An Experimental Evaluation of Multi-key Strategies for Data Outsourcing

E. Damiani[1], S. De Capitani di Vimercati[1], S. Foresti[1], S. Jajodia[2], S. Paraboschi[3], and P. Samarati[1]

[1] Università degli Studi di Milano, 26013 Crema - Italy
{damiani,decapita,foresti,samarati}@dti.unimi.it
[2] George Mason University, Fairfax, VA 22030-4444 jajodia@gmu.edu
[3] Università di Bergamo, 24044 Dalmine - Italy parabosc@unibg.it

Abstract. Data outsourcing is emerging today as a successful solution for organizations looking for a cost-effective way to make their data available for on-line querying. To protect outsourced data from unauthorized accesses, even from the (honest but curious) host server, data are encrypted and indexes associated with them enable the server to execute queries without the need of accessing cleartext. Current solutions consider the whole database as encrypted with a *single key* known only to the data owner, which therefore has to be kept involved in the query execution process. In this paper, we propose different *multi-key data encryption* strategies for enforcing access privileges. Our strategies exploit different keys, which are distributed to the users, corresponding to the different authorizations. We then present some experiments evaluating the quality of the proposed strategies with respect to the amount of cryptographic information to be produced and maintained.

1 Introduction

Data outsourcing has become increasingly popular in recent years. Its intended purpose is enabling data owners to outsource distribution of data on the open Net to *service providers* following a "database-as-a-service" paradigm. Data outsourcing promises higher availability and more effective disaster protection than in-house operations. However, since data owners physically release their information to service providers, data confidentiality and even integrity may be put at risk. Methods that protect outsourced data from unauthorized accesses are therefore needed, and data encryption techniques together with indexes associated with the data have been often used for this purpose [4, 7, 9, 10]. These techniques guarantee data confidentiality, even from (honest but curious) service providers, enabling providers to execute queries without accessing cleartext data.

Although different security aspects of the outsourced scenario have been addressed (e.g., integrity [11], inference exposure [6], physical security measures [5]), most current solutions still consider the whole database as encrypted with a single key known only to the data owner, which therefore has to be

kept involved in the query execution process. This is indeed a severe limitation, since a desirable feature of database outsourcing is a full delegation of query management to the hosting environment. In this paper, we put forward the idea of using a multi-key solution for enforcing different access privileges for different users without necessarily involving the data owner in the query processing. To this purpose, we propose to use different *multi-key data encryption* strategies that can be used for implementing access control. We then evaluate the quality of these strategies in terms of the amount of cryptographic information that needs to be stored and managed. We illustrate some experimental results, which clearly demonstrate that the use of the techniques described in the paper offers significant savings in the amount of access control information to be maintained, with a considerable increase in overall system efficiency.

The remainder of this paper is organized as follows. Section 2 describes the different approaches that can be applied to enforce selective access in an outsourced scenario. Section 3 describes the experimental setup. Section 4 presents our experiments and discusses the quality of the different approaches in terms of the amount of (public) information that needs to be managed by the system to enforce the access control policies. Section 5 concludes the paper.

2 Access Control in the Outsourced Scenario

Given a system with a set $\mathcal{U}$ of users and a set $\mathcal{T}$ of resources, the policies regulating the accesses of users on resources can be modeled via the traditional *access matrix* $\mathcal{A}$ with $|\mathcal{U}|$ rows, one for each user, and $|\mathcal{T}|$ columns, one for each resource.

For simplicity and to strengthen the relationship with previous proposals on the "database-as-a-service" paradigm, in this paper we consider tuples as resources, allowing row level access control enforcement. Note, however, that the technique presented in this paper is applicable to many scenarios, since it can support authorization at different granularity (e.g, table, column, row, or cell) and it can be used to manage access to resources stored outside the DBMS (e.g., a file service hosted by a third party).

