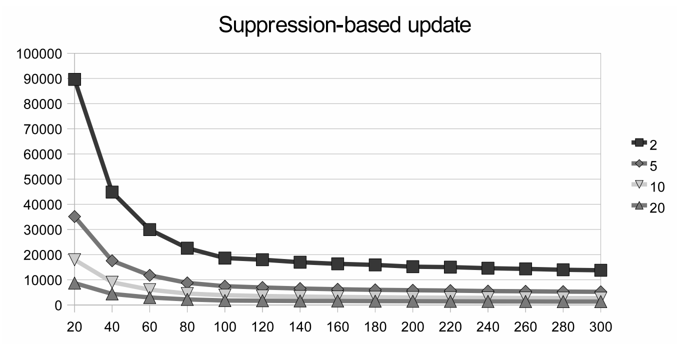


Fig. 4. Execution times of Protocol 4.1 as the parameter k increases

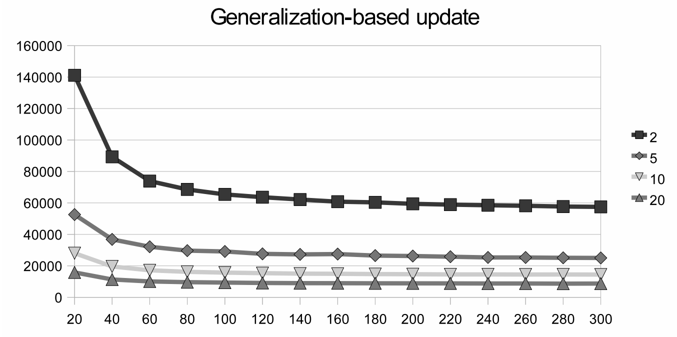


Fig. 5. Execution times of Protocol 5.1, as the parameter k increases

7 Conclusion / Future Work

In this paper, we have presented two secure protocols for privately checking whether a k-anonymous database retains its anonymity once a new tuple is being inserted to it. Since the proposed protocols ensure the updated database remains k-anonymous, the results returned from a user's (or a medical researcher's) query are also k-anonymous. Thus, the patient or the data provider's privacy cannot be violated from any query. As long as the database is updated properly using the proposed protocols, the user queries under our application domain

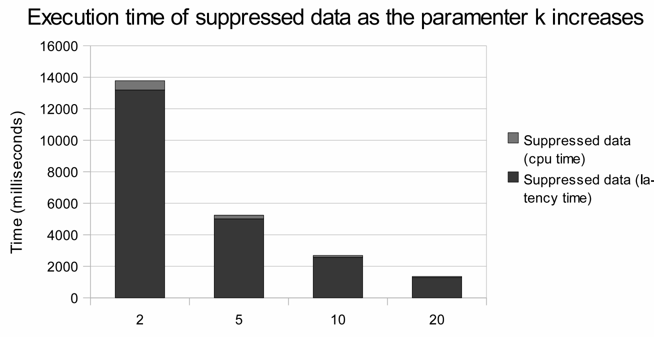


Fig. 6. Execution times of Protocol 4.1 as the parameter k increases

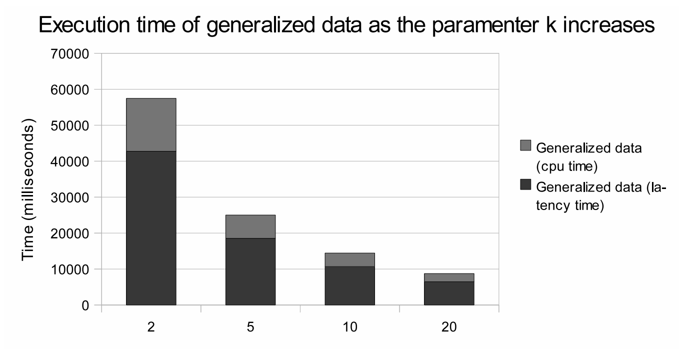


Fig. 7. Execution times of Protocol 5.1, as the parameter k increases

are always privacy-preserving.

In order for a database system to effectively perform privacy preserving updates to a k-anonymous table, Protocols 4.1 and 5.1 are necessary but clearly not sufficient. As already mentioned in Section 1, other important issues are to be addressed: (i) the definition of a mechanism for actually performing the update, once k-anonymity has been verified; (ii) the specification of the actions to take in case Protocols 4.1 or 5.1 yield a negative answer; (iii) how to initially populate an empty table. (iii) the integration with a privacy-preserving query system. In the following, we sketch the solutions developed in order to address these questions and which comprise our overall methodology for the private database update.

