Private Updates to Anonymous Databases

Alberto Trombetta* Elisa Bertino†

Abstract

Suppose that Alice, owner of a k-anonymous database, needs to determine whether her database, when adjoined with a tuple owned by Bob, is still k-anonymous. Suppose moreover that access to the database is strictly controlled, because for example data are used for experiments that need to be maintained confidential. Clearly, allowing Alice to directly read the contents of the tuple breaks the privacy of Bob; 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 adjoined with the tuple is still k-anonymous, without letting Alice and Bob know the contents of, respectively, the tuple and the database. In this paper, we propose two protocols solving this problem.

Keywords. *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 also data ownership. Data confidentiality is not however the only requirement that needs to be addressed. In fact, 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. Note that, although confidentiality and privacy are often used as synonyms, they are different concepts: data confidentiality is about the difficulty by an unauthorized user to learn anything about data stored in the database. Usually, this is achieved by enforcing an access policy, possibly using cryptographic tools. Privacy relates to what data can be safely disclosed without leaking sensitive information regarding the legitimate owner. Recently, techniques addressing the problem of obtaining privacy through data anonymization have been developed, thus making more difficult to link certain sensitive information to specific individuals. 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: *(Problem i: is the updated database still privacy-preserving?* and *(Problem ii): does the database owner need to know the data to be inserted?*. Clearly, the two problems are related: *(i) can the database owner decide whether the updated database still preserves*

*Dipartimento di Informatica e Comunicazione, Università degli Studi dell'Insubria, Via Mazzini 5, 21100 Varese, Italy, alberto.trombetta@uninsubria.it

†Department of Computer Science and CERIAS, Purdue University, 250 N. University Street, West Lafayette, IN, U.S.A., bertino@cs.purdue.edu

Proceedings of the 22nd International Conference on Data Engineering (ICDE'06)
8-7695-2570-9/06 $20.00 © 2006 IEEE

privacy of individuals, (ii) without directly knowing the new data to be inserted?. The answer we give in this work is affirmative and we introduce our proposed solutions below.

2 Anonymity and cryptographic notions

We consider a table $T = \{t_1, \ldots, t_n\}$ over the attribute set $\mathcal{A}$. The table T is *k-anonymous* [4] if for every tuple $t \in T$, there are two disjoint sets of *suppressed* attributes $\mathcal{A}_t^{supp} \subseteq \mathcal{A}$ and of *anonymous* attributes $\mathcal{A}_t^{anon} \subseteq \mathcal{A}$ such that there exist at least $k-1$ tuples $t'_1, \ldots, t'_z$ in T such that for every attribute in A_t^{supp} the corresponding value is replaced by $*$ and for every attribute in $A_1, \ldots, A_u \in \mathcal{A}_t^{anon}$ the condition $t\,[A_1, \ldots, A_u] = t'_i\,[A_1, \ldots, A_u]$ (for $1 \leq i \leq z$) is verified. Possible other attributes in $\mathcal{A}$ not in $\mathcal{A}_t^{supp}$ nor in $\mathcal{A}_t^{anon}$ are called *non-anonymous* attributes (for tuple t). That is, a k-anonymous table T is such that every tuple $t \in T$ has equal values over the anonymous attributes with at least $k-1$ other tuples in T and it has the value $*$ over the suppressed attributes. We assume that all the anonymous attributes' domains are finite. For every subset T_i, we pick a tuple δ_i in it, we delete the values of non-anonymous attributes, and we call it the *witness* of T_i. The protocols in Section 3 use a commutative, product-homomorphic encryption scheme E. We extend the definition of commutative, indistinguishable encryption scheme presented in [1], in order to allow the encryption scheme to be product-homomorphic as well. Given a finite set $\mathcal{K}$ of keys and a finite domain $\mathcal{D}$, an *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. *Commutativity*: for all key pairs $K_1, K_2 \in \mathcal{K}$ and value $d \in \mathcal{D}$, the equality $E_{K_1}(E_{K_2}(d)) = E_{K_2}(E_{K_1}(d))$ holds. *Product-homomorphism*: for every $K \in \mathcal{K}$ and every value pairs $d_1, d_2 \in \mathcal{D}$, the equality $E_K(d_1) \cdot E_K(d_2) = E_K(d_1 \cdot d_2)$ holds. *Indistiguishability*: due to lack of space, we give only an intuition of what indistiguishability is and we refer to [2, Chapter 2] for the precise definition. Informally, the indistinguishabilty of an encryption scheme guarantees that encryption and decryption without knowledge of the key is infeasible. 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 it is prime as well. Let $d \in \mathcal{D}$ and $K \in \mathcal{K} = \{0, \ldots, q-1\}$. Assuming the Decision Diffie-Hellman assumption, the function $E_K(d) \stackrel{\text{def}}{=} d^K \bmod p$ is a commutative, product-homomorphic, indistinguishable encryption scheme. Finally, following [3], we introduce a simple way of coding tuples that we will use in the second protocol: Alice and Bob agree on a set $\{g_1, g_2 \ldots, g_u\}$ of generators of $\mathcal{D}$. Let d the 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 $c(\langle d_1, d_2, \ldots, d_u \rangle) = \prod_{i=1}^{u} g_i^{d_i} \bmod q$

