A Data Outsourcing Architecture Combining Cryptography and Access Control

Sabrina De Capitani di Vimercati
DTI - Università di Milano
26013 Crema - Italy
decapita@dti.unimi.it

Sara Foresti
DTI - Università di Milano
26013 Crema - Italy
foresti@dti.unimi.it

Sushil Jajodia
CSIS - George Mason University
Fairfax, VA 22030-4444
jajodia@gmu.edu

Stefano Paraboschi
DIIMM - Università di Bergamo
24044 Dalmine - Italy
parabosc@unibg.it

Pierangela Samarati
DTI - Università di Milano
26013 Crema - Italy
samarati@dti.unimi.it

ABSTRACT

Data outsourcing is becoming today a successful solution that allows users and organizations to exploit external servers for the distribution of resources. Some of the most challenging issues in such a scenario are the enforcement of authorization policies and the support of policy updates. Since a common approach for protecting the outsourced data consists in encrypting the data themselves, a promising approach for solving these issues is based on the combination of access control with cryptography. This idea is in itself not new, but the problem of applying it in an outsourced architecture introduces several challenges.

In this paper, we first illustrate the basic principles on which an architecture for combining access control and cryptography can be built. We then illustrate an approach for enforcing authorization policies and supporting dynamic authorizations, allowing policy changes and data updates at a limited cost in terms of bandwidth and computational power.

Categories and Subject Descriptors

H.2.7 [**Database Management**]: Database Administration—*Security, integrity, and protection*; D.4.6 [**Operating Systems**]: Security and Protection—*Access control*

General Terms

Design, Security

Keywords

Outsourced architecture, access control, cryptography

1. INTRODUCTION

With the recent adoption and diffusion of the data outsourcing paradigm, where data owners store their data on external servers, there have been increasing general demands and concerns for data confidentiality. Besides well-known risks of confidentiality and privacy breaks, threats to outsourced data include improper use of information: the server could use substantial parts of a collection of data gathered and organized by the data owner, potentially harming the data owner's market for any product or service that incorporates that collection of information. Traditional access control architectures assign a crucial role to the *reference monitor* [3] for ensuring data confidentiality. The reference monitor is the system component responsible of the validation of access requests. The data outsourcing scenario however challenges one of the basic tenets of traditional access control architectures, where a trusted server is in charge of defining and enforcing access control policies. This assumption no longer holds here, because the server does not even have to know the access control policy that is defined (and can, if necessary, be modified) by the data owner. We therefore need to rethink the notion of access control in open environments, where external servers take full charge of the management of the outsourced data and are not trusted with respect to the data confidentiality.

An important opportunity for a revision for the access control architecture can be based on the use of cryptography [19]. Cryptography can be considered as a tool that transforms information in a way that its protection (confidentiality and integrity) depends only on the correct management of a compact secret (the key). Cryptography is typically used when information is transmitted on a channel, with the assumption that the channel lies outside of the trust boundary of the system. However, the improvements in cryptographic algorithms, extremely reduces the cost for the use of cryptography for stored data, producing a continuous increase in its adoption. A simple application of cryptography to stored resources can then be based on the well-known correspondence between a network and storage service: both organize the information they have to transfer/store in discrete pieces (packets in networks, blocks in storage devices). A more advanced solution takes into account that the nature of the storage service is different. For instance, in [18] the authors exploit cryptography to the aim of protecting the sensitive information plaintext represented in memory pages when a trusted process accesses it.

Indeed, the application of cryptography for the protec-

CSAW'07, November 2, 2007, Fairfax, Virginia, USA.

tion of files is today available as an option in most modern operating systems (e.g., in Microsoft Windows since Windows 2000), to make it impossible to access the information without access to the keys stored within the system. Encryption reduces the risk of loss of confidential information deriving from low-level access to the devices. The cryptographic protection can also be used to protect the swap area on disk, to reduce the risk that processes could access information they are not authorized to see by reading the content of swap pages released by processes managing confidential information. Nonetheless, the cryptography options offered by current operating systems have been designed to protect local resources and access control is still realized using the services of a reference monitor.

In this paper, we propose a novel access control model and architecture that eliminates the need for a reference monitor and relies on cryptography to ensure confidentiality of data stored on a server. Data are encrypted as the data owner stores them on an external server. Authorizations and encryption are merged thus allowing access control enforcement to be outsourced together with the data. The great advantage is that the data owner, while specifying the policy, needs not be involved in its enforcement. The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 presents a novel cryptography-based architecture. Section 4 describes a solution for representing authorizations and dealing with dynamic policy updates. Finally, Section 5 presents our conclusions.

