Authentication Hierarchy in Distributed Deductive Databases

Dorel Săvulea  
University of Craiova  
Department of Informatics  
Al.I. Cuza Street, No. 13, Craiova  
ROMANIA  
savulea@central.ucv.ro

Nicolae Constantinescu  
University of Craiova  
Department of Informatics  
Al.I. Cuza Street, No. 13, Craiova  
ROMANIA  
nikyc@central.ucv.ro

Abstract: Distributed deductive databases have become more and more popular in the past decade. Their importance is mainly due to their low costs and the high level of protecting data. For such a database there must be secured all the remote database fragments and the infrastructure in order to provide a high security level and to avoid an impersonation. We propose an algorithm for providing such a security level with less resources. The algorithm is based on authenticating the rules using digital signature.

Key–Words: Authentication; Distributed Deductive Databases; Digital Signature, Identity Prove, Deductive Database Fragmentation.

1 Introduction

Distributed deductive database security systems have been an intensive field of research in the past decade. There have been added rules to the relational database systems for including deductive capabilities. A deductive database system is a database system which can deduce or infer additional information based on rules and facts stored in the database [1]. In the mean time, a distributed database system is a collection of multiple, logically interrelated databases distributed over a computer network or over the Internet [2].

The process of distributing databases has a lot of advantages. They allow local autonomy of data so that the management is partitioned to different parts, being very important for voluminous databases [3]. Moreover, such systems provide improved reliability due to replication of database and availability of data. It also provides fast access to local data and support for incremental growth. The performance is improved by splitting the queries of the database into subqueries enhancing query processing since the answers to each subquery are computed only against those portions of the database relevant to the subquery being generated in parallel. The process of evaluating these queries and subqueries independently improves the response time and reduces the resources cost.

Using such a system is much cheaper due to the fact that it costs less to create a network of smaller computers with the power of a single large computer. The modularity is also a big advantage of the distributed systems since they can be modified, added and removed from the distributed database without affecting other modules (systems) [6, 5].

The failure of a single module does not affect the others and nor the performance of the system. All the transactions of a distributed deductive system follow the A.C.I.D property:

- Automaticity: the transaction takes place as whole or not at all;
- Consistency: any changes to values in an instance are consistent with changes to other values in the same instance; a consistency constraint is a predicate on data which serves as a precondition, post-condition, and transformation condition on any transaction.
- Isolation: each transaction sees a consistent database; the following degrees of isolation were originally described as degrees of consistency by Jim Gray [4]:
  - Durability: the ability of the system to recover committed transaction updates if either the system or the storage media fails.

An authentication stage not only provides a high security level for the system due to the fact that the users are identified, but also it ensures integrity and non-repudiation for the rules or queries that user sends. We propose a method which signs each fragment of the query and their verification is made by different servers of the system. The servers are chosen according a dictionary of predicates.
2 State of Art

Distributed databases systems are largely used in industry, banking and administration. A great interest has appeared in applying logic to databases, particularly in deductive database systems, which not only manage large facts stored in relations in a database and rules in a rulebase but also provide for deduction from given database and rulebase [2, 7]. To define relations, rules and user queries there must be used a declarative language, the most popular one being Datalog. In [8] the author presents the similarities between rules and relational views. The rules specify derived relations that are not stored in the database but that can be formed from facts through inference mechanisms based on the specifications of the rule. Unlike the views which do not involve recursion, the rules do.

Distributed deductive database systems may serve a single organization or multiple organizations who want to facilitate sharing databases and rulebases.

The rules are defined on the global relations. Rules should be defined on data fragments because data fragments and rules are logical units of allocation in distributed deductive systems. Rules can be fragmented identically as the relations (horizontally and vertically) since all the derived predicates in the global rulebase can be seen as defining relations.

The collection of rules and rules is named knowledge base. After the relations are fragmented they are allocated across the sites to speedup query processing. When the rules are incorporated in the system they can refer to fragments of relations.