3 Private update protocols

Suppose that Alice owns a k-anonymous table T and has to decide whether T when adjoined with a tuple t – owned by Bob – is still k-anonymous, without directly knowing the values in t (obviously, t and T agree on the same schema). This problem amounts to decide whether t matches on the anonymous attributes with at least one of the witnesses $\delta_1, \ldots, \delta_w$. If this is the case, then t, properly anonymized, can be adjoined to the corresponding subset of T. Otherwise, the insertion of t into T is rejected.

3.1 First protocol

1. For every δ_i, for $1 \leq i \leq w$, Alice creates the set Δ_i as follows:
   $\Delta_i = \{\bar{d} \,|\, \bar{d}[A_j] = \delta_i[A_j] \text{ if } A_j \in A_t \text{ and } \bar{d}[A_j] = v, \text{ for every } v \in \mathcal{D}_|\}$. The set Δ_i contains all the tuples having values equal to δ_i and, on suppressed attributes, the tuples' values entirely span their domains. Alice tells to Bob what are the non-anonymous attributes of δ_i. Bob deletes the values of such non-anonymous attributes from t, obtaining $\bar{t}$.

2. Steps 2–4 follow Algorithm 5.1.1 of [1]. Alice and Bob agree on an one-way hash function h – such as SHA-1 or MD5 – and on a commutative encryption scheme E. Alice hashes the elements of Δ_i with h obtaining $h(\bar{d}_1), \ldots, h(\bar{d}_z)$. Bob does the same with $\bar{t}$, obtaining $h(\bar{t})$. Alice and Bob respectively generate their keys. Alice and Bob encrypt with their keys

the elements of Δ_i and the tuple t, obtaining respectively $E_A(h(\bar{d}_1)), \ldots, E_A(h(\bar{d}_z))$ and $E_B(h(\bar{t}))$.

3. Alice sends $E_A(h(\bar{d}_1)), \ldots, E_A(h(\bar{d}_z))$ to Bob. Bob encrypts them with his key, obtaining $E_B(E_A(h(\bar{d}_1))), \ldots, E_B(E_A(h(\bar{d}_z)))$ Bob sends $E_B(E_A(h(\bar{d}_1))), \ldots, E_B(E_A(h(\bar{d}_z))), E_B(h(\bar{t}))$ to Alice.

4. Alice encrypts $E_B(h(\bar{t}))$ with her key obtaining $E_A(E_B(h(\bar{t})))$. Since E is commutative, $E_A(E_B(h(\bar{t}))) = E_B(E_A(h(\bar{t})))$. Alice checks whether $E_B(E_A(h(\bar{t})))$ is equal to one of the values $E_B(E_A(h(\bar{d}_1))), \ldots, E_B(E_A(h(\bar{d}_z)))$. If this is the case, then $\bar{t}$ belongs to Δ_i; otherwise, t does not belong to Δ_i.