2. RELATED WORK

The idea of combining access control and cryptography is in itself not new and has been proposed in different contexts. For instance, it was proposed in the context of: distributed file system [15], where a traditional client/server architecture is used; access control in a hierarchy [2, 8], where a centralized architecture is adopted; securing XML documents [17], where a "data publishing" architecture is considered and where the data owner totally loses the control on her data. However, the combination of access control and cryptography in an outsourced architecture introduces several challenges that have not been investigated in previous proposals, which are more focused on traditional architectures where the data owner and the server storing the data are in the same trusted domain.

Previous work most directly related to ours is in the database outsourcing area [12, 14]. In such a context, most proposals focused on query execution and propose to store additional indexing information together with the encrypted database. Such indexes can be used by the DBMS to select the data to be returned in response to a query [1, 6, 13]. These proposals did not investigate the data encryption issues, and assumed all users to have complete access to the whole database [5]. Adding a traditional authorization layer to the outsourcing scenario requires that when a client poses a query, both the query and its result have to be filtered by the data owner, who is in charge of enforcing the access control policy. A first attempt to solve this issue has been presented in [10], where different keys are used for encrypting different data. To our knowledge, however, there is not a solution that both allows access control enforcement to be outsourced with the data and that supports policy changes without requiring the owner to have a low-latency channel to the server, which is not realistic in many scenarios.

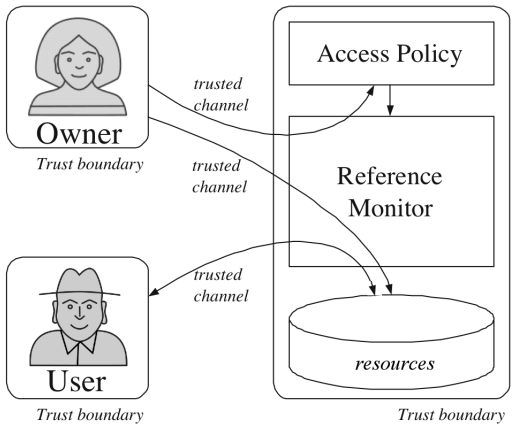


Figure 1: Reference monitor architecture

3. ACCESS CONTROL ARCHITECTURES

We first describe the traditional access control architectures and then illustrate a novel cryptography-based architecture suitable for the data outsourcing scenario.

3.1 Classical access control architectures

The reference monitor typically represents the core component of any access control system. To the aim of granting security, the reference monitor must fulfil the following three requirements [3], which are difficult to enforce, and impose strong constraints on the way sensitive information can be managed.

- *The reference monitor must be tamper-proof.* System policies have to be carefully protected and the system on which the reference monitor is executed has to be robust.
- *The reference monitor must be involved in no access request.* Resources have to lie (logically and physically) near to the reference monitor. As a consequence, the reference monitor has a crucial impact on the system performances, since any access to a resource has to be mediated by its services. Indirectly, this imposes constraints on the flexibility and expressiveness of the access policy. As a matter of fact, sophisticated policy languages may delay the verification and produce an unacceptable impact on system performances.
- *The reference monitor must be small enough to be tested.* The experience in the construction of secure systems shows a clear benefit in the minimization of the size of the trusted components. In fact, this reduces the chances of errors in system design and implementation, which could be exploited by malicious users to subvert resource protection. The impact is that, in typical systems, it becomes difficult to manage data in a distributed setting. Resources are then centralized near the location where the reference monitor can operate.

Figure 1 illustrates the typical architecture of a system using a reference monitor. The system is characterized by two kinds of users: the *data owner*, who takes advantage of the system to store her resources; and the *authorized reader*

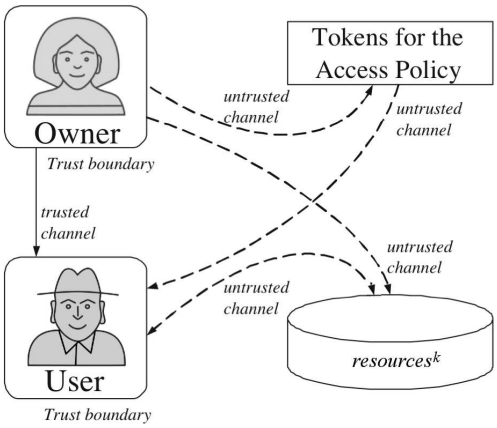


Figure 2: Data outsourcing access control architecture

(user), who accesses the system to retrieve such resources. Both users have to route their access requests through the reference monitor, which is responsible for the correct enforcement of the access policy defined by the owner. Here, the data owner and the system storing the data usually belong to the same trust domain. This architecture is therefore not suitable to operate in a data outsourcing scenario, where the data owner and the system storing the data are in two different trust domains.