To keep the distributed database system up to date there are used two methods: replicating data or duplicating data [11]. The first needs a specialized software which looks for changes that have occurred in the database. After identifying the changes the replication process will make all the databases look the same. The complexity of this process increases proportionally with the number of the system’s databases. Duplication implies identifying one database as a master and duplicating it. Such a process is recommended to be done at a set time after hours. This is to ensure that each distributed location has the same data. During duplication there cannot be made changes to any database except the master one.

The replication of rules has been studied in detail in [12], having the possibility to approach it in three ways:

- No replication: the rulebase can be partitioned into non-overlapping clusters, one for each site. In such cases the rule might compile and execute locally without needing any communications. If a rule uses rulebases from different sites compiling it might need access planned by the system from a distributed execution.

- Full replication: in such cases the global rulebase is stored fully on each site. If it is replicated fully the rule compilation is completely local, but it also needs access planning for distributed execution of the rule when it uses data from several sites.

- Controlled replication: this approach falls in between the above two approaches since the replication may be at rule or cluster level.

To exemplify the above we use the Datalog rules. Datalog is used to define rules declaratively in conjunction with an existing set of relations. A deductive database system rule $r$ is defines as:

$$p(X_1, \ldots, X_n) : - q_1(Y_1, \ldots, Y_m), \ldots, q_l(Z_1, \ldots, Z_s)$$

where $p$ is the head (predicate) of $r$ and can be derived or mixed predicate. A predicate $p$ is mixed if there is a set of ground facts for $p$, and $q$ appears as the head predicate of some rules [9]. $q_1 \ldots q_l$ form the body of $r$ and can be derived, mixed or base predicates. A base predicate corresponds to a relation in the database. The argument of a predicate is a variable or a constant.

The above formula can be simplified to:

$$p_0(X_0) : - p_1(X_1) \ldots p_n(X_n)$$

where each $p_i(X_i)$ is called a literal. The head of the rule is $p_0(X_0)$ and the other part represents the body. A rule which has an empty body and all $X_i$ are constants is a fact. A query is a rule that does not have a head. A rule is recursively if its head appears in the body. The predicate $p$ directly depends on a predicate $q$ if the latter appears in the body of $p$. The transitive dependence is called indirect dependence. $p$ and $q$ are mutually recursive if they belong to the same cycle of an indirect dependence relationship. A predicate may have multiple definitions since multiple rules can have the same head predicate. To better understand how a distributed deductive database system works we illustrate the architecture schema presented in [5]. The global schema from the top consists in defining all the global relations and rules which are contained in the distributed deductive database system as if the database and rulebase where not distributed at all.

The fragmentation schema defines the mapping between global relations and the their fragments, and rules and their fragments. Such a mapping is defined as one-to-many which means that several fragments of a relation or of a rule correspond to one global relation
Figure 1: A distributed deductive system architecture

or one global rule, respectively. In the meantime only one relation or rule can correspond to one fragment of a relation or of a rule, respectively.

After fragmenting the rules and data they are allocated to different parts of the system. Suppose these parts are several sites from a network. These sites are defined by the allocation schema. Fragments of relations and rules are logical units of allocation. The allocation schema also determines if the fragments are replicated or nonreplicated. If the fragments are replicated the mapping is one-to-many while in the nonreplication case the mapping is one-to-one.

The local mapping schema is different from site to site. Such a schema is used for mapping the physical images of the objects which are manipulated by the local deductive database systems.

Such an architecture permits the user (or an application program) to submit queries at three levels: global level, fragmentation level and allocation level [10]. In the case of a global level query the user works on global knowledge base which is defined as a collection of relations and rules. Note that at this level the user does not know anything about the fragmentation or the allocation process of the system. The distributed deductive database system transforms the initial query in subqueries, each subquery corresponding to a fragment. The query is interpreted by accessing the rulebases or databases (or both) from any site. So, a global query will never be affected by any change in fragmentation or allocation.

