Provably-Secure Time-Bound Hierarchical Key Assignment Schemes*

Giuseppe Ateniese[1], Alfredo De Santis[2], Anna Lisa Ferrara[2], and Barbara Masucci[2]

[1]Department of Computer Science, The Johns Hopkins University, Baltimore, MD 21218, USA
[2]Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy

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 both the *unconditionally secure* and the *computationally secure* settings and distinguish between two different goals: security with respect to *key indistinguishability* and against *key recovery*.

- We first present definitions of security with respect to both goals in the unconditionally secure setting and we show tight lower bounds on the size of the private information distributed to each class.
- Then, we consider the computational setting and we further 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.
- Afterwards, we prove that a recently proposed scheme is insecure against key recovery.
- 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. These appear to be the first constructions for time-bound hierarchical key assignment schemes which are simultaneously *practical* and *provably-secure*.

Keywords: Access control, key assignment, provable security.

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

*A preliminary version of this paper appears in the proceedings of ACM CCS 2006 [5].

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 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. 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 generated by the central authority, 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, 27, 30, 33, 34, 36, 39]). The problem of designing key assignment schemes for access control policies not satisfying the anti-symmetric and transitive properties of a partially ordered hierarchy was considered in [18, 35, 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 [14, 41, 48, 46]. A recent work by Crampton et al. [17] provides a detailed classification of many schemes in the literature and evaluates the respective merits of different types of scheme. The schemes are evaluated according to several parameters, such as the amount of secret data that needs to be distributed to and stored by users, the amount of data that needs to be made public, the complexity of key derivation, the complexity of key updates and the resistance to collusive attacks.

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 the scheme is equivalent to the existence of an adversary B breaking the computational assumption. The usual method of construction of B uses the adversary A as a black-box. Atallah et al. [3] proposed a first provably-secure construction based on pseudorandom functions and a second one requiring the use of a symmetric encryption scheme secure against chosen-ciphertext attacks. Their constructions also manage with dynamic changes to the access hierarchy. In particular updates do not require any private information held by users to be redistributed. Atallah et al.[4] also considered the problem of improving the efficiency of key derivation in their schemes. Recently, two constructions for provably-secure key assignment schemes, improving those in [3], as well as new techniques for reducing key derivation time, have been proposed [19]. In particular, one construction provides constant private information and public information linear in the number of the classes, whereas, both constructions support dynamic changes to the hierarchy.

All the above schemes would 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 other than 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.

As pointed out by Tzeng [43], there are several applications requiring a 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. [8], who showed how to control access to an XML document according to temporal constraints.

A basic and straightforward way to achieve a time-based access control is to require each user to memorize encryption keys assigned to all classes lower down in the hierarchy for each time period in which the user is allowed to access their data. Tzeng [43] first addressed the problem of reducing the inherent complexity of such a solution and proposed a *time-bound* hierarchical key assignment scheme that requires each user to store information whose size does not depend on the number of keys that the user has access to or on the number of time periods. 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, whereby two or more users, assigned to some classes in distinct time periods, collude to compute a key to which they are not entitled [50]. Subsequently, Chien [15] proposed an efficient time-bound hierarchical key assignment scheme based on tamper-resistant devices. However, it was shown that malicious users can collusively misuse their devices to gain unauthorized accesses in [20, 49], where countermeasures were also proposed. Another time-bound hierarchical key assignment scheme was proposed by Huang and Chang [29] and later shown to be insecure against collusive attacks [42]. An RSA-based time-bound hierarchical key assignment scheme was proposed by Yeh [47], who claimed his scheme to be secure against collusive attacks. Recently, Wang and Laih [45] and Tzeng [44] showed 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 is not clear under which assumption their schemes can be considered provably secure.

1.1 Our Results

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. The two above security goals were first introduced by Atallah et al. [3] for hierarchical key assignment schemes. We extend their definitions to include temporal constraints.

- We first consider an *information-theoretic* approach to time-bound hierarchical key assignment schemes. In this setting, the key assigned to each class at a certain time period is *unconditionally secure*, with respect to one of the above security goals, against an adversary with unlimited computing power, controlling any coalition of classes not allowed to compute such a key. We present definitions of security with respect to each goal in the unconditionally secure setting and then we prove tight lower bounds on the size of the private information distributed to each class.

- Then, we address the problem of formalizing security requirements for time-bound hierarchical key assignment schemes in the computational setting and thus based on specific computational assumptions. We consider both *static* and *adaptive* 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.

- Afterwards, we prove that a recently proposed scheme [47] is insecure against collusive attacks.

- 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. These appear to be the first constructions for time-bound hierarchical key assignment schemes which are simultaneously *practical* and *provably-secure*.

The paper is organized as follows: in Section 2 we give an informal description of what a time-bound hierarchical key assignment scheme is, whereas, in Sections 3 and 4 we consider the unconditionally and the computationally secure settings, respectively. In particular, in Section 4.1 we give formal definitions of security for time-bound hierarchical key assignment schemes and explore the relations between them. In Section 4.2 we show a security weakness of a recently proposed scheme. In Sections 4.3 and 4.4 we describe our proposals for time-bound hierarchical key assignment schemes. Section 5 concludes the paper.

2 Time-bound Hierarchical Key Assignment Schemes

Consider a set of users divided into a number of disjoint classes, called *security classes*. A security class can represent a person, a department, or a user group in an organization. A binary relation $\preceq$ that partially orders the set of classes V is defined in accordance with authority, position, or power of each class in 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, it holds that $v \preceq v$, 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, 19, 39], etc.

In this paper we consider the case where a user may be in a class for only a period of time. We consider a sequence $T = (t_1, \ldots, t_{|T|})$ 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 nonempty 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 $t \in \lambda$. The key derivation process can be either *direct* or *indirect*. In the first case, each class v can compute the key $k_{u,t}$ held by any class $u \in A_v$ in a time period t without computing the keys of all classes along a path from v to u.

An ideal time-bound hierarchical key assignment scheme should have low storage requirements and provide for efficient key derivation and key update procedures. In addition, 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 $\{(v, \lambda) \in V \times \mathcal{P} : u \notin A_v$ or $t \notin \lambda\}$, corresponding to all users which are not allowed to compute the key $k_{u,t}$.

We refer to an *unconditionally secure* time-bound hierarchical key assignment scheme if its security relies on the theoretical impossibility of breaking it, despite the computational power of the coalition, whereas, we refer to a *computationally secure* time-bound hierarchical key assignment scheme if its security relies on the computational infeasibility of breaking it, according to some specific computational assumptions. We further distinguish two security goals: against *key recovery* and with respect to *key indistinguishability*. In the key recovery case, the adversarial coalition is not able to *compute* a key that should not be accessible by any user of the coalition, whereas, in the key indistinguishability case, the adversarial coalition is not even able to *distinguish* such a key from a random string of the same length. The two above security goals were first introduced by Atallah et al. [3] for hierarchical key assignment schemes in the computational setting. We extend their definitions to include temporal constraints.

3 The Unconditionally Secure Setting

In this section, we formally define unconditionally secure time-bound hierarchical key assignment schemes by using the entropy function (we refer the reader to [16] for a complete treatment of Information Theory), mainly because this leads to a compact and simple description of the schemes and because the entropy approach takes into account all probability distributions on the keys assigned to the classes. The same approach has been used in [22] to analyze key assignment schemes without temporal constraints, whose security is guaranteed with respect to the key indistinguishability requirement.