3.2 A novel cryptography-based architecture

To partially overcome the issues above, cryptography-based access control architectures have been introduced (see Section 2). These architectures for the realization of cryptographic access protection are simply based on the use of one distinct key for the encryption of every resource and the definition of a protocol for the dialogue between the owner and each of the users, such that authorized users can receive, on a trusted channel, the required encryption key. This solution presents the crucial shortcoming that it requires the data owner to be always involved in the processing of every access request.

However, the full realization of the cryptography-based access control architecture for secure data dissemination has to correctly manage different important requirements before it could be deployed in large scale applications with large user communities. First, different users may need to see different portions of the data. To give clients different access rights, the data owner cannot then use a single key for encrypting the whole data. Instead, the data owner should be able to define access authorizations and to map them on the data through a *multi-key encryption* approach [9]. Users use then multiple keys to extract from the query results the plaintext data they are entitled to see. Although the idea of using different encryption keys for different data items is not new [17], its application in an outsourced scenario poses several challenges that need to be carefully considered. Second, it is necessary to analyze the issue of realizing an efficient management of policy updates, presenting a mechanism that should be able to offer a robust and efficient solution.

Recent research has produced important results that promise to give a significative contribution to the realization of a more flexible and efficient cryptography-based access control architecture [4, 11]. Figure 2 illustrates a novel architecture that makes use of two distinct repositories: one repository (the one on the bottom) has to store the encrypted representation of resources, the other repository is instead dedicated to the storage and management of the access policy, expressed in a way that can be managed by a public server as an open resource, with adequate protections that guarantee that a malicious user cannot gain unauthorized access to resources (i.e., to encryption keys) [4]. As we will discuss in Section 4, the basic idea is that each user is assigned a key by the owner and then a set of *tokens* is used to allow the derivation from one key to another key. The token catalog is demonstrated to be usable only by users already having access to the secret. As the figure shows, the owner first sends to a user a secret key that characterizes her. The data owner is then responsible for the creation of both the resources and the token catalog. The token catalog has to be a correct representation of the access policy the owner wants to enforce on the resources. The authorized user has to retrieve from the token catalog all the tokens that are needed to compute the resource key from her personal key, to correctly access the resource. Even if it would be possible to directly communicate each user the set of keys used to encrypt the resources she can access, such a solution would be more expensive from the data owner's point of view. As a matter of fact, in such a case the owner wold be involved in any policy update, for key distribution to users.

The advantages of the use of cryptography for stored resources increases with resource size, distribution requirements, and read-access frequency.

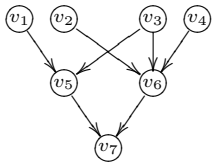
- *Large resources* can be protected more efficiently using cryptography because the cost for the verification of the access control request and the retrieval of the key can be shared among a larger number of accesses. The absence of a reference monitor removes a potential bottleneck. We note that in older computer systems a large resource would have been considered inadequate for cryptographic protection, due to its high computational cost, but in modern and future computer platforms it is reasonable to assume that clients have the required computational power.

- *Distributed resources* benefit from the use of cryptography for access protection because it permits to easily transfer resources to the network locations that are near to the users, without the need of strong trust requirements on the nodes that are used to store the encrypted resource representation. In traditional distributed systems, the enlargement of the trust domain to cover a large and often heterogeneous environment has always required a considerable investment and introduced many significant vulnerabilities.

