Measuring Inference Exposure in Outsourced Encrypted Databases

E. Damiani, S. De Capitani di Vimercati, S. Foresti, P. Samarati, M. Viviani

Università deli Studi di Milano
Dipartimento di Tecnologie dell'Informazione
Via Bramante, 65 – 26013 Crema (CR) – Italia
{damiani, decapita, foresti, samarati, viviani}@dti.unimi.it

Abstract. Database outsourcing is becoming increasingly popular introducing a new paradigm, called *database-as-a-service*, where an encrypted client's database is stored at an external service provider. Existing proposals for querying encrypted databases are based on the association, with each encrypted tuple, of additional indexing information obtained from the plaintext values of attributes that can be used in the queries. However, the relationship between indexes and data should not open the door to inference and linking attacks that can compromise the protection granted by encryption.
In this paper, we present a simple yet robust indexing technique and investigate quantitative measures to model inference exposure. We present different techniques to compute an aggregate measure from the inference exposure associated with each single index. Our approach can take into account the importance of plaintext attributes associated with indexes and/or can allow the user to weight the inference exposure values supplied in relation to their relative ordering.

1 Introduction

In most organizations databases hold sensitive information that has to be protected form unauthorized accesses. As the size of these databases is increasing very quickly, organizations may choose if to add data storage to their systems at a high rate or to outsource data to external providers. The main advantage of outsourcing is related to the costs of in-house versus outsourced hosting: outsourcing provides significant cost savings and service benefits, and promises higher availability and more effective disaster protection than in-house operations. However, database outsourcing is not free from problems: since sensitive data are not under the direct control of their owner, data confidentiality and even integrity may be put at risk. These problems are traditionally addressed by means of encryption [5]. By encrypting the information, the client is guaranteed that it alone can access the data. However, since decryption must be executed only client-side for security reasons, the remote DBMS cannot execute any query because it has not access to plaintext data. Therefore, the whole relation involved in a query would be sent back to the client for query execution, thus nullifying the advantages of outsourcing.

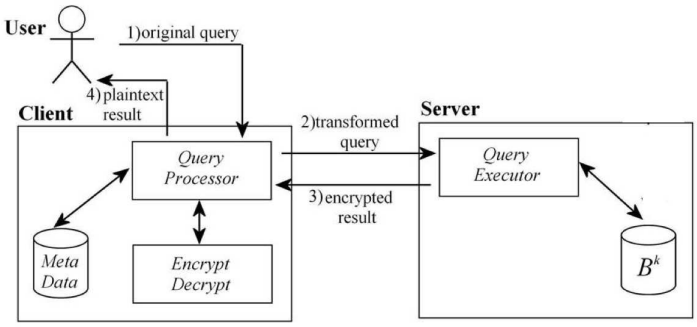

Fig. 1. DAS Scenario

A first proposal toward the solution of this problem was presented in [3, 4, 9–11] where the authors proposed storing, together with the encrypted database, additional indexing information. The scenario just described, called *database-as-a-service* (DAS), involves mainly three entities (see Figure 1):

- *User*: human entity that presents requests (queries) to the system (1);
- *Client*: front-end that transforms the user queries into queries on the encrypted data stored on the server (2) and decrypt the result of a query (4);
- *Server*: an organization that receives the encrypted data from a data owner and executes queries on them (3).

Two conflicting requirements need to be taken into consideration in the index construction: on the one side, the indexing information should be related with the data well enough to provide for an effective query execution mechanism; on the other side, the relationship between indexes and data should not open the door to inference and linking attacks that can compromise the protection granted by encryption [6]. To balance query execution efficiency and data protection from inference, it is important to provide indexing techniques able to balance these two requirements.

In this paper, after a brief explanation of data organization in the DAS scenario, we investigate quantitative measures to model inference exposure. We present different techniques to compute an aggregate measure from the inference exposure associated with each single index. The proposed techniques allow us to compute the inference exposure associated with a whole relation. The remainder of this paper is organized as follows. Section 2 describes the DAS scenario. Section 3 describes the abstract models used to compute the exposure coefficient in different scenarios. Section 4 illustrates different aggregation operators that can be used to combine the exposure coefficients associated with single indexes. Finally, Section 5 concludes the paper.