As a general approach, we separate the process of database k-anonymity checking and the actual update into two different phases, managed by two different sub-systems: the Private Checker and the Private Updater. In the first phase, the Private Checker prototype presented in Section 6 , following Protocol 4.1 or Protocol 5.1, checks whether the updated database is still k-anonymous, without knowing the content of the user's tuple. In the second phase, the Private Updater actually updates the database based on the result of the anonymity check; we refer to this step as *update execution*. At each phase, the database system and the user[1] communicate via an anonymous connection as mentioned in Section 1 by using a protocol like Crowds [27] or Onion routing [26]. Also, legitimate users are authenticated anonymously via the protocol presented in [20]. Thus, the system cannot link the user who has entered the update request to the user who actually performs it.

Concerning the actual execution of the database update, once the system has verified that the user's tuple can be safely inserted to the database without compromising k-anonymity, the user is required to send to the Private Updater the non-anonymous attributes' values to be stored in the k-anonymous database as well. The deployment of an anonymity system ensures that the system cannot associate the sender of the tuple with the subject who made the corresponding insertion's request.

Suppose that a tuple fails the tests of Protocols 4.1 and 5.1. Then, the system does not insert the tuple to the k-anonymous database, and waits until $k-1$ other tuples fail the insertion. At this point, the system checks whether such set of tuples, referred to as *pending tuple set*, are k-anonymous. Such test can be performed on encrypted data by using the methods proposed by Zhong et al. [35]. In the affirmative case, the system proceeds to insert the k-anonymous tuples to the database. In the negative case, the k-anonymization of the set of tuples failing the insertion is periodically checked, again by methods presented in [35]. Note that many issues need to be addressed for the approach described above to be effective. For instance, where and who is responsible for keeping the *pending tuple set*; how to inform and communicate with data users in order to initiate the protocol. We will address these issues in future.

In addition to the problem of falling insertion, there are other interesting and relevant issues that remain to be addressed:

- Devising private update techniques to database systems that supports notions of anonymity different than k-anonymity (see the discussion in [11]);
- Dealing with the case of malicious parties by the introduction of an untrusted, non-colluding third party [12];
- Implementing a real-world anonymous database system;
- Improving the efficiency of protocols, in terms of number of messages exchanged and in terms of their sizes, as well.

We believe that all these issues are very important and worthwhile to be pursued in the future.

1. Here, the user is referred as an abstraction. In the real world, an user may be the actual owner of the data being inserted or an agent acting on her/his behalf.

References

[1] N. R. Adam, J. C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 21(4), 1989; 515–556.
[2] G. Aggarwal, T. Feder, K. Kenthapadi, R. Motwani, R. Panigrahy, D. Thomas, A. Zhu. Anonymizing tables. In *Proc. of Int'l Conf. of Database Theory (ICDT)*, Edimburgh, UK, 2005.
[3] R. Agrawal, A. Evfimievski, R. Srikant. Information sharing across private databases. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, San Diego, California, USA, 2003.
[4] C. Blake, C. Merz. "UCI repository of machine learning databases," 1998. http://www.ics.uci.edu/ mlearn/MLRepository.html
[5] E. Bertino, R. Sandhu. Database security - Concepts, approaches and challenges. *IEEE Transactions on Dependable and Secure Computing*, 2(1), 2005; 2–19.
[6] D. Boneh. The decision Diffie-Hellman problem. In *Proc. of Int'l Algorithmic Number Theory Symposium*, *LNCS* 1423, 1998; 48–63.
[7] D. Boneh, G. di Crescenzo, R. Ostrowsky, G. Persiano. Public key encryption with keyword search. In *Proc. of Eurocrypt Conf.*, Interlaken, Switzerland, 2004.
[8] S. Brands. Untraceable off-line cash in wallets with observers. in *Proc. of CRYPTO Conf.*, Lecture Notes in Computer Science, 773, 1994; 302–318.
[9] J.W. Byun, T. Li, E. Bertino, N. Li, Y. Sohn. Privacy-preserving incremental data dissemination. *Journal of Computer Security*, 17(1), 2009, 43–68.

[10] R. Canetti, Y. Ishai, R. Kumar, m. K. Reiter, R. Rubinfeld, R. N. Wright. Selective private function evaluation with application to private statistics. In *Proc. of ACM Symposium on Principles of Distributed Computing (PODC)*, Newport, Rhode Island, USA, 2001.

[11] S. Chawla, C. Dwork, F. McSherry, A. Smith, H. Wee. Towards privacy in public databases. In *Proc. of Theory of Cryptography Conf. (TCC)*, Cambridge, Massachussets, USA, 2005.

[12] U. Feige, J. Kilian, M. Naor. A minimal model for secure computation. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, Philadelphia, Pennsylvania, USA, 1994.

[13] M. J. Freedman, M. Naor, B. Pinkas. Efficient private matching and set intersection. In *Proc. Eurocrypt Conf.*, Interlaken, Switzerland, 2004.

[14] B.C.M. Fung, K. Wang, A.W.C. Fu, J. Pei. Anonymity for continuous data publishing. In *Proc. of EDBT Conf.*, Nantes, France, 2008.

[15] O. Goldreich. *Foundations of Cryptography. Volume 1, Basic Tools*, Cambridge University Press, 2001.

[16] O. Goldreich. *Foundations of Cryptography. Volume 2, Basic Applications*, Cambridge University Press, 2004.

[17] H. Hacigümüş, B. Iyer, C. Li, S. Mehrotra. Executing SQL over encrypted data in the database-service-provider model. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, Madison, Wisconsin, USA, 2002.

[18] Y. Han, J. Pei, B. Jiang, Y. Tao, Y. Jia. Continuous privacy preserving publishing of data streams. In *Proc. EDBT Conf.*, Nantes, France, 2008.

[19] US Department of Health & Human Services, Office for Civil Rights. Summary of the HIPAA Privacy Rule, 2003.

[20] J. Li, N. Li, W. Winsborough. Policy-hiding access control in open environment. In *Proc. of ACM Conf. on Computer and Communications Security (CCS)*, Alexandria, Virginia, 2005.

[21] J. Li, B.C. Ooi, W. Wang. Anonymizing streaming data for privacy protection. In *Proc. of IEEE Int'l Conf. on Database Engineering (ICDE)*, Cancun, Mexico, 2008.

[22] U. Maurer. The role of cryptography in database security. In *Proc. of ACM SIGMOD Int'l Conf. on Management of Data*, Paris, France, 2004.

[23] A. Meyerson, R. Williams. On the Complexity of Optimal K-Anonymity. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, Paris, France, 2004.

[24] S. Micali, M. Rabin, J. Kilian. Zero-knowledge sets. In *Proc. of 44th Symposium on Foundations of Computer Science*, Cambridge, Massachussets, USA, 2003.

[25] T. Pedersen. Noninteractive and information-theoretic secure verifiable secret sharing. *Lecture Notes in Computer Science*, 576, 1991; 129–140.

[26] M. Reed, P. Syverson, D. Goldschlag. Anonymous connections and Onion routing. *IEEE Journal of Selected Areas in Communications*, 16(4), 1998; 482–494.

[27] M. K. Reiter, A. Rubin. Crowds: anonymity with Web transactions. *ACM Transactions on Information and System Security (TISSEC)*, 1(1), 1998; 66–92.

[28] P. Samarati. Protecting respondent's privacy in microdata release. *IEEE Transactions on Knowledge and Data Engineering*, vol. 13, no. 6, pp. 1010–1027, Nov./Dec. 2001.

[29] V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. of Eurocrypt Conf.*, Kostanz, Germany, 1997.

[30] D. X. Song, D. Wagner, A Perrig. Practical techniques for searches on encrypted data. In *Proc. IEEE Symposium on Security and Privacy*, Oakland, California, USA, 2000.

[31] M. Steiner, G. Tsudik, M. Waidner. Diffie-Hellman key distribution extended to group communication. In *Proc. ACM Conf. on Computer and Communication Security*, New Delhi, India, 1996.

[32] L. Sweeney. k-anonymity: a model for protecting privacy. *International Journal on Uncertainty, Fuzziness and Knowledge-based Systems*, 10(5), 557–570, 2002.

[33] A. Trombetta, E. Bertino. Private updates to anonymous databases. In *Proc. Int'l Conf. on Data Engineering (ICDE)*, Atlanta, Georgia, US, 2006.

[34] K. Wang, B. Fung. Anonymizing sequential releases. In *Proc. ACM KDD Conf.*, Philadelphia, USA, 2006.

[35] S. Zhong, Z. Yang, R. N. Wright. Privacy-enhancing k-anonymization of customer data. In *Proc. of ACM Symposium on Principles of Database Systems (PODS)*, Baltimore, Maryland, USA, 2005.