J. Cryptol. (2012) 25: 243–270
DOI: 10.1007/s00145-010-9094-6

Journal of
CRYPTOLOGY

Provably-Secure Time-Bound Hierarchical Key Assignment Schemes

Giuseppe Ateniese
The Johns Hopkins University, Baltimore, MD 21218, USA
ateniese@cs.jhu.edu

Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci
Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy
ads@dia.unisa.it; ferrara@dia.unisa.it; masucci@dia.unisa.it

Communicated by Matt Paterson

Received 23 October 2008
Online publication 23 November 2010

Abstract. A *time-bound hierarchical key assignment scheme* is a method to assign time-dependent encryption keys to a set of classes in a partially ordered hierarchy, in such a way that each class can compute the keys of all classes lower down in the hierarchy, according to temporal constraints.

In this paper we design and analyze time-bound hierarchical key assignment schemes which are provably-secure and efficient. We consider two different goals: security with respect to *key indistinguishability* and against *key recovery*. Moreover, we distinguish security against *static* and *adaptive* adversarial behaviors. We explore the relations between all possible combinations of security goals and adversarial behaviors and, in particular, we prove that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries. Finally, we propose two different constructions for time-bound key assignment schemes. The first one is based on symmetric encryption schemes, whereas the second one makes use of bilinear maps. Both constructions support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

Key words. Access control, Key assignment, Time-bound, Provable security, Efficiency.

1. Introduction

The *access control problem* deals with the ability to ensure that only authorized users of a computer system are given access to some sensitive resources. According to their competencies and responsibilities, users are organized in a hierarchy formed by a certain number of disjoint classes, called *security classes*. A hierarchy arises from the fact that some users have more access rights than others. In the real world there are several examples of hierarchies where access control is required. For example, within a hospital

system, doctors can access data concerning their patients such as diagnosis, medication prescriptions, and laboratory tests, whereas researchers can be limited to consult anonymous clinical information for studies. Similar cases abound in other areas, particularly in the government and military.

A *hierarchical key assignment scheme* is a method to assign an encryption key and some private information to each class in the hierarchy. The encryption key will be used by each class to protect its data by means of a symmetric cryptosystem, whereas the private information will be used by each class to compute the keys assigned to all classes lower down in the hierarchy. This assignment is carried out by a central authority, the Trusted Authority (TA), which is active only at the distribution phase. Such schemes assign keys that never expire and new keys are generated only after inserting or deleting classes in the hierarchy. However, in practice, it is likely that a user may be assigned to a certain class for only a certain period of time. In such cases, users need a different key for each time period, which implies that the key derivation procedure should also depend on the time period as well as the hierarchy of the classes. Once a time period expires, users in a class should not be able to access any subsequent keys if they are not authorized to do so. There are several applications requiring time-based access control. For example, a web-based electronic newspaper company could offer several types of subscription packages, covering different topics. Each user may decide to subscribe to one package for a certain period of time (e.g., a week, a month, or a year). Subscription packages could be structured to form a partially ordered hierarchy where leaf nodes represent different topics. For each time period, an encryption key is then assigned to each leaf node in the hierarchy. This key is then computed by each user that subscribes to that package and for that period of time. A similar solution was employed by Bertino et al. [11], who showed how to control access to an XML document according to temporal constraints. Hierarchical key assignment schemes which also consider temporal constraints are called *time-bound hierarchical key assignment schemes*.

1.1. *Related Works*

Akl and Taylor [2] first proposed an elegant hierarchical key assignment scheme. In their scheme each class is assigned a key that can be used, along with some public parameters, to compute the key assigned to any class lower down in the hierarchy. Subsequently, many researchers have proposed schemes that either have better performances or allow insertion and deletion of classes in the hierarchy (e.g., [3,29,31,34,35,37,40]) or support more general access control policies [21,36,48]. Despite the large number of proposed schemes, many of them lack a formal security proof and have been shown to be insecure against collusive attacks [17,41,46,48], whereby two or more classes collude to compute a key to which they are not entitled. A detailed classification of many schemes in the literature, as well as an evaluation of their merits can be found in [19].

Atallah et al. [3] first addressed the problem of formalizing security requirements for hierarchical key assignment schemes. A scheme is *provably-secure* under a complexity assumption if the existence of an adversary A breaking a scheme is equivalent to the existence of an adversary B breaking the computational assumption. Atallah et al. [3] also proposed the first provably-secure constructions based on pseudorandom functions and a symmetric encryption scheme. Constructions for provably-secure key assignment schemes, improving those in [3], have been proposed in [23]. Tzeng [43] first addressed

the problem of designing *time-bound* hierarchical key assignment schemes. However, his scheme is very costly since each user must perform expensive computations in order to compute a legitimate key. Most importantly, Tzeng's scheme has been shown to be insecure against collusive attacks [50]. Subsequently, other time-bound hierarchical key assignment schemes were proposed in [18,30,47], and later shown to be insecure against collusive attacks [7,22,42,49]. Afterwards, Wang and Laih [45] and Tzeng [44] have shown how to construct a time-bound hierarchical key assignment scheme starting from the Akl–Taylor scheme. However, since they did not formalize the definition of security and the adversarial model, it was not clear under which assumption their schemes could have been considered provably-secure. Subsequently, D'Arco et al. [20] have shown that these constructions are secure under the *RSA assumption*, provided that the parameters of the Akl–Taylor scheme are properly chosen.

1.2. *Our Contributions*

In this paper we design and analyze time-bound hierarchical key assignment schemes which are provably-secure and efficient. We consider two different security goals: security with respect to *key indistinguishability* and security against *key recovery*. Security with respect to key indistinguishability formalizes the requirement that the adversary is not able *to learn any information* about a key that it should not have access to, i.e., it is not able to distinguish it from a random string having the same length. On the other hand, security against key recovery corresponds to the requirement that an adversary is not able to *compute* a key that it should not have access to. We also provide definition of security with respect to *static* adversaries and *adaptive* ones. Informally, a static adversary chooses at random the class to attack, whereas the adaptive one makes its choices on the ground of the information he progressively learns.

The two above security goals were first introduced by Atallah et al. [3] with respect to adaptive adversaries for hierarchical key assignment schemes. We extend the definitions in [3] to include temporal constraints. Moreover, we also consider *static* adversaries and characterize the four security notions determined by all possible combinations of goals and adversarial behaviors, by exploring the relations between the resulting definitions. In particular, we prove that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries. Hence, it is sufficient to consider static adversaries in order to prove the security of a scheme with respect to a more realistic scenario modeled by adaptive adversaries. This allows to simplify the security analysis and to provide less complicated security proofs. Notice that relations and separations among the security notions also apply to key assignment schemes without temporal constraints. Indeed, these schemes can be seen as time-bound schemes with a single period of time.

Finally, we propose two different constructions for time-bound key assignment schemes. The first one, which we call *Two-Level Encryption-Based Construction* (TLEBC), is based on symmetric encryption schemes, whereas the second one, which we refer to as *Two-Level Pairing-Based Construction* (TLPBC), makes use of bilinear maps. Both constructions support updates to the access hierarchy with local changes to the public information without requiring any private information to be re-distributed. Compared with other proposals, these schemes provide a more efficient procedure to compute the keys assigned to all classes lower down in the hierarchy.

A preliminary version of our results appeared in [7]. Our model and security definitions have been adopted to develop new provably-secure constructions for time-bound hierarchical key assignment schemes [5,24]. Specifically, such constructions use as a building block any provably-secure hierarchical key assignment scheme without temporal constraints.

2. Time-Bound Hierarchical Key Assignment Schemes

Consider a set of users divided into a number of disjoint classes, called *security classes*. A binary relation $\preceq$ partially orders the set of classes V. The poset $(V, \preceq)$ is called a *partially ordered hierarchy*. For any two classes u and v, the notation $u \preceq v$ is used to indicate that the users in v can access u's data. Clearly, since v can access its own data, $v \preceq v$ holds, for any $v \in V$. We denote by A_v the set $\{u \in V : u \preceq v\}$, for any $v \in V$. The partially ordered hierarchy $(V, \preceq)$ can be represented by the directed graph $G^* = (V, E^*)$, where each class corresponds to a vertex in the graph and there is an edge from class v to class u if and only if $u \preceq v$. We denote by $G = (V, E)$ the *minimal representation* of the graph G^*, that is, the directed acyclic graph corresponding to the *transitive and reflexive reduction* of the graph $G^* = (V, E^*)$. Such a graph G has the same transitive and reflexive closure of G^*, i.e., there is a path (of length greater than or equal to zero) from v to u in G if and only if there is the edge (v, u) in E^*. Aho et al. [1] showed that every directed graph has a transitive reduction which can be computed in polynomial time and that such a reduction is unique for directed acyclic graphs. In the following we denote by Γ a family of graphs corresponding to partially ordered hierarchies. For example, Γ could be the family of the rooted trees, the family of the d-dimensional graphs [4,23,40], etc.

Let $T = (t_1, \ldots, t_{|T|})$ be the sequence composed of distinct time periods. In the following we denote by $t \in T$ the fact that the time period t belongs to the sequence T. Each user may belong to a class for a certain non-empty contiguous subsequence λ of T. Let $\mathcal{P}$ be the set of all non-empty contiguous subsequences of T. Such a set is called the *interval-set* over T. A *time-bound hierarchical key assignment scheme* is a method to assign a private information $s_{v,\lambda}$ to each class $v \in V$ for each time sequence $\lambda \in \mathcal{P}$ and an encryption key $k_{u,t}$ to each class $u \in V$ for each time period $t \in T$. The generation and distribution of the private information and keys is carried out by a trusted third party, the TA, which is connected to each class by means of a secure channel. The encryption key $k_{u,t}$ can be used by users belonging to class u in time period t to protect their sensitive data by means of a symmetric cryptosystem, whereas the private information $s_{v,\lambda}$ can be used by users belonging to class v for the time sequence λ to compute the key $k_{u,t}$ for any class $u \in A_v$ and each time period $t \in \lambda$.

A time-bound hierarchical key assignment scheme for a family of graphs Γ corresponding to partially ordered hierarchies is defined as follows.

Definition 1. A *time-bound hierarchical key assignment scheme* for Γ is a pair of algorithms (Gen, Der) satisfying the following conditions:

1. The *information generation algorithm Gen* is probabilistic polynomial time. It takes as inputs the security parameter 1^τ, a graph $G = (V, E)$ in Γ, and the

interval-set $\mathcal{P}$ over a sequence of distinct time periods T, and produces as outputs
(a) a private information $s_{u,\lambda}$, for any class $u \in V$ and any time sequence $\lambda \in \mathcal{P}$;
(b) a key $k_{u,t}$, for any class $u \in V$ and any time period $t \in T$;
(c) a public information *pub*.
We denote by (s, k, pub) the output of the algorithm *Gen* where s and k denote the sequences of private information and of keys, respectively.

