InstantDB: Enforcing Timely Degradation of Sensitive Data

Nicolas Anciaux*, Luc Bouganim*, Harold van Heerde ***, Philippe Pucheral*,**, Peter M. G. Apers***

** INRIA Rocquencourt, Le Chesnay, France*
<Fname.Lname>@inria.fr

*** PRiSM Laboratory, University of Versailles, France*
<Fname.Lname>@prism.uvsq.fr

**** CTIT, University of Twente, The Netherlands*
{heerdehjw,apers}@ewi.utwente.nl

Abstract — **People cannot prevent personal information from being collected by various actors. Several security measures are implemented on servers to minimize the possibility of a privacy violation. Unfortunately, even the most well defended servers are subject to attacks and however much one trusts a hosting organism/company, such trust does not last forever. We propose a simple and practical degradation model where sensitive data undergoes a progressive and irreversible degradation from an accurate state at collection time, to intermediate but still informative fuzzy states, to complete disappearance. We introduce the data degradation model and identify related technical challenges and open issues.**

I. Introduction

People give personal data explicitly all the time to insurance companies, hospitals, banks, and employers. Implicitly, cell phones give location information, cookies give browsing information and RFID tags may give information even more continuously. The data ends up in a database somewhere, where it can be queried for various purposes. Many people believe that they have lost all control of the usage made of their personal data and consider the situation as unavoidable.

Several solutions have been proposed to combat this privacy invasion and violation, extending the traditional access control management [6] towards usage control. Hippocratic databases [9] are a good representative of this approach where the donor's consent is collected along with the data and this consent grants permission to execute only predefined operations (i.e., usages) aiming at satisfying predefined purposes. While this approach offers a way to express privacy goals, its effectiveness relies on the trust put on the organization managing the data. Unfortunately, even the most secure servers (including those of Pentagon, FBI and NASA) are subject to attacks [3].

Beyond computer security, the trust dilemma regarding the hosting of personal records encompasses government requests and business practices. Typically, search engine companies have been urged by governments to disclose personal information about people suspected of crime, terrorism and even dissidence. Today, U.S. and E.U. regulators show important concerns about the Google-DoubleClick merging considering the unprecedented amount of information that can be gathered about the Internet activities of consumers [12]. The privacy policy that will apply after the merging to the personal data previously acquired by each partner is also questionable. Thus, however how much one trusts a hosting company, this trust can be reconsidered over time.

Existing approaches to answer these concerns rely on anonymizing the data when it complies with the acquisition purpose or on attaching a retention limit to the data storage. Data anonymization [7],[11] helps, but the more the data is anonymized, the less usage remains possible, introducing a tricky balance between application reach and privacy [10]. Besides, correctly anonymizing the data is a hard problem [2], especially when considering incremental data sets or if background knowledge is taken into account [1]. The disclosure of insufficiently anonymized data published by AOL about Web search queries conducted by 657,000 Americans exemplifies this [4]. Limited retention attaches a lifetime compliant with the acquisition purpose to the data, after which it must be withdrawn from the system. When the data retention limit is not fixed by the legislation, it is supposed to reflect the best compromise between user and company interests. In practice, the all-or-nothing behaviour implied by limited data retention leads to the overstatement of the retention limit every time a data is collected to serve different purposes [9]. As a consequence, retention limits are usually expressed in terms of years and are seen by civil rights organizations as a deceitful justification for long term storage of personal data by companies.

The approach proposed in this paper opens up a new alternative to protect personal data over time. It is based on the assumption that long lasting purposes can often be satisfied with a less accurate, and therefore less sensitive, version of the data. In our data degradation model, called Life Cycle Policy (LCP) model, data is stored accurately for a short period, such that services can make full use of it, then degraded on time progressively decreasing the sensibility of the data, until complete removal from the system. Thus, the objective is to progressively degrade the data after a given time period so that (1) the intermediate states are informative enough to serve application purposes and (2) the accurate state cannot be recovered by anyone after this period, not even by the server. The expected benefit is threefold:

- *Increased privacy wrt disclosure*: the amount of accurate personal information exposed to disclosure (e.g., after an attack, a business alliance or a governmental pressure) depends on the degradation policy but is always less than with a traditional data retention principle.
- *Increased security wrt attacks*: to be effective, an attack targeting a database running a data degradation process must be repeated with a frequency smaller than the duration of the

978-1-4244-1837-4/08/$25.00 © 2008 IEEE

shortest degradation step. Such continuous attacks are easily detectable thanks to Intrusion Detection and Auditing Systems.

- *Increased usability wrt application:* compared to data anonymization, data degradation applies to attributes describing a recorded event while keeping the identity of the donor intact. Hence, user-oriented services can still exploit the information to the benefit of the donor. Compared to data retention, data degradation steps are defined according to the targeted application purposes and the retention period in each step is defined according to privacy concerns. Then, degrading the data rather than deleting it offers a new compromise between privacy preservation and application reach.

Hence, data degradation should be considered as a new tool complementary to access control, anonymization and data retention to help better protect the privacy of personal records, with a significant expected impact on the amount of sensitive information exposed to attacks and misuses.