- *Read-mostly resources* are resources created by the owner once and then read by all the users who are not the owner, with rare or null modifications by the owner in the resource life. These resources are particularly interesting for the use of cryptographic protection, because multiple copies of the resources can be easily generated and replicated within the system. This leads to considerable benefits to read performances and scalability to allow access by large user communities. In general, this architecture provides an extremely interesting solution for the dissemination of resources.

	r_1	r_2	r_3	r_4	r_5	r_6
A	1	1	1	1	0	1
B	0	0	0	0	1	1
C	0	1	1	1	1	1
D	0	0	0	0	1	1

Figure 3: An example of access matrix



(a)

source	dest	value
$v_1.k$	$v_5.k$	$v_5.k \oplus h(v_1.k, l_5)$
$v_2.k$	$v_6.k$	$v_6.k \oplus h(v_2.k, l_6)$
$v_3.k$	$v_5.k$	$v_5.k \oplus h(v_3.k, l_5)$
$v_3.k$	$v_6.k$	$v_6.k \oplus h(v_3.k, l_6)$
$v_4.k$	$v_6.k$	$v_6.k \oplus h(v_4.k, l_6)$
$v_5.k$	$v_7.k$	$v_7.k \oplus h(v_5.k, l_7)$
$v_6.k$	$v_7.k$	$v_7.k \oplus h(v_6.k, l_7)$

(b)

Figure 4: An example of key derivation

The observations above justify the claim that the data outsourcing architecture is particularly interesting for the realization of an application for large scale dissemination of large resources (e.g., personally created multimedia resources), which are becoming increasingly common. This is an application that is becoming more and more important, as testified by the success of services like YouTube, Flickr, Box.net, Amazon Storage Services, and many others.

In terms of security, the advantage of this architecture derives from the reduction in the size of the trust domain, satisfying the classical principle for the construction of secure systems that drives designers to minimize the size of the trusted components. As it is evident from a comparison of Figure 1 with Figure 2, the reduction in size of the trust domain is extreme. This novel data outsourcing architecture is then a significant improvement compared to current solutions for the network storage and dissemination of resources, and appears interesting for a large variety of applications.

The requirements that the data outsourcing architecture asks from the servers that are responsible for the storage and distribution of resources (and tokens) can be briefly summarized as follows. The first requirement is that servers must have a clear incentive in offering a reliable service, either because they are bound by a service contract with the owner and users, or because they benefit when users access the resources on the server (e.g., advertisements are presented to users accessing a resource). Second, the servers are also interested in providing guarantees that they will not be able to violate the confidentiality of the resources they store and help disseminate, because in this way they are able to offer the service to a larger user community (privacy-conscious users or organizations who would not otherwise use public services, may now be able to use them for the storage and dissemination of confidential resources). Service providers also benefit because they are not liable for the content that the users disseminate, obtaining the same legal immunity that is provided to Internet Service Providers about the information traveling on the communication channels they offer to their clients. If the server becomes malicious, either because compromised or forced by external entities, the most damage the server can do is to not distribute the resources, but resource confidentiality is protected as long as users keep strict custody on their keys.

In the next section, we illustrate a model, which is based on the data outsourcing architecture in Figure 2, that has been proposed for representing access control policies, for managing their evolution, and that uses two layers of encryption (over-encryption).

4. OVER-ENCRYPTION

To make the proposed approach generally applicable, no assumption is made on the users and resources with respect to which authorizations are defined. We simply assume sets $\mathcal{U}$ and $\mathcal{R}$, denoting the set of users and resources in the system, respectively, are given. An *authorization policy* is a set of pairs of the form $\langle u,r \rangle$, where $u \in \mathcal{U}$ and $r \in \mathcal{R}$, meaning that user u can access resource r (we consider access by users to be read-only). A policy can then be modeled via a traditional *access matrix* $\mathcal{A}$, with a row for each user $u \in \mathcal{U}$, a column for each resource $r \in \mathcal{R}$, and $\mathcal{A}[u, r]$ is set to 1 if u can access r; 0 otherwise. In the following, we will use $acl(r)$ to denote the *access control list* of r, that is, the set of users that can access r. Figure 3 illustrates an example of access matrix with four users (A, B, C, D) and six resources (r_1, r_2, ..., r_6).

The proposed solution is based on the key derivation method presented by Atallah's et al. [4]. This method exploits the definition and computation of *public tokens* that allow the derivation of keys from other keys. Let k_i and k_j be two keys. A token, denoted $t_{i,j}$, that allows the derivation of k_j from k_i is defined as $t_{i,j} = k_j \oplus h(k_i, l_j)$, where l_j is a publicly available label associated with k_j, $\oplus$ is the bit-a-bit `xor` operator, and h is a secure hash function (e.g., HMAC [16]). Key derivation can be applied via a chain of tokens $t_{i,l}, \ldots, t_{n,j}$ such that $t_{c,d}$ follows $t_{a,b}$ in the chain only if $b = c$.