Employees

Id	Name	Age	Marital Status	Job
A1	Alice	30	Married	Manager
A2	Bob	26	Married	Director
B1	Alice	30	Married	Employee
B3	Carol	26	Single	Manager
B2	David	21	Single	Employee
A3	Alice	40	Divorced	Employee
B4	Bob	30	Single	Manager

(a)

Employees^{k}

Count	Etuple	I_1	I_2	I_3	I_4	I_5
1	r*tso/yui+	π	α	δ	μ	λ
2	hai4de-0q1	π	β	ϵ	μ	λ
3	nag+q8*L	ρ	α	δ	μ	γ
4	K/ehim*13-	σ	α	ϵ	η	λ
5	3gia*ni+aL	π	β	ϵ	η	γ
6	F0/rab1DW*	ρ	α	ϵ	μ	γ
7	Bid2*k1-l0	σ	β	δ	η	λ

(b)

Fig. 2. An example of plaintext (a) and encrypted (b) relation

2 Data Organization

We consider a relational DBMS where data are organized in tables (e.g., table `Employees` in Figure 2(a)); the underlined attribute represents the primary key of the table. In principle, database encryption may be performed at different levels of granularity: relation level, attribute level, tuple level, and element level. Both relation level and attribute level imply the communication to the user of the whole relation involved in a query. On the other hand, encrypting at element level would require an excessive workload for clients in encrypting/decrypting data. For balancing the client workload and query execution efficiency, we assume that the database is encrypted at tuple level.

The main effort of current research in this scenario is the design of a mechanism that makes it possible to directly query an encrypted database [9]. The existing proposals are based on the use of indexing information associated with each relation in the encrypted database [4, 11]. Such indexes can be used by the server to select the data to be returned in response to a query. More precisely, the server stores an encrypted table with an index for each attribute on which a query can include a condition. Each plaintext relation is represented in the encrypted database as a relation with an attribute for the encrypted tuple and as many attributes as indexes to be supported. Formally, each relation $\mathtt{r_i}$ over schema $\mathtt{R_i}(\mathtt{A_{i1}}, \mathtt{A_{i2}}, \ldots, \mathtt{A_{in}})$ in a plaintext database `DB` is mapped onto a relation $\mathtt{r_i^k}$ over schema $\mathtt{R_i^k}(\underline{\mathtt{Count}}, \mathtt{Etuple}, \mathtt{I_1}, \mathtt{I_2}, \ldots, \mathtt{I_n})$ in the encrypted database $\mathtt{DB^k}$ where, `Count` is the primary key; `Etuple` is an attribute for the encrypted tuple whose value is obtained using an encryption function E_k (k is the key); $\mathtt{I_i}$ is the index associated with the `i`-th attribute.[1] For instance, given relation `Employees` in Figure 2(a), the corresponding encrypted relation $\mathtt{Employees^k}$ is represented in Figure 2(b). As it is visible from this table, the encrypted table has the same number of rows as the original one. Let us now discuss how to represent indexing information. A trivial approach to indexing would be to use the plaintext value of each cell. This approach is obviously not suitable as plaintext data would be disclosed. An alternative approach providing the same fine-grained selection capability without disclosing plaintext values is to use the individual encrypted

[1] For the sake of simplicity, we assume that each attribute of the original relation has an index in the encrypted one.

values as index. Then, for each indexed cell the outcome of an invertible encryption function over the cell value is used. Formally, $t[\mathtt{I_i}] = E_k(t[\mathtt{A_i}])$. This solution has the advantage of preserving plaintext distinguishability, together with precision and efficiency in query execution, as all the tuples returned belong to the query set of the original query. As a drawback, however, encrypted values reproduce exactly the plaintext values distribution with respect to values' *cardinality* (i.e., the number of distinct values of the attribute) and frequencies.

A third alternative approach is to use as index the result of a *secure hash function* over the attribute values rather than straightforwardly encrypting the attributes; this way, the attribute values' distribution can be *flattened* by the hash function. A flexible characteristic of a hash function is the cardinality of its co-domain B, which allows us to adapt to the granularity of the represented data. When B is small compared with the cardinality of the attribute, the hash function can be interpreted as a mechanism that distributes tuples in $\mid B \mid$ *buckets*; a good hash function (and a secure hash has to be good) distributes uniformly the values in the buckets. For instance, the `Employees` table in Figure 2(a) can be indexed considering two buckets, α and β, for attribute `Name`, and mappings `Alice` and `Carol` in α and `Bob` and `David` in β (see Figure 2(b)).[2] With respect to direct encryption, hash-based indexing provides more protection as different plaintext values are mapped onto the same index (see Section 3). By contrast, when hashing is used, the query results will often include spurious tuples (all those belonging to the same bucket of the index) that will have to be removed by the front end receiving it.

As indexes constructed using hash or encryption functions do not preserve the domain order of the original attributes, they cannot support range queries. To this purpose, a fourth indexing approach based on *B+-trees* has been proposed in [4]. In the following sections, we will describe how to compute the inference exposure coefficient when a direct encryption method or an hash based method are used to compute the indexes.

3 Exposure Coefficient Measures

As discussed in the Introduction, it is important to be able to evaluate quantitatively the level of exposure associated with the publication of certain indexes and to determine the proper balance between index efficiency and protection. To this purpose, two different scenarios can be considered that differ in the assumption about the attacker's prior knowledge [4]. In the first scenario, called **Freq+DB**^{k}, the attacker is aware of the exact (or approximate) distribution of plaintext values in the original database in addition to knowing the encrypted database. In the second scenario, called **DB+DB**^{k}, the attacker has both the encrypted and the plaintext database. Note however that the computation of the exposure coefficient $\mathcal{E}$ depends also on the method adopted for indexing the database, that is, direct encryption or hashing. Figure 3 summarizes the abstract

[2] Here, the result of the hash function is represented as a Greek letter.

	Direct encryption	Hashing
Freq + DB^k	Quotient table	Multiple subset sum problem
DB + DB^k	RCV-graph	RCV line graph

Fig. 3. Abstract models supporting computation of exposure in the four attack scenarios

models that we used to obtain an indication of the exposure that characterizes generic databases. We now briefly describe the rationale behind these abstract models (we refer the reader to [2, 4] for a complete description of these models).

3.1 Direct Encryption Exposure

In the **Freq+DB**^{k} scenario, although the attacker does not know which index corresponds to which plaintext attribute, she can determine the actual correspondence by comparing their occurrence profiles. Intuitively, values with the same number of occurrences are indistinguishable to the attacker. The exposure of an encrypted relation to indexing inference can then be thought of in terms of an equivalence relation where indexes (and plaintext values) with the same number of occurrences belong to the same equivalence class. The measure of exposure for a single cell in the table is then equal to the inverse of the cardinality of the equivalence class to which it belongs. Consequently, the probability of disclosing a specific association (a tuple is a specific association) is the product of the inverse of the cardinalities of its cells. The exposure of the whole relation can then be estimated as the average exposure of each tuple as follows:

$$\mathcal{E} = \frac{1}{n} \sum_{i=1}^{n} \prod_{j=1}^{k} \mathrm{IC}_{i,j}$$

Here, i ranges over the tuples and j ranges over the columns, and $\mathrm{IC}_{i,j}$ denotes the exposure of value j in tuple i. The exposure coefficient can be computed in $O(n \cdot k)$, where n is the number of tuples and k is the number of attributes.

In the **DB+DB**^{k} scenario, the model of the attack is based on the definition of *RCV-graphs*. Given a relational table, the corresponding 3-colored undirected graph $G = (V, E)$, called the *RCV-graph* (i.e., the row-column-value–graph), is a graph where the set V of vertexes contains one vertex for every attribute, one vertex for every tuple, and one vertex for every distinct value in each of the attributes; if the same value appears in different attributes, a distinct vertex is introduced for every attribute in which the value appears. The set E of edges contains both edges connecting each vertex representing a value with the vertex representing the column in which the value appears and edges connecting each vertex representing a value with the vertexes representing tuples in which the value appears. This graph has an important property, that is, the RCV-graph built starting from a plaintext table is identical to the RCV-graph built starting

from the corresponding encrypted table. The identification of the correspondence between plaintext and index values requires then to establish a correspondence between the encrypted vertex labels and the plaintext values. This correspondence is strongly related to the presence of automorphisms in the RCV-graph. We used the *Nauty* algorithm [13] to produce a concise representation of all the automorphisms. The automorphisms over a graph constitutes a group that, for undirected graphs, can be described by the coarsest *equitable partition* [13] of the vertexes, where each element of the partition (each subset appearing in the partition) contains vertexes that can be considered interchangeable in an automorphism. The Nauty algorithm starts, for the group definition, from a partition on the vertexes that can be immediately derived grouping all the vertexes with the same color and connected by the same number of edges. This partition is then iteratively refined. From the structure of the partition $(C_1 \dots C_i)$, it derives that the vertexes appearing in the generic partition element C_j are equivalently substitutable in all the automorphisms, as they have exactly the same characteristics. From this observation, it derives that the probability p_i of a correct identification of a vertex $v_i \in C_j$ is equal to the inverse of the cardinality of C_j. Then, given $\mid C_j \mid$ vertexes in the partition element C_j, n elements in the equitable partition, and a total number m of vertexes, the exposure coefficient of the table is:

$$\mathcal{E} = \sum_{i=1}^{m} p_i/m = \sum_{j=1}^{n} \sum_{v_i \in C_j} p_i/m = \sum_{j=1}^{n} \sum_{v_i \in C_j} 1/(\mid C_j \mid \ m) = \sum_{j=1}^{n} 1/m = n/m$$

The exposure coefficient can be computed in $O(n^2 \log n)$, where n is the number of vertexes in the RCV-graph.

3.2 Hashing Exposure

It is important to note that collisions due to hashing increase protection from inference. The hash function is then characterized by a *collision factor* denoting the number of attribute values that on average collide on the same index value. As an example, consider the relation in Figure 2. Here, `Alice` and `Carol` are mapped on the same value α. The abstract models used for the computation of the inference exposure consider separately each attribute of the table; the inference exposure for the whole table can be obtained by aggregating the values associated with each single attribute (see Section 4). Note that while direct encryption indexing methods preserve the association of values of attributes within the tuples, the hash based methods do not preserve this association and therefore a potential intruder cannot use such information. Consequently, the exposure index computation is performed at attribute level and then each single value is aggregated to derive the exposure associated with the whole table.

In the **Freq+DB**^{k} scenario, the goal of the attacker is to find a mapping from plaintext values to indexing values that satisfies the constraints given by the attacker's prior knowledge which is represented by the occurrences of each plaintext value and each hashed one. The exposure coefficient is then computed

as follows. We first enumerate the different mappings by using an adaptation of Pisinger's algorithm for the subset sum problem. We then compute the exposure coefficient for each mapping and we take the average of these exposure coefficients. The exposure coefficient can be computed in $O(n^k)$, where n and k are values related to the number of different items in the index domain and their frequency in the encrypted table.

In the $\mathbf{DB}$+$\mathbf{DB}^k$ scenario, the exposure coefficient is computed by extending the RCV-graph described in the previous section. As before, identifying the correct correspondence between plaintext and hash values requires finding a matching between each vertex of the plaintext RCV-graph and a vertex of the corresponding encrypted RCV-graph. When collisions occur, the two graphs are not identical, as different vertexes of the plaintext RCV-graph may collapse to the same encrypted RCV-graph vertex. We can observe that the number of edges connecting row vertexes to value vertexes in the plaintext and encrypted RCV-graph is the same. Therefore, the problem can be viewed as finding a correct matching between the edges of the plaintext RCV-graph and the edges of the encrypted RCV-graph. Following this observation, we compute the exposure coefficient as the average of the exposure coefficients associated with an attribute in correspondence of each matching. The exposure coefficient can be computed in $O(n!n)$, where n is the number of nodes in the graph.

4 Exposure Coefficient Measures based on Aggregation Operators

The *aggregation operators* are mathematical objects that have the function of combining a set of numbers into a unique representative (or meaningful) number. As specified in the previous section, we are interested in computing the exposure coefficient associated with a whole table when indexes have been obtained by applying a hash-based method. To this purpose, it is possible to use one of the many operators that satisfy the definition of aggregation operator [14]. Formally, an aggregation operator is defined as follows.

Definition 1. *An operator* $A : \cup_{n\in\mathbb{N}}[0,1]^n \rightarrow [0,1]$ *is an* aggregation operator *on the unit interval if the following conditions hold:*

Identity property: $A(x) = x$,[3]
Boundary conditions: $A(0,\ldots,0) = 0$ *and* $A(1,\ldots,1) = 1$,
Monotonicity: $A(x_1,\ldots,x_n) \leq A(y_1,\ldots,y_n)$ *if* $(x_i \leq y_i)\ \forall i = 1,\ldots,n$.

Note that additional properties (*mathematical* and *behavioral*) may also be added [7, 8]. Although many aggregation operators [18] satisfy these properties, we consider the *Weighted Mean* (WM) [1, 16] and the *Ordered Weighted Averaging operator* (OWA) [19]. These two operators combine the input values according to a single set of weights. Therefore, to apply these operators, we first

[3] This property is required when the argument of the aggregation operator is a unary vector.

need to associate a weight (or set of weights) with each attribute of a relational table. The determination of these weights is usually done in an heuristic way (after trial and error) or asking an expert to supply them. We now describe the use of these operators more in details.

4.1 Weighted Mean

The Weighted Mean allows the system to compute an aggregate value from the ones corresponding to the exposure coefficient associated with each single index of a given table. This operator can take into consideration the risk connected to the disclosure of an attribute due to the inference from the corresponding index. The formal definition of a Weighted Mean operator is as follows.

Definition 2. *Let* $\mathbf{p} = [p_1\ p_2\ ...\ p_n]$ *be a weighting vector of dimension* n *such that:* $p_i \in [0,1]$; $\sum_i^n p_i = 1$. *A mapping* $f_{\mathrm{WM}} : \mathbb{R}^n \rightarrow \mathbb{R}$ *is a* Weighted Mean (WM) *operator of dimension* n *if*

$$f_{\mathrm{WM}}(a_1, a_2, \ldots, a_n) = \sum_i p_i a_i \ . \tag{1}$$

The weighting vector $\mathbf{p}$ is here used to reflect the *sensitivity* of the attributes in the original table. More precisely, an attribute is considered more "sensitive" than another attribute if its disclosure puts more at risk the outsourced database. As above-mentioned, there are several ways to choose the weights and we assume that a domain expert provides a vector depending on the context. This vector multiplied by the values of the exposure coefficients permits the evaluation of the robustness of the indexing method: a higher protection of the most sensitive attributes (a low exposure coefficient for the connected index) leads to a lower global exposure coefficient value; on the contrary, a lower protection of the most sensitive attributes leads to a higher global exposure coefficient value. The main advantage of using a weighted mean with respect to the classical mean is that it allows to make a distinction among the attributes of a table. For instance, if the exposure coefficients associated with the indexes computed from the more sensitive attributes is low, we would expect a low global exposure coefficient. Vice versa, if the exposure coefficients associated with the indexes computed from the more sensitive attributes is high, we would expect a high global exposure coefficient. If we use the classical mean operator, it is possible that a similar global exposure coefficient is obtained in both situations because it considers the attributes equivalent.

Example 1. Consider the tables in Figure 2 and suppose that the most sensitive attribute is `Name` followed by `Age`, `Marital Status`, `Job`, and `Id`. A possible weighting vector reflecting the sensitivity of the attributes is, for example, $\mathbf{p}$= [.05 .40 .30 .15 .10]. Suppose now that the exposure coefficient associated with each index is computed as discussed in Section 3.2: $\mathcal{E}_{\mathtt{I}}$ = [1/7! 1/36 1/36 1/8 1/8]. According to the WM definition, the global exposure coefficient is:

$$\left(\frac{1}{7!}\right)(.05) + \left(\frac{1}{36}\right)(.40) + \left(\frac{1}{36}\right)(.30) + \left(\frac{1}{8}\right)(.15) + \left(\frac{1}{8}\right)(.10) \cong .0507$$

If the exposure coefficients associated with indexes are $\mathcal{E}_{\mathtt{I}} = [1/7!\ 1/8\ 1/8\ 1/36\ 1/36]$, we would obtain $\mathcal{E} \cong .0944$ and we can conclude that the protection is worse than the first case because sensitive attributes are less protected.

It is important to note that the primary key of a table is always well protected because its values are indistinguishable. For this reason, in the above example, the weight associated with the primary key `Id` is very low.

4.2 Ordered Weighted Averaging Operator

The OWA operator allows the user to weight the input values in relation to their relative ordering. In this way a system can give more importance to a subset of the input values than to another subset. The OWA operator is formally defined as follows.

Definition 3. *Let* $\mathbf{w} = [w_1\ w_2\ ...\ w_n]$ *be a weighting vector of dimension* n *such that:* $w_i \in [0, 1]$; *and* $\sum_i^n w_i = 1$. *A mapping* $f_{\text{OWA}} : \mathbb{R}^n \rightarrow \mathbb{R}$ *is an* Ordered Weighted Averaging (OWA) *operator of dimension* n *if*

$$f_{\text{OWA}}(a_1, a_2, ..., a_n) = \sum_i w_i a_{\sigma(i)} \tag{2}$$

where $\{\sigma(1), \sigma(2), ..., \sigma(n)\}$ *is a permutation of* $\{1, 2, ..., n\}$ *such that* $a_{\sigma(i-1)} \geq a_{\sigma(i)}$ *for all* $i = 2, ..., n$.

The exposure coefficient associated with an index reflects how the corresponding attribute is protected. An higher exposure coefficient (always a value between 0 and 1) indicates that a particular attribute has a low protection and vice versa. Using an OWA operator, it is therefore possible to highlight this fact by choosing an appropriate weighting vector $\mathbf{w}$. In particular, there are two strategies that the data owner may adopt:

- *Maximal protection*: a table is considered protected only if all attributes are well protected (low exposure coefficient). Even a single not well protected attribute may cause a poor evaluation of the hash function adopted. In this case, it is necessary that the highest exposure coefficients have an higher weight to amplify their contribution to the final result ($\mathbf{w}$ has to be decreasing).
- *Minimal protection*: a table is considered protected even if just one of its attributes is well protected. In this case, it is necessary that the lowest exposure coefficients have an higher weight ($\mathbf{w}$ has to be increasing).

There are also many other intermediate strategies between these two ones, depending on the policy that the data owner has decided to adopt.

In summary, by comparing these two aggregation operators (WM and OWA), it is easy to see that in the weighted mean, weights measure the importance of an attribute with independence of the corresponding exposure coefficient. On the other hand, in the OWA operator weights measure the importance of an exposure coefficient (in relation to other values), independently from the attribute with which it is associated.

Example 2. Consider the table in Figure 2 and the exposure coefficients $\mathcal{E}_{\mathtt{I}} = [1/7!\ 1/36\ 1/36\ 1/8\ 1/8]$. We first order these coefficients thus obtaining the permutation: $\mathcal{E}_{\mathtt{I}\sigma} = \{1/8, 1/8, 1/36, 1/36, 1/7!\}$. If the data owner wants to apply a maximal protection strategy, she has to define a decreasing weighting vector such as $\mathbf{w} = [.30\ .30\ .20\ .15\ .05]$. In this case the exposure coefficient for the whole table is:

$$\left(\frac{1}{8}\right)(.30) + \left(\frac{1}{8}\right)(.30) + \left(\frac{1}{36}\right)(.20) + \left(\frac{1}{36}\right)(.15) + \left(\frac{1}{7!}\right)(.05) \cong .0848$$

On the contrary, if the data owner wants to apply a minimal protection strategy she has to choose an increasing weighting vector such as $\mathbf{w} = [.05\ .15\ .20\ .30\ .30]$. In this case, the exposure coefficient for the whole table is: $\mathcal{E} \cong .03894$ and is lower than the previous one, as the low value in $\mathcal{E}_{\mathtt{I}\sigma}$ has an high weight.