An important question is whether data degradation can be reasonably implemented in a DBMS. As pointed out in [8], even guaranteeing that data cannot be recovered after a regular delete as performed by traditional databases is not easy. Indeed, every trace of deleted data must be physically cleaned up in the data store, the indexes and the logs. Data degradation is a more complex process which includes physical data deletion but impacts more thoroughly the data storage, indexation, logging and locking mechanisms to deal with data traversing a sequence of states of accuracies.

We propose below a simple and effective data degradation model, identify the technical challenges and outline some open issues.

II. LCP Degradation Model

In our data degradation model, data is subject to a progressive degradation from the accurate state to intermediate less detailed states, up to disappearance from the database. The degradation of each piece of information (typically an attribute) is captured by a *Generalization Tree*. Given a *domain generalization hierarchy* [5] for an attribute, a generalization tree (GT) for that attribute gives, at various levels of accuracy, the values that the attribute can take during its lifetime (see Figure 1). Hence, a path from a particular node to the root of the GT expresses all degraded forms the value of that node can take in its domain. Furthermore, for simplicity we assume that for each domain there is only one GT.

A *life cycle policy* (LCP) governs the degradation process by fixing how attribute values navigate from the GT leaves up to the root. While we may consider complex life cycle policies where state transitions are triggered by events, combine different attributes and are user defined, we adopt the following simplifying assumptions:

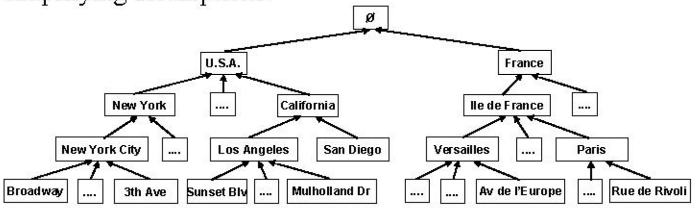

Fig. 1 Generalization tree of the location domain.

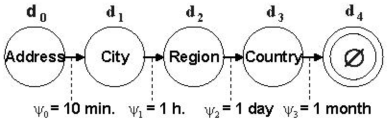

Fig 2. An example of an attribute's LCP.

- LCPs express degradation triggered by time
- LCPs are defined per degradable attribute
- LCPs apply to all tuples of the same data store uniformly

A Life Cycle Policy for an attribute is modelled by a deterministic finite automaton as a set of degradable attribute states $\{d_0, \ldots, d_n\}$ denoting the levels of accuracy of the corresponding attribute d, a set of transitions between those states and the associated time delays (Ψ_i) after which these transitions are triggered. The following figure shows an example of a LCP defined for the location attribute.

A tuple is a composition of *stable attributes* which do not participate in the degradation process and *degradable attributes*. The combination of LCPs of all degradable attributes makes that, at each independent attribute transition, the tuple as a whole reaches a new *tuple state* t_k, until all degradable attributes have reached their final state. A tuple LCP is thus derived from the combination of each individual attributes' LCP (see Figure 3).

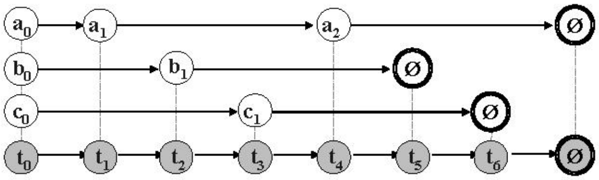

Fig. 3 Tuple LCP based on attribute LCP

Due to degradation, the dataset *DS* is divided into subsets ST_k of tuples within the same tuple state t_k, having a strong impact on the selection and projection operators of queries. These operators have to take accuracy into account, and have to return a coherent and well defined result. To achieve this goal, data subject to a predicate P expressed on a demanded accuracy level k, will be degraded before evaluating P, using a degradation function f_k (based on the generalization tree(s)). Given f, P and k, we define the select and project operators $\sigma_{P,k}$ and $\pi_{*,k}$ as:

$$\sigma_{\mathrm{P},k}(DS) = \sigma_{\mathrm{P}}\left(f_k\left(\bigcup_{i=0}^{k} ST_i\right)\right) \qquad \pi_{*,k}(DS) = \pi_{*}\left(f_k\left(\bigcup_{i=0}^{k} ST_i\right)\right)$$

The accuracy level k is chosen such that it reflects the declared *purpose* for querying the data. Then, queries can be expressed with no change on the SQL syntax as illustrated below:

```
DECLARE PURPOSE STAT SET ACCURACY LEVEL COUNTRY
FOR P.LOCATION, RANGE1000 FOR P.SALARY
SELECT * FROM PERSON WHERE LOCATION LIKE"%FRANCE%"
AND SALARY = '2000-3000'
```

The semantics of update queries is as follows: delete query semantic is unchanged compared to a traditional database, except for the selection predicates which are evaluated as explained above. Thus, the delete semantics is similar to the deletion through SQL views. When a tuple is deleted, both stable and degradable attributes are deleted. We made the assumptions that insertions of new elements are granted only in the most accurate state. Finally, we make the assumption that updates of degradable attributes are not granted after the tuple creation has been committed. On the other hand, updates of stable attributes are managed as in a traditional database.