2. The *key derivation algorithm Der* is deterministic polynomial time. It takes as inputs the security parameter 1^τ, a graph $G = (V, E)$ in Γ, the interval-set $\mathcal{P}$ over a sequence of distinct time periods T, two classes u and v such that $v \in A_u$, a time sequence $\lambda \in \mathcal{P}$, the private information $s_{u,\lambda}$ assigned to class u for the time sequence λ, a time period $t \in \lambda$, and the public information *pub*, and produces as output the key $k_{v,t}$ assigned to class v at time period t.

   We require that for each class $u \in V$, each class $v \in A_u$, each time sequence $\lambda \in \mathcal{P}$, each time period $t \in \lambda$, each private information $s_{u,\lambda}$, each key $k_{v,t}$, each public information *pub* which can be computed by *Gen* on inputs 1^τ, G, and $\mathcal{P}$, it holds that

$$Der\big(1^\tau, G, \mathcal{P}, u, v, \lambda, s_{u,\lambda}, t, pub\big) = k_{v,t}.$$

Notice that in Definition 1 we have not specified the structure of the public information *pub* and of the graph G. In order to improve the efficiency of key derivation, *pub* and G could be structured in such a way that, whenever class u performs key derivation to compute the key of a class $v \in A_u$, it does not need to input the algorithm *Der* with the whole *pub* and G, but only those parts of them involved in the computation.

2.0.1. *Security Requirement*

Unauthorized users should not be able to compute keys to which they have no access right. More precisely, for each class $u \in V$ and each time period $t \in T$, the key $k_{u,t}$ should be protected against a coalition of users belonging to each class v such that $u \notin A_v$ in all time periods, and users belonging to each class w such that $u \in A_w$ in all time periods but t. We denote by $F_{u,t}$ the set $\{(v, \lambda) \in V \times \mathcal{P} : u \notin A_v \text{ or } t \notin \lambda\}$, corresponding to all users which are not allowed to compute the key $k_{u,t}$.

2.0.2. *Evaluation Criteria*

The efficiency of a time-bound hierarchical key assignment scheme is evaluated according to different parameters: storage requirements, which correspond to the amount of secret data that need to be distributed and stored by the users and the amount of data that need to be made public; the complexity of both key derivation and key update procedures (it is desirable that updates to the access hierarchy require only local changes to the public information and do not need any private information to be re-distributed); the computational assumption on which the security of the scheme holds (it is desirable to employ standard assumptions).

3. Notions of Security

We use the standard notation to describe probabilistic algorithm and experiments following [28]. If $A(\cdot,\cdot,\ldots)$ is any probabilistic algorithm then $a \leftarrow A(x, y, \ldots)$ denotes the experiment of running A on inputs $x, y, \ldots$ and letting a be the outcome, the probability being over the coins of A. Similarly, if X is a set then $x \leftarrow X$ denotes the experiment of selecting an element uniformly from X and assigning x this value. If w is neither an algorithm nor a set then $x \leftarrow w$ is a simple assignment statement.

A function $\epsilon : N \rightarrow R$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

We consider two different security goals: with respect to *key indistinguishability* and against *key recovery*. We also provide definitions of security with respect to static and adaptive adversaries.

A *static* adversary, given a class u and a time period t, is allowed to access the private information assigned to all users not allowed to compute the key $k_{u,t}$, as well as all public information. An *adaptive* adversary is first allowed to access all public information as well as all private information of a number of users of its choice; afterwards, it chooses the class u it wants to attack and the time period t for which the attack will be mounted. We explore the relationships of the security notions determined by all possible combinations of goals (key indistinguishability/key recovery) and adversarial behaviors (static/adaptive). In particular, we show whether one notion implies another and vice versa. Figure 1 summarizes our results.

3.0.1. *Security with Respect to Key Indistinguishability*

We first consider the case where there is a *static adversary* $\mathtt{STAT}_{u,t}$, which attacks a class $u \in V$ at a certain time period $t \in T$ and which is able to corrupt *all* users not allowed to compute the key $k_{u,t}$. We define an algorithm $Corrupt_{u,t}$ which, on input the private information s generated by the algorithm *Gen*, extracts the secret values $s_{v,\lambda}$ associated to all pairs $(v, \lambda) \in F_{u,t}$. We denote by *corr* the sequence output by $Corrupt_{u,t}(s)$. The computations performed by the adversary involve all public information generated by the algorithm *Gen*, as well as the private information *corr* held by the corrupted users. Two experiments are considered. In the first one, the adversary is given the key $k_{u,t}$, whereas in the second one it is given a random string ρ having the same length as $k_{u,t}$. It is the adversary's job to determine whether the received

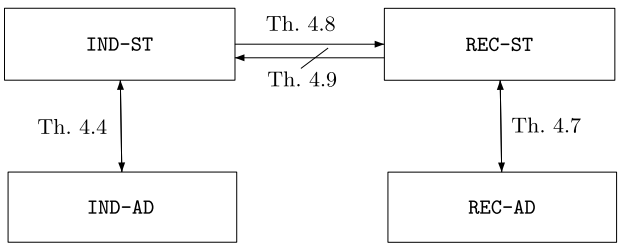

Fig. 1. Hierarchy of security notions for time-bound hierarchical key assignment schemes.

challenge corresponds to $k_{u,t}$ or to a random string. We require that the adversary will succeed with probability only negligibly different from 1/2.

Definition 2 [IND-ST]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $\mathtt{STAT}_{u,t}$ be a static adversary which attacks a class $u \in V$ in a time period $t \in T$. Consider the following two experiments:

Experiment $\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$corr \leftarrow Corrupt_{u,t}(s)$
$d \leftarrow \mathtt{STAT}_{u,t}(1^\tau, G, \mathcal{P}, pub, corr, k_{u,t})$
return d

Experiment $\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$corr \leftarrow Corrupt_{u,t}(s)$
$\rho \leftarrow \{0,1\}^{\text{length}(k_{u,t})}$
$d \leftarrow \mathtt{STAT}_{u,t}(1^\tau, G, \mathcal{P}, pub, corr, \rho)$
return d

The advantage of $\mathtt{STAT}_{u,t}$ is defined as

$$\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P}) = \left| Pr\left[\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P}) = 1\right] - Pr\left[\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P}) = 1\right]\right|.$$

The scheme is said to be *secure in the sense of* IND-ST if, for each graph $G = (V, E)$ in Γ, each sequence of distinct time periods T, each class $u \in V$ and each time period $t \in T$, the function $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$ is negligible, for each static adversary $\mathtt{STAT}_{u,t}$ whose time complexity is polynomial in τ.

Now, consider the case where an *adaptive adversary* ADAPT first gets all public information generated by the algorithm *Gen*, and then chooses, in an adaptive order, a number of users to be corrupted. We assume the existence of an oracle which can provide the adversary with the private information held by the corrupted users. Each adversary's query to the oracle consists of a pair $(v, \lambda) \in V \times \mathcal{P}$, which the oracle answers with the private information $s_{v,\lambda}$. Afterwards, the adversary chooses the class u it wants to attack and the time period t for which the attack will be mounted, among the classes and time periods such that the corresponding key $k_{u,t}$ cannot be computed by the corrupted users. Two experiments are considered. In the first one, the adversary is given the key $k_{u,t}$, whereas in the second one it is given a random string ρ having the same length as $k_{u,t}$. After this stage, the adversary is still allowed to corrupt other users of its choice, among those who cannot compute the key $k_{u,t}$, making queries to the oracle. The adversary is challenged to determine whether the received value corresponds to $k_{u,t}$ or to a random string. We require that the adversary will succeed with probability only negligibly different from 1/2.

Definition 3 [IND-AD]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ be an adaptive

adversary that is given access to the oracle $\mathcal{O}_s(\cdot)$ during both stages of the attack, where s is the private information computed by *Gen*. Consider the following two experiments:

Experiment $\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$(u, t, state) \leftarrow \mathtt{ADAPT}_1^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub)$
$d \leftarrow \mathtt{ADAPT}_2^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub, u, t, state, k_{u,t})$
return d

Experiment $\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$(u, t, state) \leftarrow \mathtt{ADAPT}_1^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub)$
$\rho \leftarrow \{0,1\}^{\text{length}(k_{u,t})}$
$d \leftarrow \mathtt{ADAPT}_2^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub, u, t, state, \rho)$
return d

It is required that the pair (u, t) output by $\mathtt{ADAPT}_1$ is such that $(v, \lambda) \in F_{u,t}$, for any pair (v, λ) already queried to the oracle $\mathcal{O}_s(\cdot)$. Moreover, it is also required that $\mathtt{ADAPT}_2$ never queries the oracle $\mathcal{O}_s(\cdot)$ on a pair $(v, \lambda) \in V \times \mathcal{P}$ such that $u \in A_v$ and $t \in \lambda$. The advantage of ADAPT is defined as

$$\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P}) = \big|Pr[\mathbf{Exp}^{\mathtt{IND}-1}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P}) = 1] - Pr\big[\mathbf{Exp}^{\mathtt{IND}-0}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P}) = 1\big]\big|.$$

The scheme is said to be *secure in the sense of* IND-AD if for each graph $G = (V, E)$ in Γ and each sequence of distinct time periods T, the function $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P})$ is negligible, for each adaptive adversary ADAPT whose time complexity is polynomial in τ.

3.0.2. *Security Against Key Recovery*

We first consider the case where there is a *static adversary* $\mathtt{STAT}_{u,t}$ which wants to *compute* the key assigned to a class $u \in V$ at a certain time period $t \in T$. As done before, we denote by *corr* the sequence output by the algorithm $Corrupt_{u,t}$, on input the private information s generated by the algorithm *Gen*. The adversary, on input all public information generated by the algorithm *Gen*, as well as the private information *corr*, outputs a string $k'_{u,t}$ and succeeds whether $k'_{u,t} = k_{u,t}$. We require that the adversary will succeed with probability only negligibly different from $1/2^{\text{length}(k_{u,t})}$.

Definition 4 [REC-ST]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $\mathtt{STAT}_{u,t}$ be a static adversary which attacks a class u in a time period t. Consider the following experiment:

Experiment $\mathbf{Exp}^{\mathtt{REC}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$corr \leftarrow Corrupt_{u,t}(s)$
$k'_{u,t} \leftarrow \mathtt{STAT}_{u,t}(1^\tau, G, \mathcal{P}, pub, corr)$
return $k'_{u,t}$

The advantage of $\mathtt{STAT}_{u,t}$ is defined as

$$\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P}) = Pr\big[k'_{u,t} = k_{u,t}\big].$$

The scheme is said to be *secure in the sense of* REC-ST if, for each graph $G = (V, E)$ in Γ, each sequence of distinct time periods T, each class $u \in V$ and each time period $t \in T$, the function $\mathbf{Adv}^{\mathrm{REC}}_{\mathrm{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$ is negligible, for each static adversary $\mathrm{STAT}_{u,t}$ whose time complexity is polynomial in τ.