Graphically, a set of keys $\mathcal{K}$ and a set of tokens $\mathcal{T}$ can be represented through a graph, having a vertex v_i associated with each key in the system, denoted $v_i.k$, and an edge (v_i, v_j) connecting v_i and v_j if token $t_{i,j}$ belongs to $\mathcal{T}$. Chains of tokens then correspond to paths in the graph. For instance, Figure 4 represents an example of graph corresponding to a set of 7 keys, along with its public token catalog $\mathcal{T}$ composed of 7 items. An example of token chain in the graph is represented by $t_{2,6}$ and $t_{6,7}$, that allow to derive $v_6.k$ from $v_2.k$ and $v_7.k$ from $v_6.k$, respectively.

A key derivation graph can be exploited for enforcing an access control policy by associating each user with one key (a node in the graph), and encrypting then each resource with a key that can be directly or indirectly derived only by the set of users belonging to its *acl*.

We now ready describe the two-layers model.

4.1 Two-layers model

Our solution introduces two separate layers of encryption, both adopting the key derivation method previously described (although some adaptations are necessary as explained in the following): a *Base Encryption Layer* (BEL), performed by the data owner before transmitting data to the server; and a *Surface Encryption Layer* (SEL), performed by the server over the resources already encrypted by the data owner. In the two-layer model, each user u receives two keys: one to access BEL and the other to access SEL, to be used for decryption one after the other.[1] At each layer, each

[1]Note that, for simplicity, the key at SEL layer can be computed from the key at BEL layer by applying on it a hash function. In this case, at initialization time, the data owner

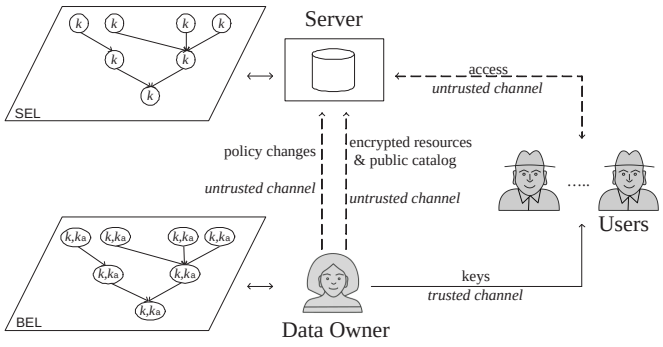
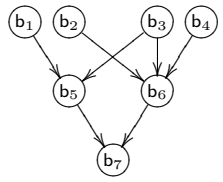


Figure 5: Over-encryption in the data outsourcing access control architecture



u	key
A	$b_1.k$
B	$b_2.k$
C	$b_3.k$
D	$b_4.k$

r	key
r_1	$b_1.k_a$
r_2,r_3,r_4	$b_5.k_a$
r_5	$b_6.k_a$
r_6	$b_7.k_a$

(a) **(b)** **(c)**

Figure 6: BEL encryption policy enforcing the authorization policy in Figure 3

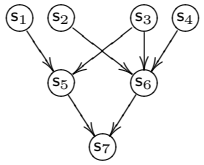
resource is encrypted by using a single key. Figure 5 illustrates the two-layer model translated in the data outsourcing architecture.

BEL. It enforces encryption on the resources according to the policy existing at initialization time. We distinguish two kinds of keys: *access keys* are used to encrypt resources, and *derivation keys* are used to provide the derivation capability via tokens. Each derivation key k is always associated with an access key k_a that is obtained through a secure hash function from k (i.e., $k_a = h(k)$). The rationale for this evolution is to distinguish the two roles associated with keys: enabling key derivation via the corresponding tokens, and enabling resource access. It is then up to the data owner to define a policy and to generate the corresponding encryption keys and tokens in such a way that each user can derive the *right set of decryption keys*, meaning that each user can decrypt all and only the resources for which she has the authorization (see [10]). The resulting set of keys and tokens is called BEL *encryption policy*.

The key derivation relationship is again represented through a graph, where now there is a vertex b for each pair of keys $\langle k, k_a \rangle$ and an edge (b_i, b_j) connects b_i and b_j if there is a token in the public catalog allowing the derivation of either $b_j.k$ or $b_j.k_a$ from $b_i.k$.

Figure 6 illustrates the BEL encryption policy enforcing the authorization policy in Figure 3. In particular, Figure 6(a) shows the graph representing the key derivation relationship, Figure 6(b) shows the keys communicated to each user, and Figure 6(c) shows the keys used for encrypting the resources. It is easy to see that, for example, since user A knows key $b_1.k$ and is able to derive keys $b_5.k$ and $b_7.k$ that in turn allow the computation of keys $b_1.k_a$, $b_5.k_a$, and $b_7.k_a$, she is allowed to access the set of resources $\{r_1,r_2,r_3,r_4,r_6\}$, which are exactly the resources that user A is authorized to access.

has to communicate to each user a single BEL key and to the server the SEL keys.



u	key
A	$s_1.k$
B	$s_2.k$
C	$s_3.k$
D	$s_4.k$

r	key
r_1	$s_1.k$
r_2,r_3,r_4	$s_5.k$
r_5	$s_6.k$
r_6	$s_7.k$

(a) **(b)** **(c)**

Figure 7: SEL encryption policy enforcing the authorization policy in Figure 3

SEL. It enforces the dynamic changes over the policy and it is initialized to repeat exactly the BEL policy: for each pair of keys $\langle k, k_a \rangle$ in BEL a corresponding key is defined in SEL; for each token in BEL, a corresponding token is defined in SEL. The key derivation relationship is then represented through a graph that is isomorphic to the graph existing at the BEL level. Each user is assigned the unique key corresponding to her key at BEL layer. Each resource is encrypted with the unique key corresponding to the unique access key at BEL layer. Since according to this strategy, which we call Full_SEL, the SEL encryption policy models exactly the BEL policy, by definition it is correct with respect to the original authorization policy. Figure 7 illustrates the SEL encryption policy enforcing the authorization policy in Figure 3.

Note that an alternative strategy that can be adopted at the SEL layer consists in not carrying out any encryption when the BEL layer correctly enforces the policy for a resource (i.e., no encryption is performed on resources) [11]. A double encryption will be enforced only when needed to enforce a policy change. This strategy, which we call Delta_SEL, is however characterized by a greater information exposure than the Full_SEL (see Section 4.3). By contrast, the Full_SEL always requires double encryption to be enforced.

4.2 Policy changes enforcement

A policy change can require the insertion/deletion of a user/resource or the grant/revoke of an authorization. Without loss of generality, we focus on grant/revoke operations, since the insertion (resp. deletion) of a user/resource can be modeled via a set of grant (resp. revoke) operations.

We assume also that SEL layer operates in the Full_SEL mode.

A *grant* operation on user u for resource r is performed by analyzing the status of r compared with the current configuration of BEL and SEL. First, $\mathcal{A}[u, r]$ is set to 1 (i.e., $acl(r) = acl(r) \cup \{u\}$) and if user u cannot derive the BEL key used for encrypting r, a BEL token from the BEL key associated with u to the access key used for encrypting r is added. Finally, the SEL level receives an over-encryption request for resource r to synchronize the SEL layer with the policy change, meaning that the graph corresponding to the key derivation relationship needs to be properly updated to reflect the new policy [9]. To this aim, it may be necessary to create a new SEL key, along with the tokens for its derivation. Analogously, whenever a SEL key is not used for encrypting any resource, it can be removed from the layer, along with the corresponding public tokens.

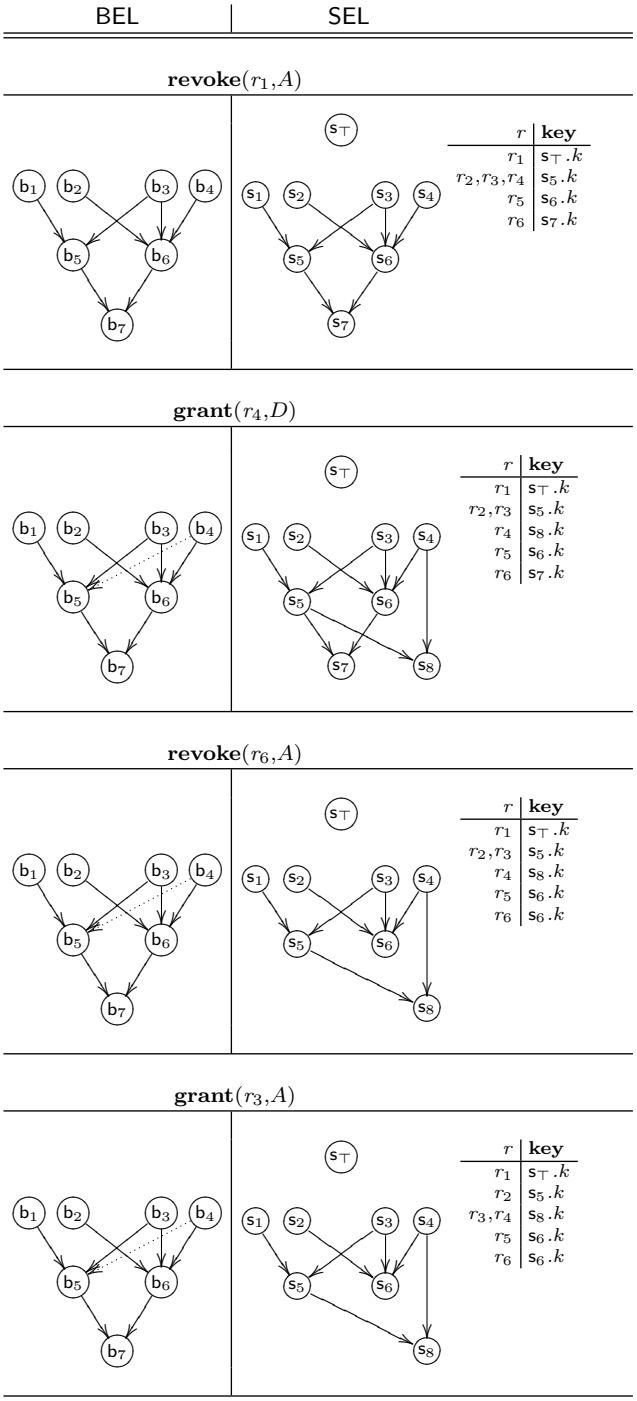


Figure 8: An example of grant and revoke operations

A *revoke* operation for user u on resource r is performed by setting $\mathcal{A}[u, r]$ to 0 (i.e., $acl(r) = acl(r) - \{u\}$) and by requesting the SEL layer to make r accessible only to the new set of users that are authorized to access r.

As an example, consider the initial configuration of both BEL and SEL encryption policies represented in Figure 6 and Figure 7, respectively. Figure 8 illustrates the evolution of both BEL and SEL encryption policies to accommodate a series of grant and revoke operations. Note that, in the graphical representation, dashed edges represent tokens leading to access keys.

- **revoke**(r_1,A): user A is removed from $acl(r_1)$. Since this acl becomes empty, resource r_1 has to be over-encrypted with a key that no user can compute. Consequently, a new vertex $\mathsf{s}_\top$ is created and its key is used to encrypt r_1.

- **grant**(r_4,D): key $b_5.k_\mathsf{a}$ used to encrypt r_4 cannot be derived from $b_4.k$ assigned to D. The data owner therefore adds a BEL token $t_{4,5}$. Also, $s_5.k$ used to over-encrypt r_4 cannot be derived from $s_4.k$ assigned to D. Therefore, a new vertex s_8 is added to the SEL, along with two tokens, $t_{5,8}$ and $t_{4,8}$. Resource r_4 is then over-encrypted by using $s_8.k$, which is accessible to A, C, and D.

- **revoke**(r_6,A): user A is removed from $acl(r_6)$. While the encryption policy at BEL layer remains unchanged, r_6 needs to be over-encrypted with a key accessible only to B, C, and D. Since $s_6.k$ can be accessed by these three users only, r_6 is over-encrypted by using $s_6.k$. After r_6 over-encryption, no resource is over-encrypted using $s_7.k$. Consequently, s_7 is removed from SEL.

- **grant**(r_3,D): key $b_5.k_\mathsf{a}$ used to encrypt r_3 can be derived from $b_4.k$ assigned to D. Consequently, BEL remains unchanged. By contrast, key $s_5.k$ used to over-encrypt r_3 cannot be derived from $s_4.k$, assigned to D. Resource r_3 needs to be over-encrypted with a key derivable from A, C, and D. Since $s_8.k$ is accessible by exactly these users, r_3 is over-encrypted by using $s_8.k$.