Each entry $\mathcal{A}[\mathtt{u},\mathtt{t}]$ in the matrix contains the list of actions that user `u` is authorized to execute over resource `t`. Since we take into consideration `read` actions only, each entry in the access matrix can simply assume two values: $\mathcal{A}[\mathtt{u},\mathtt{t}]$=1 if `u` can read `t`; 0 otherwise. Figure 1 reports an example of access matrix for a system with 7 tuples ($\mathtt{t}_1 \ldots \mathtt{t}_7$) and 5 users (**A**, **B**, **C**, **D**, and **E**). Given an access matrix $\mathcal{A}$, $acl_{\mathtt{t}}$ denotes the access control list for tuple `t`, that is, the set of users that can read `t`; $cap_{\mathtt{u}}$ denotes the capability list of user `u`, that is, the set of resources `u` can read. For instance, with reference to Fig. 1, $acl_{\mathtt{t}_1}$=$\{$**A**,**B**,**C**$\}$ and $cap_{\mathbf{A}}$=$\{\mathtt{t}_1,\mathtt{t}_2,\mathtt{t}_3,\mathtt{t}_5,\mathtt{t}_7\}$.

In the above scenario, the enforcement of access control policies cannot be delegated to the remote server, which is not trusted for accessing neither database content nor the access control policies themselves. To tackle this issue,

	t_1	t_2	t_3	t_4	t_5	t_6	t_7
A	1	1	1	0	1	0	1
B	1	1	1	0	1	1	1
C	1	0	0	1	1	1	1
D	0	1	0	1	1	1	0
E	0	0	1	1	0	1	0

Fig. 1. An example of access matrix

we propose to enforce access control by using *multi-key* techniques [3, 13–15]. Basically, these techniques consist in using different keys for encrypting data and in giving to each user a *key set* such that she can access all and only the resources she is authorized to access.

A straightforward solution for adopting these approaches consists in encrypting each tuple $\mathtt{t}$ with a different key and communicating to each user $\mathtt{u}$ the set of keys used to encrypt the tuples in $cap_{\mathtt{u}}$. This solution correctly implements the policy represented by access matrix $\mathcal{A}$, but it is expensive to manage, due to the high number of keys that users are required to manage. We therefore combine a multi-key technique together with *key derivation methods* [1,2,8,12]. These methods operate on a hierarchy where each of its elements is associated with a key; the keys of lower-level elements can be computed based on the keys of their predecessors and on information publicly available. In our context, these methods permit to reduce the number of keys that need to be directly communicated to each user. Among the different key derivation methods proposed in the literature, Atallah's method [2] is well adapted to our context. This method is based on the concept of *token*, which is defined as follows. Given two keys, $\mathtt{k_i}$ and $\mathtt{k_j}$, and a public label $\mathtt{l_j}$ associated with $\mathtt{k_j}$, a token from $\mathtt{k_i}$ to $\mathtt{k_j}$ is defined as $T_{i,j}=\mathtt{k_j}\oplus\mathtt{h}(\mathtt{k_i},\mathtt{l_j})$, where $\oplus$ is the n-ary *xor* operator and $\mathtt{h}$ is a secure hash function. Given $T_{i,j}$, any user from the knowledge of $\mathtt{k_i}$ and with access to public label $\mathtt{l_j}$ can compute (derive) $\mathtt{k_j}$. All tokens $T_{i,j}$ in the system are stored in a *public catalog*. The combination of a multi-key technique with this key derivation method can be performed according to different strategies, which we generically call *multi-key data encryption* strategies. Implementing these strategies involves several technicalities; for the sake of clarity, in the remainder of this section we shall outline the algorithms we have designed at the level of detail needed to carry out a comparison between the approaches.

The first and simplest strategy assigns a label and a key to each tuple $\mathtt{t}\in\mathcal{T}$ and a key to each user $\mathtt{u}\in\mathcal{U}$. For each entry $\mathcal{A}[\mathtt{u},\mathtt{t}]$ such that $\mathcal{A}[\mathtt{u},\mathtt{t}]=1$, token $T_{\mathtt{u},\mathtt{t}}$ is computed and stored in the public catalog. This strategy drastically reduces the number of keys that each user has in her key set (each user has exactly one key), but introduces a huge public catalog of tokens. For instance, with respect to the access matrix in Fig. 1, the public catalog contains 23 tokens because the access matrix contains 23 entries equal to 1. Every time user $\mathtt{u_i}$ has to access tuple $\mathtt{t_j}$, $\mathtt{u_i}$ retrieves $\mathtt{t_j}$, $\mathtt{l_j}$, and $T_{i,j}$ from the public catalog. Tuple $\mathtt{t_j}$ can then be decrypted using the key obtained computing $T_{i,j}\oplus\mathtt{h}(\mathtt{k_i},\mathtt{l_j})$, where