Now, we consider an *adaptive adversary*. As done before, we assume the existence of an oracle which can provide the adversary with the private information held by the corrupted users. We require that the adversary will guess the key $k_{u,t}$ with probability only negligibly different from $1/2^{\mathrm{length}(k_{u,t})}$.

Definition 5 [REC-AD]. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ. Let $\mathrm{ADAPT} = (\mathrm{ADAPT}_1, \mathrm{ADAPT}_2)$ be an adaptive adversary that is given access to the oracle $\mathcal{O}_s(\cdot)$ during both stages of the attack, where s is the private information computed by Gen. Consider the following experiment:

Experiment $\mathbf{Exp}^{\mathrm{REC}}_{\mathrm{ADAPT}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$(u, t, state) \leftarrow \mathrm{ADAPT}_1^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub)$
$k'_{u,t} \leftarrow \mathrm{ADAPT}_2^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub, u, t, state)$
return $k'_{u,t}$

It is required that the pair (u, t) output by ADAPT_1 is such that $(v, \lambda) \in F_{u,t}$, for any pair (v, λ) already queried to the oracle $\mathcal{O}_s(\cdot)$. Moreover, it is also required that ADAPT_2 never queries the oracle $\mathcal{O}_s(\cdot)$ on a pair $(v, \lambda) \in V \times \mathcal{P}$ such that $u \in A_v$ and $t \in \lambda$. The advantage of ADAPT is defined as

$$\mathbf{Adv}^{\mathrm{REC}}_{\mathrm{ADAPT}}\big(1^\tau, G, \mathcal{P}\big) = Pr[k'_{u,t} = k_{u,t}].$$

The scheme is said to be *secure in the sense of* REC-AD if, for each graph $G = (V, E)$ in Γ, each sequence of distinct time periods T, each class $u \in V$ and each time period $t \in T$, the function $\mathbf{Adv}^{\mathrm{REC}}_{\mathrm{ADAPT}}(1^\tau, G, \mathcal{P})$ is negligible, for each adaptive adversary ADAPT whose time complexity is polynomial in τ.

3.1. *Implications and Separations*

In the following we prove that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries.

Theorem 1 [IND-ST⇔IND-AD]. *Let Γ be a family of graphs corresponding to partially ordered hierarchies. A time-bound hierarchical key assignment scheme for Γ is secure in the sense of* IND-ST *if and only if it is secure in the sense of* IND-AD.

Proof. The implication IND-AD⇒IND-ST is trivial, since any adaptive adversary could behave as a static one attacking a class u in a time period t, simply by querying the oracle $\mathcal{O}_s(\cdot)$ on all pairs $(v, \lambda) \in F_{u,t}$ and by choosing the pair (u, t) in the first stage of the attack.

Now we prove that IND-ST⇒IND-AD. Let (Gen, Der) be a time-bound hierarchical key assignment scheme for Γ secure in the sense of IND-ST and assume by contradiction the existence of an adaptive adversary $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ whose advantage $\mathbf{Adv}^{\mathrm{IND}}_{\mathtt{ADAPT}}$ on input a given graph $G' = (V', E')$ in Γ and an interval-set $\mathcal{P}'$ over a sequence of distinct time periods T is non-negligible. Let (u, t) be a pair output by $\mathtt{ADAPT}_1$ with probability at least $\frac{1}{|V'|\cdot|T|}$, where the probability is taken over the coin flips of Gen and $\mathtt{ADAPT}_1$. This means that (u, t) belongs to the set of the most likely choices made by $\mathtt{ADAPT}_1$. We show how to construct a static adversary $\mathtt{STAT}_{u,t}$, using ADAPT, such that $\mathbf{Adv}^{\mathrm{IND}}_{\mathtt{STAT}_{u,t}}$ on input G' and $\mathcal{P}'$ is non-negligible. In particular, we show that $\mathtt{STAT}_{u,t}$'s advantage is polynomially related to ADAPT's advantage.

The algorithm $\mathtt{STAT}_{u,t}$, on inputs the graph G', the interval-set $\mathcal{P}'$, the public information *pub* output by the algorithm *Gen*, the private information *corr* assigned by *Gen* to all corrupted users, and a challenge value x, corresponding either to the key $k_{u,t}$ or to a random value having the same length as $k_{u,t}$, runs the algorithm $\mathtt{ADAPT}_1$, on inputs G', $\mathcal{P}'$, and *pub*. Notice that $\mathtt{STAT}_{u,t}$ is able to simulate the interaction between $\mathtt{ADAPT}_1$ and the oracle $\mathcal{O}_s(\cdot)$, for each query $(v, \lambda) \in F_{u,t}$. Indeed, $\mathtt{STAT}_{u,t}$ simply retrieves from *corr* the private information $s_{v,\lambda}$ and gives it to $\mathtt{ADAPT}_1$. On the other hand, if $\mathtt{ADAPT}_1$ queries the oracle on a pair (v, λ) such that $u \in A_v$ and $t \in \lambda$, then $\mathtt{STAT}_{u,t}$ outputs 0, because it is not able to reply with the private information $s_{v,\lambda}$, which is not included in *corr*. In such a case (u, t) cannot be the pair output by $\mathtt{ADAPT}_1$. Let $(v, t', state)$ be the triple output by $\mathtt{ADAPT}_1$. If $u = v$ and $t = t'$, then $\mathtt{STAT}_{u,t}$ outputs the same output as $\mathtt{ADAPT}_2$, on inputs G', $\mathcal{P}'$, *pub*, u, t, *state* and x. On the other hand, if $u \neq v$ or $t \neq t'$, $\mathtt{STAT}_{u,t}$ outputs 0.

It is easy to see that whether $G = G'$ and $\mathcal{P} = \mathcal{P}'$, it holds that

$$\mathbf{Adv}^{\mathrm{IND}}_{\mathtt{STAT}_{u,t}}\left(1^{\tau}, G, \mathcal{P}\right) = Pr[u = v \text{ and } t = t'] \cdot \mathbf{Adv}^{\mathrm{IND}}_{\mathtt{ADAPT}}\left(1^{\tau}, G, \mathcal{P}\right).$$

Since (u, t) is chosen by $\mathtt{ADAPT}_1$ with probability at least $\frac{1}{|V'|\cdot|T|}$ and $\mathbf{Adv}^{\mathrm{IND}}_{\mathtt{ADAPT}}$ on input G' and $\mathcal{P}'$ is non-negligible, it follows that also $\mathbf{Adv}^{\mathrm{IND}}_{\mathtt{STAT}_{u,t}}$ on input G' and $\mathcal{P}'$ is non-negligible. Contradiction. □

The next result can be proved following the lines of the proof of Theorem 1.

Theorem 2 [REC-ST⇔REC-AD]. *A time-bound hierarchical key assignment scheme for a family of graphs Γ is secure in the sense of* REC-ST *if and only if it is secure in the sense of* REC-AD.

It is easy to see that security with respect to key indistinguishability implies security against key recovery, whilst the opposite does not hold (see [6] for formal proofs).

Theorem 3 [IND-ST⇒REC-ST]. *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If a time-bound hierarchical key assignment scheme for Γ is secure in the sense of* IND-ST, *then it is also secure in the sense of* REC-ST.

Theorem 4 [REC-ST⇏IND-ST]. *Let Γ be a family of graphs corresponding to partially ordered hierarchies. If there exists a time-bound hierarchical key assignment*

scheme for Γ which is secure in the sense of REC-ST, *there exists a time-bound hierarchical key assignment scheme for Γ which is secure in the sense of* REC-ST *but which is not secure in the sense of* IND-ST.

Figure 1 shows the hierarchy of security definitions for time-bound key assignment schemes, resulting from Theorems 1, 2, 3, and 4.

4. Constructions

A time-bound hierarchical key assignment scheme can be easily obtained by applying an instance of a key assignment scheme without temporal constraint to each time period. Unfortunately, this naive solution results in impractical schemes. In particular, these schemes present the following drawbacks:

1. users need to store up to $|T|$ private keys, where $|T|$ is the total number of time periods;
2. the amount of public information depends not only on the number of classes but also on the number of time periods;
3. the key derivation procedure is expensive.

Our constructions improve on the above naive solution in order to overcome its drawbacks. In particular, we aim to provide schemes with a very efficient key derivation procedure. Unfortunately, we are not able to design a scheme which is optimal with respect of all three items at the same time. The first proposal provides a solution to drawbacks 1 and 3. Indeed, each user is required to store only one key and the key derivation procedure consists of only one symmetric decryption. The second proposal provides a solution to drawbacks 2 and 3. Indeed, the public information depends only on the number of classes and the key derivation procedure consists of only one pairing evaluation. Both solutions are optimal in terms of key derivation complexity.

4.1. *A Scheme Based on Symmetric Encryption Schemes*

In this section we show how to construct a time-bound key assignment scheme using as a building block a symmetric encryption scheme. We refer to the resulting scheme as the *Two-Level Encryption-Based Construction* (TLEBC). Such a scheme provides an efficient key derivation procedure and requires each user to store only a single private key. On the other hand, the public information could be very large since it depends not only on the number of classes but also on the number of time periods. We prove that the security property of such a scheme depends on the security property of the underlying encryption scheme. We need to recall the definition of a symmetric encryption scheme and its notions of security.

4.1.1. *Symmetric Encryptions and Their Security Notions*

We first recall the definition of a symmetric encryption scheme.

Definition 6. A *symmetric encryption scheme* is a triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of algorithms satisfying the following conditions:

1. The *key-generation algorithm* $\mathcal{K}$ is probabilistic polynomial time. It takes as input the security parameter 1^τ and produces as output a string *key*.
2. The *encryption algorithm* $\mathcal{E}$ is probabilistic polynomial time. It takes as inputs 1^τ, a string *key* produced by $\mathcal{K}(1^\tau)$, and a message $m \in \{0,1\}^*$, and produces as output the ciphertext y.
3. The *decryption algorithm* $\mathcal{D}$ is deterministic polynomial time. It takes as inputs 1^τ, a string *key* produced by $\mathcal{K}(1^\tau)$, and a ciphertext y, and produces as output a message m. We require that for any string *key* which can be output by $\mathcal{K}(1^\tau)$, for any message $m \in \{0,1\}^*$, and for all y that can be output by $\mathcal{E}(1^\tau, key, m)$, we have $\mathcal{D}(1^\tau, key, y) = m$.

Now, we define what we mean by a *secure* symmetric encryption scheme. We consider two different security goals: with respect to *plaintext indistinguishability* and against *plaintext recovery*.