For any class $u \in V$ and any time sequence $\lambda \in \mathcal{P}$, we denote by $S_{u,\lambda}$ and $K_{u,t}$ the sets of all possible values that $s_{u,\lambda}$ and $k_{u,t}$ can assume, respectively. Given a set of pairs $X =$

$\{(u_1, \lambda_1), \cdots, (u_\ell, \lambda_\ell)\} \subseteq V \times \mathcal{P}$, we denote by S_X the set $S_{u_1,\lambda_1} \times \cdots \times S_{u_\ell,\lambda_\ell}$. In the following, with a boldface capital letter, say $\mathbf{Y}$, we denote a random variable taking values on a set, denoted by the corresponding capital letter Y, according to some probability distribution $\{Pr_{\mathbf{Y}}(y)\}_{y \in Y}$. The values such a random variable can take are denoted by the corresponding lower case letter. Given a random variable $\mathbf{Y}$, we denote by $H(\mathbf{Y})$ the Shannon entropy of $\{Pr_{\mathbf{Y}}(y)\}_{y \in Y}$. An unconditionally secure time-bound hierarchical key assignment scheme for a family Γ of graphs, corresponding to partially ordered hierarchies, is defined as follows.

Definition 3.1 *Let $G = (V, E)$ be a graph in Γ, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let $0 \leq \alpha \leq 1$. An α-unconditionally secure time-bound hierarchical key assignment scheme for Γ is a method to assign a private information $s_{u,\lambda}$ to each class $u \in V$, 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 $t \in T$, in such a way that the following two properties are satisfied:*

Correctness Each user *can compute* the key held by any class lower down in the hierarchy for each time period in which it belongs to its class.
Formally, for each class $v \in V$, each class $u \in A_v$, each time sequence $\lambda \in \mathcal{P}$, and each time period $t \in \lambda$, it holds that $H(\mathbf{K}_{u,t} | \mathbf{S}_{v,\lambda}) = 0$.

Security Any coalition of users *cannot compute / have absolutely no information about* any key the coalition is not entitled to obtain.
Formally, for each class $u \in V$, each time period $t \in T$, and each coalition of users $X \subseteq F_{u,t}$, it holds that $H(\mathbf{K}_{u,t} | \mathbf{S}_X) \geq \alpha \cdot H(\mathbf{K}_{u,t})$.

Notice that the correctness requirement is equivalent to saying that the values of the private information $s_{v,\lambda}$ held by each user belonging to a class $v \in V$ for a time sequence $\lambda \in \mathcal{P}$ correspond to a unique value of the key $k_{u,t}$, for each class $u \in A_v$ and each time period $t \in \lambda$. Moreover, the security requirement has different meanings, depending on the value of the parameter α. Indeed, if $\alpha = 1$, the requirement formalizes security with respect to key indistinguishability and is equivalent to saying that the probability that the unauthorized key is equal to $k_{u,t}$, given the values of the private information s_X held by the users in the coalition, is the same as the a priori probability that the key is $k_{u,t}$, i.e., the random variables $\mathbf{K}_{u,t}$ and $\mathbf{S}_X$ are statistically independent. On the other hand, if $0 < \alpha < 1$, the requirement formalizes security against key recovery and is equivalent to saying that the coalition is not able to *compute* the unauthorized key $k_{u,t}$, but could obtain some *partial information* about it, for example, it could be able to compute part of the key. Clearly, if $\alpha = 0$, Definition 3.1 does not formalize any security requirement, since the conditional entropy of $\mathbf{K}_{u,t}$ given $\mathbf{S}_X$ is always greater than or equal to zero.

In the following we show a tight lower bound on the size of the private information distributed to each user in any α-unconditionally secure time-bound hierarchical key assignment scheme. We will use the next definition.

Definition 3.2 *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 $0 \leq \alpha \leq 1$. In any α-unconditionally secure time-bound hierarchical key assignment scheme for Γ, a sequence of pairs $((u_1, r_1), \ldots, (u_\ell, r_\ell)) \in V \times T$ is called* well_ordered *if either $\ell = 1$, or $\ell > 1$ and for each $j = 2, \ldots, \ell$, it holds that $\{(u_i, r_i) \in V \times T : 1 \leq i \leq j - 1\} \subseteq F_{u_j, r_j}$.*

The next lemma will be a useful tool to prove our results.

Lemma 3.3 *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 $0 \le \alpha \le 1$. In any α-unconditionally secure time-bound hierarchical key assignment scheme for Γ, if $((u_1, r_1), \ldots, (u_\ell, r_\ell))$ is a well_ordered sequence of pairs in $V \times T$, then it holds that*

$$H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_\ell,r_\ell}) \ge H(\mathbf{K}_{u_1,r_1}) + \alpha \cdot \sum_{j=2}^{\ell} H(\mathbf{K}_{u_j,r_j}).$$

Proof. Let $X_j = \{(u_i, r_i) \in V \times T : 1 \le i \le j-1\}$, for any $j = 2, \ldots, \ell$. Since $((u_1, r_1), \ldots, (u_\ell, r_\ell))$ is a well_ordered sequence of pairs in $V \times T$, from Definition 3.2 we have that, $X_j \subseteq F_{u_j,r_j}$. Therefore, from the security requirement of Definition 3.1 it holds that

$$H(\mathbf{K}_{u_j,r_j} | \mathbf{S}_{X_j}) \ge \alpha \cdot H(\mathbf{K}_{u_j,r_j}). \tag{1}$$

From the correctness requirement of Definition 3.1 it holds that $H(\mathbf{K}_{u_i,r_i} | \mathbf{S}_{u_i,r_i}) = 0$ and from (13) we have that

$$H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{S}_{X_j}) \le \sum_{i=1}^{j-1} H(\mathbf{K}_{u_i,r_i} | \mathbf{S}_{X_j}) \le \sum_{i=1}^{j-1} H(\mathbf{K}_{u_i,r_i} | \mathbf{S}_{u_i,r_i}) = 0.$$

Hence, from (12) it follows that

$$H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{K}_{u_j,r_j} \mathbf{S}_{X_j}) \le H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{S}_{X_j}) = 0. \tag{2}$$

Consider the mutual information $I(\mathbf{K}_{u_j,r_j}; \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{S}_{X_j})$. From (11) it holds that

$$\begin{aligned} H(\mathbf{K}_{u_j,r_j} | \mathbf{S}_{X_j}) - H(\mathbf{K}_{u_j,r_j} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} \mathbf{S}_{X_j}) &= \\ H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{S}_{X_j}) - H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} | \mathbf{K}_{u_j,r_j} \mathbf{S}_{X_j}). \end{aligned} \tag{3}$$

Hence, from (2) and (3) it follows that

$$H(\mathbf{K}_{u_j,r_j} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} \mathbf{S}_{X_j}) = H(\mathbf{K}_{u_j,r_j} | \mathbf{S}_{X_j}). \tag{4}$$

Therefore, from (8) it holds that

$$\begin{aligned} H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_\ell,r_\ell}) &= H(\mathbf{K}_{u_1,r_1}) + \sum_{j=2}^{\ell} H(\mathbf{K}_{u_j,r_j} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}}) \\ &\ge H(\mathbf{K}_{u_1,r_1}) + \sum_{j=2}^{\ell} H(\mathbf{K}_{u_j,r_j} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{j-1},r_{j-1}} \mathbf{S}_{X_j}) \text{(from (12))} \\ &= H(\mathbf{K}_{u_1,r_1}) + \sum_{j=2}^{\ell} H(\mathbf{K}_{u_j,r_j} | \mathbf{S}_{X_j}) \text{(from (4))} \\ &\ge H(\mathbf{K}_{u_1,r_1}) + \alpha \cdot \sum_{j=2}^{\ell} H(\mathbf{K}_{u_j,r_j}) \text{(from (1))}. \end{aligned}$$

□

The next theorem shows a lower bound on the size of the private information distributed to each user. Such a result applies to the general case of arbitrary entropies of keys, but, for the sake of simplicity, we consider the case when all entropies of keys are equal. We denote this common entropy by $H(\mathbf{K})$.

Theorem 3.4 *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 $0 \leq \alpha \leq 1$. In any α-unconditionally secure time-bound hierarchical key assignment scheme for Γ, for any pair $(u, \lambda) \in V \times \mathcal{P}$, it holds that*

$$H(\mathbf{S}_{u,\lambda}) \geq (1 - \alpha + \alpha \cdot |A_u| \cdot |\lambda|) \cdot H(\mathbf{K}),$$

where $|\lambda|$ denotes the number of time periods in the time sequence λ.