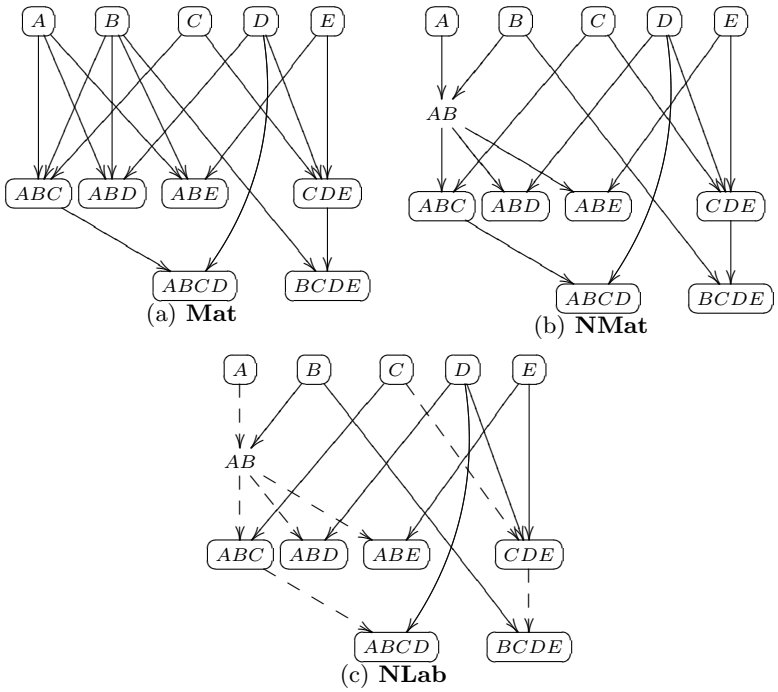


Fig. 2. Examples of UH hierarchies based on the access matrix in Fig. 1

$\mathtt{k_i}$ is the key assigned to user $\mathtt{u_i}$. Due to the high number of tokens, we do not consider further this strategy. Instead, we propose to use other approaches described in the remainder of this section.

AM approach. The first approach we propose improves on the direct representation of the access matrix by encrypting with the same key all the tuples having the same *acl* (e.g., tuples $\mathtt{t_1}$ and $\mathtt{t_7}$ in Fig. 1). The reduction in the number of tokens provided by this approach depends on the number of resources that share the same access profile. In the example, the use of this strategy reduces the number of tokens to 20, because the same tokens can be used to access $\mathtt{t_1}$ and $\mathtt{t_7}$. To access a tuple, user $\mathtt{u_i}$ will have to retrieve the encrypted tuple, its label $\mathtt{l_j}$ (which characterizes all the tuples with the same *acl*) and token $T_{i,j}$.

To further reduce the number of tokens in the public catalog, we propose to apply the Atallah's approach to a *user key derivation hierarchy* UH, where the elements correspond to sets of users in the system (i.e., *acls* that can be defined on $\mathcal{U}$) and the partial order is naturally induced on it by the subset containment relationship. Each element in the hierarchy has its own key, while each arc has its token in the public catalog. Each tuple is then encrypted with the key associated with the element representing its *acl*, and each user is associated with an element (and therefore a key) representing herself in the hierarchy.

Since the number of tokens in the public catalog depends on the number of arcs in the hierarchy, we need to carefully select the elements and the arcs in the hierarchy. To this purpose, we develop an algorithm that takes an access matrix as input and returns a user key derivation hierarchy as output. This hierarchy can be computed in different ways.