We start with the definition of security with respect to *plaintext indistinguishability*, which is an adaption of the notion of *polynomial security* as given in [27]. We imagine an adversary $A = (A_1, A_2)$ running in two stages. In advance of the adversary's execution, a random key *key* is chosen and kept hidden from the adversary. During the first stage, the adversary A_1 outputs a triple $(x_0, x_1, state)$, where x_0 and x_1 are two messages of the same length, and *state* is some state information which could be useful later. One message between x_0 and x_1 is chosen at random and encrypted to give the challenge ciphertext y. In the second stage, the adversary A_2 is given y and *state* and has to determine whether y is the encryption of x_0 or x_1. Informally, the encryption scheme is said to be secure with respect to a non-adaptive chosen plaintext attack, denoted by `IND-P1-C0` in [33], if every polynomial-time adversary A, which has access to the encryption oracle only during the first stage of the attack and never has access to the decryption oracle, succeeds in determining whether y is the encryption of x_0 or x_1 with probability only negligibly different from $1/2$.

Definition 7 `[IND-P1-C0]`. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let τ be a security parameter. Let $A = (A_1, A_2)$ be an adversary that has access to the encryption oracle only during the first stage of the attack and never has access to the decryption oracle. Consider the following two experiments:

Experiment $\mathbf{Exp}^{\mathtt{IND-P1-C0}-1}_{\Pi,A}(1^\tau)$	Experiment $\mathbf{Exp}^{\mathtt{IND-P1-C0}-0}_{\Pi,A}(1^\tau)$
key $\leftarrow \mathcal{K}(1^\tau)$	key $\leftarrow \mathcal{K}(1^\tau)$
$(x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{\text{key}}(\cdot)}(1^\tau)$	$(x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{\text{key}}(\cdot)}(1^\tau)$
$y \leftarrow \mathcal{E}_{\text{key}}(x_1)$	$y \leftarrow \mathcal{E}_{\text{key}}(x_0)$
$d \leftarrow A_2(1^\tau, y, state)$	$d \leftarrow A_2(1^\tau, y, state)$
return d	**return** d

The advantage of A is defined as

$$\mathbf{Adv}^{\mathtt{IND-P1-C0}}_{\Pi,A}(1^\tau) = \left|Pr\left[\mathbf{Exp}^{\mathtt{IND-P1-C0}-1}_{\Pi,A}(1^\tau) = 1\right] - Pr\left[\mathbf{Exp}^{\mathtt{IND-P1-C0}-0}_{\Pi,A}(1^\tau) = 1\right]\right|.$$

The scheme is said to be *secure* in the sense of IND-P1-C0 if the advantage function $\mathbf{Adv}^{\text{IND-P1-C0}}_{\Pi,A}(1^\tau)$ is negligible, for any adversary A whose time complexity is polynomial in τ.

In the following we consider a weaker definition of security, namely *plaintext recovery*. We imagine an adversary A whose goal is to recover the plaintext corresponding to a given ciphertext. In advance of the adversary's execution, both a random key *key* and a random message x, having a certain length, are chosen and kept hidden from the adversary. The message x is then encrypted and given to the adversary as the challenge ciphertext y. Informally, the encryption scheme is said to be secure with respect to a non-adaptive chosen plaintext attack, which we denote by PR-P1-C0,[1] if every polynomial-time adversary A, which has access to the encryption oracle and never has access to the decryption oracle, succeeds in determining the plaintext x corresponding to the challenge ciphertext y with probability only negligibly different from $1/2^{\text{length}(x)}$.

Definition 8 [PR-P1-C0]. Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let τ be a security parameter. Let A be an adversary that has access to the encryption oracle and has never access to the decryption oracle. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{PR-P1-C0}}_{\Pi,A}(1^\tau)$
$key \leftarrow \mathcal{K}(1^\tau)$
$x \leftarrow \{0,1\}^\tau$
$y \leftarrow \mathcal{E}_{\text{key}}(x)$
$x' \leftarrow A^{\mathcal{E}_{\text{key}}(\cdot)}(y)$
if $x = x'$ **then return** 1
else return 0

The advantage of A is defined as

$$\mathbf{Adv}^{\text{PR-P1-C0}}_{\Pi,A}(1^\tau) = Pr\left[\mathbf{Exp}^{\text{PR-P1-C0-1}}_{\Pi,A}(1^\tau) = 1\right].$$

The scheme is said to be *secure* in the sense of PR-P1-C0 if the advantage function $\mathbf{Adv}^{\text{PR-P1-C0}}_{\Pi,A}(1^\tau)$ is negligible, for any adversary A whose time complexity is polynomial in τ.

4.1.2. *The Two-Level Encryption-Based Construction*

We consider a graph transformation, starting from the graph $G = (V, E)$ and $\mathcal{P}$. The output of such a transformation is a graph $G_{\mathcal{P}T} = (V_{\mathcal{P}T}, E_{\mathcal{P}T})$, where $V_{\mathcal{P}T} = V_{\mathcal{P}} \cup V_T$ and $V_{\mathcal{P}} \cap V_T = \emptyset$, constructed as follows:

- for each class $u \in V$ and each time sequence $\lambda \in \mathcal{P}$, we place a class u_λ in $V_{\mathcal{P}}$;
- for each class $u \in V$ and each time period $t \in T$, we place a class u_t in V_T;
- for each class $u \in V$, each time sequence $\lambda \in \mathcal{P}$, and each time period $t \in \lambda$, we place an edge between u_λ and u_t in $G_{\mathcal{P}T}$, i.e., $(u_\lambda, u_t) \in E_{\mathcal{P}T}$;

[1] The security notion PR-P1-C0 is known in literature as PR-CPA. For instance, its description may be found in [8].

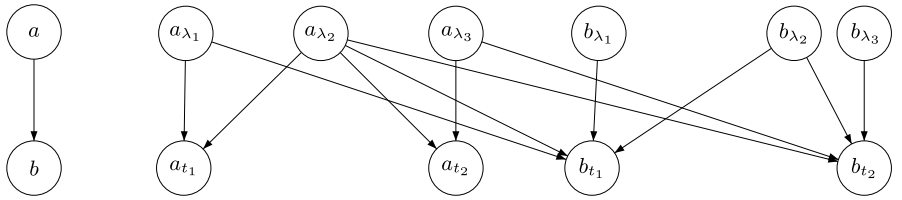

Fig. 2. The graph transformation used in our construction.

– for each pair of classes u and v connected by a path in G, each time sequence $\lambda \in \mathcal{P}$, and each time period $t \in \lambda$, we place an edge between u_λ and v_t in $G_{\mathcal{PT}}$, i.e., $(u_\lambda, v_t) \in E_{\mathcal{PT}}$.

Figure 2 shows an example of the graph transformation described above, where $\mathcal{P} = \{\lambda_1, \lambda_2, \lambda_3\}$, $\lambda_1 = (t_1)$, $\lambda_2 = (t_1, t_2)$, and $\lambda_3 = (t_2)$.

In the following we describe the TLEBC. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme.

Algorithm. $Gen(1^\tau, G, \mathcal{P})$

1. Perform a graph transformation in order to obtain the two-level partially ordered hierarchy $G_{\mathcal{PT}} = (V_{\mathcal{PT}}, E_{\mathcal{PT}})$, where $V_{\mathcal{PT}} = V_{\mathcal{P}} \cup V_T$;
2. For each class u_λ in $V_{\mathcal{P}}$, let $s_{u,\lambda} \leftarrow \mathcal{K}(1^\tau)$;
3. For each class u_t in V_T, randomly choose a secret value $k_{u,t} \in \{0,1\}^\tau$;
4. Let s and k be the sequences of private information and keys, respectively, computed in the previous two steps;
5. For any pair of classes $(u_\lambda, v_t) \in V_{\mathcal{P}} \times V_T$ such that $(u_\lambda, v_t) \in E_{\mathcal{PT}}$, compute the public information $p_{(u,\lambda),(v,t)} = \mathcal{E}_{s_{u,\lambda}}(k_{v,t})$;
6. Let *pub* be the sequence of public information computed in the previous step;
7. Output (s, k, pub).

Algorithm. $Der(1^\tau, G, \mathcal{P}, u, v, \lambda, s_{u,\lambda}, t, pub)$

1. Extract the public value $p_{(u,\lambda),(v,t)}$ from *pub*;
2. Output the key $k_{v,t} = \mathcal{D}_{s_{u,\lambda}}(p_{(u,\lambda),(v,t)})$.

4.1.3. *Analysis of the Scheme*

In the following we show that the security property of the TLEBC depends on the security property of the underlying encryption scheme. We prove that if the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of `IND-P1-C0` (`PR-P1-C0`, respectively), then the TLEBC is secure in the sense of `IND-ST` (`REC-ST`, respectively).

Theorem 5. *If the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of* IND-P1-C0, *then the TLEBC is secure in the sense of* IND-ST.

Proof. Let Γ be a family of graphs corresponding to partially ordered hierarchies and let $G = (V, E)$ be any graph in Γ. The proof uses a standard hybrid argument. Let $u_{t^*} \in V_T$ be a class and assume there exist m classes in $V_{\mathcal{P}}$ which are able to access u_{t^*}. W.l.o.g., let $u_{1_{\lambda_1}}, \ldots, u_{m_{\lambda_m}}$ be such classes. Let $\mathtt{STAT}_{u,t^*}$ be a static adversary attacking class u_{t^*}. We construct a sequence of $m+1$ experiments $\mathbf{Exp}^1_{u,t^*}, \ldots, \mathbf{Exp}^{m+1}_{u,t^*}$, all defined over the same probability space. In each experiment we modify the way the view of $\mathtt{STAT}_{u,t^*}$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $q = 1, \ldots, m+1$, experiment $\mathbf{Exp}^q_{u,t^*}$ is defined as follows:

Experiment $\mathbf{Exp}^q_{u,t^*}(1^\tau, G, \mathcal{P})$
  $(s, \alpha, pub) \leftarrow Gen^q(1^\tau, G, \mathcal{P})$
  $corr \leftarrow Corrupt_{u,t^*}(s)$
  $d \leftarrow \mathtt{STAT}_{u,t^*}(1^\tau, G, \mathcal{P}, pub, corr, \alpha_{u,t^*})$
  return d

The algorithm Gen^q used in $\mathbf{Exp}^q_{u,t^*}$ is the same algorithm Gen used in the TLEBC with the following modification: for any $h = 1, \ldots, q-1$, the public value $p_{(v_h,\lambda_h),(u,t^*)}$ is computed as the encryption, with the key s_{v_h,λ_h}, of a random value $\beta_q \in \{0,1\}^\tau$, instead of the encryption of the key assigned to u_{t^*}, which is denoted by α_{u,t^*}. Notice that experiment $\mathbf{Exp}^1_{u,t^*}$ is the same as $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,t^*}}$. Indeed, the adversary $\mathtt{STAT}_{u,t^*}$ is given the value α_{u,t^*} and for each $h = 1, \ldots, m$, the public value $p_{(v_h,\lambda_h),(u,t^*)}$ computed by Gen^1 corresponds to the encryption of α_{u,t^*}. On the other hand, experiment $\mathbf{Exp}^{m+1}_{u,t^*}$ is the same as $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,t^*}}$. Indeed, the adversary $\mathtt{STAT}_{u,t^*}$ is given the value α_{u,t^*} and, for each $h = 1, \ldots, m$, the public value $p_{(v_h,\lambda_h),(u,t^*)}$ computed by Gen^{m+1} corresponds to the encryption of the value β_{m+1}.