Proof. Let u be a class and consider the directed acyclic graph $G' = (V', E')$ induced by G and involving all the classes in A_u. Moreover, let $(u_{|A_u|}, \ldots, u_1)$ be the sequence of classes output by the topological sorting on G'. This sequence has the property that for each pair of classes u_i, $u_j \in A_u$ such that $(u_j, u_i) \in E'$, the class u_j appears before that u_i in the ordering. Let $\lambda = (r_1, \ldots, r_{|\lambda|})$ be a time sequence. It is easy to see that the sequence of pairs $((u_1, r_1), \ldots (u_1, r_{|\lambda|}), \ldots, (u_{|A_u|}, r_1), \ldots, (u_{|A_u|}, r_{|\lambda|}))$ is well_ordered. Therefore, from (13) and from the correctness requirement of Definition 3.1, we have that

$$\begin{aligned} H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}} | \mathbf{S}_{u,\lambda}) &\leq \sum_{i=1}^{|A_u|} \sum_{j=1}^{|\lambda|} H(\mathbf{K}_{u_i,r_j} | \mathbf{S}_{u,\lambda}) \\ &\leq \sum_{i=1}^{|A_u|} \sum_{j=1}^{|\lambda|} H(\mathbf{K}_{u_i,r_j} | \mathbf{S}_{u_i,r_j}) \\ &= 0. \end{aligned} \tag{5}$$

Consider the mutual information $I(\mathbf{S}_{u,\lambda}; \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}})$. From (9) we have that

$$\begin{aligned} &H(\mathbf{S}_{u,\lambda}) - H(\mathbf{S}_{u,\lambda} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}}) \\ &\qquad = H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}}) - H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}} | \mathbf{S}_{u,\lambda}). \end{aligned} \tag{6}$$

Since $H(\mathbf{S}_{u,\lambda} | \mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}}) \geq 0$, from (5) and (6) it follows that

$$H(\mathbf{S}_{u,\lambda}) \geq H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}}).$$

From Lemma 3.3 we get

$$\begin{aligned} H(\mathbf{K}_{u_1,r_1} \ldots \mathbf{K}_{u_{|A_u|},r_{|\lambda|}}) &\geq H(\mathbf{K}) + \alpha \cdot (|A_u| \cdot |\lambda| - 1) \cdot H(\mathbf{K}) \\ &= (1 - \alpha + \alpha \cdot |A_u| \cdot |\lambda|) \cdot H(\mathbf{K}). \end{aligned}$$

Hence, the theorem follows. □

Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E) \in \Gamma$, let T be a sequence of distinct time periods, let $\mathcal{P}$ be the interval-set over T, and let $0 \leq \alpha \leq 1$ be a rational number, say $\alpha = a/b$, with a and b integers and $b \neq 0$. Moreover, let $q \geq 1$.

Initialization

1. If $a \neq b$, the TA randomly chooses a string $\eta \in \{0,1\}^{(b-a)\cdot q}$; if $a = b$, let η be the empty string;
2. Afterwards, if $a \neq 0$, for any class $v \in V$ and any time period $t \in T$, the TA randomly chooses a string $k'_{u,t} \in \{0,1\}^{a \cdot q}$; if $a = 0$, let $k'_{u,t}$ be the empty string;
3. Then, the TA computes the key $k_{u,t} = \eta || k'_{u,t}$, where $||$ denotes string concatenation;
4. When a user is assigned to a class $u \in V$ for a time sequence $\lambda \in \mathcal{P}$, the TA delivers to the user, by means of a secure channel, the private information $s_{u,\lambda}$, containing the string η, as well as the string $k'_{v,t}$, for any class $v \in A_u$ and any time period $t \in \lambda$.

Key derivation

Each user belonging to a class $u \in V$ for a time sequence $\lambda \in \mathcal{P}$ can use its private information $s_{u,\lambda}$ to compute the key $k_{v,t} = \eta || k'_{v,t}$ for any class $v \in A_u$ and any time period $t \in \lambda$, since both strings η and $k'_{v,t}$ are contained in $s_{u,\lambda}$.

Figure 1: An α-unconditionally secure key assignment scheme.

The bound of Theorem 3.4 is tight. Indeed, in Figure 1 we describe an α-unconditionally secure key assignment scheme which meets it with equality.

In the following we show that the scheme of Figure 1 satisfies the security requirement of Definition 3.1. Indeed, let $u \in V$ be a class, $t \in T$ be a time period, and $X \subseteq F_{u,t}$ be a coalition of corrupted users trying to compute the key $k_{u,t}$. We distinguish two cases:

1. Case $\alpha = 1$.
   The key $k_{u,t}$ is independent from the private information s_X held by the coalition, hence, the corrupted users have absolutely no information about $k_{u,t}$.

2. Case $0 < \alpha < 1$.
   The key $k_{u,t}$ is equal to $\eta || k'_{u,t}$. Since the string η is part of the private information s_X and the string $k'_{u,t}$ is randomly chosen by the TA, the uncertainty on $\mathbf{K}_{u,t}$, given $\mathbf{S}_X$, is equal to the uncertainty on $\mathbf{K}'_{u,t}$. Since $H(\mathbf{K}'_{u,t}) = a \cdot q$ and $H(\mathbf{K}_{u,t}) = b \cdot q$, it follows that $H(\mathbf{K}_{u,t} | \mathbf{S}_X) = \alpha \cdot H(\mathbf{K}_{u,t})$.

It is easy to see that the scheme of Figure 1 meets the bound of Theorem 3.4 with equality. Consider a user belonging to a class $u \in V$ for a time sequence $\lambda \in \mathcal{P}$. If $\alpha = 1$, the size of the private information $s_{u,\lambda}$ is equal to $|A_u| \cdot |\lambda| \cdot a \cdot q$ bits, whereas, the size of each key is equal to $a \cdot q$ bits. On the other hand, when $\alpha = 0$, $s_{u,\lambda}$ contains a key, having size $b \cdot q$ bits, which is the same for each class and each time period. Finally, if $0 < \alpha < 1$, $s_{u,\lambda}$ consists of $(b - a + a \cdot |A_u| \cdot |\lambda|) \cdot q$ bits, whereas, the size of each key is equal to $b \cdot q$ bits.

4 The Computationally Secure Setting

In this section, we consider time-bound key assignment schemes based on specific computational assumptions. In this setting, we obtain several results of interest. We first provide notions of security with respect to *key indistinguishability* and *key recovery* and consider attacks carried out by *static* or *adaptive* adversaries. Then, we explore all possible relations between different notions of security and types of adversary. We show that security against adaptive adversaries is polynomially equivalent to security against static ones. Afterwards, we prove that Yeh's scheme [47] is insecure against collusive attacks. Finally, motivated by the need for *provably-secure* schemes, we propose two different constructions of 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 are provably-secure and efficient.

4.1 Notions of Security

We use the standard notation to describe probabilistic algorithm and experiments following [26]. 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 $\epsilon : N \to R$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

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

Definition 4.1 *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 *on inputs* 1^τ*,* G*, and* $\mathcal{P}$*, 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

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

Notice that in Definition 4.1 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 with those parts of them involved in the computation.

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.

Security against adaptive adversaries for hierarchical key assignment schemes with no temporal constraints has been first considered by Atallah et al. [3]. 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 viceversa. Figure 2 summarizes our results.

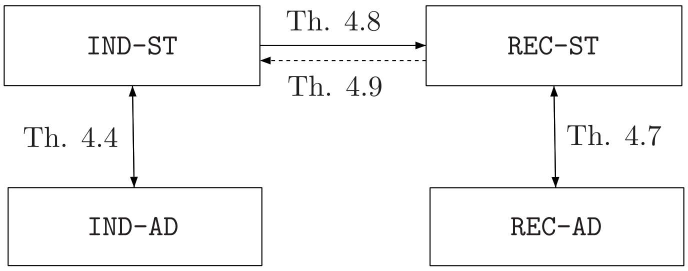

Figure 2: Hierarchy of security notions for time-bound hierarchical key assignment schemes. A solid line from notion A to notion B means that any scheme meeting notion A also meets notion B, whereas, a broken line indicates that a scheme meeting notion A does not necessarily meet notion B.

4.1.1 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 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 4.2 `[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 $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$	Experiment $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$
$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$	$(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$
$corr \leftarrow Corrupt_{u,t}(s)$	$corr \leftarrow Corrupt_{u,t}(s)$
$d \leftarrow \mathtt{STAT}_{u,t}(1^\tau, G, \mathcal{P}, pub, corr, k_{u,t})$	$\rho \leftarrow \{0,1\}^{length(k_{u,t})}$
return d	$d \leftarrow \mathtt{STAT}_{u,t}(1^\tau, G, \mathcal{P}, pub, corr, \rho)$
	return d

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

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

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 in 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. It is the adversary's job to determine whether the received 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 4.3 `[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 $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P})$ $(s, k, pub) \leftarrow Gen(1^\tau, G, \mathcal{P})$ $(u, t, state) \leftarrow \mathtt{ADAPT}_1^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub)$ $\rho \leftarrow \{0,1\}^{length(k_{u,t})}$ $d \leftarrow \mathtt{ADAPT}_2^{\mathcal{O}_s(\cdot)}(1^\tau, G, \mathcal{P}, pub, u, t, state, \rho)$ **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*

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

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.*

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

Theorem 4.4 [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 $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{ADAPT}}$ 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 $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,t}}$ 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

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

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

4.1.2 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 $t \in T$ and which is able to corrupt *all* users not allowed to compute the key $k_{u,t}$. 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* held by corrupted users, 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 $1/2^{length(k_{u,t})}$.

Definition 4.5 [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:*

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

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

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

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* $\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{STAT}_{u,t}}(1^\tau, G, \mathcal{P})$ *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 in 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. Finally, it 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 $1/2^{length(k_{u,t})}$.

Definition 4.6 [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* $\mathtt{ADAPT} = (\mathtt{ADAPT}_1, \mathtt{ADAPT}_2)$ *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 experiment:*

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

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*

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

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 $\mathbf{Adv}^{\mathtt{REC}}_{\mathtt{ADAPT}}(1^\tau, G, \mathcal{P})$ is negligible, for each adaptive adversary* ADAPT *whose time complexity is polynomial in τ.*

The next result can be proved following the lines of the proof of Theorem 4.4.

Theorem 4.7 [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 any static adversary which breaks the security of the key assignment scheme for Γ in the sense of REC-ST can be easily turned into an adversary which breaks the security of the key assignment scheme for Γ in the sense of IND-ST. Hence, the next result holds.

Theorem 4.8 [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*.*

In the following we show that security against key recovery does not necessarily imply security with respect to key indistinguishability. Let (Gen, Der) be a time-bound hierarchical key assignment scheme for a family of graphs Γ which is secure in the sense of REC-ST. We construct another scheme (Gen', Der') for Γ and we show that it is secure in the sense of REC-ST but is not secure in the sense of IND-ST. Let $G = (V, E)$ be a graph in Γ, let $u \in V$ be a class and let $t \in T$ be a time period. Let $k_{u,t}$ be the key assigned by Gen to u in time period t. Algorithm Gen' randomly chooses a bit b and computes the key $k'_{u,t}$ by concatenating b and $k_{u,t}$. All other values computed by Gen' are exactly the same as the ones computed by Gen, with the exception of the public information pub', which also includes the bit b. Algorithm Der' differs from Der in the fact that the bit b contained in pub' has also to be considered when deriving $k_{u,t}$. Let $\mathtt{STAT}_{u,t}$ be a static adversary that simply checks whether the first bit x_0 of the challenge x, which corresponds either to the key $k'_{u,t}$ or to a random string having the same length as $k'_{u,t}$, is equal to the bit b, which is included in pub'. If $x_0 = b$, then $\mathtt{STAT}_{u,t}$ outputs 1, otherwise, it outputs 0. It is easy to see that $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_{u,t}}(1^\tau)$ is non-negligible, hence (Gen', Der') is not secure in the sense of IND-ST. On the other hand, (Gen', Der') is secure in the sense of REC-ST. Assume by contradiction that (Gen', Der') is not secure in the sense of REC-ST. It follows that also (Gen, Der) is not secure in the sense of REC-ST. This is a contradiction. Hence, the next result holds.

Theorem 4.9 [REC-ST$\not\Rightarrow$IND-ST] *Let Γ be a family of graphs corresponding to partially ordered hierarchies. 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 2 shows the hierarchy of security definitions for time-bound key assignment schemes, resulting from Theorems 4.4, 4.7, 4.8, and 4.9.

4.2 A Collusion Attack to Yeh's Scheme

In this section we show a security weakness of Yeh's scheme, relying in the fact that in some cases a coalition of users is able to compute some encryption keys that they should not be able to access. Yeh's scheme is described in Figure 3.

In order to show our attack we need the next lemma, which is a simple generalization of a result due to Shamir [40].

Lemma 4.10 *Let n be the product of two distinct large primes. Given four integers $\alpha, \beta \in Z_n^*$ and $x, y \in Z$, such that $\beta^x = \alpha^y \bmod n$, it is easy to compute $\gamma \in Z_n^*$ such that $\gamma^x \bmod n = \alpha^{gcd(x,y)} \bmod n$.*

Proof. Let $\delta = gcd(x, y)$. By the extended Euclid's algorithm it is easy to compute two integers a and b such that $\delta = ax + by$. Let $\gamma = \beta^b \cdot \alpha^a$. It is easy to see that $\gamma^x \bmod n = \alpha^\delta \bmod n$. Indeed

$$\begin{aligned}(\beta^b \cdot \alpha^a)^x \bmod n &= \alpha^{by+ax} \bmod n \text{ (since } \beta^x = \alpha^y \bmod n) \\ &= \alpha^{by+ax-\delta} \cdot \alpha^\delta \bmod n \\ &= \alpha^\delta \bmod n.\end{aligned}$$

Thus, the lemma holds. □

Consider the hierarchy of Figure 4 and let A and B be two users assigned to classes u and v in time sequences (t_1, t_2) and (t_2, t_3), respectively. Moreover, let $gcd(e_u, g_{t_3}) = 1$. In the following we show how users A and B can collude to compute the key k_{u,t_3}, that they should not be able to obtain. Let $k_{u,t_2} = k_0^{d_u d_w h_{t_2}} \bmod n$ and $s_{v,(t_2,t_3)} = k_0^{d_v d_w h_{t_2} h_{t_3}} \bmod n$. Moreover, let $\beta = k_{u,t_2}$ and $\alpha = (s_{v,(t_2,t_3)})^{e_v} \bmod n$. Since $\beta^{e_u} = \alpha^{g_{t_3}} \bmod n$, from Lemma 4.10 users A and B can efficiently compute the value γ such that $\gamma^{e_u} \bmod n = \alpha$, that is, $\gamma = k_0^{d_u d_w h_{t_2} h_{t_3}} \bmod n$. Thus, they can compute the key $k_{u,t_3} = k_0^{d_u d_w h_{t_3}} \bmod n = \gamma^{g_{t_2}} \bmod n$.

More generally, let u and v be two distinct classes such that $v \notin A_u$ and $A_u \setminus \{u\} \subseteq A_v$. Let A and B be two users assigned to classes u and v in time sequences $(t_x, \ldots, t_y)$ and $(t_i, \ldots, t_j)$, respectively, where $t_1 \le t_x < t_i \le t_y < t_j \le t_{|T|}$. Let $t_i \le t \le t_y$ and $g_t = \delta \cdot \rho$, where $\rho \ge 1$ and

Let Γ be a family of graphs corresponding to partially ordered hierarchies. Let $G = (V, E) \in \Gamma$ be a graph, let $T = (t_1, \ldots, t_{|T|})$ be a sequence of distinct time periods, and let $\mathcal{P}$ be the interval-set over T.

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

1. Randomly choose two distinct large primes p and q and compute $n = p \cdot q$, having bitlength τ, and $\phi(n) = (p-1)(q-1)$;
2. For each class $u \in V$, randomly choose a public integer e_u such that $gcd(e_u, \phi(n)) = 1$;
3. For each time period $t \in T$, randomly choose a public integer g_t such that $gcd(g_t, \phi(n)) = 1$;
4. Let pub be the sequence of public information computed in the previous two steps;
5. For each class $u \in V$, compute the secret integer d_u, such that $e_u \cdot d_u = 1 \bmod \phi(n)$;
6. For each time period $t \in T$, compute the secret integer h_t such that $g_t \cdot h_t = 1 \bmod \phi(n)$;
7. Choose a random integer k_0, where $1 < k_0 < n$, and for each class $u \in V$ compute a class key $k_u = k_0^{\prod_{v \in A_u} d_v} \bmod n$;
8. For each class $u \in V$ and each time sequence $\lambda \in \mathcal{P}$, compute the private information $s_{u,\lambda} = k_u^{\prod_{r \in \lambda} h_r} \bmod n$;
9. For each class $u \in V$ and each time period $t \in T$, compute the key $k_{u,t} = k_u^{h_t} \bmod n$;
10. Let s and k be the sequences of private information and keys, respectively, computed in the previous steps;
11. Output (s, k, pub).

Algorithm $Der(1^\tau, G, \mathcal{P}, u, v, \lambda, s_{u,\lambda}, t, pub)$
Compute the key $k_{v,t}$ as

$$(s_{u,\lambda})^{\prod_{w \in A_u \setminus A_v} e_w \prod_{r \in \lambda \ \& \ r \neq t} g_r} = k_v^{h_t} \bmod n = k_{v,t}.$$

Figure 3: Yeh's time-bound hierarchical key assignment scheme.

$\delta = gcd(e_u, g_{t'})$, for some $t_y < t' \leq t_j$. In the following we show how users A and B can collude to compute the key $k_{u,t'}$, that they should not be able to obtain. Let $\beta = k_{u,t}$ and

$$\alpha = \left(s_{v,(t_i,t_j)}\right)^{\prod_{r \in (t_i, \ldots, t_j) \ \& \ r \neq t, t'} g_r \prod_{w \in A_v \setminus A_u} e_w} \bmod n.$$

It is easy to see that

$$\beta^{e_u} = k_0^{h_t \prod_{w \in A_u \setminus \{u\}} d_w} \bmod n = \alpha^{g_{t'}} \bmod n.$$

Hence, from Lemma 4.10 users A and B can efficiently compute the value γ such that $\gamma^{e_u} \bmod n = \alpha^\delta \bmod n$, that is, $\gamma = k_0^{h_t \cdot h_{t'} \cdot \delta \prod_{w \in A_u} d_w} \bmod n$. Afterwards, they can compute the value

$$\gamma^\rho \bmod n = k_0^{h_{t'} \prod_{w \in A_u} d_w} \bmod n = k_{u,t'}.$$

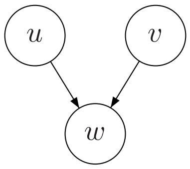

Figure 4: A partially ordered hierarchy.

4.3 A Scheme based on Symmetric Encryption Schemes

In this section we first show how to construct a time-bound key assignment scheme (Gen, Der) using as a building block a symmetric encryption scheme. Afterwards, we prove that the security property of the resulting time-bound key assignment scheme depends on the security property of the underlying encryption scheme. We first recall the definition of a symmetric encryption scheme.

Definition 4.11 *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 that* $\mathcal{D}(1^\tau, key, y) = m$*.*

In Figure 6 we describe a time-bound hierarchical key assignment scheme using as a building block a symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. The first step of the algorithm *Gen* performs 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., $(u_\lambda, u_t) \in E_{\mathcal{P}T}$;
- 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 $G_{\mathcal{P}T}$, i.e., $(u_\lambda, v_t) \in E_{\mathcal{P}T}$.

Figure 5 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 $\lambda_3 = (t_2)$.

Notice that in the two-level partially ordered hierarchy obtained by the above transformation the classes at the first level do not need to be assigned encryption keys, since they have no data to be protected. On the other hand, the classes at the second level do not need to perform key derivations, since there are no classes that can be accessed by them.

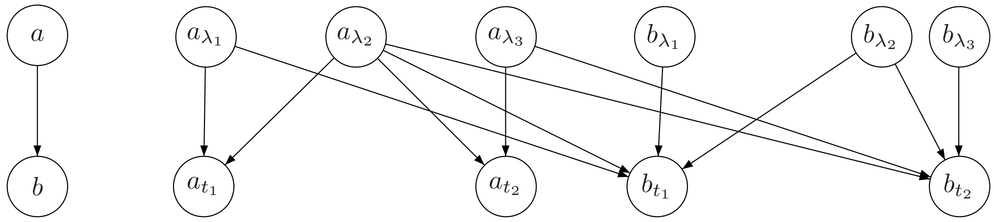

Figure 5: The graph transformation used in our construction.

4.3.1 Analysis of the Scheme

In the following we show that the security property of the time-bound key assignment scheme of Figure 6 depends on the security property of the underlying encryption scheme. We first need to 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 [25]. 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 [32], if every polynomial-time adversary A, which has access to the encryption oracle only during the first stage of the attack and has never 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 4.12 [`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 has never access to the decryption oracle. Consider the following two experiments:*

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

The advantage of A is defined as

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

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

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 $G_{\mathcal{P}T} = (V_{\mathcal{P}T}, E_{\mathcal{P}T})$, where $V_{\mathcal{P}T} = V_{\mathcal{P}} \cup V_T$;
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 $(u_\lambda, v_t) \in E_{\mathcal{P}T}$, 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 $k_{v,t} = \mathcal{D}_{s_{u,\lambda}}(p_{(u,\lambda),(v,t)})$.

Figure 6: A time-bound hierarchical key assignment scheme based on a symmetric encryption scheme.

In the following we consider a weaker definition of security. 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, denoted by `PR-P1-C0`, if every polynomial-time adversary A, which has access to the encryption oracle and has never access to the decryption oracle, succeeds in determining the plaintext x corresponding to the challenge ciphertext y with probability only negligibly different from $1/2^{length(x)}$.

Definition 4.13 `[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:*

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

The advantage of A is defined as

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

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

Now we are ready to show that if the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of IND-P1-CO (PR-P1-CO, respectively), then our time-bound key assignment scheme is secure in the sense of IND-ST (REC-ST, respectively).

Theorem 4.14 *If the encryption scheme* $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ *is secure in the sense of* IND-P1-CO*, then the time-bound key assignment scheme of Figure 6 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 $u_{t^*} \in V_{\mathcal{T}}$ 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 scheme of Figure 6 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 q-th one. Hence, it follows that also 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 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-CO. The algorithm A_1, on input 1^τ, makes queries to the encryption oracle $\mathcal{E}_{key}(\cdot)$ 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 $A_1^{\mathcal{E}_{key}(\cdot)}(1^\tau)$
$x_0, x_1 \leftarrow \{0,1\}^\tau$
//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 $w_t \in V_\mathcal{T} \setminus \{u_{t^*}\}$
$k_{w,t} \leftarrow \{0,1\}^\tau$
//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 $w_t \in V_\mathcal{T} \setminus \{u_{t^*}\}$ such that $(v_{q_{\lambda_q}}, w_t) \in E_{\mathcal{P}T}$
$p_{(v_q,\lambda_q),(w,t)} \leftarrow \mathcal{E}_{key}(k_{w,t})$
for any two classes $z_\lambda \in V_\mathcal{P} \setminus \{v_{q_{\lambda_q}}\}$ and $w_t \in V_\mathcal{T} \setminus \{u_{t^*}\}$ 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 to the adversary's view is exactly the same as the one associated to 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 to 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 q-th one. Therefore, the adversary's view in experiment $\mathbf{Exp}_{\mathtt{STAT}_{u,t^*}}^{\mathtt{IND-1}}$ is indistinguishable from the adversary's view in experiment $\mathbf{Exp}_{\mathtt{STAT}_{u,t^*}}^{\mathtt{IND-0}}$. This concludes the proof. □

Following the lines of Theorem 4.14 we can prove that the next result also holds.

Theorem 4.15 *If the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of* `PR-P1-C0`*, then the time-bound key assignment scheme of Figure 6 is secure in the sense of* `REC-ST`*.*

4.3.2 Performance Evaluation

In this section we evaluate the scheme of Figure 6, taking into account several parameters, such as space requirements for public and private information storage, computational requirements for key derivation, and security. Regarding space requirements, the scheme requires a public value for each edge in the graph $G_{\mathcal{PT}}$ used in the construction. It is easy to see that $|E_{\mathcal{PT}}| = O(|V|^2) \cdot \sum_{i=1}^{|T|} i \cdot (|T| - i + 1) = O(|V|^2 \cdot |T|^3)$. More precisely, $|E_{\mathcal{PT}}| = O(|E^*| \cdot |T|^3)$, 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 scheme secure in the sense of `IND-ST` we construct an encryption scheme secure in the sense of `IND-P1-C0`. To this aim, we could use a *pseudorandom function family*, an important cryptographic primitive originally defined by Goldreich, Goldwasser, and Micali [24]. Loosely speaking, a distribution of functions is pseudorandom if it satisfies the following requirements: 1) It is easy to sample a function according to the distribution and to evaluate it at a given point; 2) It is hard to tell apart a function sampled according to the distribution from a uniformly distributed function, given access to the function as a block-box. Since pseudorandom functions have a wide range of applications, the problem of designing efficient constructions for such functions has received considerable attention. A first construction, based on pseudorandom generators, was proposed in [24]. It is well known that pseudorandom generators can be constructed from one-way functions [9, 28]. The two more efficient constructions were proposed by Naor and Reingold [38]. In their constructions, the cost of evaluating such functions is comparable to two modular exponentiations.

Consider the following construction, called the *XOR construction* [7], of a symmetric encryption scheme $\Pi_{XOR,\mathcal{F}} = (\mathcal{K}_{XOR}, \mathcal{E}_{XOR}, \mathcal{D}_{XOR})$ which is based on a pseudorandom function family $\mathcal{F} : \{0,1\}^\tau \times \{0,1\}^\tau \to \{0,1\}^\tau$:

- The key generation algorithm $\mathcal{K}_{XOR}$ outputs a random τ-bit key ρ for the pseudorandom function family $\mathcal{F}$, thus specifying a function F_ρ of the family.
- The encryption algorithm $\mathcal{E}_{XOR}$ considers the message x to be encrypted as a sequence of τ-bits blocks $x = x_1 \cdots x_n$ (padding is done on the last block, if necessary), chooses a random string r of τ bits and computes, for $i = 1, \ldots, n$ the value $y_i = F_\rho(r + i) \oplus x_i$. The ciphertext is $r||y_1 \cdots y_n$, where $||$ denotes string concatenation.
- The decryption algorithm $\mathcal{D}_{XOR}$, on input a ciphertext z, parses it as $r||y_1 \cdots y_n$ and computes, for $i = 1, \ldots, n$ the value $x_i = F_\rho(r + i) \oplus y_i$. The corresponding plaintext is $x = x_1 \cdots x_n$.

The encryption scheme $\Pi_{XOR,\mathcal{F}}$ has been shown to be secure in the sense of `IND-P1-C0` (see [7, 32]), assuming that $\mathcal{F}$ is a pseudorandom function family. Therefore, $\Pi_{XOR,\mathcal{F}}$ could be used to obtain a time-bound hierarchical key assignment scheme secure in the sense of `IND-ST`. An efficient implementation of the resulting time-bound hierarchical key assignment scheme could be obtained by using the HMAC [6] to realize the pseudorandom function family $\mathcal{F}$.

Notice that if the message x to be encrypted has length τ, the XOR construction reduces to compute the ciphertext as $r||y$, where $y = F_\rho(r) \oplus x$ and r is a random string of τ bits. Such a construction has been used by Atallah et al. [3] to design a hierarchical key assignment scheme without temporal constraints. In their scheme, for each edge $(u, v) \in E$ there is a public value $y_{u,v} = F_{k_u}(\ell_v) \oplus k_v$, corresponding to the encryption of the key k_v assigned to class v, where the key k_u specifies a function F_{k_u} of the pseudorandom function family $\mathcal{F}$, and ℓ_v is a public label associated to v.

4.3.3 Handling Dynamic Changes

In this section 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 $i \leq x \leq y$, 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 $j = 1, \ldots, |T|$, 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.

4.4 A Scheme based on Bilinear Maps

In this section we design a time-bound hierarchical key assignment where the amount of public information does not depend on the number of time periods. Our scheme uses as a building block a bilinear map between groups. Bilinear maps have been used in cryptography to construct key exchange schemes [31], public-key cryptosystems [10, 11, 13], signature schemes [12], etc. We first recall the definition of a bilinear map.

Definition 4.16 *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 two groups G_1 and G_2 of order q, and the description of a bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The running time of $\mathcal{G}$ is polynomial in τ. We denote the output of $\mathcal{G}$ by $\mathcal{G}(1^\tau) =< q, G_1, G_2, e >$.

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 [37]). 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 in a two-level partially ordered hierarchy having the above features, using a technique proposed in [18]. 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';
- 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 7 shows an example of the graph transformation described above.

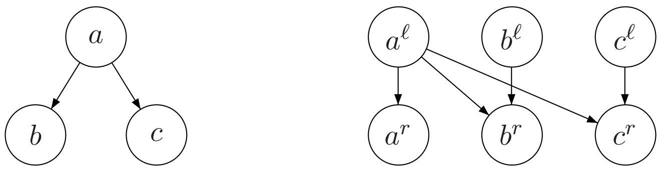

Figure 7: The graph transformation used in our construction.

Our scheme based on bilinear maps is described in Figure 8.

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

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

Figure 8: A time-bound hierarchical key assignment scheme based on a bilinear map.

4.4.1 Analysis of the Scheme

The *Bilinear Diffie-Hellman Problem (BDH)* in $< G_1, G_2, e >$ is as follows: 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 $e(g, g)^{\alpha \cdot \beta \cdot \gamma} \in G_2$. Such a problem has been introduced in [11].

Definition 4.17 (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*

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

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 $\mathbf{Adv}^{\mathtt{BDH}}_{\mathcal{G},A}(1^\tau)$ 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: 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 $x = e(g,g)^{\alpha\cdot\beta\cdot\gamma}$. Such a problem has been introduced in [11].

Definition 4.18 (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*

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

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 $\mathbf{Adv}^{\mathtt{BDDH}}_{\mathcal{G},A}(1^\tau)$ is negligible, for each randomized algorithm A whose time complexity is polynomial in 1^τ.

Now we are ready to prove that if the BDDH problem is hard in groups generated by $\mathcal{G}$, then our time-bound key assignment scheme is secure in the sense of `IND-ST`.

Theorem 4.19 *The time-bound hierarchical key assignment scheme of Figure 8 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 $\mathtt{STAT}_{u,t^*}$ 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 to 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 to 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 to 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.

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. Clearly, such secret values cannot be computed by the adversary A, but we have outlined the correspondence between each class and its secret value in order to fuel intuition over the reader. Figure 4.4.1 shows the two-level hierarchy of Figure 7 with the public information constructed by A and the secret values corresponding to the classes, assuming b^r is the attacked class.

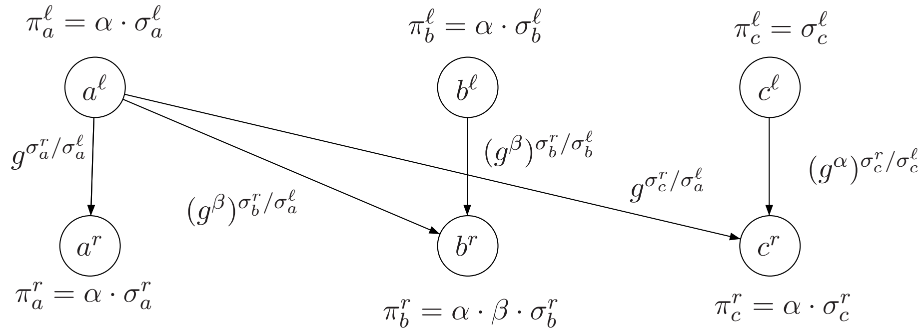

Figure 9: The two-level hierarchy of Figure 7 with the public information constructed by A and the secret values corresponding to the classes.

In order to construct the private information *corr* held by corrupted classes, to be given as input to $\texttt{STAT}_{u,t^*}$, 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 to 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 to 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. □

4.4.2 Performance Evaluation

With respect to storage requirements, notice that the scheme 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 satify the BDH and BDDH assumptions can be efficiently constructed from the (modified) Weil [10] and Tate pairings [23] defined within elliptic or hyperelliptic curves over finite fields.

4.4.3 Handling Dynamic Changes

In this section 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 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 considered both the *unconditionally secure* and the *computationally secure* settings and we have distinguished between two different goals: security with respect to *key indistinguishability* and against *key recovery*. In the computational setting, we have further distinguished security against *static* and *adaptive* adversarial behaviors. After showing that a recently-proposed scheme is insecure, 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. Figure 10 shows a summary of the constructions proposed in this paper.

Scheme	Public info.	Private info.	Key derivation	Operation type	Computational assumption
Unconditionally secure	None	$O(\|V\| \cdot \|T\|)$	Direct	String concat.	None
Encryption based	$O(\|V\|^2 \cdot \|T\|^3)$	One	Direct	Decryption	IND-P1-C0 secure encryption
Pairing based	$O(\|V\|^2)$	$O(\|T\|)$	Direct	Pairing eval.	BDDH

Figure 10: Summary of the constructions proposed in this paper.

Building on this work, and using some constructions for hierarchical key assignment schemes without time constraints, recently proposed in [19], new constructions for time-bound hierarchical key assignment schemes have been proposed in [21]. Such schemes exhibit a tradeoff among the amount of secret data that needs to be distributed and stored by the users, the amount of data that needs to be made public, the complexity of key derivation, and the computational assumption on which the security of the scheme is based. An open problem would be to find a time-bound hierarchical key assignment scheme which optimizes all parameters at the same time.

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 and it is the same for any time period. However, all our results could be easily extended for the more general setting.

References

[1] A. V. Aho, M. R. Garey, and J. D. Ullman. The Transitive Reduction of a Directed Graph. *SIAM Journal on Computing*, 1:131–137, 1972.

[2] Selim G. Akl and Peter D. Taylor. Cryptographic Solution to a Problem of Access Control in a Hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, 1983.

[3] Mikhail J. Atallah, Marina Blanton, Nelly Fazio, and Keith B. Frikken. Dynamic and Efficient Key Management for Access Hierarchies. In *CERIAS Technical Report TR 2006-09, Purdue University. Preliminary version in Proc. of the ACM Conference on Computer and Communications Security*, pages 190–202, 2006.

[4] Mikhail J. Atallah, Marina Blanton, and Keith B. Frikken. Key Management for Non-Tree Access Hierarchies. In *Proc. of the ACM Symposium on Access Control Models and Technologies*, pages 11–18, 2006.

[5] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci. Provably-Secure Time-Bound Hierarchical Key Assignment Schemes. In *Proc. of the ACM Conference on Computer and Communications Security*, pages 288–297, 2006.

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying Hash Functions for Message Authentication. In *Proc. of Advances in Cryptology, Crypto, Lecture Notes in Computer Science*, pages 1–15, 1996.

[7] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. A Concrete Security Treatment of Symmetric Encryption. In *Proc. of the 38th IEEE Symposium on Foundations of Computer Science*, pages 394–403, 1997.

[8] Elisa Bertino, Barbara Carminati, and Elena Ferrari. A Temporal Key Management Scheme for Secure Broadcasting of XML Documents. In *Proc. of the ACM Conference on Computer and Communications Security*, pages 31–40, 2002.

[9] Manuel Blum and Silvio Micali. How to Generate Cryptographically Strong Sequences of Pseudo-Random Bits. *SIAM Journal on Computing*, 13(4):850–864, 1984.

[10] Dan Boneh and Xavier Boyen. Efficient Selective-ID Secure Identity-Based Encryption Without Random Oracles. In *Advances in Cryptology - Eurocrypt, Lecture Notes in Computer Science*, pages 223–238, 2004.

[11] Dan Boneh and Matthew K. Franklin. Identity-Based Encryption from the Weil Pairing. *SIAM Journal on Computing*, 32(3):586–615, 2003.

[12] Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Journal of Cryptology*, 17(4):297–319, 2004.

[13] Ran Canetti, Shai Halevi, and Jonathan Katz. A Forward-Secure Public-Key Encryption Scheme. In *Proc. of Advances in Cryptology - Eurocrypt, Lecture Notes in Computer Science, 2656*, pages 255–271, 2003.

[14] T. Chen and Y. Chung. Hierarchical Access Control based on Chinese Remainder Theorem and Symmetric Algorithm. *Computers & Security.*, 21(6):565–570, 2002.

[15] Hung-Yu Chien. Efficient Time-Bound Hierarchical Key Assignment Scheme. *IEEE Transaction on Knowledge and Data Engineering.*, 16(10):1301–1304, 2004.

[16] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* John Wiley & Sons, 1991.

[17] J. Crampton, K. Martin, and P. Wild. On Key Assignment for Hierarchical Access Control. In *Proc. of the 19th IEEE Computer Security Foundations Workshop*, pages 98–111, 2006.

[18] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Cryptographic Key Assignment Schemes for any Access Control Policy. *Information Processing Letters*, 92(4):199–205, 2004.

[19] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Efficient Provably-Secure Hierarchical Key Assignment Schemes. *Available as Report 2006/479 at the IACR Cryptology ePrint Archive*, 2006.

[20] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Enforcing the Security of a Time-Bound Hierarchical Key Assignment Scheme. *Information Sciences*, 176(12):1684–1694, 2006.

[21] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. New Constructions for Provably-Secure Time-Bound Hierarchical Key Assignment Schemes. *Available as Report 2006/483 at the IACR Cryptology ePrint Archive*, 2006.

[22] Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci. Unconditionally Secure Key Assignment Schemes. *Discrete Applied Mathematics*, 154(2):234–252, 2006.

[23] S. D. Galbraith, K. Harrison, and D. Soldera. Implementing the Tate Pairing. In *Proc. of the Algorithmic Number Theory Symposium, Lecture Notes in Computer Science*, pages 385–394, 2000.

[24] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to Construct Random Functions. *Journal of the ACM*, 33(4):792–807, 1986.

[25] Shafi Goldwasser and Silvio Micali. Probabilistic Encryption. *Journal of Computer and System Sciences*, 28(2):270–299, 1984.