Mat approach. A first hierarchy-based approach consists in selecting a set M of elements, called *material*, that contains the elements representing single users and the elements corresponding to *acls* in $\mathcal{A}$. The algorithm then connects the material elements using a set of arcs (i.e., tokens) with no redundant arcs (e.g., if we consider node ABCD in Fig. 2(a), we observe that among its 6 potential ancestors, only ABC and D have an outgoing arc reaching it). Figure 2(a) represents the UH obtained by applying the **Mat** approach to the access matrix in Fig. 1. In this case the public catalog contains 16 tokens, one token for each arc in the hierarchy. As an example of key derivation, suppose that user **A** needs to read tuple $\mathtt{t_5}$, which is encrypted with key $\mathtt{k_{ABCD}}$ ($acl_{\mathtt{t_5}}$={**A**,**B**,**C**,**D**}). User **A** can use her key $\mathtt{k_A}$ and token $T_{A,ABC}$ to derive key $\mathtt{k_{ABC}}$, which in turn can be used together with token $T_{ABC,ABCD}$ to derive key $\mathtt{k_{ABCD}}$.

NMat approach. Another hierarchy-based approach consists in using other elements in addition to the material ones. To this purpose, our algorithm selects a set NM of elements, called *non material*, that are useful to reduce the number of tokens. Intuitively, a non material element can reduce the total number of tokens if it allows to have different paths in the hierarchy with a common node. Figure 2(b) represents the UH obtained by applying the **NMat** approach to the access matrix in Fig. 1. In this case, the public catalog contains 15 tokens.

NLab approach. A third hierarchy-based approach consists in allowing the presence of arcs in UH without a corresponding token. In this case, the hierarchy obtained by applying the **NMat** approach is modified by assigning a randomly chosen key only to *root elements*, that is, elements without incoming arcs. For each non root element, the algorithm chooses an incoming arc and computes the key of the element through a traditional key derivation method operating on trees [12]. For instance, given the arc ($\mathtt{v_i}$,$\mathtt{v_j}$) for element $\mathtt{v_j}$, key $\mathtt{k_j}$ is computed by applying a predefined hash function to $\mathtt{k_i}$. The remaining arcs are associated with tokens. Figure 2(c) illustrates the same hierarchy obtained with the **NMat** approach, where the arcs without a token are represented with a dashed line. In this case, the public catalog contains 8 tokens.

3 Experimental Setup

We perform some experiments aimed at assessing the quality of the different approaches in terms of the number of tokens that need to be managed. We

Category of Users	Notation	Cardinality
Team Managers	$TM_1 \dots TM_t$	t
Players	$P_1 \dots P_p$	$p = t \cdot pt$
Writers	$W_1 \dots W_w$	$w = \lceil t/tw \rceil$
Managers of writers	$WM_1 \dots WM_m$	$m = \lceil w/wm \rceil$
Subscribers	$S_1 \dots S_s$	s

pt: number of players of each team
tw: number of teams assigned to a writer
wm: number of writers in a group

Fig. 3. Categories of users in the system

consider a sport news database, with t *teams* of pt *players* each and s *subscribers* (i.e., team supporters). The league is also followed by a number of *writers*, each working with tw teams. The writers are grouped into sets of wm elements and one *manager* is assigned to each set. The set of users $\mathcal{U}$ is partitioned into five categories summarized in Fig. 3.

Analogously, the set of resources $\mathcal{T}$ in the database is partitioned into two subsets: *player news* $PN_1 \dots PN_p$ (tuples describing players); and *team news* $TN_1 \dots TN_t$ (tuples describing teams).
We then define two classes of authorizations. The first set contains authorizations assigned to users on the basis of the tuples that they need to access for playing their role (e.g., each team manager needs to access the team news for the teams she follows; we omit the formal description of this authorization set).

The second set of authorizations contains access rights assigned on the basis of subscribers' requests (e.g., each subscriber can choose the teams and players she wants to follow). In particular, we define two different configurations to better evaluate the scalability of the different approaches. The first configuration, denoted C_1, is characterized by a great variability in the authorization set because each subject can subscribe to whatever resource she wants to. The second configuration, denoted C_2, is instead more static because each subject cannot choose to subscribe to a single resource, but can only be associated with a predefined set of access rights, defined by the data owner. The second configuration is more similar to a real-life application, where it is required to manage subscriptions to news. The first scenario has been designed with the goal to be difficult to manage by the approaches we have designed, in order to put them to a significant test.