When a user submits a query at a fragmentation level he is aware of the fragmentation of relations and rules, but not of the allocation of them. As you may have already figured, such a query is not affected by any change in the allocation process but it depends on the changes from the fragmentation process since the fragmentation structure is incorporated in the query. For interpreting a query at a fragmentation level the system determines the site of accessing rulebases or databases.

When a user submits a query at the allocation level he is aware of both fragmentation and allocation of a knowledge base. Despite this the user must specify at which site the query interprets. So, in the case of such a query the access is routed by the system to a specific site.

The management system of a database normally runs on top of an operating system which provides the security of the database. The security features of an operating system include memory and file protection, resource access control and user authentication [15]. Most of the distributed database systems does not have implemented other security algorithms [16]. So the security of such deductive database systems relies on the security of the operating system.

Our method provides an authentication process for all the subqueries that the global query was fragmentated in. Each subquery is verified by a different server, independent of the operating system. The next section describes how the fragmentation of the global query is made and how the servers are chosen.

### 3 Rules Authentication with Digital Authentication

Using distributed deductive database systems has a lot of advantages especially regarding their low costs and the high level of protecting data. However this kind of systems need extra attention for their security since all the remote database fragments and the infrastructure must be securized. To do so we recommend encrypting the network links between remote sites.

Another security problem that must be taken in consideration for such systems is providing access to users (or application programs). Not all the users must have the same rights on the system. There must be a master administrator who can modify, update, and delete data and rules from the entire system, regardless of their location. Each local part of the distributed deductive database system has a local administrator which has the permission to modify, update or delete only data and rules from his local database. The other users of such a system can be:

- users who can access only their local database;
- users who can access the local database and the other ones from the system.
User access to shared databases are formulated as transaction, which are units of execution that satisfy the A.C.I.D properties. Concurrency control involves the synchronizations of accesses to the distributed database, such that the integrity of the database is maintained. The most popular consistency control algorithms are locking-based. Such a scheme implies placing a lock on some unit of storage whenever a transaction attempts to access it. These locks depend on the lock compatibility rules. An important theorem says that:

"No lock on behalf of a transaction should be set once a lock previously held by the transaction is released."

Such a process is named a two-phase locking since transactions go through a growing phase when they obtain locks and through a shrinking phase when they release them. In general, most of the concurrency control algorithms are strict in holding their locks until the transaction ends. This is because releasing it before the transaction ends may cause serious problems. The locking process may be:

1. Centralized locking: in such cases there is used a single lock table for the entire distributed database; this lock is placed on one of the sites being controlled by only one manager.

2. Primary copy locking: the locking is made only on a primary copy in order to access the respective item; this method is useful in replicated databases where an item may have several copies and one of them (the primary one) has a special design.

3. Distributed locking: in this case the lock management duty is shared by all the sites in the system; the transaction execution involves lock managers from several sites, locks being obtained at each site where the transaction accesses an item.

Concurrency control algorithms based on locking may cause deadlocks whose detection and management in a distributed system is very difficult. However they are more performant and simpler than timestamp-based algorithms.

In a distributed system we can have four types of failures:

1. transaction failures: can be caused by an error in the transaction due to an input data or by an error in the transaction code;

2. site (system) failures: are caused by hardware failure or by a software failure;

3. media failures: are caused by introducing redundancy of storage devices and maintaining archival copies of the database;

4. communication failures: are caused by error in the messages, improperly ordered messages, lost or undelivered messages and line failures.

To avoid all the above problems and provide a high security level we propose a method for authenticating each user such that the management system provides a hierarchal access to data and rules. So each user or administrator has an account which helps him authenticate himself to the system. For the authentication part we propose using a digital signature algorithm. Such an algorithm must have three steps: generating keys, generating signature and verifying signature. Using such an algorithm the system manager will be able to recognize the owner of each query by verifying the signatures. Suppose the administrator $A_1$ of the site $s_1$ wants to submit a query:

$$-p_1(X_1) \ldots p_n(X_n)$$

We note $S$ the central server and $S_i \forall 1 \leq i \leq n$ are the connected servers where are stored the deductive databases. We define an information dictionary in $S$ which contains a list consisting in predicates that are solvable by a single partition $S_i$, where $p_j$ is the head of one or more rules. So $S_i \forall 1 \leq i \leq n$ contains a set $\{p_1^i, p_2^i, \ldots, p_m^i\}$ where all the predicates can be solved independently of other partitions.

