Enforcing Subscription-Based Authorization Policies in Cloud Scenarios

Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Giovanni Livraga

▶ **To cite this version:**

Sabrina De Capitani Di Vimercati, Sara Foresti, Sushil Jajodia, Giovanni Livraga. Enforcing Subscription-Based Authorization Policies in Cloud Scenarios. 26th Conference on Data and Applications Security and Privacy (DBSec), Jul 2012, Paris, France. pp.314-329, 10.1007/978-3-642-31540-4_24 . hal-01534756

HAL Id: hal-01534756
https://inria.hal.science/hal-01534756

Submitted on 8 Jun 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Enforcing Subscription-based Authorization Policies in Cloud Scenarios

Sabrina De Capitani di Vimercati[1], Sara Foresti[1],
Sushil Jajodia[2], and Giovanni Livraga[1]

[1] Università degli Studi di Milano, 26013 Crema, Italy
firstname.lastname@unimi.it
[2] George Mason University, Fairfax, VA 22030-4444, USA
jajodia@gmu.edu

Abstract. The rapid advances in the Information and Communication Technologies have brought to the development of on-demand high quality applications and services allowing users to easily access resources anywhere anytime. Users can pay for a service and access the resources made available during their subscriptions until the subscribed periods expire. Users are then forced to download such resources if they want to access them also after the subscribed periods. To avoid this burden to the users, we propose the adoption of a subscription-based access control policy that combines a flexible key derivation structure with selective encryption. The publication of new resources as well as the management of subscriptions are accommodated by adapting the key derivation structure in a transparent way for the users.

Keywords: access control, subscription-based policies, data outsourcing

1 Introduction

The advances in the Information and Communication Technologies (ICTs) have driven the users into the Globalization era, where the techniques for processing, storing, and accessing information have radically changed. New emerging computing paradigms (e.g., data outsourcing and cloud computing) offer enormous advantages to both users and organizations. Users can now subscribe to a variety of services, and access them anywhere anytime: at home from their laptop, on the train from their tablet, or while waiting in a queue from their smartphone. Organizations are more and more resorting to external elastic storage and computational services for creating and running business over the Internet in new ways. Organizations can then provide large-scale cloud data services widely accessible to a variety of users, possibly restricting access to resources on the basis of users' subscriptions. These services can be offered at affordable prices, thanks to the use of external cloud storage servers for the management of data. As a side effect of this trend, security requirements are becoming more complex since cloud storage servers are typically trusted neither to access the resources content nor to restrict access to the services according to users' subscriptions.

Emerging approaches in the data outsourcing scenario regulate access to resources through selective encryption, meaning that they translate the privilege to access a resource into the knowledge of the key used to encrypt the resource itself (e.g., [10]). These approaches, however, while representing important steps towards the support of flexible access control solutions in data outsourcing, are still in their infancy. In fact, they cannot easily support a scenario where both the set of users who can access a resource and the set of resources change frequently over time, due to new subscriptions and the publication of new resources. Also access control solutions developed for publish/subscribe systems (e.g., [11, 20]), which may seem to have some similarities with the publication scenario we consider, are not suitable since they have been developed for a different problem. We take into account scenarios where users pay for a service and then can freely access the resources made available during their subscriptions. In this context, to access resources also after the expiration of their subscriptions, users can download the resources for which they are authorized to their local machine. Our proposal aims at avoiding this burden to the users allowing them to maintain the right to access such resources without the worry that they will lose this right after the expiration of their subscriptions. For instance, users who have purchased an annual subscription for 2012 for a magazine should be able to access all and only the issues of the magazine published in 2012, and should be able to access them even after December 31, 2012. We therefore propose an approach that takes advantage of selective encryption to guarantee that users who subscribe for a service can access all and only the resources published during their subscriptions, while allowing the resources to self-enforce the subscription-based restrictions. Before being stored on the cloud storage server, the resources are encrypted, and a key derivation structure is built to guarantee that they can be decrypted only by authorized users. The key derivation structure is updated whenever new resources are published, new subscriptions are received, or users withdraw from their subscriptions.

The remainder of this paper is organized as follows. Section 2 describes the considered scenario and the protection requirements that the access control system should satisfy. Section 3 formalizes the concept of subscription-based policy. Section 4 presents our techniques for enforcing a subscription-based policy. Section 5 illustrates how the system publishes resources and manages subscriptions. Section 6 discusses related work. Finally, Section 7 reports our conclusions.

2 Scenario, Protection Requirements, and Motivation

We consider a scenario where a *resource provider* uses an external *cloud storage server* for storing its resources, thus taking advantage of the cost savings that the cloud storage server can provide. The resource provider periodically publishes new resources that should be able to self-enforce restrictions on who can access them and should not be accessible to the cloud storage server. *Users* can subscribe to the services offered by the resource provider for different periods of time, and can withdraw from a subscription at any time. We assume that

users are trusted, that is, they do not redistribute resources they can access to unauthorized users.

In the considered scenario, accesses to resources should be regulated by a *subscription-based access control policy* according to which users are authorized to access all and only the resources that have been published by the resource provider during their subscribed periods. A peculiarity of our scenario is that user authorizations remain valid also after the expiration of their subscriptions. The subscription-based access control policy takes then into consideration both the subscriptions of the users and the time when resources have been published. Existing solutions result limited for our scenario. We can classify such existing solutions in two main categories.

- *Account-based*. Traditional access control solutions (e.g., [17]), including those emerging in the data outsourcing scenario (e.g., [10]), are based on the assumption that when users leave the system their authorizations terminate and they cannot access the resources anymore. Furthermore, access control solutions for data outsourcing cannot easily support a dynamic scenario where resources are continuously created, and new users can join the system and old users can leave the system at any time.
- *Time-based*. Temporal-based access control solutions (e.g., [4]) enforce time restrictions in a way that is different from what we need. In fact, these solutions consider a scenario where resources are stored and managed by the party who creates them, and assume that authorizations apply only to specific time intervals and/or that authorizations can be applied following a periodic pattern (e.g., a user can access a file only during the working days from 8:00 a.m. to 5:00 p.m.).

We then put forward the idea of using the same principles at the basis of the access control solutions developed for the data outsourcing scenario (which encrypt resources for protecting their confidentiality from the storage server and adopt key derivation techniques for efficiently combining authorization-based access control and cryptographic protection) to enforce a subscription-based access control policy without delegating it to the cloud storage server. Our solution should guarantee the correct enforcement of the subscription-based access control policy (i.e., users should be able to access the resources made available during their subscribed periods also after the expiration of their subscriptions) and the *forward* and *backward* protection requirements. Forward protection means that users cannot access resources published before the beginning of their subscriptions (e.g, users who subscribe to a magazine for 2012 cannot access the issues of the magazine published before January 1, 2012). Backward protection means that users cannot access resources published after the expiration of their subscriptions (e.g., users who subscribe to a magazine for 2012 cannot access the issues of the magazine published during 2013). Like for the data outsourcing scenario, with our solution the published resources are encrypted so that they self-enforce the subscription-based access restrictions. In addition to the correct enforcement of the subscription-based policy and the satisfaction of the

forward and backward protection requirements mentioned above, our solution should avoid re-encryption of resources and re-distribution of keys whenever users subscribe to services or withdraw from their subscriptions.