3.1 Configuration of Scenario C_1

In scenario C_1 the set of authorizations associated with subscribers is completely random, as well as some authorizations associated with team managers. These authorizations are formally defined as follows.

– $\mathcal{A}[TM_{2i}, TN_j] = 1, i = 1 \dots \lceil t/2 \rceil$, $j = 1 \dots t$: half team managers can access all the team news of the league.

- $\mathcal{A}[TM_{2i-1}, TN_j] = 1$, $i = 1 \dots \lceil t/2 \rceil$, for each value of i, j is randomly chosen in $\{1, \dots, t\}$: half team managers (the ones that cannot access all the team news) can access the team news of another team of the league.
- $\mathcal{A}[TM_{2i-1}, PN_l] = 1$, $i = 1 \dots \lceil t/2 \rceil$, $l = ((j-1) \cdot pt + 1) \dots (j \cdot pt)$: these team managers have also access to the player news of the same team, that is, for each i, the corresponding j is the one chosen for the previous item.
- Each subscriber S_i in the system can access $f(i)$ team news, and player news of the players of the same team. Function f is a *Zipf* distribution that increases with the number s of subscribers. In our experiments, the first $s/3$ subscribers can view just a team news, $2s/9$ subscribers can access two team news, and so on. For each subscriber S_i, once computed $f(i)$, we randomly choose the team news that she can access. It is important to note here that we avoid to assign the same subscriber twice to the same team.
- Each subscriber S_i in the system can access also $f(i) \cdot pt$ player news, randomly chosen from the set of player news in the system that she cannot access.

3.2 Configuration of Scenario $\mathsf{C_2}$

Scenario C_2 is characterized by pre-defined sets of authorizations to which team managers and subscribers can subscribe. We define two sets for team managers, and two sets for subscribers. The two sets for team managers contain a team news and all the player news associated with the players of the considered team. The first set for subscribers contains the news of two teams together with their player news and twelve other player news; the second set contains three team news together with their player news and twelve other player news. These authorizations are formally defined as follows.

- $\mathcal{A}[TM_{2i}, TN_j] = 1, i = 1 \dots \lceil t/2 \rceil$, $j = 1 \dots t$: half team managers can access all the team news of the league.
- Half team managers (the ones that cannot access all team news) can subscribe to one of the two sets defined for them.
- Each subscriber S_i in the system can subscribe to one of the two sets defined for them.

Although in C_1 and C_2 authorizations are differently distributed among users and resources, their total number is nearly the same.

4 Experimental Results

The main goal of this set of experiments is that of analyzing the behavior of the different approaches for creating a key derivation hierarchy when the size of the system grows, that is, when both $\mathcal{T}$ and $\mathcal{U}$ increase. We ran experiments by varying the number t of teams from 5 to 50 and by considering the following different cases: 0, 10, 20, and 30 subscribers (s), $pt = 5$ players per team, $tw = 5$ teams followed by each writer, and $wm = 5$ writers followed by each manager.

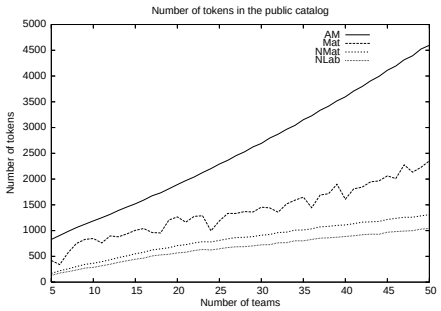
Note that we focus our investigation more on the increase in the number of team rather than on the number of subscribers, because in this way we are able to increase at the same time the number of users and the number of resources. With 50 teams, the model creates more than 300 distinct users, in the roles of team players, team managers, and writers.

For each combination of values for t and s, we evaluate: the number of tokens in the public catalog; and the number of elements in the hierarchy, distinguishing between material and non material. We decide to measure these two parameters since they have a great impact in the access control enforcement: a huge catalog causes great key derivation costs. The number of material and non material elements in the hierarchy is also important to evaluate the quality of UH because its structure determines the number of tokens in the public catalog. The same experimental setup has been adopted in configuration C_1 and configuration C_2, where the approaches described in Sect. 2 (i.e., **AM**, **Mat**, **NMat**, and **NLab**) have been evaluated.