**Definition 1** Independent Rules Partition. A partition $S_i$ is Independent Rules Partition (I.R.P) if for all $p_j^i$, where $p_j^i$ is the head of a rule from $S_i$, we have:

$$p_j^i : q_1^i, \ldots, q_l^i$$

where $q_1^i, \ldots, q_l^i$ can be solved with rules and facts only from $S_i$.

Suppose the administrator $A_1$ wants to submit the query:

$$-p_1(X_1) \ldots p_n(X_n)$$

Before signing it, the query is fragmented in $m$ ordered I.R.Ps:

$$p_1^{i_1} \ldots p_1^{i_{l_1}}, p_2^{i_2} \ldots p_2^{i_{l_2}}, \ldots, p_m^{i_m}$$

where $S_i$'s are the connected servers where are stored the deducational databases. We define an information dictionary $S_i$ which contains a list consisting in predicates that are solvable by a single partition $S_i$, where $p_j$ is the head of one or more rules. So $S_i$ contains a set $\{p_1^i, p_2^i, \ldots, p_m^i\}$ where all the predicates can be solved independently of other partitions.
Each subquery is signed and submitted to its corresponding partition:

\[
[\text{Sign}(A_1, (p_1^{i_1} \ldots p_1^{i_{m_1}})), (p_1^{i_1} \ldots p_1^{i_{m_1}})] \rightarrow S_{i_1}
\]

\[
[\text{Sign}(A_1, (p_2^{i_2} \ldots p_2^{i_{m_2}})), (p_2^{i_2} \ldots p_2^{i_{m_2}})] \rightarrow S_{i_2}
\]

\[
\ldots
\]

\[
[\text{Sign}(A_1, (p_n^{i_n} \ldots p_n^{i_{m_n}})), (p_n^{i_n} \ldots p_n^{i_{m_n}})] \rightarrow S_{i_n}
\]

When the management system receives one subquery, it verifies the signature to see the rights on the database of the owner. If the signature does not match with any user the management system sends an error message back to the user and stops the evaluating process for the received subquery. If the signature is valid the management system continues the evaluating process.

Each subquery is evaluated with an inference system in the rulebase and with a relational database system in the database. To perform such an evaluation may be used two methods: the interpretative method which evaluates a query tuple at a time in the rulebase and then evaluates it in the database. The latter one is much more advantageous than the former one mainly because it generates fewer database system requests.

The general answer is obtained after evaluating all the subqueries and collecting the partial results. When the evaluation process ends the management system verifies if the global query response fits in its owner rights on the system. Suppose the query that \( A_1 \) submitted was attempting to modify data from site\(_2\). When the management system evaluates the response it will send an error messaging because \( A_1 \) does not have the right to modify no other database except site\(_1\). If the query attempts to modify site\(_1\) the management system will end the process successfully and will send the output to the administrator.

### 3.1 Signing and Verifying Signature

For signing the subqueries we chose the ElGamal signature algorithm [14]. This algorithm allows that a verifier can confirm the authenticity of a subquery sent by the signer over an insecure channel. For this system we need three parameters: \( H \) a hash function which is resistant to collisions, \( p \) a large prime such that the discrete logarithms modulo \( p \) are very difficult to compute, \( g \) a randomly chosen generator for \( \mathbb{Z}_p^* \). Then the signer follows the steps:

1. choose a random secret key \( 1 < x < p - 1 \)
2. \( y = g^x \mod p \)
3. the public key is \( (p, g, y) \) and the private one is \( x \)