3 Subscription-based Policy

A resource provider offers a set $\mathcal{P}$ of services to which users can subscribe. Each service $p \in \mathcal{P}$ consists in a period of publication of resources, and each user subscribing to service p can access all the resources published for p during her subscription. We denote with $\mathcal{U}$ and $\mathcal{R}$ the set of users subscribed to service p and the set of published resources for p, respectively. For simplicity, but without loss of generality, we focus on the management of accesses to a single service. We also note that, although in this paper we consider time-based subscriptions, our approach can be easily adapted to other scenarios where subscriptions to a service can be defined on the basis of different criteria (e.g., topic of interest, geographical region).

Given a *time domain* $(\mathcal{TS},\leq)$, with $\mathcal{TS}$ a set of time instants and $\leq$ a total order relationship on $\mathcal{TS}$ [5], the resource provider assigns to each resource $r \in \mathcal{R}$ a timestamp $r.t$ in $\mathcal{TS}$ that represents the time when the resource has been published. The resource provider may combine contiguous time instants into time windows, defined on arbitrary granularities, forming a *time hierarchy*. Intuitively, these time windows represent the periods of time for which the resource provider allows users to subscribe to the service offered. Formally, a time hierarchy $\mathcal{H}_\mathcal{T}$ is a pair $(\mathcal{T},\succeq)$, where $\mathcal{T}$ is a set of time windows, and $\succeq$ is a partial order relationship over $\mathcal{T}$. A time window T_i in $\mathcal{T}$ is a pair $[t_i^s,t_i^e]$ of time instants and represents the set of time instants $t \in \mathcal{TS}$ such that $t_i^s \leq t \leq t_i^e$. Given two time windows T_i and T_j in $\mathcal{T}$, T_i *dominates* T_j, denoted $T_i \succeq T_j$, if $t_i^s \leq t_j^s$ and $t_j^e \leq t_i^e$ (i.e., the time instants in T_j represent a subset of the time instants in T_i). The leaves of the time hierarchy correspond to time instants in $\mathcal{TS}$, which can be seen as time windows with $t^s = t^e$. The time hierarchy can be graphically represented as a directed acyclic graph with vertices representing time windows in $\mathcal{T}$ and edges representing direct dominance relationships. For simplicity, but without loss of generality, in this paper we assume $\mathcal{H}_\mathcal{T}$ to be a tree. As an example, consider resource provider Condé Nast, monthly publishing magazine *Glamour* and offering the possibility to buy subscriptions for a month (single issue), a trimester, a semester, or a year. Figure 1 illustrates the time hierarchy defined by the resource provider. For the sake of readability, in the figure we denote leaves with the time instant they represent. Each user $u \in \mathcal{U}$ can subscribe to the service offered by the resource provider for an arbitrary set, denoted $u.\mathcal{S}$, of time windows in $\mathcal{H}_\mathcal{T}$ (i.e., $u.\mathcal{S} \subseteq \mathcal{T}$).

The timestamps assigned to resources along with the user subscriptions establish the set of resources that each user can access: user $u \in \mathcal{U}$ can access resource $r \in \mathcal{R}$ if she subscribed for a time window including $r.t$. Formally, the subscription-based policy regulating access to the resources is defined as follows.

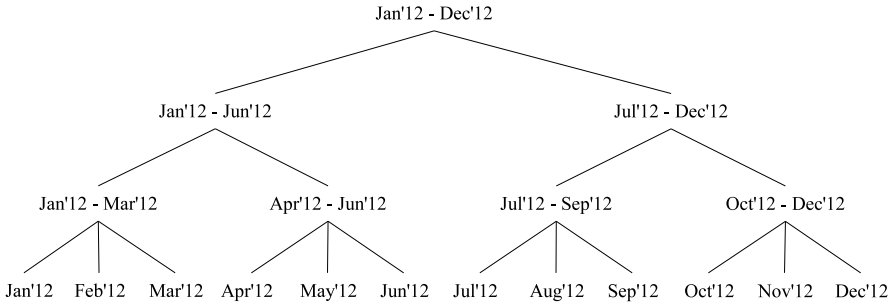


Fig. 1: An example of time hierarchy

Definition 1 (Subscription-based policy). *Let $\mathcal{H}_{\mathcal{T}}(\mathcal{T},\succeq)$ be a time hierarchy defined on time domain $(\mathcal{TS},\leq)$, $\mathcal{U}$ be a set of users with $u.\mathcal{S}\subseteq\mathcal{T}$ for all $u\in\mathcal{U}$, and $\mathcal{R}$ be a set of resources with $r.t\in\mathcal{TS}$ for all $r\in\mathcal{R}$. The* subscription-based policy $\mathcal{A}$ *on $\mathcal{U}$ and $\mathcal{R}$ grants $u\in\mathcal{U}$ access to $r\in\mathcal{R}$ iff $\exists[t^s,t^e]\in u.\mathcal{S}$ s.t. $t^s\leq r.t\leq t^e$.*

Example 1. Suppose that three issues of magazine *Glamour* have been published with timestamp Jan'12, Feb'12, and Mar'12, respectively (i.e., $\mathcal{R}$={*Glam-01*,*Glam-02*,*Glam-03*}). Assume now that two users $\mathcal{U}$={*Alice*, *Barbara*} subscribe to the magazine for the first trimester of 2012 ([Jan'12,Mar'12]), and for the first issue of the year ([Jan'12,Jan'12]), respectively. The subscription-based policy grants *Alice* access to all the issues of the magazine in $\mathcal{R}$, while it grants *Barbara* access only to the first issue *Glam-01*.

4 Graph Modeling of the Subscription-based Policy

We propose to enforce the subscription-based policy by combining *selective encryption* [10] with a key derivation technique that uses a key derivation structure based on public *tokens* [1]. Given two keys k_i and k_j in a set $\mathcal{K}$ of keys, token $d_{i,j}=k_j\oplus h(k_i,l_j)$, with l_j a public label associated with k_j, $\oplus$ the bitwise `xor` operator, and h a deterministic cryptographic function, permits to derive k_j from the knowledge of k_i and label l_j. The derivation relationship between keys can be either *direct*, via a single token, or *indirect*, via a chain of tokens. Our idea consists in defining a key derivation structure so that each resource is encrypted only once with a single key, and each user receives only one key from which she can derive all and only the keys used for encrypting the resources that she can access according to the subscription-based policy. To fix ideas and make the discussion clear, we consider the system at a specific point in time when some resources have been published and some users have subscribed to the service offered by the resource provider. We first discuss how resources are encrypted and then describe how to model users' subscriptions.