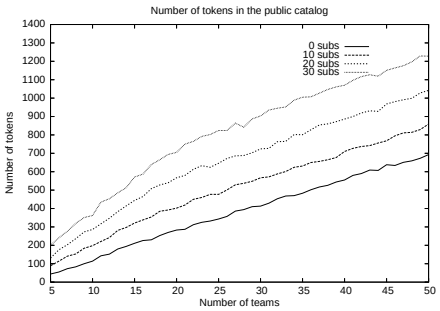
4.1 Number of Tokens in the Public Catalog

Figure 4(a) illustrates the number of tokens in the public catalog according to the four approaches **AM**, **Mat**, **NMat**, and **NLab** in configuration C_1, as the number of teams varies and the number of subscribers s is equal to 20. As the graph shows, there is a substantial gap between **AM** curve and the other curves. The curves **NMat** and **NLab** are very close and the gap between them is relatively constant, because in these two cases UH contains the same set of elements; the only difference is that **NLab** suppresses some tokens. By contrast, **Mat** is based on a different set of elements, which contains only material elements, and therefore UH is different from the hierarchy obtained by applying **NMat** and **NLab**.

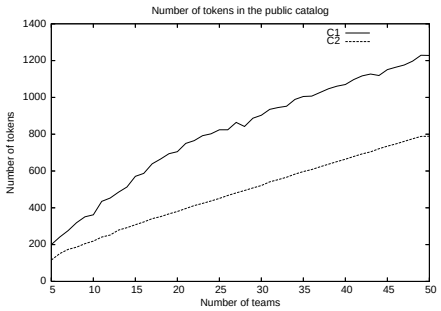
Note that in Fig. 4(a) we consider only the case where the number of subscribers is equal to 20, because the graphs associated with all the other configurations (i.e., with a number of subscribers equal to 0, 10, and 30, respectively) exhibit an almost identical behavior. As a proof of this statement, consider, for example, the **NLab** approach: Fig. 4(b) reports the number of tokens in the public catalog according to the four different values of s in C_1, as the number of teams varies. Here, the curves have exactly the same trend and, as expected, the number of tokens increases with the number of teams in the system. We observe that **Mat**, **NMat**, and **NLab** scale well with the system size and, as expected, the best solution is obtained with the **NLab** approach. These experiments have also been performed with configuration C_2 and they confirm the considerations above-mentioned. In particular, by comparing the number of tokens in C_1 and C_2 we observe that, as expected, configuration C_2 is significantly more frugal than configuration C_1. Figure 4(c) compares the number of tokens in C_1 and C_2 when the **NLab** approach is applied and the number of subscribers is equal to 30. As the graph shows, the number of tokens, as well as the gap between the two curves, increase with the number of teams. This is because in C_2 au-



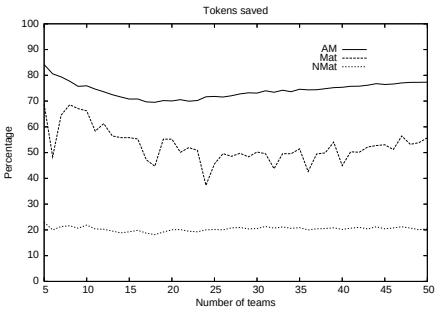
(a) C_1 $s = 20$



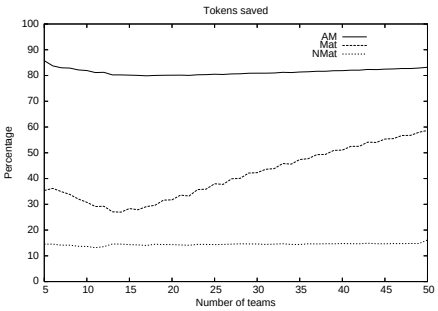
(b) C_1 **NLab**



(c) C_1 vs C_2 (**NLab**,$s = 30$)



(d) C_1(**NLab**,$s = 20$)

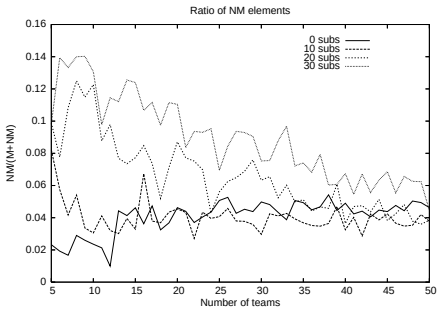


(e) C_2(**NLab**,$s = 20$)

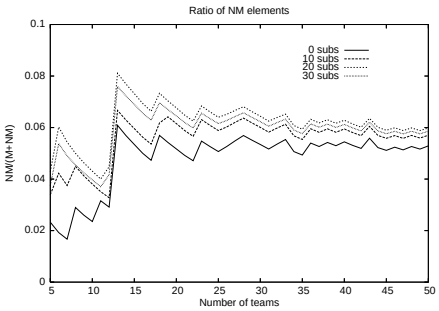
Fig. 4. Number of tokens in the public catalog (a)-(c) and percentage of tokens saved with **NLab** (d)-(e)

thorizations are less variable than in C_1 and therefore it is easier to determine a good solution in terms of the hierarchy that the algorithm is able to create.

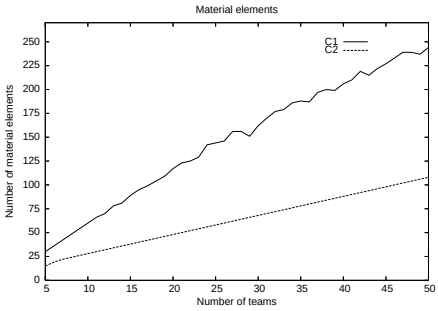
The advantage of approach **NLab** compared with the other three approaches **AM**, **Mat**, and **NMat** is much more visible in Fig. 4(d) and Fig. 4(e), which report the percentage of tokens saved with the **NLab** approach in C_1 and C_2, respectively, and when the number of subscribers is equal to 20. As the graphs



(a) C_1



(b) C_2



(c) C_1 vs C_2 ($s = 20$)

Fig. 5. Ratio of the number of non material elements in the system (a)-(b) and number of non material elements as the number of teams varies (c)

show, this percentage has a slow but growing trend with respect to the **AM** approach while it is quite constant with respect to the **NMat** approach (around 20% in C_1 and 15% in C_2). With respect to the **Mat** approach, the percentage is variable in C_1 and has a well defined increasing trend in C_2. The main reason for this is that the hierarchy in the **NMat** and **NLab** approach is exactly the same, while it may be very different in the **Mat** approach, depending on how authorizations are distributed in the system.

4.2 Number of Material and Non Material Elements in UH

Another important aspect that should be considered in evaluating and comparing the different approaches proposed is the number of elements in the final hierarchy. In particular, it is relevant the ratio between non material and material elements in the structure to better understand whether non material elements are useful to reduce the number of tokens. Figure 5(a) and Fig. 5(b) show the ratio between the number of non material elements and the total number of elements of the hierarchy obtained for C_1 and C_2, respectively, as the number

of teams increases. Note that this measure is available when the hierarchy is built by applying the **NMat** or **NLab** approaches only (**AM** and **Mat** do not consider non material elements), which are characterized by the same hierarchy. By comparing the graphs in Fig. 5(a) and Fig. 5(b), we can immediately note that they are very different. Configuration C_1 presents a great variability with respect to the considered measure. More precisely, when the number of subscribers is equal to 0 or 10, the ratio increases with the number of teams; it decreases when the number of subscribers is equal to 20 or 30. This behavior is mainly due to the fact that material elements vary on the basis of the authorizations initially defined. By contrast, in configuration C_2 the considered measure follows a trend that is similar for the different values of s and this ratio decreases with the number of teams. As it is also visible from these graphs, both C_1 and C_2 present a ratio that tends to converge to a value between 4% and 6%, and the gaps among the different curves in each configuration decrease with the number of teams.

Figure 5(c) illustrates the number of material elements in C_1 and C_2, as the number of teams varies and the number of subscribers s is equal to 20 (note that the graphs that we can obtain for the other possible values of s have basically the same trend). As the graph shows, the number of material elements increases with the number of teams and the curve increases more rapidly in C_1. This is because the number of material elements in a hierarchy depends on the different *acl* values. Therefore, if the resources to be protected are characterized by similar access profiles, the corresponding *acls* will be the same and the number of material elements will be low. Since in C_1 authorizations are randomly chosen, it is more likely to have a lot of different *acl* values and consequently a lot of material elements. By contrast, in C_2 authorizations follow pre-defined patterns, and the number of different *acl* values is lower. In addition, in C_2 the number of material elements follows the same trend for all possible values of s; it varies in C_1.

5 Conclusions

In this paper, we presented different strategies for enforcing selective access in an outsourced database scenario. We then performed some experiments to evaluate their quality with respect to the amount of cryptographic information to be produced and maintained. The results of the experiments demonstrate the significant savings produced by the use of **NLab** compared to **AM** approach. We designed two experimental scenarios that, even if based on the same data, present significantly distinct access profiles; the fact that good results have been obtained in the two scenarios, and specifically in scenario C_1, is a strong indication that the savings produced by the **NLab** approach can be achieved in most applications.

Acknowledgements

This work was supported in part by the European Union under contract IST-2002-507591, by the Italian Ministry of Research Fund for Basic Research (FIRB) under project "RBNE05FKZ2", and by the Italian MIUR under project 2006099978. The work of Sushil Jajodia was partially supported by the National Science Foundation under grants CT-0627493, IIS-0242237, and IIS-0430402.

References

1. S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer System*, **1**(3), 239–248 (August 1983).
2. M.J. Atallah, K.B. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *Proc. of the ACM CCS*, Alexandria, VA, USA (November 2005).
3. J.C. Birget, X. Zou, G. Noubir, and B. Ramamurthy. Hierarchy-based access control in distributed environments. In *Proc. of the IEEE International Conference on Communications*, Helsinki, Finland (June 2002).
4. D. Boneh, G.D. Crescenzo, R. Ostrovsky, and G. Persiano. Public-key encryption with keyword search. In *Proc. Eurocrypt 2004*, Interlaken, Switzerland (May 2004).
5. L. Bouganim and P. Pucheral. Chip-secured data access: confidential data on untrusted servers. In *Proc. of the 28th VLDB Conference*, Hong Kong, China (August 2002).
6. A. Ceselli, E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Modeling and assessing inference exposure in encrypted databases. *ACM TISSEC*, **8**(1), 119–152 (February 2005).
7. E. Damiani, S. De Capitani di Vimercati, S. Jajodia, S. Paraboschi, and P. Samarati. Balancing confidentiality and efficiency in untrusted relational DBMSs. In *Proc. of the ACM CCS*, Washington, DC, USA (October 2003).
8. E. Gudes. The design of a cryptography based secure file system. *IEEE Transactions on Software Engineering*, **6**(5), 411–420 (September 1980).
9. H. Hacigümüs, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proc. of the ICDE*, San Jose, CA, USA (February 2002).
10. H. Hacigümüs, B. Iyer, S. Mehrotra, and C. Li. Executing SQL over encrypted data in the database-service-provider model. In *Proc. of the ACM SIGMOD*, Madison, Wisconsin, USA (June 2002).
11. E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced database. In *Proc. of the 11th Annual Network and Distributed System Security Symposium*, San Diego, California, USA (February 2004).
12. R.S. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, **27**(2), 95–98 (April 1988).
13. Y. Sun and K.J.R. Liu. Scalable hierarchical access control in secure group communications. In *Proc. of the IEEE Infocom*, Hong Kong, China (March 2004).
14. H. Tsai and C. Chang. A cryptographic implementation for dynamic access control in a user hierarchy. *Computer and Security*, **14**(2), 159–166 (September 1995).
15. C.K. Wong, M. Gouda, and S.S. Lam. Secure group communications using key graphs. In *Proc. of the ACM SIGCOMM*, Vancouver, British Columbia (September 1998).