5 Conclusions and Future Work

We presented different measures for evaluating the robustness of indexing techniques against inference attacks. Issues to be investigated will include the analysis of more complex operators for the computation of the exposure coefficient of a whole table such as non linear operators and the WOWA operator.

References

1. Aczél, J.: On Weighted synthesis of judgments. Aequationes Math. **27** (1984) 288–307
2. Ceselli, A., Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Modeling and assessing inference exposure in encrypted databases. ACM Transactions on Information and System Security (TISSEC) **8(1)** (February 2005) 119–152
3. Damiani, E., De Capitani di Vimercati, S., Finetti, M., Paraboschi, S., Samarati, P., Jajodia, S.: Implementation of a storage mechanism for untrusted DBMSs. In Proc. of the Second International IEEE Security in Storage Workshop, Washington DC, USA (May 2003)
4. Damiani, E., De Capitani di Vimercati, S., Jajodia, S., Paraboschi, S., Samarati, P.: Balancing confidentiality and efficiency in untrusted relational DBMSs. In Proc. of the 10th ACM Conference on Computer and Communications Security, Washington, DC, USA (October 2003)

5. Davida, G.I., Wells, D.L., Kam, J.B.: A database encryption system with subkeys. ACM Transactions on Database Systems **6(2)** (June 1981) 312–328
6. Denning, D.E.: Cryptography and Data Security. Addison-Wesley (1982)
7. Fodor, J., Marichal, J.L., Roubens, M.: Characterization of the Ordered Weighted Averaging Operators. IEEE Transactions on Fuzzy Systems **3(2)** (1995) 236–240
8. Grabisch, M.: Fuzzy integral in multicriteria decision making. Fuzzy Sets and Systems **69** (1995) 279–298
9. Hacigümüs, H., Iyer, B., Mehrotra, S.: Providing database as a service. In Proc. of the 18th International Conference on Data Engineering, San Jose, California, USA (February 2002)
10. Hacigümüs, H., Iyer, B., Mehrotra, S.: Ensuring integrity of encrypted databases in database as a service model. In Proc. of the IFIP Conference on Data and Applications Security, Estes Park Colorado (August 2003)
11. Hacigümüs, H., Iyer, B., Mehrotra, S., Li, C.: Executing SQL over encrypted data in the database-service-provider model. In Proc. of the ACM SIGMOD'2002, Madison, Wisconsin, USA (June 2002)
12. Hore, B., Mehrotra, S., Tsudik, G.: A privacy-preserving index for range queries. In Proc. of the 30th VLDB Conference, Toronto, Canada (2004)
13. McKay, B.D.: Practical graph isomorphism. Congressus Numerantium, **30** (1981) 45–87
14. Mesiar, R., Komorníková, M.: Aggregation Operators. Proceeding of the XI Conference on applied Mathematics PRIM' 96, Herceg D., Surla K. (eds.), Institute of Mathematics, (1997) 193–211
15. Mykletun, E., Narasimha, M., Tsudik, G.: Authentication and integrity in outsourced database. In Proc. of the 11th Annual Network and Distributed System Security Symposium, San Diego, California, USA (February 2004)
16. Torra, V.: The weighted OWA operator. International Journal of Intelligent Systems **12(2)** (1997) 153–166
17. Torra, V.: On the learning of weights in some aggregation operators: the weighted mean and the OWA operators. Mathware and Soft Computing **6** (1999) 249–265
18. Xu, Z.S., Da, Q.L.: An Overview of Operators for Aggregating Information. International Journal of Intelligent Systems **18** (2003) 953–969
19. Yager, R.: On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on Systems, Man and Cybernetics **18(1)** (1988) 183–190