In the following we show that, for any $q = 2, \ldots, m+1$, the adversary's view in the $(q-1)$th experiment is indistinguishable from the adversary's view in the qth one. Hence, it follows that the adversary's views in experiments $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,t^*}}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,t^*}}$ are also indistinguishable.

Assume by contradiction that there exists a polynomial-time distinguisher B_q which is able to distinguish between the adversary $\mathtt{STAT}_{u,t^*}$'s views in experiments $\mathbf{Exp}^{q-1}_{u,t^*}$ and $\mathbf{Exp}^q_{u,t^*}$ with non-negligible advantage. We show how to construct a polynomial-time adversary $A = (A_1, A_2)$, using B_q, which breaks the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of IND-P1-C0. The algorithm A_1, on input 1^τ, makes queries to the encryption oracle $\mathcal{E}_{\mathrm{key}}(\cdot)$ and outputs a triple $(x_0, x_1, state)$, where $x_0, x_1 \in \{0,1\}^\tau$, and *state* is some state information.

Algorithm $A_1^{\mathcal{E}_{\text{key}}(\cdot)}(1^\tau)$
$x_0, x_1 \leftarrow \{0,1\}^\tau$
//construction of secret values
for each class $w_\lambda \in V_\mathcal{P} \setminus \{v_{q_{\lambda_q}}\}$
$s_{w,\lambda} \leftarrow \mathcal{K}(1^\tau)$
for each class $w_t \in V_T \setminus \{u_{t^*}\}$
$k_{w,t} \leftarrow \{0,1\}^\tau$
//construction of public values
for $h = 1, \ldots, q-1$
$p_{(v_h,\lambda_h),(u,t^*)} \leftarrow \mathcal{E}_{s_{v_h,\lambda_h}}(x_1)$
for $h = q+1, \ldots, m$
$p_{(v_h,\lambda_h),(u,t^*)} \leftarrow \mathcal{E}_{s_{v_h,\lambda_h}}(x_0)$
for any class $w_t \in V_T \setminus \{u_{t^*}\}$ such that $(v_{q_{\lambda_q}}, w_t) \in E_{\mathcal{P}T}$
$p_{(v_q,\lambda_q),(w,t)} \leftarrow \mathcal{E}_{\text{key}}(k_{w,t})$
for any two classes $z_\lambda \in V_\mathcal{P} \setminus \{v_{q_{\lambda_q}}\}$ and $w_t \in V_T \setminus \{u_{t^*}\}$ such that $(z_\lambda, w_t) \in E_{\mathcal{P}T}$
$p_{(z,\lambda),(w,t)} \leftarrow \mathcal{E}_{s_{z,\lambda}}(k_{w,t})$
//construction of the view
$pub' \leftarrow$ all public values constructed as above
$corr \leftarrow$ secret values held by classes in the set $\{w_\lambda \in V_\mathcal{P} : (w_\lambda, u_{t^*}) \notin E_{\mathcal{P}T}\}$
$state \leftarrow (pub', corr, x_0, x_1)$
return $(x_0, x_1, state)$

Let y be the challenge for the algorithm A, corresponding to the encryption of either x_0 or x_1 with the unknown key key. The algorithm A_2 constructs the view for the distinguisher B_q, adding the value $p_{(v_q,\lambda_q),(u,t^*)} = y$ to the public information pub' constructed by A_1, and outputs the same output as B_q on inputs such a view, the class u, the time period t^*, and x_0. More formally, the algorithm A_2 is defined as follows:

Algorithm $A_2(1^\tau, y, state)$
let $state = (pub', corr, x_0, x_1)$
$pub \leftarrow pub'$ with $p_{(v_q,\lambda_q),(u,t^*)}$ set equal to y
$d \leftarrow B_q(1^\tau, G, \mathcal{P}, pub, corr, x_0)$
return d

Notice that if y corresponds to the encryption of x_1, then the random variable associated with the adversary's view is exactly the same as the one associated with the adversary view in experiment $\mathbf{Exp}_{u,t^*}^{q-1}$, whereas, if y corresponds to the encryption of x_0, it has the same distribution as the one associated with the adversary's view in experiment $\mathbf{Exp}_{u,t^*}^{q}$.

Hence, if the algorithm B_q is able to distinguish between such views with non-negligible advantage, it follows that algorithm A is able to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of `IND-P1-C0`. Contradiction.

Hence, for any $q = 2, \ldots, m+1$, the adversary's view in the $(q-1)$th experiment is indistinguishable from the adversary's view in the qth one. Therefore, the adversary's view in experiment $\mathbf{Exp}_{\text{STAT}_{u,t^*}}^{\texttt{IND-1}}$ is indistinguishable from the adversary's view in experiment $\mathbf{Exp}_{\text{STAT}_{u,t^*}}^{\texttt{IND-0}}$. This concludes the proof. □

Following the lines of Theorem 5 we can prove that the next result also holds.

Theorem 6. *If the encryption scheme* $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ *is secure in the sense of* PR-P1-C0, *then the TLEBC is secure in the sense of* REC-ST.

4.1.4. *Performance Evaluation*

In this section we evaluate the TLEBC with respect to different criteria. Regarding space requirements, the scheme requires a public value for each edge in the graph $G_{\mathcal{P}T}$ used in the construction. It is easy to see that $|E_{\mathcal{P}T}| = O(|V|^2) \cdot \sum_{i=1}^{|T|} i \cdot (|T| - i + 1) = O(|V|^2 \cdot |T|^3)$. More precisely, $|E_{\mathcal{P}T}| = O(|E^*| \cdot |T|^3)$, where $G^* = (V, E^*)$ is the directed graph that can be obtained from $G = (V, E)$ by adding to E all self-loops and edges which are implied by the property of the transitive closure. On the other hand, each user belonging to a certain class for a time sequence has to store a single secret value. Moreover, users are required to perform a single decryption in order to derive a key.

To obtain a time-bound hierarchical key assignment scheme secure in the sense of IND-ST we can use the IND-P1-C0 secure symmetric encryption scheme, called the *XOR construction*, defined in [10]. This construction makes use of a function family $\mathcal{F}$. Assuming that $\mathcal{F}$ is a pseudorandom function family [26], the XOR construction has been shown to be secure in the sense of IND-P1-C0 (see [10,33]). The most efficient constructions for pseudorandom function families were proposed by Naor and Reingold [39]. In their proposals, the cost of evaluating a pseudorandom function is comparable to two modular exponentiations. An efficient implementation of the resulting time-bound hierarchical key assignment scheme could be obtained by using the HMAC [9] to realize the function family $\mathcal{F}$. More details about the instantiation of the TLEBC with the XOR construction may be found in [6].

In the following we show how to manage changes to the hierarchy, such as addition and deletion of nodes and edges, in such a way that no private information held by users need to be re-computed by the TA. Indeed, such updates can be handled by local changes to the public information.

Insertion of an edge Let (u, v) be an edge to be added to the hierarchy, starting from time period t_i through $t_{|T|}$. Such an update can be managed by the TA by adding to the public information *pub* the public value $p_{(u,\lambda),(v,t_j)} = \mathcal{E}_{s_{u,\lambda}}(k_{v,t_j})$, for each sequence of time periods $\lambda = (t_x, \ldots, t_y) \in \mathcal{P}$, where $i \leq x \leq j \leq y$.

Deletion of an edge Let (u, v) be an edge to be deleted from the hierarchy, starting from time period t_i through $t_{|T|}$. In order to forbid users belonging to class u from computing the key of class v in any time period t_j, where $j = 1, \ldots, |T|$, the TA has to choose a new key $k'_{v,t_j} \in \{0,1\}^\tau$ for class v at time period t_j. On the other hand, in order to allow authorized users to compute such a new key, the TA has to update the public information *pub*, by recomputing the public value $p_{(w,\lambda),(v,t_j)} = \mathcal{E}_{s_{w,\lambda}}(k'_{v,t_j})$, for each edge $(w, v) \in E$ and each time sequence $\lambda = (t_x, \ldots, t_y) \in \mathcal{P}$, where $i \leq x \leq j \leq y$.

Insertion of a node Let u be a node to be added to the hierarchy, along with new incoming and outgoing edges, starting from time period t_i through $t_{|T|}$. For each $j = i, \ldots, |T|$, the TA first chooses a random key $k_{u,t_j} \in \{0,1\}^\tau$. Then, for each time sequence

$\lambda = (t_x, \ldots, t_y) \in \mathcal{P}$, where $i \leq x \leq y \leq |T|$, the TA computes the private information $s_{u,\lambda} \leftarrow \mathcal{K}(1^\tau)$ and uses it to compute the public value $p_{(u,\lambda),(u,t_j)} = \mathcal{E}_{s_{u,\lambda}}(k_{u,t_j})$, for any $j = x, \ldots, y$, which is added to the public information *pub*. Finally, the updates involving the addition of incoming and outgoing edges are managed by using the above procedure for edge insertions.

Deletion of a node Let u be a node to be deleted by the hierarchy, starting from time period t_i through $t_{|T|}$. The TA first uses the above procedure for edge deletions to delete all edges incident on u and then removes the node from V.

4.2. *A Scheme Based on Bilinear Maps*

In this section we design a time-bound hierarchical key assignment which provides an efficient key derivation procedure and the amount of public information does not depend on the number of time periods. On the other hand, the private information could be as large as the number of time periods. Our scheme uses as a building block a bilinear map between groups. We refer to such a scheme as the *Two-Level Pairing-Based Construction* (TLPBC). Bilinear maps have been used in cryptography to construct key exchange schemes [32], public-key cryptosystems [12,14,16], signature schemes [15], etc. We need to recall the definition of a bilinear map and the computational assumption on which the security of our scheme is based.

4.2.1. *Bilinear Maps, BDH and BDDH Assumptions*

We first recall the definition of a bilinear map.

Definition 9. A function $e : G_1 \times \hat{G}_1 \rightarrow G_2$ is said to be a *bilinear map* if the following properties are satisfied:

1. G_1 and $\hat{G}_1$ are two groups of the same prime order q;
2. For each $\alpha, \beta \in Z_q$, each $g \in G_1$, and each $h \in \hat{G}_1$, the value $e(g^\alpha, h^\beta) = e(g, h)^{\alpha\beta}$ is efficiently computable;
3. The map is non-degenerate (i.e., if g generates G_1 and h generates $\hat{G}_1$, then $e(g, h)$ generates G_2).

Typically, the group G_1 is a subgroup of the additive group of points of an elliptic curve $E(F_p)$, where p denotes the size of the field where the elliptic curve is defined. The group $\hat{G}_1$ is a subgroup of $E(F_{p^\eta})$, where $\eta > 0$ is the *embedding degree* of the map, whereas the group G_2 is a subgroup of the multiplicative group of the finite field $F^*_{p^\eta}$. Given a security parameter τ, let $\mathcal{G}$ be a randomized algorithm, called a *BDH parameter generator*, which, on input 1^τ, outputs a prime number q of τ bits, the description of three groups G_1, $\hat{G}_1$ and G_2 of order q, and the description of a bilinear map $e : G_1 \times \hat{G}_1 \rightarrow G_2$. The running time of $\mathcal{G}$ is polynomial in τ. We denote the output of $\mathcal{G}$ by $\mathcal{G}(1^\tau) = \langle q, G_1, \hat{G}_1, G_2, e \rangle$.

The *Bilinear Diffie–Hellman Problem (BDH)* in $\langle G_1, G_2, e \rangle$ is as follows [14]: given the tuple $(g, g^\alpha, g^\beta, g^\gamma)$, for randomly chosen $\alpha, \beta, \gamma \in Z^*_q$, and a random generator g of G_1, compute $e(g, g)^{\alpha \cdot \beta \cdot \gamma} \in G_2$.

Definition 10 (BDH assumption). Let $\mathcal{G}$ be a BDH parameter generator. The advantage of an algorithm A in solving the BDH Problem for $\mathcal{G}$ is defined as

$$\mathbf{Adv}^{\mathrm{BDH}}_{\mathcal{G},A}(1^\tau) = Pr\big[A\big(g, g^\alpha, g^\beta, g^\gamma\big) = e(g,g)^{\alpha\cdot\beta\cdot\gamma}\big],$$

where the probability is over the random choices of $\mathcal{G}(1^\tau)$, the random choice of g in G_1^*, the random choice of α, β, γ in Z_q^*, and the random bits of A.

The BDH problem is said to be hard in groups generated by $\mathcal{G}$ if the function $\mathbf{Adv}^{\mathrm{BDH}}_{\mathcal{G},A}(1^\tau)$ is negligible, for each randomized algorithm A whose time complexity is polynomial in 1^τ.

The *Bilinear Decisional Diffie–Hellman Problem (BDDH)* in $< G_1, G_2, e >$ is as follows [14]: given the tuple $(g, g^\alpha, g^\beta, g^\gamma, x)$, for randomly chosen $\alpha, \beta, \gamma \in Z_q^*$, $x \in G_2$, and a random generator g of G_1, decide whether $x = e(g,g)^{\alpha\cdot\beta\cdot\gamma}$.

Definition 11 (BDDH assumption). Let $\mathcal{G}$ be a BDH parameter generator. The advantage of an algorithm A in solving the BDDH Problem for $\mathcal{G}$ is defined as

$$\begin{aligned}\mathbf{Adv}^{\mathrm{BDDH}}_{\mathcal{G},A}(1^\tau) = \big|&Pr\big[A\big(g, g^\alpha, g^\beta, g^\gamma, x\big) = 1\big] \\ &- Pr\big[A\big(g, g^\alpha, g^\beta, g^\gamma, e(g,g)^{\alpha\cdot\beta\cdot\gamma}\big) = 1\big]\big|,\end{aligned}$$

where the probability is over the random choices of $\mathcal{G}(1^\tau)$, the random choice of g in G_1^*, the random choice of α, β, γ in Z_q^*, the random choice of x in G_2, and the random bits of A.

The BDDH problem is said to be hard in groups generated by $\mathcal{G}$ if the function $\mathbf{Adv}^{\mathrm{BDDH}}_{\mathcal{G},A}(1^\tau)$ is negligible, for each randomized algorithm A whose time complexity is polynomial in 1^τ.

4.2.2. *The Two-Level Pairing-Based Construction*

In the following we describe a time-bound hierarchical key assignment scheme based on a bilinear map. For simplicity, we focus on symmetric bilinear maps (i.e., such that $G_1 = \hat{G}_1$), but our scheme works in the more general asymmetric setting (in particular, this implies that we could use the highly efficient MNT curves [38]). We consider a two-level partially ordered hierarchy, where each level contains the same number of classes and there are no edges between classes at the same level. We remark that this is not a restriction, since any directed graph representing an access control policy can be transformed into a two-level partially ordered hierarchy having the above features, using a technique proposed in [21]. For the reader's convenience, we first explain how such a graph transformation works. Let $G = (V, E)$ be the graph corresponding to a partially ordered hierarchy. We can construct a two-level partially ordered hierarchy $G' = (V', E')$, where $V' = V_\ell \cup V_r$ and $V_\ell \cap V_r = \emptyset$, as follows:

- for each class $u \in V$, we place two classes u^ℓ and u^r in V', where $u^\ell \in V_\ell$ and $u^r \in V_r$;
- for each class $u \in V$, we place the edge (u^ℓ, u^r) in E';

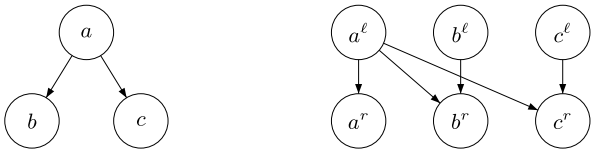

Fig. 3. The graph transformation used in our construction.

– for each pair of classes v and u connected by a path in G, we place the edge (v^ℓ, u^r) in E'.

It is easy to see that the graphs G and G' define exactly the same access control policy. Figure 3 shows an example of the graph transformation described above.

We stress that the graph transformation has the only purpose of simplifying the description of the scheme. It is not required in practice. Therefore, it does not introduce any extra efficiency cost.

In the following we describe the TLPBC. Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, let $G' = (V', E')$ be the two-level partially ordered hierarchy obtained from G, and let $\mathcal{G}$ be a BDH parameter generator.

Algorithm. $Gen(1^\tau, G', \mathcal{P})$

1. Run $\mathcal{G}(1^\tau)$ to generate a prime q, two groups G_1 and G_2 of order q and a bilinear map $e : G_1 \times G_1 \to G_2$;
2. Choose a generator $g \in G_1^*$;
3. For each class $u^\ell \in V_\ell$, randomly choose a secret value $\pi_u^\ell \in Z_q$;
4. For each class $v^r \in V_r$, randomly choose a secret value $\pi_v^r \in Z_q$;
5. For each pair of classes $u^\ell \in V_\ell$ and $v^r \in V_r$ connected by an edge, i.e., such that $(u^\ell, v^r) \in E'$, compute the public information $p_{u,v} = g^{\pi_v^r/\pi_u^\ell}$;
6. Let *pub* be the sequence of public information computed in the previous step, along with the bilinear map e and the generator g;
7. For each time period $t \in T$, randomly choose a secret value $\delta_t \in Z_q$;
8. For each class $u^\ell \in V_\ell$ and each time period $t \in T$, compute the private information $s_{u,t} = g^{\pi_u^\ell \cdot \delta_t}$;
9. For each class $u^\ell \in V_\ell$ and each time sequence $\lambda \in \mathcal{P}$, where $\lambda = (t_x, \ldots, t_y)$, compute the private information $s_{u,\lambda} = (s_{u,t_x}, \ldots, s_{u,t_y})$;
10. For each class $v^r \in V_r$ and each time period $t \in T$, compute the key $k_{v,t} = e(g, g)^{\pi_v^r \cdot \delta_t}$;
11. Let s and k be the sequences of private information and keys, respectively, computed in previous steps;
12. Output (s, k, pub).

Algorithm. $Der(1^\tau, G', \mathcal{P}, u^\ell, v^r, \lambda, s_{u,\lambda}, t, pub)$

1. Extract the public value $p_{u,v} = g^{\pi_v^r/\pi_u^\ell}$ from *pub*;

2. Compute the key $k_{v,t}$ as follows

$$\begin{aligned} e(s_{u,t}, p_{u,v}) &= e\big(g^{\pi_u^\ell \cdot \delta_t}, g^{\pi_v^r/\pi_u^\ell}\big) \\ &= e(g,g)^{\pi_v^r \cdot \delta_t} \\ &= k_{v,t}. \end{aligned}$$

4.2.3. *Analysis of the Scheme*

Now we are ready to prove that if the BDDH (BDH, respectively) problem is hard in groups generated by $\mathcal{G}$, then the TLPBC is secure in the sense of IND-ST (REC-ST, respectively).

Theorem 7. *The TLPBC is secure in the sense of* IND-ST, *assuming the BDDH problem is hard in groups generated by* $\mathcal{G}$.

Proof. We show that any polynomial-time adversary breaking the security of the scheme in the sense of IND-ST can be turned into a polynomial-time adversary solving the BDDH problem. Let Γ be a family of graphs corresponding to partially ordered hierarchies and let $G = (V, E)$ be any graph in Γ. Assume there exists a static adversary $\mathtt{STAT}_{v,t^*}$ whose advantage $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{v,t^*}}(1^\tau, G)$ is non-negligible. In the following we show how to construct a polynomial-time adversary A that, given an instance $(g, g^\alpha, g^\beta, g^\gamma, x)$ of the BDDH problem, uses the adversary $\mathtt{STAT}_{v,t^*}$ to decide whether $x = e(g,g)^{\alpha\cdot\beta\cdot\gamma}$. The adversary A, on input the instance $(g, g^\alpha, g^\beta, g^\gamma, x)$, constructs the inputs for the adversary $\mathtt{STAT}_{v,t^*}$ by means of a simulation of the scheme, as shown in the following. In order to construct the public information *pub* to be given as input to $\mathtt{STAT}_{v,t^*}$, the adversary A performs the following steps:

1. For each class $u^\ell \in V_\ell$, randomly chooses a value $\sigma_u^\ell \in Z_q$;
2. For each class $v^r \in V_r$, randomly chooses a value $\sigma_v^r \in Z_q$;
3. For each pair of classes connected by an edge, computes the public information according to the three following distinct cases:
   (a) For each class $u^\ell \in V_\ell$ such that $(u^\ell, v^r) \in E'$, computes the value $p_{u,v} = (g^\beta)^{\sigma_v^r/\sigma_u^\ell}$. Note that this means that the secret values π_u^ℓ and π_v^r associated with the classes u^ℓ and v^r during the initialization phase of the simulated scheme correspond to the values $\alpha \cdot \sigma_u^\ell$ and $\alpha \cdot \beta \cdot \sigma_v^r$, respectively.
   (b) For each pair of classes $(u^\ell, w^r) \in V_\ell \times V_r \setminus \{v^r\}$ such that $(u^\ell, w^r) \in E'$ and $(u^\ell, v^r) \in E'$, computes the public information $p_{u,w} = g^{\sigma_w^r/\sigma_u^\ell}$. Note that this means that the secret values π_w^r and π_u^ℓ associated with the classes w^r and u^ℓ during the initialization phase of the simulated scheme correspond to the values $\alpha \cdot \sigma_w^r$ and $\alpha \cdot \sigma_u^\ell$, respectively.
   (c) For each pair of classes $(u^\ell, w^r) \in V_\ell \times V_r \setminus \{v^r\}$ such that $(u^\ell, w^r) \in E'$ and $(u^\ell, v^r) \notin E'$, computes the public information $p_{u,w} = (g^\alpha)^{\sigma_w^r/\sigma_u^\ell}$. Note that this means that the secret values π_w^r and π_u^ℓ associated with the classes w^r and u^ℓ during the initialization phase of the simulated scheme correspond to the values $\alpha \cdot \sigma_w^r$ and σ_u^ℓ, respectively.

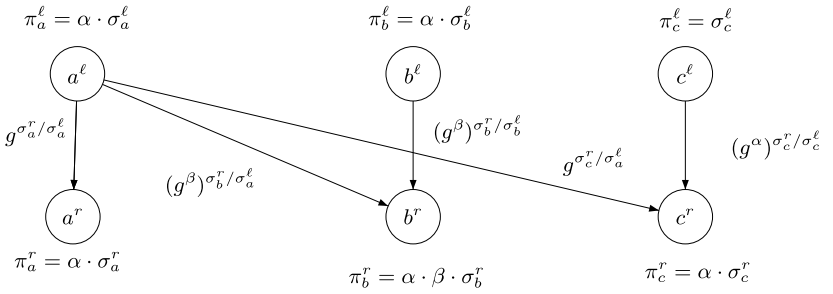

Fig. 4. The two-level hierarchy of Fig. 3 with the public information constructed by A and the secret values corresponding to the classes.

Observe that each pair of classes connected by an edge in E' is involved in exactly one of the above three cases. On the other hand, each single class may be involved in more than one case. However, it is easy to see that the secret value corresponding to each class is consistent with the others.

Figure 4 shows the two-level hierarchy of Fig. 3 with the public information constructed by A and the secret values corresponding to the classes, assuming b is the attacked class.

In order to construct the private information *corr* held by corrupted classes, to be given as input to $\mathtt{STAT}_{v,t^*}$, the adversary A performs the following steps:

1. For each time period $t \neq t^*$, randomly chooses a value $\delta_t \in Z_q$ and for each class $u^\ell \in V_\ell$, computes the private information $s_{u,t} = g^{\pi_u^\ell \cdot \delta_t}$, where the value π_u^ℓ corresponds either to $\alpha \cdot \sigma_u^\ell$ or to σ_u^ℓ according to the above construction. More precisely, we distinguish the following two cases:
   (a) For each class $u^\ell \in V_\ell$ such that $(u^\ell, v^r) \in E'$, A computes the value $s_{u,t} = (g^\alpha)^{\sigma_u^\ell \cdot \delta_t}$;
   (b) For each class $u^\ell \in V_\ell$ such that $(u^\ell, v^r) \notin E'$, A computes the value $s_{u,t} = g^{\sigma_u^\ell \cdot \delta_t}$.
2. For the time period t^*, randomly chooses a value $\varphi \in Z_q$ and for each class $u^\ell \in V_\ell$ such that $(u^\ell, v^r) \notin E'$, computes the private information $s_{u,t^*} = (g^\gamma)^{\sigma_u^\ell \cdot \varphi}$. Note that this means that the secret value δ_{t^*} associated with the time period t^* during the initialization phase of the simulated scheme corresponds to the value $\gamma \cdot \varphi$.

The last input for $\mathtt{STAT}_{v,t^*}$, corresponding either to the key k_{v,t^*} or to a random value having the same length as k_{v,t^*}, is computed as $x^{\sigma_v^r \cdot \varphi}$.

It is easy to see that the adversary $\mathtt{STAT}_{v,t^*}$'s view in the above simulation cannot be distinguished from the one obtained in a real execution of the scheme, since the random variables associated with such views are exactly the same. Moreover, all the computations needed to construct $\mathtt{STAT}_{v,t^*}$'s view can be performed in polynomial time.

Clearly, since $\mathtt{STAT}_{v,t^*}$ distinguishes the key k_{v,t^*} from a random string having the same length, with non-negligible advantage, it follows that the adversary A decides

whether x is equal to $e(g, g)^{\alpha \cdot \beta \cdot \gamma}$ with non-negligible advantage. Hence, the theorem holds. □

Following the lines of Theorem 7 we can prove that the next result also holds.

Theorem 8. *The TLPBC is secure in the sense of* REC-ST, *assuming the BDH problem is hard in groups generated by* $\mathcal{G}$.

4.2.4. *Performance Evaluation*

With respect to storage requirements, notice that the TLPBC requires a public value for each edge in the graph $G' = (V', E')$ used in the construction, thus the total number of public values is $|E'| = |E^*| = O(|V|^2)$, which does not depend on the number $|T|$ of time periods. This means that the number of time periods for which the scheme must be active does not need to be known in advance. Moreover, we stress that each public value is typically 171 bits long. On the other hand, each user belonging to a certain class for a time sequence has to store as many secret values as the number of time periods in the sequence. Hence, the number of private values for each user is $O(|T|)$. Moreover, users are required to evaluate the bilinear map at two given points, in order to perform key derivations. Finally, notice that BDH parameter generators believed to satisfy the BDH and BDDH assumptions can be efficiently constructed from the (modified) Weil [12] and Tate pairings [25] defined within elliptic or hyperelliptic curves over finite fields.

In the following, we show how to manage changes to the hierarchy, such as addition and deletion of nodes and edges, in such a way that no private information held by users need to be re-computed by the TA. Indeed, such updates can be handled by local changes to the public information.

Insertion of an edge Let (u, v) be an edge to be added to the hierarchy, starting from time period t_i through $t_{|T|}$. Such an update can be managed by the TA by adding the value $p_{u,v} = g^{\pi_v^r / \pi_u^\ell}$ to the public information *pub*.

Deletion of an edge Let (u, v) be an edge to be deleted from the hierarchy, starting from time period t_i through $t_{|T|}$. In order to forbid users belonging to class u from computing the key of class v in time period t_j, where $j = i, \ldots, |T|$, the TA has to assign a new key k'_{v,t_j} to v. This is done by choosing a new secret value for $\pi_v^r \in Z_q$ and computing k'_{v,t_j} according to such a value. On the other hand, in order to allow authorized users to compute such a new key, the TA has to update the public information *pub*, by recomputing the public value $p_{(w,v)}$, for each edge $(w, v) \in E$ according to the new value of $\pi_v^r \in Z_q$.

Insertion of a node Let u be a node to be inserted to the hierarchy, along with new incoming and outgoing edges, starting from time period t_i through $t_{|T|}$. The TA first chooses two random values $\pi_u^\ell, \pi_u^r \in Z_q$ and then computes the value $p_{(u,u)} = g^{\pi_u^r / \pi_u^\ell}$, which is added to the public information *pub*. Finally, the updates involving the addition of incoming and outgoing edges are managed by using the above procedure for edge insertions.

Deletion of a node Let u be a node to be deleted from the hierarchy, starting from time period t_i through $t_{|T|}$. The TA first uses the above procedure for edge deletions to delete all edges incident on u and then removes the node from V.

5. Comparisons

This section shows a comparison among the constructions proposed in this paper and all other provably-secure proposals which are summarized in Table 1.

Wang and Laih [45] and Tzeng [44] have shown how to construct a time-bound hierarchical key assignment scheme starting from the Akl–Taylor scheme. Since they did not formalize the definition of security and the adversarial model, it was not clear under which assumption their schemes could have been considered provably-secure. Subsequently, D'Arco et al. [20] have shown that these constructions are secure under the *RSA assumption*, provided that the parameters of the Akl–Taylor scheme are properly chosen. Unfortunately, the Akl–Taylor based schemes perform key derivation in a very expensive way. Indeed, although the key derivation procedure consists of only one modular division and one modular exponentiation, such operations become very impractical for most hierarchies because the numbers involved are very large. Moreover, Akl–Taylor based schemes only provide security with respect to key recovery.

Building on the present work, and using some constructions for hierarchical key assignment schemes without time constraints proposed in [23], two families of time-bound hierarchical key assignment schemes have been proposed in [24]: Time-Bound Encryption-Based Family (TBEBF) that uses as building block the encryption-based hierarchical key assignment scheme in [23]; Time-Bound Broadcast Encryption-Based Family (TBBEBF) which makes use of the broadcast encryption-based scheme in [23]. Such families exhibit a tradeoff among the amount of secret data that need to be distributed and stored by the users, the amount of data that need to be made public, the complexity of key derivation, and the computational assumption on which the security of the scheme is based.

Table 1. Comparisons among provably-secure time-bound key assignment schemes.

Scheme	Public info.	Private info.	Key derivation	Computational assumption	Security Notion
TLEBC §4.1	At most $\|V\|^2 \cdot \|T\|^3$	One	One decrypt.	IND-P1-C0 secure enc.	IND-ST
TLPBC §4.2	At most $\|V\|^2$	At most $\|T\|$	One pairing eval.	BDDH	IND-ST
Akl–Taylor based schemes [20,44,45]	$\|V\| \cdot \|T\|$	One	One modular division and one exponent.	RSA	REC-ST
TBEBF [24]	At least $\|E\| \cdot \|T\|$	At least one	At least PathLength decrypt.	IND-P1-C0 secure enc.	IND-ST
TBBEBF [24]	At least $\|V\| \cdot \|T\|$	At least one	One (complex) decrypt.	$(\|V\| \cdot \|T^2\|)$-BDDHE	IND-ST
Atallah et al. schemes [5]	At least $\|E\| \cdot \|T\|$	At most three	At least PathLength decrypt. and PRF eval.	IND-P1-C0 secure enc. + PRF	IND-ST

Atallah et al. [5] have shown a different framework to construct time-bound hierarchical key assignment schemes starting from any scheme without temporal constraints. By using as building block a scheme in [3] they have shown time-bound hierarchical key assignment schemes where each user has to store at most three private information and the amount of public information is inversely proportional to the complexity of the key derivation.

Table 1 shows lower bounds on the parameters exhibited by the TBEBF, TBBEBF and the Atallah's et al. schemes. We refer the reader to [5,24] for the exact values.

Compared to our schemes, all such constructions require a more expensive key derivation procedure. The key derivation procedures of both TBEBC and Atallah et al.'s schemes require a user in a class u to perform at least as many operations as the length of the path `PathLength` between class u and a class v, in order to derive a temporal key corresponding to class v. In particular, besides `PathLength` decryption operations, the key derivation procedure of the Atallah et al.'s schemes also needs to perform `PathLength` pseudorandom function (PRF) evaluations. Notice that the complex decryption needed by the key derivation procedure of the schemes in TBBEBF may require $O(|V| \cdot |T^2|)$ group operations. Moreover, the computational assumption required by the constructions in TBBEBC, which is the *m-Bilinear Decisional Diffie–Hellman Exponent* assumption (m-BDDHE) introduced in [13], is not as well studied as the assumptions on which the other proposals rely.

We remark that our solutions are optimal with respect to key derivation complexity and that no scheme in Table 1 is superior to the others with respect to all parameters. An open problem would be to find a time-bound hierarchical key assignment scheme which optimizes all parameters at the same time.

6. Summary and Extensions

In this paper we have designed and analyzed time-bound hierarchical key assignment schemes that are provably-secure and efficient. We have distinguished between two different goals: security with respect to *key indistinguishability* and against *key recovery*. We have also distinguished security against *static* and *adaptive* adversarial behaviors. Then, we have introduced two different constructions for time-bound key assignment schemes. The first one is based on symmetric encryption schemes, whereas the second one makes use of bilinear maps. Both schemes support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

In this paper we have considered *hierarchical* time-bound key assignment schemes, however, the model could be extended to the case where the graph G represents a general access control policy (i.e., which cannot be represented by a partially ordered hierarchy). Moreover, we have considered the case where the graph G has the same structure for any time period, since it represents the same access control policy. The model could be generalized to the case where there are different access control policies, one for each time period. For example, consider a web-based electronic newspaper company which offers several types of subscription packages, organized as a partially ordered hierarchy, where leaf nodes represent different topics. Assume that the newspaper company is going to offer some subscription packages in some fixed time periods. In such a case a user

may subscribe to a package only for the time periods in which the newspaper company offers it. Such a situation can be modeled by using a different graph to describe the access control policy for each time period. More precisely, for any $i = 1, \ldots, |T|$, we could represent the access control policy for time period t_i by a graph $G_i = (V_i, E_i)$, where V_i denotes the set of classes affected by the policy at time period t_i, whereas E_i represent the access relation between the classes.

Throughout this paper, for the sake of simplicity, we have analyzed the case usually considered in literature where the access control policy can be represented by a partially ordered hierarchy which is the same for any time period. However, all our results could be routinely extended to a more general setting.

Acknowledgements

We would like to thank the anonymous referees for their careful reading and useful comments.

References

[1] A.V. Aho, M.R. Garey, J.D. Ullman, The transitive reduction of a directed graph. *SIAM J. Comput.* **1**, 131–137 (1972)

[2] S.G. Akl, P.D. Taylor, Cryptographic solution to a problem of access control in a hierarchy. *ACM Trans. Comput. Syst.* **1**(3), 239–248 (1983)

[3] M.J. Atallah, M. Blanton, N. Fazio, K.B. Frikken, Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.* **12**(3) (2009). Preliminary version in Proc. of the ACM Conference on Computer and Communications Security 2006

[4] M.J. Atallah, M. Blanton, K.B. Frikken, Key management for non-tree access hierarchies, in *Proc. of the ACM Symposium on Access Control Models and Technologies* (2006), pp. 11–18

[5] M.J. Atallah, M. Blanton, K.B. Frikken, Incorporating temporal capabilities in existing key management schemes, in *ESORICS* (2007), pp. 515–530

[6] G. Ateniese, A. De Santis, A.L. Ferrara, B. Masucci, Provably-secure time-bound hierarchical key assignment schemes. Rep. 2006/225 at the IACR Cryptology ePrint Archive.

[7] G. Ateniese, A. De Santis, A.L. Ferrara, B. Masucci, Provably-secure time-bound hierarchical key assignment schemes, in *Proc. of the ACM Conference on Computer and Communications Security* (2006), pp. 288–297

[8] M. Bellare, P. Rogaway, Introduction to modern cryptography. Available as http://www.cs.ucdavis.edu/~rogaway/classes/227/fall03/book/index.html

[9] M. Bellare, R. Canetti, H. Krawczyk, Keying hash functions for message authentication, in *Proc. of Advances in Cryptology, Crypto*. Lecture Notes in Computer Science (1996), pp. 1–15

[10] M. Bellare, A. Desai, E. Jokipii, P. Rogaway, A concrete security treatment of symmetric encryption, in *Proc. of the 38th IEEE Symposium on Foundations of Computer Science* (1997), pp. 394–403

[11] E. Bertino, B. Carminati, E. Ferrari, A temporal key management scheme for secure broadcasting of XML documents, in *Proc. of the ACM Conference on Computer and Communications Security* (2002), pp. 31–40

[12] D. Boneh, X. Boyen, Efficient selective-ID secure identity-based encryption without random oracles, in *Advances in Cryptology—Eurocrypt*. Lecture Notes in Computer Science (2004), pp. 223–238

[13] D. Boneh, X. Boyen, E.-J. Goh, Hierarchical identity based encryption with constant size ciphertext, in *EUROCRYPT* (2005), pp. 440–456

[14] D. Boneh, M.K. Franklin, Identity-based encryption from the Weil pairing. *SIAM J. Comput.* **32**(3), 586–615 (2003)

[15] D. Boneh, B. Lynn, H. Shacham, Short signatures from the Weil pairing. *J. Cryptol.* **17**(4), 297–319 (2004)

[16] R. Canetti, S. Halevi, J. Katz, A forward-secure public-key encryption scheme, in *Proc. of Advances in Cryptology—Eurocrypt*. Lecture Notes in Computer Science, vol. 2656 (2003), pp. 255–271
[17] T. Chen, Y. Chung, Hierarchical access control based on Chinese remainder theorem and symmetric algorithm. *Comput. Secur.* **21**(6), 565–570 (2002)
[18] H.-Y. Chien, Efficient time-bound hierarchical key assignment scheme. *IEEE Trans. Knowl. Data Eng.* **16**(10), 1301–1304 (2004)
[19] J. Crampton, K. Martin, P. Wild, On key assignment for hierarchical access control, in *Proc. of the 19th IEEE Computer Security Foundations Workshop* (2006), pp. 98–111
[20] P. D'Arco, A. De Santis, A.L. Ferrara, B. Masucci, Variations on a theme by Akl and Taylor: security and tradeoffs. *Theor. Comput. Sci.* **411**(1), 213–227 (2010)
[21] A. De Santis, A.L. Ferrara, B. Masucci, Cryptographic key assignment schemes for any access control policy. *Inf. Process. Lett.* **92**(4), 199–205 (2004)
[22] A. De Santis, A.L. Ferrara, B. Masucci, Enforcing the security of a time-bound hierarchical key assignment scheme. *Inf. Sci.* **176**(12), 1684–1694 (2006)
[23] A. De Santis, A.L. Ferrara, B. Masucci, Efficient provably-secure hierarchical key assignment schemes, in *MFCS*, ed. by L. Kucera, A. Kucera. Lecture Notes in Computer Science, vol. 4708 (Springer, Berlin, 2007), pp. 371–382
[24] A. De Santis, A.L. Ferrara, B. Masucci, New constructions for provably-secure time-bound hierarchical key assignment schemes. *Theor. Comput. Sci.* **407**(1–3), 213–230 (2008)
[25] S.D. Galbraith, K. Harrison, D. Soldera, Implementing the tate pairing, in *Proc. of the Algorithmic Number Theory Symposium*. Lecture Notes in Computer Science (2000), pp. 385–394
[26] O. Goldreich, S. Goldwasser, S. Micali, How to construct random functions. *J. ACM* **33**(4), 792–807 (1986)
[27] S. Goldwasser, S. Micali, Probabilistic encryption. *J. Comput. Syst. Sci.* **28**(2), 270–299 (1984)
[28] S. Goldwasser, S. Micali, R.L. Rivest, A digital signature scheme secure against adaptive chosen-message attacks. *SIAM J. Comput.* **17**(2), 281–308 (1988)
[29] L. Harn, H.Y. Lin, A cryptographic key generation scheme for multilevel data security. *Comput. Secur.* **9**(6), 539–546 (1990)
[30] H.F. Huang, C.C. Chang, A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Comput. Stand. Interfaces* **26**, 159–166 (2004)
[31] M.S. Hwang, A cryptographic key assignment scheme in a hierarchy for access control. *Math. Comput. Model.* **26**(1), 27–31 (1997)
[32] A. Joux, A one round protocol for tripartite Diffie-Hellman, in *Proc. of the Algorithmic Number Theory Symposium* (2000), pp. 385–394
[33] J. Katz, M. Yung, Characterization of security notions for probabilistic private-key encryption. *J. Cryptol.* **19**(1), 67–95 (2006)
[34] H.T. Liaw, S.J. Wang, C.L. Lei, A dynamic cryptographic key assignment scheme in a tree structure. *Comput. Math. Appl.* **25**(6), 109–114 (1993)
[35] C.H. Lin, Dynamic key management schemes for access control in a hierarchy. *Comput. Commun.* **20**, 1381–1385 (1997)
[36] I.C. Lin, M.S. Hwang, C.C. Chang, A new key assignment scheme for enforcing complicated access control policies in hierarchy. *Future Gener. Comput. Syst.* **19**, 157–462 (2003)
[37] S.J. MacKinnon, P.D. Taylor, H. Meijer, S.G. Akl, An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Trans. Comput.* **34**(9), 797–802 (1985)
[38] A. Miyaji, M. Nakabayashi, S. Takano, New explicit conditions for elliptic curve traces for FR-reduction. *IEICE Trans. Fundam.* **E-84**(5) (2001)
[39] M. Naor, O. Reingold, Number-theoretic constructions of efficient pseudo-random functions. *J. ACM* **51**(2), 231–262 (2004)
[40] R.S. Sandhu, Cryptographic implementation of a tree hierarchy for access control. *Inf. Process. Lett.* **27**(2), 95–98 (1988)
[41] V. Shen, T. Chen, A novel key management scheme based on discrete logarithms and polynomial interpolations. *Comput. Secur.* **21**(2), 164–171 (2002)
[42] Q. Tang, C.J. Mitchell, Comments on a cryptographic key assignment scheme. *Comput. Stand. Interfaces* **27**, 323–326 (2005)
[43] W.-G. Tzeng, A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Trans. Knowl. Data Eng.* **14**(1), 182–188 (2002)

[44] W.-G. Tzeng, A secure system for data access based on anonymous and time-dependent hierarchical keys, in *Proc. of the ACM Symposium on Information, Computer and Communications Security* (2006), pp. 223–230
[45] S.-Y. Wang, C.- Laih, Merging: an efficient solution for a time-bound hierarchical key assignment scheme. *IEEE Trans. Dependable Secure Comput.* **3**(1), 91–100 (2006)
[46] T. Wu, C. Chang, Cryptographic key assignment scheme for hierarchical access control. *Int. J. Comput. Syst. Sci. Eng.* **1**(1), 25–28 (2001)
[47] J. Yeh, An RSA-based time-bound hierarchical key assignment scheme for electronic article subscription, in *Proc. of the ACM CIKM International Conference on Information and Knowledge Management* (2005), pp. 285–286
[48] J. Yeh, R. Chow, R. Newman, A key assignment for enforcing access control policy exceptions, in *Proc. of the International Symposium on Internet Technology* (1998), pp. 54–59
[49] X. Yi, Security of chien's efficient time-bound hierarchical key assignment scheme. *IEEE Trans. Knowl. Data Eng.* **17**(9), 1298–1299 (2005)
[50] X. Yi, Y. Ye, Security of Tzeng's time-bound key assignment scheme for access control in a hierarchy. *IEEE Trans. Knowl. Data Eng.* **15**(4), 1054–1055 (2003)