He then generates the signature for signing the subqueries:

1. choose a random \( k \) such that \( \gcd(k, p - 1) = 1 \) where \( 0 < k < p - 1 \)
2. \( r = g^k(\mod p) \)
3. \( s = (H(sq) - xr)k^{-1}(\mod p - 1) \)
4. the signature for the subquery \( sq \) is \( (r, s) \)

When the management system verifies the signature it follows:

1. verifies if \( 0 < r < p \) and \( 0 < s < p - 1 \)
2. verifies if \( g^{H(sq)} = y^r r^s(\mod p) \)

If both the above conditions are true then the signature is valid, and if at least one of them is false the management system rejects the signature.

### 3.2 How it works

Suppose the \( S \) dictionary (defined for Intensional Databases \( S^I \) and Extensional Databases \( S^E \)) is:

- \( S^I_1 = \{p_2, p_3, p_5\} \)
- \( S^I_2 = \{T(3), T(6), T(9), T(12), T(18), T(24), T(27)\} \)
- \( S^E_1 = \{p_1\} \)
- \( S^E_2 = \{R(5), R(10), R(15)\} \)
- \( S^E_3 = \{p_4\} \)
- \( S^E_4 = \{F(7), F(14), F(21)\} \)

and the rules are:

\[
\begin{align*}
p_1(a, b) & : = \ R(a). \\
p_1(a, b) & : = \ R(b). \\
p_2(a) & : = \ p_3(a, 3), T(a). \\
p_3(a, b) & : = \ p_5(a), T(b). \\
p_4(a, b) & : = \ F(a), F(b). \\
p_5(a) & : = \ T(a), T(a - 3).
\end{align*}
\]

The administrator of the site\(_2\), \( A_2 \), sends the following query:

\[
: \ p_2(6), p_3(9, 12), p_1(5, 0), p_4(7, 14), p_5(12)
\]

Next we fragmentate the query in subqueries respecting the ordinated partitions:

\[
\begin{array}{lllll}
p_2(6) & p_3(9, 12) & p_1(5, 0) & p_4(7, 14) & p_5(12)
\end{array}
\]

Each subquery is signed and submitted to the corresponding partition:
After verifying the signatures by each of the servers $S_1, S_2, S_3$, they resolve the subqueries. The partial answers are sent to the management system from which is obtained the answer, noted $Ans$ for the initial query that $A_2$ has submitted. If $A_2$ has the right to do $Ans$ then the management system resolves the request according with $Ans$, else an error message is sent to $A_2$. We can see that none of the above rules does not have predicates which can be solved by other servers than the corresponding one, for example:

$$p_6 : \neg p_2(b), p_4(7, 14).$$

This rule cannot be solved only by one server because $p_2(b)$ can be solved only by $S_1$ and $p_4(0, 1)$ can be solved only by $S_3$.

## 4 Conclusions and Future Work

Distributed deductive database systems have become a reality in the past decades. They provide functionality of centralized database management systems in an environment where data is distributed over the sites of a computer network or the nodes of a multiprocessor system. Distributed databases have allowed working with voluminous databases by simply adding new machines. In order to meet a high level of performance and data protection the distributed deductive database systems need to be designed with special consideration for protocols and strategy. Authenticating the users is highly recommended for providing the system hierarchy, improving concurrency control. Using a digital signature scheme is the easiest way to provide a high security for the system. The digital signature scheme may be an elliptic curve algorithm or a classic one. The former may be considered more performant since it uses much smaller keys. As future work we plan to study such an authentication system for a multidatabase deductive system. A multidatabase system is a system which provides access independently designed and implemented, and possibly heterogeneous, databases [13]. We also plan to apply such a system to rules that are not solvable by only one server. In this way, we intend to apply the security issue in more complex knowledge database which can be treated in accordance with the representations from [17, 18]. We work on the performance requirements of the system since applying it to rules that depend on various servers implies much more computations and complicates the evaluation process.

## References:


