13

An information theoretic analysis of architectures for multilevel secure databases

J. E. Aisbett
Dept. Applied Computing & Mathematics
University of Tasmania
Launceston, Tasmania, Australia 7250
Email: Janet.Aisbett@appcomp.utas.edu.au
Telephone: 61 363 243401 Fax: 61 363 243368

Abstract

This paper develops a framework for assessing the security cost of implementations of a class of distributed database architectures, a framework which has previously been lacking in the literature. The value of information in a relational database is first introduced, i.e. a value is ascribed to the thing the security is to protect. In identifying sources of security costs, both the hiding of information from authorised users and the disclosure of information to unauthorised users are considered. Parameters which describe the effects of implementation and system usage on the security of the architecture are then determined. Finally, guidelines for estimating the information values and the implementation parameters required in calculating the security cost model are provided.

For simplicity only two security classes are considered, although the method would extend to any access control policies based on hierarchical classes. The cost model assumes nothing about the assurance of the implementation of the access control policy, and so applies equally to privacy considerations in the design of health care database systems or to national security considerations in military databases.

Keywords

Multilevel secure database, distributed databases, information security, information measure

1 INTRODUCTION

This paper develops a model of the cost of security of a database architecture, parametrised to describe operational patterns of data classification, database usage and access control security policy. Using this model, the relative security risk of an existing system can be established and compared with alternate database architectures.

P. Samarati et al. (eds.), *Database Security*

If enough is known about characteristics of the data to be protected, the model can also be used in the initial design stage, to select the architecture with least security cost.

A key notion in the analysis is the value of information in a relational database. Information, whether in a statistical or a logical semantic sense, is defined in terms of relative reduction in uncertainty (eg. (Carnap, 1963; Devlin, 1992; Lozinskii, 1994; Mackay, 1969)). There are three aspects to consider. Firstly, there is what has been called the **information capacity** of a database schema (Duzi, 1992; Hull, 1986), related to its structure. Then there is the impact of partial population of that schema. Finally, there is the **economic** aspect of information, as a factor of production used to improve decision making (Demski, 1980) whereby its value relates to its usefulness to the tasks at hand.

Another key idea in the modelling is that security cost involves factors other than disclosure which in the past have been neglected because policies of least privilege and disclosure prohibition have been considered paramount. Today's information systems are being developed in a spirit of maximal sharing, and so MLS database design can be seen to bring in issues of data **availability**. Thus our security cost model will take account of the cost of information hiding, which is equal to the value of the information withheld. It also takes account of correctness, confidentiality, and existence disclosure.

The remainder of the paper is organised as follows. Section 2 defines the information value of a relational database state, and Section 3 describes a generic MLS database architecture. Section 4 develops the notion of the security cost of such an architecture. Section 5 discusses how the necessary parameters and information values might be estimated. Section 6 illustrates application of these costings.

2 THE VALUE OF INFORMATION IN A RELATIONAL DATABASE

The security costing developed in this paper uses a measure of the information in an information system. This section looks at various definitions of the value of information, and outlines how statistical information theory can be modified to include more of the information semantics, that is, the underlying meaning. Here, the recipient of data is taken to be the 'average authorised user' of the system containing the data. (The value of the data to unauthorised users may of course differ.)

The classical formulation of statistical information theory (Shannon and Weaver, 1949) is

$$\text{information in } a = -\log_2 P(a). \tag{1}$$

where a is an information object belonging to a finite domain D from which individuals are drawn according to a probability P. The statistical approach is limited by its treatment of data as signals, which does not take into account semantics. The same limitations apply to the logical measures of information developed by Hintakka and Lozinskii and others (Carnap, 1963; Hintikka and Suppes, 1970; Lozinskii, 1994). Translated into the database setting, the logical formalisms define the probability of a fully populated database to be the reciprocal of the number of different ways the schema can be populated; and according to the Closed World Assumption of relational theory (Papoulis, 1965), every database is fully populated. The information value is then given by applying (1).

A user knows that the Closed World Assumption does not apply to real-world databases: just because a fact is not in the database does not mean its negation is true. Therefore in practice the information in a system increases as the database becomes

populated, and any practical measure of information must modify approaches based on frequency of occurrence to take into account the extent of population of the database.

Estimating frequencies of occurrence of records requires a fairly detailed knowledge of the underlying domains of the attributes. Standard relational database definition of domains is coarse: eg. integer, time, or $:c. Codd professes early confusion over whether columns in relational tables represent domains (Codd, 1990), and his later works argue that lack of support for domains is one of the most serious omissions of current database management systems. A salary given in dollars in a Bank's EMPLOYEE PERSONAL DETAILS table should be considered as belonging to a different domain to a salary in a BORROWERS PERSONAL DETAILS table, from a statistical information viewpoint because the distribution of salaries amongst employees will not be the same as amongst borrowers, and because semantically, the two concepts differ. We therefore define:

Definition 1

A database domain is a finite set Δ and a probability distribution P: $\Delta \rightarrow [0, 1]$ where P is **the probability of occurrence**.

The probability of occurrence need not be a good representation of semantic information content. For example, in the BORROWERS PERSONAL DETAILS table, the salary distribution will be flatter than the single-organisation employee database, thus the probability of any particular salary is less and, using (1), the information in a salary figure is greater. However, because computation of the bank's risk in lending to an individual will not be sensitive to small salary changes, the information value to the bank this figure is not as high as indicated by simple frequencies. Domains of character datatype are particularly likely to carry semantics not expressed by simple frequencies. Thus, in many cases an address field carries no information, being essentially part of the unique identifier of an individual, while for a market research company, the information in an address might be in the socio-economic status of the suburb.

To model the distortion of statistical information values introduced by a user's prior knowledge and current tasks, we need to provide for functional relationships between the domain underlying the database field and the domain of semantic interest. We also need a mechanism to focus attention on some members of a domain at the expense of others, according to their relative interest to the user.

In cases where a transformation between domains better reflects the informational interests of a system's users, then the system is not designed appropriately, or requirements have changed, or performance and other external factors have moderated the design. In any event, a domain transformation function would in practice be established through interview with the system users. There will typically be multiple users with different duties within the organisation, and so the transformation function must be some consensus of the individual transformations.

If users are performing repeatable tasks, it may be possible to list the facts of interest to them when undertaking their tasks, and their relative importance to those tasks. In (Aisbett and Gibbon, 1996) this is modelled as a set of statements with accompanying weights, where each statement is potentially derivable from the database, but may or may not be supported by the actual data. For example, if a database records individuals employers' and their international travel details, the statement 'John Smith works for Organisation X and travels to Laos more than 3 times a year' may be true, false or undecidable according to the data in the database. We will assume that the statements users are interested in are the database records (which assumes that the database designed is reasonably tailored to their requirements). Then the relative importance of such statements can be estimated from the number of accesses to database records in

the transaction log, in an established system with appropriate audit. The proportion of accesses to records with a particular attribute set gives a weight function on records which we call the **utility** of the information. These two functions are formally described:

Definition 2

A **domain transformation** is a function from a database domain to another domain (not necessarily finite).

A **utility function** w is a non-negative real-valued function on a database domain Δ. If P is the probability function on Δ and $\Im = \sum\{w(d): d \in \Delta\}$ then the domain is of **utility** $\Im$. If $w(d_1) > w(d_2)$, the occurrence of the value d_1 has **more utility** than that of d_2.

Suppose a table T with N rows is defined on product domain $\Delta_0 \, x \, \Delta_1$, with primary key Δ_0 (possible also a product of domains). Suppose members of the domain Δ_1 occur according to probability P. The table row $T(a, b)$ is a representation of a random variable $\boldsymbol{x}$ distributed according to P, and carries the statistical information -log $P(\boldsymbol{x}(a))$. The information in the table T, I_T, is

$$I_T = -\sum (\log P(\boldsymbol{x}(a): a \in \Delta_0), \tag{2}$$

summed over rows. If the frequency of values in the table accords with that defined by P, then

$$I_T = -N \sum\{P(x)\log(P(x)): x \in \Delta_1\}. \tag{3}$$

If Δ is a product domain Δ_1 x Δ_2 x ....x Δ_k and $x = (x_1, x_2, ...x_k) \in \Delta$, then by Bayes,

$$\log(P(x)) = \log P(x_1, x_2, ..., x_k) = \sum\{\log P(x_i | x_1, ..., x_{i-1}): 1 \le i \le k\}. \tag{4}$$

The extension of these equations to cater for domain transformations and utilities is straightforward. Suppose f is a domain transformation from Δ to Δ^f and w is a utility function on Δ^f and P^f is the probability of occurrences in Δ^f. The expected information I_T in a row of the table T is:

$$I_T = -\sum\{P(x)\, w(x)\, \log(P^f(f(x))): x \in \Delta\}. \tag{5}$$

If Δ is a product domain, then in cases such as the salaries example, f can be represented as a vector of monovariate functions: $f(x_1, x_2,...x_j,..., x_k) = (f_1(x_1), ... f_j(x_j),..., f_k(x_k))$. Similarly, if Δ^f is a product domain, w will in many cases be a vector of such monovariate functions.

Because we have ascribed utility to single database records rather than joins etc., the information in a database is at most the sum of the information in each of the tables. If a table in optimal normal form, say, has multiple dependent attributes, it can be decomposed into a set of binary tables without affecting the normalisation. But if two tables have the same primary key associated with different random variables, then their

join can have less information than the sum of the information in the two tables. To make the information value unique, we assume that there are no referential dependencies linking primary keys. Additional constraints which functionally define field values in terms of values in different tables reduce the total information in the database; a common example would be through arithmetic relationships. This effect may be difficult to estimate.

If such dependencies are not involved, a simple estimate of the information in a database with K binary tables, in which table i has N_i records and the dependent variable comes from a domain with M_i discrete values, all equally likely, is

$$-\Sigma\{ N_i \Sigma\{(1/M_i) \log (1/M_i): j = 1,...M_i\}: I = 1,... K\} = \Sigma\{N_i \log M_i: I = 1,... K\}. \quad (6)$$

This estimate is readily computable from the statistics in the database catalog.

3 DEFINITIONS AND ASSUMPTIONS ABOUT THE SYSTEM

We suppose data and users belong to two comparable security levels, High and Low, for which there are two physical security zones, with users having read/write privileges for the zone for which they are cleared. Traffic between the High and Low zones is policed by a security gateway, protecting both the confidentiality and integrity of the High zone data, the assurance of which depends on the gateway's implementation.

We assume the following about the operation of the system:

(i) There is no disinformation from High to Low: if High users update Low databases, they enter correct, if incomplete, information.
(ii) Low users are trustworthy at the Low level: if Low users update databases, they enter information they believe to be correct, if incomplete, information.
(iii) Inter-zone traffic is minimised: if Low labelled data are available in the High zone they will be accessed by a High user there rather than in the Low zone.
(iv) If High users can read both High and Low labelled versions of information about the same entity, then there is no cost if the information differs.

The last assumption effectively claims there is no cost to polyinstantiation, only a cost associated with non-availability. The security risk associated with polyinstantiation is generally treated as being due to loss of integrity. In practice, the ambiguity of polyinstantiation is resolved by assuming there is an official version of the database, usually that with the highest visible classification. The argument against this position is that in some cases lower classified users might have more up-to-date information. However, as long as the database architecture permits High users to update High label data with any Low label data readable by them, this argument flies in the face of the conventional assumption that all databases contain the best version of information available to the database maintainers.

Assumption (i) excludes a policy of sanitisation by disinformation (for example, by cover stories inconsistent with the real-world facts). Such a policy may enable inferences to be made through confirmation that there is something 'to hide', for example when fields which were populated are nulled, as proposed in (Jajodia and Sandhu, 1991).

The following notation distinguishes between the zone in which a datum is held (the system classification), the level at which it is classified (the labelled classification), and the security level of the information the datum contains (the semantic classification):

Definition 3

High$_{\text{physical}}$ **data** are data stored in the High zone.
High$_{\text{label}}$ **data** are data classified as High, that is, data with a High label.
High$_{\text{semantic}}$**data** are data with an information content that is sensitive.
Similarly for **Low**$_{\text{physical}}$, **Low**$_{\text{label}}$ and **Low**$_{\text{semantic}}$.

Best data are the subset of facts and rules in the full dataset which constitute the 'best' version of information available (in the sense of best reflecting the current state of the world modelled by the database). The ability to determine this subset is assumed to be a property of High users if the architecture permits them to see all the information in the system.

Conforming with usual practice, we assume information may be overclassified, but is never underclassified. Figure 1 then illustrates the relationships between the data sets.

The generic relationship between the data readable by the users in the respective zones and the data sets described above is illustrated in Figure 2.

We introduce some information value parameters. Recall that for these are with respect to the authorised users.

Definition 5

I is the value of the total information in the system -- that is, the information value of the best data.

P_H is the information value of the High$_{\text{physical}}$ data and P_L that of the Low$_{\text{physical}}$ data.

L_H is the information value of the High$_{\text{label}}$ data.

S_L is the information value of the Low$_{\text{semantic}}$ data.

V_H is the total value of information readable by the High user, and V_L that readable by the Low user, including information that can be inferred. V_{inf} is the value of the information in the High labelled data that can be inferred by the Low user.

$I - S_L$ will not be equal to the value of information that is semantically classified High, because the set of sensitive facts and rules and the set of those that are not sensitive will in general not be independent.

Under our assumptions the following relationships hold:

(i) $I \geq V_H \geq P_H$
(ii) $S_L \geq V_L - V_{\text{inf}} \geq P_L$
(iii) $P_H + P_L \geq I$.

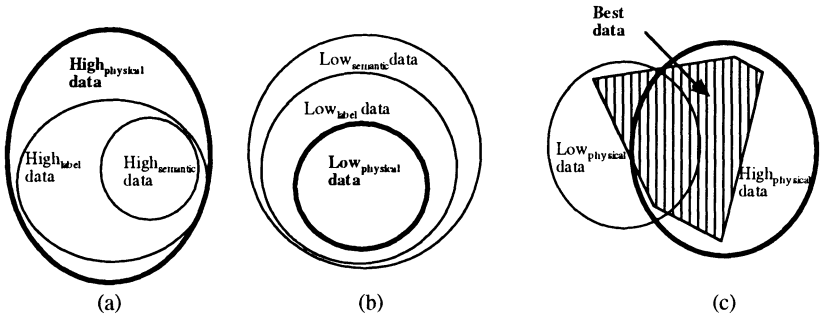

Figure 1 Assumptions about the data sets: (a) High physical data may include High and Low labelled data; semantically High datum is always labelled correctly. (b) Data may be overclassified, so that Low labelled data may not include all semantically Low data. Low labelled data may be held in the High zone. (c) Best data are drawn from the full information set.

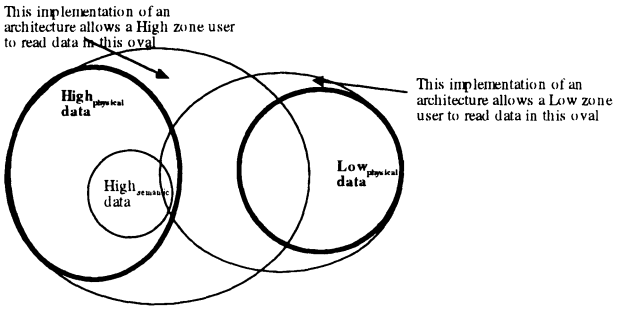

Figure 2 High zone users can read all the physically High data and may, depending on the information system architecture, read some or all of the physically Low data. Low zone users can read all the data in the Low zone (ie. physically Low data) and may, depending on the architecture, read some Low labelled data stored in the High zone. They may also be able to read semantically High data if the system implementation affords disclosure opportunities in breach of the stated access control security policy.

4 THE SECURITY COST MODEL

Our analysis costs loss of availability (information hiding), confidentiality, completeness and correctness.

Information lost to the Low user, of value S_L - V_L, results from mismatch between the database schema and the natural underlying data models. While field-level labelling can be implemented through binary tables in systems which support row-level labelling, overriding normal design criteria comes at the cost of performance (Jajodia and Mukkamala, 1993). Information loss also occurs when the best possible

version of information is not available to a Low user because the architecture does not support sanitisation.

Loss of information to a High user occurs when the database violates the principle of **completeness** and not all available information can be accessed -- that is, when High users cannot access the Low zone and information entered by Low users is not replicated in the High zone. The information hidden from High users is then $I - V_H$.

Any architecture which permits Low zone updates to High databases is, other things equal, more open to integrity violations than are other architectures. When the system automatically performs the update in a high assurance implementation such as SINTRA (Kang, Froscher, et al, 1994), this risk may be negligible; but when users perform updates on systems with weak integrity filters on the gateway, the risk may be significant.

High zone database writes issued by Low users may also generate High to Low traffic through error/acknowledgment messages, the curtailment of which narrows covert signaling bandwidths at a useability cost. Regardless of the assurance of the system, in practice any High to Low traffic introduces non-zero disclosure possibilities, which intuitively increase with the volume of traffic.

Therefore some security cost is incurred by each of the four possible types of gateway traffic, viz. Read or Write from High (resp. Low) to Low (resp. High). We assume the security cost of each of these types of traffic is proportional to the anticipated value of information passed through the gateway via each traffic type, which in turn is proportional to the information value of the type being passed. For example, the security cost incurred by reads from High of physical Low data is proportional to the value of information in the Low zone, P_L, with constant of proportionality we denote by R_{HL} that depends on the architecture and the implementation. Permissible reads from Low to High and permissible writes from High to Low can only involve Low label information, the value of which is $P_H - L_H$; and it is to this that the constants of proportionality R_{LH} and W_{HL} refer. Thus the security cost due to confidentiality and integrity (correctness) violation is modelled by the term:

$$R_{HL}(P_L) + W_{LH}(P_L) + W_{HL}(P_H - L_H) + R_{LH}(P_H - L_H). \tag{7}$$

Not all implementations of architectures will allow all these types of accesses. So set R_{HL} to zero if Read from High to Low is not permitted by an architecture's access policy and is prevented by the implementation; and similarly for the other parameters.

A source of disclosure not captured in the above is inference, the value of which we have denoted V_{inf}.

A weighting, call it ∂, must be introduced to reflect the relativity between cost of release of information to unauthorised users and cost of non-release of information to authorised users, which we have equated to the information value to those users. Since the weighting can be lumped into the parameters R_{HL}, etc., we explicitly apply it only to V_{inf}.

Taking into account the cost of information hiding, the total security cost of a system architecture becomes

$$\text{COST} = I - V_H + S_L - V_L + \partial V_{inf} + R_{HL}(P_L) + W_{LH}(P_L) + W_{HL}(P_H - L_H) + R_{LH}(P_H - L_H). \tag{8}$$

5 ESTIMATING THE VARIABLES IN THE SECURITY COST MODEL

This section looks at the estimation of the variables in equation (8).

Whether replication is at the table, row, or (conceptual) field level, the value of replicated data can be estimated using the equations presented in section 2. At the table level, the value of the replicated information is given by equations such as (4). With row or field level replication, a hypothesis must be made about the distribution of members of the domain of the replicated dependent variables. If replication is at the row level and replicated members come from the same distribution as the original table, then the value of the replicated information is just the fraction of rows replicated times the value of the original table. If not, the original table can be treated as two tables defined on different domains, one of which is fully replicated. The field level case is analogous.

Polyinstantiation can be treated as a generalisation of replication, potentially requiring a more subtle analysis since its effect on information values depends on the relationship between the versions. The enhanced role of domains in our approach allows them to carry the semantics of classification. If the salary column in an EMPLOYEE PERSONAL DETAILS table for a bank's employees is classified High then this fact is carried in the domain description. If the effect of the sanitisation rule relating salaries to rounded salaries, say, has been taken into explicit consideration, then the Low version in this case does not add anything to the value of the information in the total database because it is functionally determined by the High version. When the High database maintainers do not have access to the Low database because of the architecture, then the policy of maximal sharing is violated and as previously noted there will be information loss.

Computation of P_H and P_L is, in most architectures, straightforward, in the sense that these are the value of information in independent databases, and can be estimated as in section 2, with a rough estimate possible through equation (6). Computation of the other information values relies on careful understanding of the labelling and underlying classification of the data.

There are many ways that information can be sensitive, as examined for example in (Smith, 1992). We consider only the following:

(i) classification of a fact regardless of instantiation;
(ii) classification of a fact when it involves certain domain members;
(iii) classification of a particular alphanumeric value regardless of the domain or the entity to which it refers.

Case (i) includes the classification of columns and tables in databases, and (ii) covers most cases of classification of fields or rows on the basis of the particular instantiation. For example, the fact that a company keeps records on the HIV status of its employees may be classified. In other cases, only the values 'HIV positive' and 'AIDS confirmed' might be classified. The case (iii) is uncommon in practice: an example might be classification of the string 'HIV positive' as High wherever it appears in the database.

We need to understand the distribution of data between classifications. Suppose P is the probability on the dependent variables in a table which has N records and $k+1$ columns, of which the first is primary key. Suppose the jth dependent column in this table is defined on domain Δ_j and is labelled High (ie. is an example of type (i)) and the other columns are Low. The information in the High label data is just the difference between the information value in the full record, call it i_{H+L}, and that in the declassified record, call it i_L.

If the distribution of values in Δ_j is independent of those in other columns, the value of High labelled information in this table can be computed from the expressions for $i_{H+L}$and i_L, derived as in section 2:

$$i_{H+L} - i_L = -N \sum\{P_j(x)\log(P^{f_j}_j(f_j(x))): x \in \Delta_j\}. \quad (9)$$

(For simplicity, we ignore utility weightings.) Note that if the distribution is NOT independent, release of this table with only the column removed to Low users allows inferences about High zone data to be made. The information conveyed by the table is equal to the reduction in uncertainty about the value in the jth column which occurs because of knowledge of the remaining columns. This will be impractible to compute in practice, and indeed should not be necessary to compute as the database security administrators should eliminate such inference channels through appropriate classification.

We henceforth assume semantic High information to be independent of semantic Low information. This limits inference to existence information, treated presently.

If only particular values in a domain Δ_j are classified High, as in type (ii), then High information in the table will, on average, have value

$$i_H = -N \sum\{P_j(x)\log(P^{f_j}_j(f_j(x))): x \in \Delta_j \text{ and } x \text{ is a High value}\}. \quad (10)$$

Type (iii) extends this equation to cover all entries in the table. The generalisation of equations (9) and (10) to the case when multiple columns in a record are labelled High is equally obvious.

In going from the estimation of information in a single datatable to that in the whole database, we have to take into account duplication. The estimation of the total information value I in the High and Low databases is otherwise straightforward, as is that of S_L.

The calculation of V_{inf} is made feasible by our assumption that the security labelling of data is well managed and the only inference information is through existence information made available by some architectures. Consider a table T with dependence domain $\Delta 1$ which can be partitioned into a High classified subset of members and a Low subset. The details of the High subset are unknown to the Low user who can determine existence, and so this subset is represented as a single element in the domain, occurring with probability $\sum (P(x): x \in \Delta 1$ and x is a High value). Thus, if existence information can be determined at all, its expected value in a table is

$$V_{inf} = -\sum (P(x): x \in \Delta_H \text{ and } x \text{ is a High value}) \log(\sum (P(x): x \in \Delta_H \text{ and } x \text{ is a High value})). \quad (11)$$

The value of information whose existence is known is not directly comparable as a cost with the value of information hidden from a user, because the former relates to disclosure which is usually considered to be more serious than information loss. It is for this reason that the parameter ∂ was introduced in the previous section. Since ∂ is the heuristic ratio, **potential damage caused by disclosure/potential damage caused by information hiding**, it must be reevaluated for each particular application, taking into account the sensitivity of the classified information and the harm done by its disclosure; taking into account probable **a priori** information which Low users might have from other unclassified sources to build on the existence information; and taking into account the damage which could be caused by information loss in this situation.

Such a calculation is inevitably subjective and possibly unrepeatable -- as in much estimation in risk assessment.

The four parameters R_{HL}, W_{HL}, R_{LH} and W_{LH} link the risk of disclosure or integrity violation inherent in the four associated operations to the value of the information to which the operation could be applied. It is reasonable to assume that each of the parameters is dependent on the actual activity rate, defined as the number of users times the average activity per user per information element, and is dependent on the actual information system operational characteristics. They are also affected by the trust of the system, including the trustworthiness of the gateway to prevent the passage of invalid data or requests. Any quantification of such trust is implementation-dependent. For example, classified data might be appended by rogue software in the High side to a legitimate query: some gateways might detect such appendages. The actual likelihood of such rogue software being in place is another implementation-dependent factor which relates to the system and application software development environment.

The security value of the information is a third factor in the determination of the cross-gateway traffic parameters, and is not explicitly captured in our notion of information value. Although it could be incorporated in the utility weighting it seems cleaner to factor it in separately. Thus the likelihood of covert channels actually being used are a function of the potential damage of release of High data, and this can be used to magnify these parameters.

As well as the system and implementation dependencies which effectively scale the four cross-gateway traffic parameters, there are some specific factors which assist in estimating the appropriate relative sizes of the four parameters. So let us consider each parameter in turn.

The disclosure channel in reads from High to Low must be via the actual command, and could be a covert signalling channel in the timing of reads, or coded signalling in the content of the read requests. The risk to the integrity of the High data is in material passed back in response to a query.

The disclosure channel in writes from High to Low is in the actual maintenance command, which will also include any amended data involved in the transaction. There is very little integrity risk, being associated only with error or other advisory messages.

The disclosure risk in reads from Low to High is in the data passed in response to the query. For some systems, a very large volume could be involved, if for example only whole rows are allowed to be passed, and joins are carried out on the Low zone side. The integrity risk is very low.

The disclosure risk in writes from Low to High is low, being in the advisory messages if such are allowed. The integrity risk is relatively high. However, as we have said, in general integrity risks (which are one form of preventing an authorised user accessing correct information) are considered less costly than disclosure (allowing an unauthorised user to obtain information).

6 EXAMPLES

The architectural components considered are:

(i) data architectures, in terms of what classification data is contained in what zones;
(ii) access control security models, in terms of read/write permissions.

Any data architecture will only be compatible with some security policies, where these policies refer to operations on **data zones** and to **data operations** eg. read-down

means that a High zone user can query High or Low zone data (typically through the SQL **select** and **project** commands), write-up means that a Low zone user can update High zone databases (through **update**, **insert**, or **delete**). Strictly interpreted, a read-down, write-at-level policy would prohibit error messages on failure of a physical High zone query of a physical Low database. However, some error messages and handshaking may be allowed to cross the gateway from High to Low, so that the actual restrictions on the usual functionality available to a database user will depend on the implementation.

The traditional model of data in which whole documents are classified either as High or Low permits a variant in which all the Low objects are replicated in the High zone, thereby reducing the amount of read traffic from the High to Low zones. This data architecture, illustrated in Figure 3, we call the **basic replicated** data architecture. Suppose a security policy which reverses the standard Bell-Padula policy to read-at-level, write-up, which denies the High user update privileges to the Low database. There may also be information hiding. The cost of this architecture is

$$\mathrm{COST}(B) = S_L - P_L + W_{LH}(P_L). \tag{12}$$

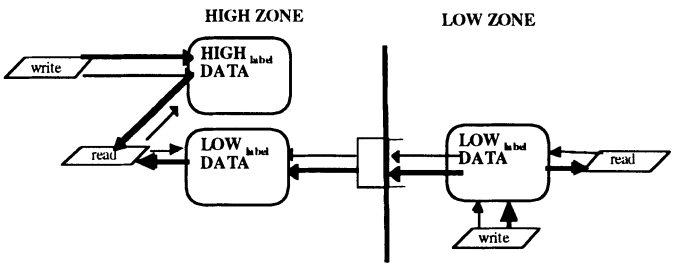

Figure 3 The basic replicated architecture: write-up, read-at-level.

In the paper world, Low users are sometimes permitted to see sanitised versions of High documents where sensitive words, paragraphs or sections are 'blacked out'. Similarly, a **replicated** database architecture has a complete version of the database located in the High zone, and a Low classification subset of it replicated in the Low zone. This allows censoring of precisely the information which is sensitive, but permits only one version of information, so Low users cannot be given a sanitised version of information. The architecture is fitted naturally by two security policies: the first, call it $R1$, being write-down, read-at-level; the second, $R2$, being write-up, read-at-level. Figures 4 and 5 illustrate. $R1$ places all the data maintenance load on the High zone users, which is in many situations operationally unsatisfactory since typically most users and most data are at the Low level. Problems also arise with architecture $R2$, because in most operational circumstances integrity of the High zone data must be guaranteed by prohibition of Low users from intentionally or unintentionally overwriting High label data. But if the Low user is prevented from updating their own zone database because of failure of an update known to be valid, then the existence of High labelled data is revealed, opening an inference channel.

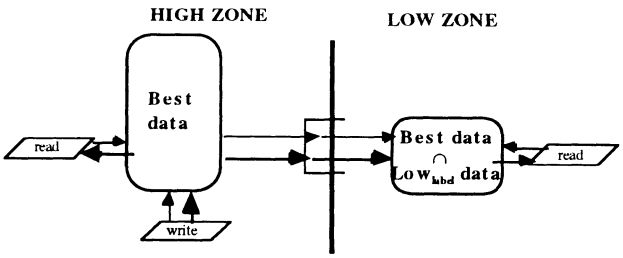

Figure 4 Database architecture $R1$: The replicated data architecture with write-down, read-at-level.

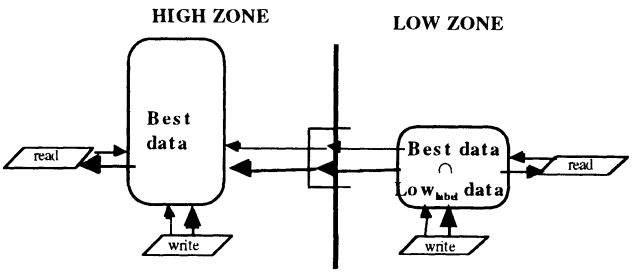

Figure 5 Database architecture $R2$: The replicated data architecture with write-up, read-at-level.

With either setup, the High user can see all information ($V_H = I$). Although the architecture is designed to maximise the amount of Low data available to a Low user, it still may result in information hiding because of the prohibition of sanitised versions of information.

For $R1$: Only the High user has update privileges ($W_{HL} \neq 0$) . There are no inference channels, so $V_{inf} = 0$ and $V_L = P_L$, and

$$\text{COST}(R1) = S_L - P_L + W_{HL}(P_L). \tag{13}$$

For $R2$: Low users can update Low information in both databases providing it does not overwrite existing High information. $V_L - V_{inf} = P_L$.

$$\text{COST}(R2) = S_L - P_L + \partial V_{inf} + W_{LH}(P_L). \tag{14}$$

7 CONCLUSION

This paper presented a framework for assessing the 'security cost' of implementations of a class of distributed database architectures. Its contribution was on two fronts:

(i) introducing an information theoretic approach to support a systematic analysis of MLS database architectures;
(ii) explicitly costing hidden information, in line with the growing movement toward maximised sharing security policies rather than narrowly-defined need-to-know.

8 REFERENCES

Aisbett, J. and Gibbon, G. (1996), A practical measure of the information in a logical theory. Submitted to *J. Exp and Theoretical AI.*

Codd, E. (1990) *The relational model for database management Version 2*. Addison Wesley.

Carnap, R. (1963) *Logical foundations of probability*. University Chicago Press.

Demski, J. (1980) *Information Analysis*. Addison Wesley.

Devlin, K. (1992) *Information and Logic*. Cambridge Univ. Press.

Duzi, M. (1992) Semantic information connected with data. *Lecture Notes in Computer Science*, **646**, 376-90.

Hintikka, J. and Suppes, P. (ed.) (1970) *Information and Inference*. D. Riedel., Dordrecht.

Hull, R. (1986) Relative information capacity of simple relational database schemata. *SIAM J. Computing*, **15(3)**, 856-85.

Jajodia, S., Sandhu, S. and R. (1991) Toward a multilevel secure relational data model. *ACM SIGMOD* , 50-59.

Jajoida, S. and Mukkamala, R. (1993) Effects of SeaView decomposition of multilevel relations on database performance, in *Database Security* (ed. Landwehr, C. and Jajoida, S.), North Holland.

Kang, M.H., Froscher, J., McDermott, J., Costich, O., and Peyton, R. (1994) Achieving database security through data replication: the SINTRA prototype. *Proceedings of the 17th National Computer Security Conference*, September 1994.

Lozinskii, E., (1994) Information and evidence in logic systems, *J. Exp and Theoretical AI.*, **6**, 163-93.

Mackay, D. (1969) *Information, mechanism and meaning*. MIT Press.

Papoulis, A. (1965) *Probability, random variables and stochastic processes*. McGraw-Hill.

Reiter, R. (1984) Towards a logical reconstruction of relational database theory, in *On Conceptual Modelling*, (ed. Brodie, M., Myopoulos, J. and Schmidt, J.), Springer-Verlag, New York.

Shannon, C. and Weaver, W. (1949) *The mathematical theory of communication.* University of Illinois Press.

Smith, G. (1992) Classifying and downgrading: is a human needed in the loop?, in *Research Directions in Database Security,* (ed. Lunt, T.), Springer Verlag.