III. Technical Challenges

Whenever an extension is proposed to a database model, and whatever the merits of this extension is, the first and legitimate question which comes in mind is how complex will the technology be to support it. Identifying the impact of making a DBMS data-degradation aware leads to several important questions briefly discussed below:

How does data degradation impact transaction semantics?

User transaction inserting tuples with degradable attributes generates effects all along the lifetime of the degradation process, that is from the transaction commit up to the time where all inserted tuples have reached a final LCP state for all their degradable attributes. This significantly impacts transaction atomicity and durability and even isolation considering potential conflicts between degradation steps and reader transactions.

How to enforce timely data degradation?

Degradation updates, as well as final removal from the database have to be timely enforced. As pointed out in [8], traditional DBMSs cannot even guarantee the non-recoverability of deleted data due to different forms of unintended retention in the data space, the indexes and the logs. In our context, the problem is particularly acute considering that each tuple inserted in the database undergoes as many degradation steps as tuple states. The storage of degradable attributes, indexes and logs have thus to be revisited in this light.

How to speed up queries involving degradable attributes?

Traditional DBMSs have been designed to speed up either OLTP or OLAP applications. OLTP workloads induce the need of few indexes on the most selective attributes to get the best trade-off between selection performance and insertion/update/deletion cost. By contrast, in OLAP workloads, insertions are done off-line, queries are complex and the data set is very large. This leads to multiple indexes to speed up even low selectivity queries thanks to bitmap-like indexes. Data degradation can be useful in both contexts. However, data degradation changes the workload characteristics in the sense that OLTP queries become less selective when applied to degradable attributes and OLAP must take care of updates incurred by degradation. This introduces the need for indexing techniques supporting efficiently degradation.

IV. Conclusion

The life cycle policy model is a promising new privacy model. Data degradation provides guarantees orthogonal and complementary to those brought by traditional security services. A clear and intuitive semantics has been defined for this model and related technical challenges have been identified.

In this model, we consider that state transitions are fired at predetermined time intervals and apply to all tuples of the same table uniformly. However, other forms of data degradation make sense and could be the target of future work. For instance, state transitions could be caused by events like those traditionally captured by database triggers. They could also be conditioned by predicates applied to the data to be degraded. In addition, since users do not have the same perception of their privacy, letting paranoid users defining their own LCP makes sense.

To answer a query expressed on a tuple state, we chose to consider only the subset of tuples for which this state is computable, thus making the semantics of the query language unambiguous and intuitive. However, more complex query semantics could be devised. In particular, selection predicates expressed at a given accuracy could also be evaluated on tuples exhibiting a lower accuracy. In addition, qualified tuples for which the projected attributes reveal a weaker accuracy than expected by the query could be projected on the most accurate computable value.

Finally, we proscribe insertions and updates in states other than the most accurate. This choice was motivated by avoiding the fuzziness incurred by allowing users modifying "past events". However, modifying degraded tuples may make sense when, for instance, incorrect values have been collected.

This paper is a first attempt to lay the foundation of future data degradation enabled DBMS. It introduces a set of new problems ranging from the definition of the degradation model up to the optimization of the DBMS engine, opening up an exciting research agenda.

References

[1] A. Machanavajjhala, J. Gehrke, D. Kifer, M. Venkitasubramaniam. "L-diversity: Privacy beyond k-anonymity," in *ICDE'06*, 2006.

[2] A. Meyerson, R. Williams, "On the complexity of optimal k-anonymity," in *PODS'04*, 2004.

[3] (2006) Computer Security Institute website, CSI/FBI Computer Crime and Security Survey, [Online]. Available: http://www.gocsi.com

[4] D. Hillyard, M. Gauen, "Issues around the protection or revelation of personal information," *Knowledge, Technology and Policy*, vol. 20(2), 2007.

[5] R. J., Hilderman, H. J., Hamilton, N. Cercone, Data Mining in Large Databases Using Domain Generalization Graphs. *J. Intell. Inf. Syst.*, *13*, 3 (Nov. 1999).

[6] J. W. Byun, E. Bertino, "Micro-views, or on how to protect privacy while enhancing data usability: concepts and challenges," *SIGMOD Record*, vol. 35(1), 2006.

[7] L. Sweeney, "K-anonymity: A model for protecting privacy," *Int. Journal on Uncertainty Fuzziness and Knowledge-based Systems*, vol. 10(5), 2002.

[8] P. Stahlberg, G. Miklau, B.N. Levine, "Threats to privacy in the forensic analysis of database systems," in *SIGMOD'07*, 2007.

[9] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, "Hippocratic databases," in *VLDB'02*, 2002.

[10] S. Chawla, C. Dwork, F. McSherry, A. Smith, H. Wee, "Toward privacy in public databases," in *Theory of Cryptography Conference*, 2005.

[11] X. Xiao, Y. Tao, "Personalized privacy preservation," in *SIGMOD'06*, 2006.

[12] EDRI, "German DP Commissioner against Google-Doubleclick deal," [Online].Available: http://www.edri.org/edrigram/number5.19/german-dp-google, Oct. 2007