4.3 Collusion risks

The two-layer approach guarantees the protection of confidentiality with respect to both the server (for the encryption performed at BEL) and the users (for the encryption performed at SEL). Collusion takes place anytime two entities, combining their knowledge (i.e., their keys) can acquire knowledge that neither of them had access to. Before any policy update is made, the system is collusion free. Dynamic policy updates could instead expose to collusion, although according to our analysis it is limited and identifiable. In particular, from our analysis [11], we can conclude that there is a risk of collusion when a grant is executed that assigns to a user u access privilege to only a portion of the resources encrypted with the same key at the BEL layer. All the other resources to which access is not granted can be decrypted if user u colludes with the server. As an example, resource r_2 is exposed to the risk of collusion between the server and user D since, after policy updates depicted in Figure 8, D can compute the value of $b_5.k_\mathsf{a}$. Due to the separation between secrets and keys for each vertex, the exposure is however well defined.

5. CONCLUSIONS

The paper explored many important issues that arise when enforcing access control in a scenario where data are stored and offered to clients by an external server. We then presented a novel data outsourcing access control architecture for supporting flexible applications, preserving privacy

and empowering the user. We also described an approach for policy evolution that takes into account the main features of the scenario and is able to guarantee in most cases confidentiality of the information in the presence of significant policy updates, clearly identifying the exposure to collusion when this risk may arise. Other issues to be investigated include the integration with the Web paradigm, and the efficient execution of queries.

6. ACKNOWLEDGEMENTS

This work was supported in part by the European Union under contract IST-2002-507591, and by the Italian MIUR under PRIN 2006 project ID:2006099978 "Basi di dati crittografate".

7. REFERENCES

[1] R. Agrawal, J. Kierman, R. Srikant, and Y. Xu. Order preserving encryption for numeric data. In *Proc. of ACM SIGMOD 2004*, Paris, France, June 2004.

[2] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM TOCS*, 1(3):239–248, August 1983.

[3] J. Anderson. Computer security planning study. Technical Report 73-51, Air Force Electronic System Division, 1972.

[4] M. Atallah, K. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *Proc. of the 12th ACM CCS05*, Alexandria, VA, USA, November 2005.

[5] L. Bouganim, F. D. Ngoc, P. Pucheral, and L. Wu. Chip-secured data access: Reconciling access rights with data encryption. In *Proc. of the 29th VLDB Conference*, Berlin, Germany, September 2003.

[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] V. Ciriani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Fragmentation and encryption to enforce privacy in data storage. In *Proc. of the 12th ESORICS*, Dresden, Germany, September 2007.

[8] J. Crampton, K. Martin, and P. Wild. On key assignment for hierarchical access control. In *Proc. of the 19th IEEE CSFW'06*, Venice, Italy, July 2006.

[9] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Key management for multiuser encrypted databases. In *Proc. of the International Workshop on Storage Security and Survivability*, Fairfax, Virginia, USA, November 2005.

[10] E. Damiani, S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. An experimental evaluation of multi-key strategies for data outsourcing. In *Proc. of the 22nd IFIP TC-11 International Information Security Conference*, South Africa, May 2007.

[11] S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Over-encryption: Management of access control evolution on outsourced data. In *Proc. of the 33rd VLDB Conference*, Vienna, Austria, September 2007.

[12] H. Hacigümüs, B. Iyer, and S. Mehrotra. Providing database as a service. In *Proc. of 18th ICDE*, San Jose, CA, USA, February 2002.

[13] H. Hacigümüs, B. Iyer, and S. Mehrotra. Efficient execution of aggregation queries over encrypted relational databases. In *Proc. of the 9th International Conference on Database Systems for Advanced Applications*, Jeju Island, Korea, March 2004.

[14] 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 2002*, Madison, Wisconsin, USA, June 2002.

[15] A. Harrington and C. Jensen. Cryptographic access control in a distributed file system. In *Proc. of the 8th SACMAT*, Como, Italy, June 2003.

[16] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*. Internet Request for Comments RFC-2104, February 1997.

[17] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *Proc. of the 29th VLDB Conference*, Berlin, Germany, September 2003.

[18] N. Provos . Encrypting virtual memory. In *Proc. of the 9th USENIX Security Symposium*, Denver, Colorado, USA, August 2000.

[19] J. Saltzer and M. Schroeder. The protection of information in computer systems. *Communications of the ACM*, 17(7), July 1974.