3.2 Second protocol

1. Alice creates the tuple $0_i^* = \langle 0, 0, \ldots, *, \ldots, 0 \rangle$ of length u, containing all zeros except the value of the j-th attribute of δ_i, which is denoted with $*$. Alice codes 0_i^* into $c(0_i^*)$, encrypts it with her key A, obtaining $E_A(c(0_i^*))$ and sends $E_A(c(0_i^*))$ to Bob.

2. Bob codes his tuple t into $c(t)$, encrypts $c(t)$ and $E_A(c(0_i^*))$ with his key B and sends $E_B(c(t))$ and $E_B(E_A(c(0_i^*)))$ to Alice.

3. Since E is a commutative encryption scheme, $E_B(E_A(c(0_i^*))) = E_A(E_B(c(0_i^*)))$, Alice decrypts $E_A(E_B(c(0_i^*)))$ to $E_B(c(0_i^*))$. Alice computes $E_B(c(t)) \cdot E_B(c(0_i^*))$. Since E is a product-homomorphic encryption scheme, $E_B(c(t)) \cdot E_B(c(0_i^*)) = E_B(c(t) \cdot c(0_i^*))$. We remind that t is equal to $\langle v_1, v_2, \ldots, v_u \rangle$. The value $E_B(c(t) \cdot c(0_i^*))$ can be rewritten as $E_B(c(\langle v_1, \ldots, v_u \rangle) \cdot c(\langle 0, \ldots, *, \ldots, 0 \rangle))$. By the definition of coding $c(\cdot)$ (see end of Section 2), the last expression is equal to $E_B(c(\langle v_1, v_2, \ldots, v_j + *, \ldots, v_u \rangle))$.

4. Bob encrypts the values $v_1, v_2, \ldots, v_u$, obtaining $E_B(c(v_1)), E_B(c(v_2)), \ldots, E_B(c(v_u))$. Bob sends the tuple $\langle E_B(c(v_1)), E_B(c(v_2)), \ldots, E_B(c(v_u)) \rangle$ to Alice.

5. Alice knows which are, among the encrypted values sent by Bob, the values corresponding to the suppressed attribute and to non-anonymous attributes. Thus Alice computes $\frac{E_B(c(\langle v_1, v_2, \ldots, v_j + *, \ldots, v_u \rangle))}{E_B(c(v_j)) E_B(c(v_{na1})) \cdots E_B(c(v_{nar}))}$. because E is a product-homomorphic encryption scheme, the last expression is equal to $E_B\left(\frac{c(\langle v_1, v_2, \ldots, v_j + *, \ldots, v_u \rangle)}{c(v_j)\, c(v_{na_1}) \cdots c(v_{na_r})}\right)$, which is in turn equal to $E_B\left(c(\langle v_1, v_2, \ldots, *, \ldots, v_{u-r} \rangle)\right)$. We denote the last expression with $E_B(c(t_i^*))$.

6. The last steps are inspired by Algorithm 5.1.1 of [1]. Alice codes her anonymized tuple δ_i into $c(\delta_i)$. Then, she encrypts $c(\delta_i)$ with her private key and sends $E_A(c(\delta_i))$ to Bob. Bob encrypts $E_A(c(\delta_i))$ and sends $E_B(E_A(c(\delta_i)))$ to Alice. Since E is commutative, $E_B(E_A(c(\delta_i))) = E_A(E_B(c(\delta_i)))$ holds. Alice decrypts with her private key, obtaining $E_B(c(\delta_i))$. Finally, Alice checks whether the equality $E_B(c(\delta_i)) = E_B(c(t_i^*))$ holds. If this is the case, then the tuple t (properly anonymized) can be safely adjoined to table T. Otherwise, when adjoined to T, tuple t breaks its k-anonymity.

This material is based upon work partially supported by the National Science Foundation under Grant No. 0430274 and the sponsor of CERIAS

References

[1] 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.

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

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

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