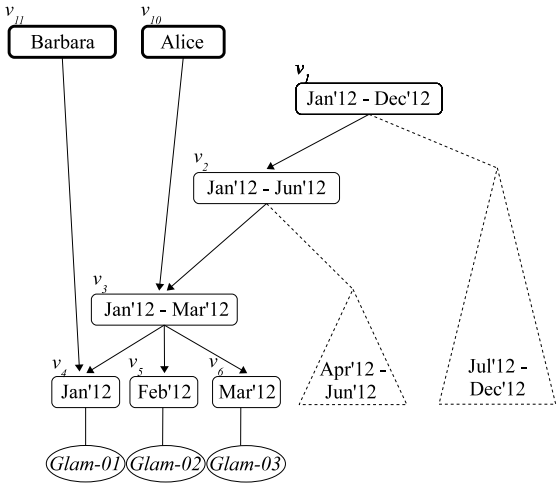
The techniques developed for enforcing an access control policy in the data outsourcing scenario build a key derivation structure on the basis of the sets of

users that can access resources. In our scenario, such sets of users vary frequently over time, and therefore it is not convenient to exploit them for building the key derivation structure. We then use the time hierarchy $\mathcal{H}_{\mathcal{T}}$ defined by the resource provider as a key derivation structure where each time window is associated with a key, and each edge corresponds to a token. The timestamp associated with a published resource, therefore, identifies the time window in the time hierarchy representing the key used to encrypt the resource itself. The keys associated with time windows including more than a time instant (i.e., internal vertices) are not used for encrypting resources, but only for derivation purposes. Clearly, not all the time windows in the time hierarchy are necessary for enforcing the subscription-based policy, but only those corresponding to the timestamps of published resources along with all the time windows dominating them. For instance, with respect to Example 1, the time windows that must be represented in the key derivation structure are Jan'12, Feb'12, and Mar'12, which are the timestamps of the three published resources, and all the time windows dominating them in the time hierarchy in Figure 1, that is, [Jan'12,Mar'12], [Jan'12,Jun'12], and [Jan'12,Dec'12]. In this way, from the knowledge, for example, of the key associated with [Jan'12,Mar'12] we can derive the keys used for encrypting all the resources published during the first trimester of 2012.

For each user in the system, the resource provider generates a new key and communicates it to the user. With this unique key, the user should be able to access all and only the resources for which she is authorized according to her subscriptions. The idea is to "hook the user" through a token on each time window T for which she subscribed. In this way, the user can adopt her key to directly derive the key associated with time window T. From this key she can directly or indirectly derive the keys used to encrypt all and only the resources whose timestamp is included in T. For instance, according to the subscriptions in Example 1, *Alice* can access all the resources published in the first trimester of 2012. The resource provider then creates a token from *Alice*'s key to the key associated with [Jan'12,Mar'12]. By construction, all resources published in Jan'12, Feb'12, and Mar'12 will be encrypted with a key derivable from the key associated with [Jan'12,Mar'12], which *Alice* can derive. Note that it may happen that a user subscribes for a time window for which no resource has been published (e.g., a user subscribes to a magazine for April'12 and the issue of April has not been published yet). The key derivation structure must then include also the time windows representing users' subscriptions, along with their ancestors in $\mathcal{H}_{\mathcal{T}}$. The resulting key derivation structure, which we call *user and resource graph*, can be formally defined as follows.

Definition 2 (User and resource graph). *Let $\mathcal{H}_{\mathcal{T}}(\mathcal{T},\succeq)$ be a time hierarchy on time domain $(\mathcal{TS},\leq)$, $\mathcal{U}$ be a set of users with $u.\mathcal{S}\subseteq\mathcal{T}$ for all $u\in\mathcal{U}$, and $\mathcal{R}$ be a set of resources with $r.t\in\mathcal{TS}$ for all $r\in\mathcal{R}$. A* user and resource graph *over $\mathcal{U}$, $\mathcal{R}$, and $\mathcal{H}_{\mathcal{T}}$ is a graph $G(V,E)$, with:*

- $V = \mathcal{T}_r \cup \mathcal{T}_s \cup \mathcal{T}_p \cup \mathcal{U}$, *with* $\mathcal{T}_r = \bigcup_{r\in\mathcal{R}}[r.t, r.t]$, $\mathcal{T}_s = \bigcup_{u\in\mathcal{U}} u.\mathcal{S}$, *and* $\mathcal{T}_p = \{T \in \mathcal{T} \mid \exists T' \in \mathcal{T}_s \cup \mathcal{T}_r$ *such that* $T \succeq T'\}$



(a)

Id	Enc_Resource
1	E(Glam-01,$v_4.k$)
2	E(Glam-02,$v_5.k$)
3	E(Glam-03,$v_6.k$)

(b)

From	To	Token
v_1	v_2	$v_2.k \oplus h(v_1.k, v_2.l)$
v_2	v_3	$v_3.k \oplus h(v_2.k, v_3.l)$
v_3	v_4	$v_4.k \oplus h(v_3.k, v_4.l)$
v_3	v_5	$v_5.k \oplus h(v_3.k, v_5.l)$
v_3	v_6	$v_6.k \oplus h(v_3.k, v_6.l)$
v_{10}	v_3	$v_3.k \oplus h(v_{10}.k, v_3.l)$
v_{11}	v_4	$v_4.k \oplus h(v_{11}.k, v_4.l)$

(c)

Fig. 2: An example of user and resource graph (a), published resources (b), and token catalog (c)

– $E = \{(u,T) \mid u \in \mathcal{U} \wedge T \in V \setminus \mathcal{U} \wedge T \in u.\mathcal{S}\} \cup$
$\{(T_i,T_j) \mid T_i,T_j \in V \setminus \mathcal{U} \wedge T_i \succeq T_j \wedge (\nexists T_z \in V \setminus \mathcal{U},\ T_i \succeq T_z \succeq T_j \wedge T_z \neq T_i \neq T_j)\}$

The vertices in the user and resource graph represent the keys of the system, while the edges represent the tokens in the *token catalog* $\mathcal{D}$ stored at the external cloud storage server together with the encrypted resources.

Example 2. Consider the time hierarchy in Figure 1 and the subscription-based policy in Example 1. Figure 2(a) shows the corresponding user and resource graph, where dotted triangles represent subtrees of the time hierarchy that are not associated with a vertex in the graph. For the sake of clarity, the figure also reports the published resources, represented as ovals connected with the vertices in the graph representing their timestamp and whose keys are used to encrypt them. Figure 2(b) shows the encrypted resources stored at the external cloud storage server, with **Id** the resource identifier and **Enc_Resource** the encrypted resource ($E(r,k)$ denotes the encryption of r with k), and Figure 2(c) illustrates the token catalog resulting from the user and resource graph in Figure 2(a).

The user and resource graph in Definition 2 guarantees the correct enforcement of the subscription-based policy since each user can decrypt all and only the resources with a timestamp included in at least one of the time windows in the user's subscriptions. This is formalized by the following theorem, whose proof is omitted from the paper for space constraints.

Theorem 1 (Correct enforcement of subscription-based policy). *Let $\mathcal{H}_\mathcal{T}(\mathcal{T},\succeq)$ be a time hierarchy on time domain $(\mathcal{TS},\leq)$, $\mathcal{U}$ be a set of users with*

```
PUBLISH_RESOURCE(r)
 1: R := R ∪ {r}
 2: v := Get_Vertex([r.t,r.t]) /* retrieve the vertex representing the timestamp of the resource */
 3: Encrypt(r,v.k)
 4: publish the encrypted resource

GET_VERTEX(T)
 5: if T∈V then /* T already belongs to G */
 6:   let v∈V be the vertex with v=T
 7:   return(v)
 8: generate vertex v := T
 9: generate encryption key v.k
10: generate public label v.l
11: V := V ∪ {v} /* insert the vertex into the user and resource graph */
12: let T_i∈T: T_i⊒T ∧ ∄T_j: T_i⊒T_j⊒T, T_j≠T_i≠T /* determine the direct ancestor of T in H_T */
13: if T_i≠NULL then
14:   v_i := Get_Vertex(T_i) /* retrieve the vertex in G that represents T_i */
15:   E := E ∪ {(v_i,v)} /* insert the edge connecting T_i to T in G */
16:   D := D ∪ {v.k⊕h(v_i.k,v.l)} /* publish the corresponding token */
17: return(v)
```

Fig. 3: Pseudocodes of procedure **Publish_Resource** and function **Get_Vertex**

$u.\mathcal{S} \subseteq \mathcal{T}$ for all $u \in \mathcal{U}$, and $\mathcal{R}$ be a set of resources with $r.t \in \mathcal{T}\mathcal{S}$ for all $r \in \mathcal{R}$. The user and resource graph $G(V,E)$ correctly enforces *a subscription-based policy $\mathcal{A}$ on $\mathcal{U}$ and $\mathcal{R}$ when $\forall u \in \mathcal{U}$, $\forall r \in \mathcal{R}$:*

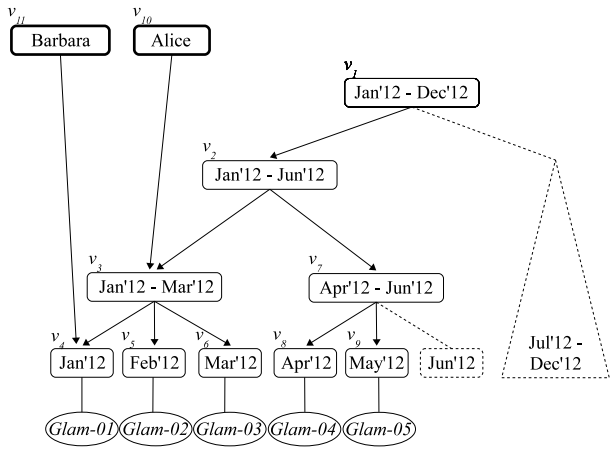
$$\exists [t^s,t^e] \in u.\mathcal{S} \text{ s.t. } t^s \leq r.t \leq t^e \iff \langle u,[r.t,r.t]\rangle \text{ is a path in } G.$$

5 Management of Resources and Subscriptions

Whenever there is a change in the subscription-based policy (e.g., a new resource is published, a user subscribes to a service for a specific time window, or a user decides to withdraw from a subscription), the user and resource graph has to be updated accordingly. In the following, we discuss how changes to the policy can be managed in a transparent way for the users.

5.1 Resource Publishing

At initialization time, the user and resource graph is empty (no key is necessary for resource encryption) and it is dynamically built as resources are published. Figure 3 illustrates the pseudocode of procedure **Publish_Resource** that the resource provider calls whenever it needs to publish a resource. The procedure takes a resource r as input and publishes its encrypted representation. The procedure first calls function **Get_Vertex** on time window $T=[r.t,r.t]$ (line 2). This function checks whether the vertex representing $[r.t,r.t]$ is in the user and resource graph, since its key has to be used for encrypting r. If such a vertex exists, the function returns it (lines 5-7). Otherwise, the function first creates a vertex v representing T, along with the corresponding encryption key $v.k$ and public label $v.l$, and inserts v into the set V of vertices of the user and resource



Id	Enc_Resource	
1	E(Glam-01,$v_4.k$)	
2	E(Glam-02,$v_5.k$)	
3	E(Glam-03,$v_6.k$)	
4	E(Glam-04,$v_8.k$)	*
5	E(Glam-05,$v_9.k$)	*

(b)

From	To	Token	
v_1	v_2	$v_2.k \oplus h(v_1.k, v_2.l)$	
v_2	v_3	$v_3.k \oplus h(v_2.k, v_3.l)$	
v_2	v_7	$v_7.k \oplus h(v_2.k, v_7.l)$	*
v_3	v_4	$v_4.k \oplus h(v_3.k, v_4.l)$	
v_3	v_5	$v_5.k \oplus h(v_3.k, v_5.l)$	
v_3	v_6	$v_6.k \oplus h(v_3.k, v_6.l)$	
v_7	v_8	$v_8.k \oplus h(v_7.k, v_8.l)$	*
v_7	v_9	$v_9.k \oplus h(v_7.k, v_9.l)$	*
v_{10}	v_3	$v_3.k \oplus h(v_{10}.k, v_3.l)$	
v_{11}	v_4	$v_4.k \oplus h(v_{11}.k, v_4.l)$	

(c)

Fig. 4: User and resource graph (a), published resources (b), and token catalog after *Glam-04* and *Glam-05* are published (c)

graph (lines 8-11). To guarantee that the time window T_i directly dominating T in the time hierarchy is represented in the user and resource graph, function **Get_Vertex** recursively calls itself on T_i, obtaining the vertex v_i representing T_i in the graph (lines 12-14). The function inserts into G edge (v_i,v) and publishes the corresponding token (lines 15-16). We note that the recursive nature of function **Get_Vertex** guarantees that all the ancestors of T in $\mathcal{H}_{\mathcal{T}}$ are represented by a vertex in the user and resource graph, and that each vertex is connected to all its direct descendants represented in the graph. The function then returns vertex v representing $[r.t,r.t]$ (line 17). Finally, procedure **Publish_Resource** encrypts r with $v.k$ and publishes the resulting encrypted resource (lines 3-4).

Example 3. Consider the user and resource graph, published resources, and token catalog in Figure 2 and assume that Condé Nast publishes the fourth issue of *Glamour* in April'12. The resource provider calls procedure **Publish_Resource** on resource *Glam-04* that in turn calls function **Get_Vertex** on [Apr'12,Apr'12]. The function inserts vertex v_8 representing [Apr'12,Apr'12] and its direct ancestor v_7 representing [Apr'12,Jun'12]. Procedure **Publish_Resource** then encrypts *Glam-04* with the key of vertex v_8. Assume now that Condé Nast publishes the fifth issue of *Glamour* in May'12, calling procedure **Publish_Resource** on resource *Glam-05*. Function **Get_Vertex** inserts vertex v_9 representing [May'12,May'12] and directly connects it to [Apr'12,Jun'12], since it is already included in the graph. Resource *Glam-05* is encrypted with the key of vertex v_9. Figure 4 illustrates the resulting user and resource graph, published resources, and token catalog, where new resources and tokens are denoted with a $*$.

SUBSCRIBE(u,T)
1: **if** $u \notin \mathcal{U}$ **then** /* u is a new user in the system */
2: $\mathcal{U} := \mathcal{U} \cup \{u\}$
3: generate vertex $v_u := u$
4: generate encryption key $v_u.k$
5: generate public label $v_u.l$
6: $V := V \cup \{v_u\}$
7: **else** let $v_u \in V$ be the vertex with $v_u = u$
8: $u.\mathcal{S} := u.\mathcal{S} \cup \{T\}$
9: $v_T :=$ **Get_Vertex**(T)
10: $E := E \cup \{(v_u, v_T)\}$
11: $\mathcal{D} := \mathcal{D} \cup \{v_T.k \oplus h(v_u.k, v_T.l)\}$
12: let $T_i \in \mathcal{T}$: $T_i \succeq T \wedge (\nexists T_j$: $T_i \succeq T_j \succeq T$, $T_j \neq T_i \neq T)$ /* determine the direct ancestor of T in $\mathcal{H}_\mathcal{T}$ */
13: $\mathcal{T}' := \{T_j \in u.\mathcal{S} \mid T_i \succeq T_j \wedge (\nexists T_z \in \mathcal{T}$: $T_i \succeq T_z \succeq T_j$, $T_i \neq T_z \neq T_j)\}$
14: **if** $\bigcup_{T_j \in \mathcal{T}'} T_j = T_i$ **then**
15: $u.\mathcal{S} := u.\mathcal{S} \setminus \mathcal{T}'$
16: $E := E \setminus \{(v_i, v_j) \mid v_i = u \wedge v_j = T_j,\ T_j \in \mathcal{T}'\}$
17: $\mathcal{D} := \mathcal{D} \setminus \{v_j.k \oplus h(v_i.k, v_j.l) \mid v_i = u \wedge v_j = T_j,\ T_j \in \mathcal{T}'\}$
18: **Subscribe**(u,T_i)

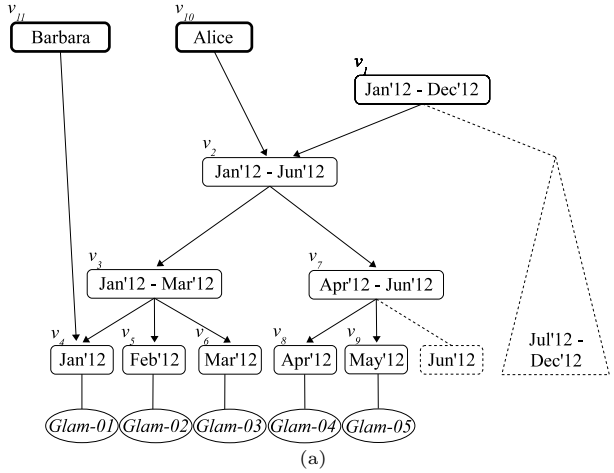
Fig. 5: Pseudocode of procedure **Subscribe**

5.2 New Subscription

Both new and existing users can subscribe to a service for a time window at any point in time (i.e., before the beginning, during, or even after the expiration of the window). Figure 5 illustrates procedure **Subscribe** that manages new subscriptions. The procedure takes a user u and a time window T as input and works as follows. If u is a new user, the procedure creates a vertex v_u representing u, her encryption key $v_u.k$, and public label $v_u.l$ (lines 1-6). Otherwise, the procedure identifies the vertex v_u representing the user in G (line 7). The procedure then inserts T into $u.\mathcal{S}$, calls function **Get_Vertex** on T so that the vertex v_T representing T and its ancestors are possibly added to the graph, and inserts edge (v_u, v_T) in the user and resource graph, publishing the corresponding token (lines 8-11). Through this token, the user can directly derive from her key the key of the time window to which she is subscribing.

To keep the number of tokens under control, the procedure verifies whether the set $u.\mathcal{S}$ of subscriptions includes all the time windows directly dominated by T_i that in turn directly dominates T in $\mathcal{H}_\mathcal{T}$ (e.g., a user may be subscribed for three issues of a magazine that correspond to a trimester). In this case, instead of maintaining a token from u to all the direct descendants of T_i, it is possible to replace them with a single token from vertex u to T_i. To this purpose, procedure **Subscribe** identifies the direct ancestor T_i of the time window T to which u is subscribing and checks if $u.\mathcal{S}$ includes all the descendants $T_j, \ldots, T_l$ of T_i (lines 12-14). In this case, it removes $T_j, \ldots, T_l$ from $u.\mathcal{S}$, the edges connecting v_u to the vertices representing them, and the corresponding tokens (lines 15-17). The procedure then recursively calls itself to subscribe u to T_i to possibly propagate up in the graph this factorization (line 18).

Example 4. Consider the user and resource graph, published resources, and token catalog in Figure 4, and assume that *Alice* renews her subscription to *Glam-*



(a)

Id	Enc_Resource
1	E(Glam-01,$v_4.k$)
2	E(Glam-02,$v_5.k$)
3	E(Glam-03,$v_6.k$)
4	E(Glam-04,$v_8.k$)
5	E(Glam-05,$v_9.k$)

(b)

From	To	Token	
v_1	v_2	$v_2.k \oplus h(v_1.k, v_2.l)$	
v_2	v_3	$v_3.k \oplus h(v_2.k, v_3.l)$	
v_2	v_7	$v_7.k \oplus h(v_2.k, v_7.l)$	
v_3	v_4	$v_4.k \oplus h(v_3.k, v_4.l)$	
v_3	v_5	$v_5.k \oplus h(v_3.k, v_5.l)$	
v_3	v_6	$v_6.k \oplus h(v_3.k, v_6.l)$	
v_7	v_8	$v_8.k \oplus h(v_7.k, v_8.l)$	
v_7	v_9	$v_9.k \oplus h(v_7.k, v_9.l)$	
~~v_{10}~~	~~v_3~~	~~$v_3.k \oplus h(v_{10}.k, v_3.l)$~~	
v_{10}	v_2	$v_2.k \oplus h(v_{10}.k, v_2.l)$	*
v_{11}	v_4	$v_4.k \oplus h(v_{11}.k, v_4.l)$	

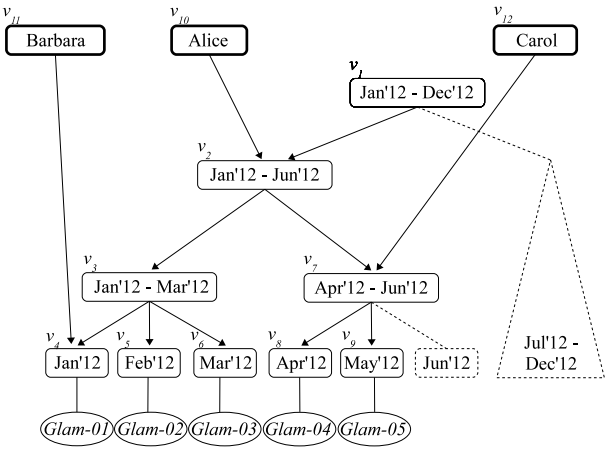
(c)

Fig. 6: User and resource graph (a), published resources (b), and token catalog after *Alice* subscribes for [Apr'12,Jun'12] (c)

our for trimester [Apr'12,Jun'12]. Since both *Alice* and [Apr'12,Jun'12] are already in the graph (vertices v_{10} and v_7, respectively), procedure **Subscribe** only inserts edge (v_{10},v_7) and publishes the corresponding token. Renewing her subscription, *Alice* is now subscribed for the first semester of year 2012. Procedure **Subscribe** factorizes the two subscriptions for [Jan'12,Mar'12] and [Apr'12,Jun'12] in a unique subscription for [Jan'12,Jun'12]. Figure 6 illustrates the resulting user and resource graph, published resources, and token catalog (removed tokens are ~~crossed out~~). Assume now that *Carol* joins the system and subscribes for [Apr'12,Jun'12]. Procedure **Subscribe** first inserts vertex v_{12} representing *Carol* in the graph, and communicates her the corresponding key. It then inserts edge (v_{12},v_7) in the graph. Figure 7 illustrates the resulting user and resource graph, published resources, and token catalog.

5.3 Withdrawal from a Subscription

As our system provides high flexibility in defining the time windows available for subscription, withdrawal from a subscription represents an exception in the working of the system and must be managed as a special case. In fact, no action is needed when a subscription naturally expires. When a user withdraws from a subscription for time window [t^s,t^e], starting from time instant t, the resource provider must guarantee that: *i)* she cannot access the resources with timestamp in (t,t^e] (backward protection), and *ii)* she continues to access the resources with timestamp in [t^s,t]. For instance, consider Example 4. In May'12 *Alice* could decide to withdraw from her subscription for the first semester of year 2012. In this case, she should not be able to decrypt the issue of June of the magazine, while she will continue to access the issues of January, February, March, April, and May. Clearly, a user can withdraw from her subscription at time t only



(a)

Id	Enc_Resource
1	E(Glam-01,$v_4.k$)
2	E(Glam-02,$v_5.k$)
3	E(Glam-03,$v_6.k$)
4	E(Glam-04,$v_8.k$)
5	E(Glam-05,$v_9.k$)

(b)

From	To	Token
v_1	v_2	$v_2.k \oplus h(v_1.k, v_2.l)$
v_2	v_3	$v_3.k \oplus h(v_2.k, v_3.l)$
v_2	v_7	$v_7.k \oplus h(v_2.k, v_7.l)$
v_3	v_4	$v_4.k \oplus h(v_3.k, v_4.l)$
v_3	v_5	$v_5.k \oplus h(v_3.k, v_5.l)$
v_3	v_6	$v_6.k \oplus h(v_3.k, v_6.l)$
v_7	v_8	$v_8.k \oplus h(v_7.k, v_8.l)$
v_7	v_9	$v_9.k \oplus h(v_7.k, v_9.l)$
v_{10}	v_2	$v_2.k \oplus h(v_{10}.k, v_2.l)$
v_{11}	v_4	$v_4.k \oplus h(v_{11}.k, v_4.l)$
v_{12}	v_7	$v_7.k \oplus h(v_{12}.k, v_7.l)$ *

(c)

Fig. 7: User and resource graph (a), published resources (b), and token catalog after *Carol* subscribes for [Apr'12,Jun'12] (c)

if no resource with timestamp in $(t,t^e]$ has been published yet, since otherwise she could have accessed it before withdrawal. To guarantee that withdrawals are transparent for all the users and cause a limited overhead to the resource provider, our approach avoids re-keying and re-encryption operations.

Figure 8 illustrates procedure **Withdraw_Subscription**, which takes a user u and a time instant t as input, and updates the user and resource graph. The procedure first identifies the vertex v_u representing the user in G and the time window $[t^s,t^e]$ in $u.\mathcal{S}$ that includes t (lines 1-2). If such a time window does not exist or if at least a resource with timestamp in $(t,t^e]$ has been published, the procedure terminates notifying the problem to the resource provider (line 3). Otherwise, procedure **Withdraw_Subscription** removes the subscription by first substituting $[t^s,t^e]$ with $[t^s,t]$ in $u.\mathcal{S}$ (line 4). Since user u already knows the keys of the vertices along the path from vertex $[t^s,t^e]$ to t if they are represented in the user and resource graph, the resource provider must guarantee that all the resources with a timestamp following t will be encrypted with a key that is not derivable from the keys along this path. To this purpose, the procedure updates the time window $[t_i^s,t_i^e]$ that each of these vertices represents by setting t_i^e to t, creates a new set of vertices representing the time windows that has been changed, and connects them in a path of the user and resource graph. Also, the procedure inserts an edge between each new vertex $[t_i^s,t_i^e]$ to vertex $[t_i^s,t]$ since $[t_i^s,t_i^e]$ clearly dominates $[t_i^s,t]$. Finally, for each user u such that $[t_i^s,t_i^e] \in u.\mathcal{S}$, the procedure substitutes the token (and corresponding edge) between u and $[t_i^s,t]$ (i.e., the vertex that represented $[t_i^s,t_i^e]$ before the change performed by procedure **Withdraw_Subscription**) with the token (and corresponding edge) between u and the new vertex representing $[t_i^s,t_i^e]$, to preserve her ability to derive all the keys of the time windows dominated by $[t_i^s,t_i^e]$.

WITHDRAW_SUBSCRIPTION(u,t)
1: let $v_u \in V$ be the vertex with $v_u = u$
2: let $T=[t^s,t^e] \in u.\mathcal{S}$ s.t. $t^s \leq t \leq t^e$
3: **if** $T=$NULL $\vee$ ($\exists r \in \mathcal{R}$ s.t. $t<r.t \leq t^e$) **then exit**
4: $u.\mathcal{S} := u.\mathcal{S} \setminus \{T\} \cup \{[t^s,t]\}$ /* update the time window in user subscriptions */
5: let $v_T \in V$ be the vertex with $v_T = T$
6: **while** $t^e \neq t \wedge t^s \neq t^e \wedge T \in V$ **do** /* visit the path from T to $[t,t]$ */
7: $T_{new} := [t^s,t^e]$
8: $v_T := [t^s,t]$ /* update the label of the vertex */
9: $v_{new} :=$ **Get_Vertex**(T_{new}) /* create a vertex representing T_{new} */
10: $E := E \cup \{(v_{new},v_T)\}$ /* $[t^s,t^e]$ dominates $[t^s,t]$ */
11: $\mathcal{D} := \mathcal{D} \cup \{v_T.k \oplus h(v_{new}.k, v_T.l)\}$
12: **for each** (v_u,v_T) s.t. $v_u \in \mathcal{U} \setminus \{u\}$ **do** /* update users' subscriptions */
13: $E := E \cup \{(v_u,v_{new})\} \setminus \{(v_u,v_T)\}$
14: $\mathcal{D} := \mathcal{D} \cup \{v_{new}.k \oplus h(v_u.k, v_{new}.l)\} \setminus \{v_T.k \oplus h(v_u.k, v_T.l)\}$
15: let $T=[t^s,t^e] \in \mathcal{T}$ s.t. $T_{new} \succeq T \wedge t^s \leq t \leq t^e \wedge \nexists T_j$: $T_{new} \succeq T_j \succeq T$, $T_j \neq T_{new} \neq T$
16: let $v_T \in V$ be the vertex with $v_T = T$

Fig. 8: Pseudocode of procedure **Withdraw_Subscription**

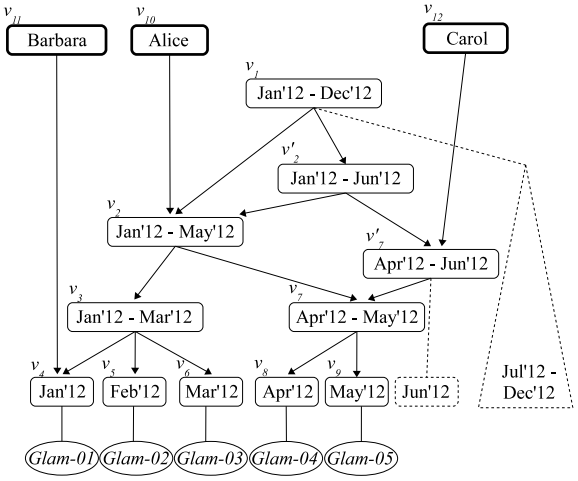
Note that the keys along the path from T to t, whose time windows have been updated by procedure **Withdraw_Subscription**, are not affected. Therefore, users who have already computed these keys can still use their local copy. The number of additional vertices and edges in the user and resource graph is limited and is at most h-1 and 2(h-1), respectively, where h is the height of the time hierarchy. The number of updated edges is $|\mathcal{U}|$-1 in the worst case.

Example 5. Consider the user and resource graph, published resources, and token catalog in Figure 7, and assume that *Alice* withdraws from her subscription in May'12. Procedure **Withdraw_Subscription** updates her subscription for [Jan'12,Jun'12] to [Jan'12,May'12], and visits the path from vertex v_2 (representing [Jan'12,Jun'12]) to the vertex representing [May'12,May'12]. First, it visits vertex v_2, updates its time window to [Jan'12,May'12], creates a new vertex v_2' for time window [Jan'12,Jun'12], and inserts edge (v_2',v_2) in the user and resource graph. The procedure executes the same operations when visiting v_7. Since *Carol* should still be able to access all the issues of *Glamour* published in [Apr'12,Jun'12], the procedure substitutes edge (v_{12},v_7) with edge (v_{12},v_7'). From her key *Alice* can derive, after this update, the keys used to encrypt the issues published in [Jan'12,May'12], while *Carol* can still derive keys used to encrypt issues published in [Apr'12,Jun'12]. Figure 9 illustrates the user and resource graph, published resources, and token catalog after *Alice*'s withdrawal.

5.4 Correctness

The procedures described in this section correctly enforce changes to the subscription-based policy. This is formally stated by the following theorem, whose proof is omitted from the paper for space constraints.

Theorem 2 (Correct enforcement of policy updates). *Let $\mathcal{H}_{\mathcal{T}}(\mathcal{T},\succeq)$ be a time hierarchy on time domain $(\mathcal{TS},\leq)$, $\mathcal{U}$ be a set of users with $u.\mathcal{S} \subseteq \mathcal{T}$ for all*



(a)

Id	Enc_Resource
1	E(Glam-01,$v_4.k$)
2	E(Glam-02,$v_5.k$)
3	E(Glam-03,$v_6.k$)
4	E(Glam-04,$v_8.k$)
5	E(Glam-05,$v_9.k$)

(b)

From	To	Token	
v_1	v_2	$v_2.k \oplus h(v_1.k, v_2.l)$	
v_1	v'_2	$v'_2.k \oplus h(v_1.k, v'_2.l)$	
v_2	v_3	$v_3.k \oplus h(v_2.k, v_3.l)$	
v_2	v_7	$v_7.k \oplus h(v_2.k, v_7.l)$	
v'_2	v_2	$v_2.k \oplus h(v'_2.k, v_2.l)$	*
v'_2	v'_7	$v'_7.k \oplus h(v'_2.k, v'_7.l)$	*
v_3	v_4	$v_4.k \oplus h(v_3.k, v_4.l)$	
v_3	v_5	$v_5.k \oplus h(v_3.k, v_5.l)$	
v_3	v_6	$v_6.k \oplus h(v_3.k, v_6.l)$	
v_7	v_8	$v_8.k \oplus h(v_7.k, v_8.l)$	
v_7	v_9	$v_9.k \oplus h(v_7.k, v_9.l)$	
v'_7	v_7	$v_7.k \oplus h(v'_7.k, v_7.l)$	*
v_{10}	v_2	$v_2.k \oplus h(v_{10}.k, v_2.l)$	
v_{11}	v_4	$v_4.k \oplus h(v_{11}.k, v_4.l)$	
~~v_{12}~~	~~v_7~~	~~$v_7.k \oplus h(v_{12}.k, v_7.l)$~~	
v_{12}	v'_7	$v'_7.k \oplus h(v_{12}.k, v'_7.l)$	*

(c)

Fig. 9: User and resource graph (a), published resources (b), and token catalog after *Alice* withdraws from her subscription in May'12 (c)

$u \in \mathcal{U}$, $\mathcal{R}$ be a set of resources with $r.t \in \mathcal{TS}$ for all $r \in \mathcal{R}$, and $G(V,E)$ be the user and resource graph over $\mathcal{U}$, $\mathcal{R}$, and $\mathcal{H}_{\mathcal{T}}$.

1. *Procedure* **Publish_Resource**(r) *generates a user and resource graph that correctly enforces the subscription-based policy on $\mathcal{U}$ and $\mathcal{R} \cup \{r\}$.*
2. *Procedure* **Subscribe**(u,T) *generates a user and resource graph that correctly enforces the subscription-based policy on $\mathcal{U} \cup \{u\}$ and $\mathcal{R}$, with $u.\mathcal{S} \cup \{T\}$.*
3. *Procedure* **Withdraw_Subscription**(u,t) *generates a user and resource graph that correctly enforces the subscription-based policy on $\mathcal{U}$ and $\mathcal{R}$, with $u.\mathcal{S} \setminus \{[t^s,t^e]\} \cup \{[t^s,t]\}$.*

6 Related Work

Previous work close to ours is in the area of data outsourcing [18], where many approaches focused on efficient query evaluation at the external server (e.g., [8, 12, 21]), and on guaranteeing data integrity and authenticity (e.g., [15]). Recent works have also addressed access control enforcement (e.g., [10, 14, 23]), but these approaches are not suited for the scenario considered in this paper, as they assume the sets of users, resources, and authorizations not to change frequently.

The problem of enforcing access control policies with time-based restrictions has been widely studied (e.g., [4, 19]). However, these works restrict access to resources depending on the time when the access is requested. Recently, time-based access control restrictions have been enforced also in the data outsourcing

scenario, by integrating them in the key derivation process (e.g., [2, 3, 7]). The solutions in [2, 3] allow users to derive encryption keys only within the time windows for which they are authorized. The approach in [7] proposes instead a more general model for enforcing any interval-based restriction (e.g., time and space). These solutions mainly focus on the security of key derivation and on minimizing the number of edges in the key derivation graph. Our proposal is instead aimed at correctly enforcing a subscription-based policy and at guaranteeing transparency for users in subscription management and resource publishing.

Our work may bring some resemblance with access control in publish/subscribe systems, characterized by a set of users who publish events, a set of users who subscribe to the system declaring their interests, and a service responsible to deliver published events to the users whose interests match with the event attributes [11, 20]. However, in publish/subscribe systems the access control policy depends on some properties related to the events. Also, publish/subscribe systems typically rely on a trusted party that can access events and enforce access restrictions.

Another related but different line of work addresses the problem of enforcing time-based restrictions to users when accessing broadcasting services (e.g., [6, 22]). These approaches are not applicable in our scenario where we assume to publish persistent resources as opposed to data streams.

7 Conclusions

We proposed an approach for effectively restricting access to published resources based on the subscriptions of the users to a service. Our solution is based on selective encryption so that the encrypted resources self-enforce the subscription-based restrictions. A key derivation structure is also used for easily enforcing changes in the subscription-based policy due to the addition of new users and resources, and to the withdrawal of users from their subscriptions.

Acknowledgements. We would like to thank Pierangela Samarati for discussions, suggestions, and comments. This work was partially supported by the Italian Ministry of Research within the PRIN 2008 project "PEPPER" (2008SY2PH4). The work of Sushil Jajodia was partially supported by the National Science Foundation under grants CCF-1037987 and CT-20013A.

References

1. M. Atallah, M. Blanton, N. Fazio, and K. Frikken. Dynamic and efficient key management for access hierarchies. *ACM TISSEC*, 12(3):18:1–18:43, January 2009.
2. M. Atallah, M. Blanton, and K. Frikken. Incorporating temporal capabilities in existing key management schemes. In *Proc. of ESORICS 2007*, Dresden, Germany, September 2007.
3. G. Ateniese, A. De Santis, A.L. Ferrara, and B. Masucci. Provably-secure time-bound hierarchical key assignment schemes. *Journal of Cryptology*, 25(2):243–270, April 2012.

4. E. Bertino, C. Bettini, E. Ferrari, and P. Samarati. An access control model supporting periodicity constraints and temporal reasoning. *ACM TODS*, 23(3):231–285, September 1998.
5. C. Bettini, C. Dyreson, W. Evans, R. Snodgrass, and X. Wang. A glossary of time granularity concepts. In O. Etzion, S. Jajodia, and S. Sripada, editors, *Temporal Databases: Research and Practice*, volume 1399 of *LNCS*. Springer-Verlag, 1998.
6. M. Blanton and K. Frikken. Efficient multi-dimensional key management in broadcast services. In *Proc. of ESORICS 2010*. Athens, Grece, September 2010.
7. J. Crampton. Practical and efficient cryptographic enforcement of interval-based access control policies. *ACM TISSEC*, 14(1):14:1–14:30, June 2011.
8. 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 CCS 2003*, Washington, DC, USA, October 2003.
9. S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. A data outsourcing architecture combining cryptography and access control. In *Proc. of CSAW 2007*, Fairfax, VA, USA, November 2007.
10. S. De Capitani di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Encryption policies for regulating access to outsourced data. *ACM TODS*, 35(2):12:1–12:46, April 2010.
11. P.T. Eugster, P. A. Felber, R. Guerraoui, and A. Kermarrec. The many faces of publish/subscribe. *ACM CSUR*, 35(2):114–131, June 2003.
12. H. Hacigümüs, B. Iyer, S. Mehrotra, and C. Li. Executing SQL over encrypted data in the database-service-provider model. In *Proc. of the SIGMOD 2002*, Madison, WI, USA, June 2002.
13. R. Jhawar, V. Piuri, and M. D. Santambrogio. A comprehensive conceptual system-level approach to fault tolerance in cloud computing. In *Proc. of IEEE SysCon 2012*, Vancouver, BC, Canada, March 2012.
14. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *Proc. of VLDB 2003*, Berlin, Germany, September 2003.
15. E. Mykletun, M. Narasimha, and G. Tsudik. Authentication and integrity in outsourced databases. *ACM TOS*, 2(2):107–138, May 2006.
16. S. Preda, N. Cuppens-Boulahia, F. Cuppens, and L. Toutain. Architecture-aware adaptive deployment of contextual security policies. In *Proc. of ARES 2012*, Krakow, Poland, 2010.
17. P. Samarati and S. De Capitani di Vimercati. Access control: Policies, models, and mechanisms. In R. Focardi and R. Gorrieri, editors, *Foundations of Security Analysis and Design*, volume 2171 of *LNCS*. Springer-Verlag, 2001.
18. P. Samarati and S. De Capitani di Vimercati. Data protection in outsourcing scenarios: Issues and directions. In *Proc. of ASIACCS 2010*, China, April 2010.
19. M. Toahchoodee and I. Ray. On the formalization and analysis of a spatio-temporal role-based access control model. *JCS*, 19(3):399–452, May 2011.
20. C. Wang, A. Carzaniga, D. Evans, and A. Wolf. Security issues and requirements for internet-scale publish-subscribe systems. In *Proc. of HICSS 2002*, Big Island, HI, USA, January 2002.
21. H. Wang and L.V.S. Lakshmanan. Efficient secure query evaluation over encrypted XML databases. In *Proc. of VLDB 2006*, Seoul, Korea, September 2006.
22. C.K. Wong, M. Gouda, and S.S. Lam. Secure group communications using key graphs. *IEEE/ACM TON*, 8(1):16–30, February 2000.
23. S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *Proc. of INFOCOM 2010*, San Diego, CA, USA, March 2010.