[26] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A Digital Signature Scheme Secure against Adaptive Chosen-Message Attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[27] L. Harn and H. Y. Lin. A Cryptographic Key Generation Scheme for Multilevel Data Security. *Computers & Security.*, 9(6):539–546, 1990.

[28] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-Way Function. *SIAM Journal of Computing*, 28(4):1364–1396, 1999.

[29] H. F. Huang and C. C. Chang. A New Cryptographic Key Assignment Scheme with Time-Constraint Access Control in a Hierarchy. *Computer Standards & Interfaces.*, 26:159–166, 2004.

[30] M. S. Hwang. A Cryptographic Key Assignment Scheme in a Hierarchy for Access Control. *Mathematical and Computational Modeling.*, 26(1):27–31, 1997.

[31] Antoine Joux. A One Round Protocol for Tripartite Diffie-Hellman. In *Proc. of the Algorithmic Number Theory Symposium*, pages 385–394, 2000.

[32] Jonathan Katz and Moti Yung. Characterization of Security Notions for Probabilistic Private-Key Encryption. *Journal of Cryptology*, 19(1):67–95, 2006.

[33] H. T. Liaw, S. J. Wang, and C. L. Lei. A Dynamic Cryptographic Key Assignment Scheme in a Tree Structure. *Computers and Mathematics with Applications*, 25(6):109–114, 1993.

[34] C. H. Lin. Dynamic Key Management Schemes for Access Control in a Hierarchy. *Computer Communications*, 20:1381–1385, 1997.

[35] I. C. Lin, M. S. Hwang, and C. C. Chang. A New Key Assignment Scheme for Enforcing Complicated Access Control Policies in Hierarchy. *Future Generation Computer Systems.*, 19:157–462, 2003.

[36] Stephen J. MacKinnon, Peter D. Taylor, Henk Meijer, and Selim G. Akl. An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy. *IEEE Transaction on Computers*, 34(9):797–802, 1985.

[37] A. Miyaji, M. Nakabayashi, and S. Takano. New Explicit Conditions for Elliptic Curve Traces for FR-Reduction. *IEICE Transactions Fundamentals*, E-84(5), 2001.

[38] Moni Naor and Omer Reingold. Number-Theoretic Constructions of Efficient Pseudo-Random Functions. *Journal of ACM*, 51(2):231–262, 2004.

[39] Ravi S. Sandhu. Cryptographic Implementation of a Tree Hierarchy for Access Control. *Information Processing Letters*, 27(2):95–98, 1988.

[40] Adi Shamir. On the Generation of Cryptographically Strong Pseudorandom Sequences. *ACM Transaction on Computer Systems*, 1(1):38–44, 1983.

[41] V. Shen and T. Chen. A Novel Key Management Scheme based on Discrete Logarithms and Polynomial Interpolations. *Computers & Security*, 21(2):164–171, 2002.

[42] Q. Tang and C. J. Mitchell. Comments on a Cryptographic Key Assignment Scheme. *Computer Standards & Interfaces*, 27:323–326, 2005.

[43] W.-G. Tzeng. A Time-Bound Cryptographic Key Assignment Scheme for Access Control in a Hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 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*, pages 223–230, 2006.

[45] S.-Y. Wang and C.-Laih. Merging: an Efficient Solution for a Time-Bound Hierarchical Key Assignment Scheme. *IEEE Transactions on Dependable and Secure Computing*, 3(1):91–100, 2006.

[46] T. Wu and C. Chang. Cryptographic Key Assignment Scheme for Hierarchical Access Control. *International Journal of Computer Systems Science and Engineering*, 1(1):25–28, 2001.

[47] J. Yeh. An RSA-Based Time-Bound Hierarchical Key Assignment Scheme for Electronic Article Subscription., booktitle = Proc. of the ACM CIKM International Conference on Information and Knowledge Management, year = 2005, pages = 285-286,.

[48] J. Yeh, R. Chow, and R. Newman. A Key Assignment for Enforcing Access Control Policy Exceptions. In *Proc. of the International Symposium on Internet Technology*, pages 54–59, 1998.

[49] X. Yi. Security of Chien's Efficient Time-Bound Hierarchical Key Assignment Scheme. *IEEE Transactions on Knowledge and Data Engineering*, 17(9):1298–1299, 2005.

[50] X. Yi and Y. Ye. Security of Tzeng's Time-Bound Key Assignment Scheme for Access Control in a Hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1054–1055, 2003.

Appendix

In this Appendix we review the basic concepts of Information Theory used in our definitions and proofs. For a complete treatment of the subject the reader is advised to consult [16].

Given a probability distribution $\{Pr_{\mathbf{X}}(x)\}_{x \in X}$ on a set X, we define the *entropy* [1] of $\mathbf{X}$, $H(\mathbf{X})$, as

$$H(\mathbf{X}) = -\sum_{x \in X} Pr_{\mathbf{X}}(x) \log Pr_{\mathbf{X}}(x).$$

The entropy satisfies the following property

$$0 \leq H(\mathbf{X}) \leq \log |X|,$$

where $H(\mathbf{X}) = 0$ if and only if there exists $x_0 \in X$ such that $Pr_{\mathbf{X}}(x_0) = 1$; whereas, $H(\mathbf{X}) = \log |X|$ if and only if $Pr_{\mathbf{X}}(x) = 1/|X|$, for all $x \in X$.

Given two sets X and Y and a joint probability distribution on their cartesian product, the *conditional entropy* $H(\mathbf{X}|\mathbf{Y})$, is defined as

$$H(\mathbf{X}|\mathbf{Y}) = -\sum_{y \in Y} \sum_{x \in X} Pr_{\mathbf{Y}}(y) Pr(x|y) \log Pr(x|y).$$

From the definition of conditional entropy it is easy to see that

$$H(\mathbf{X}|\mathbf{Y}) \geq 0.$$

Given n sets $X_1, \ldots, X_n$ and a joint probability distribution on their cartesian product, the entropy of $\mathbf{X}_1 \ldots \mathbf{X}_n$ can be expressed as

$$H(\mathbf{X}_1 \ldots \mathbf{X}_n) = H(\mathbf{X}_1) + \sum_{i=2}^{n} H(\mathbf{X}_i|\mathbf{X}_1 \ldots \mathbf{X}_{i-1}). \tag{7}$$

Given $n+1$ sets $X_1, \ldots, X_n, Y$ and a joint probability distribution on their cartesian product, the entropy of $\mathbf{X}_1 \ldots \mathbf{X}_n$ given $\mathbf{Y}$ can be expressed as

$$H(\mathbf{X}_1 \ldots \mathbf{X}_n|\mathbf{Y}) = H(\mathbf{X}_1|\mathbf{Y}) + \sum_{i=2}^{n} H(\mathbf{X}_i|\mathbf{X}_1 \ldots \mathbf{X}_{i-1}\mathbf{Y}). \tag{8}$$

The *mutual information* $I(\mathbf{X};\mathbf{Y})$ between $\mathbf{X}$ and $\mathbf{Y}$ is defined by

$$I(\mathbf{X};\mathbf{Y}) = H(\mathbf{X}) - H(\mathbf{X}|\mathbf{Y}) \tag{9}$$

and satisfies the following properties:

$$I(\mathbf{X};\mathbf{Y}) = I(\mathbf{Y};\mathbf{X})$$

and $I(\mathbf{X};\mathbf{Y}) \geq 0$, from which one gets

$$H(\mathbf{X}) \geq H(\mathbf{X}|\mathbf{Y}). \tag{10}$$

[1] All log's in this paper denote basis 2 logarithms.

Given three sets X, Y, Z and a joint probability distribution on their cartesian product, the *conditional mutual information* $I(\mathbf{X};\mathbf{Y}|\mathbf{Z})$ between $\mathbf{X}$ and $\mathbf{Y}$ given $\mathbf{Z}$ is

$$I(\mathbf{X};\mathbf{Y}|\mathbf{Z}) \quad = \quad H(\mathbf{X}|\mathbf{Z}) - H(\mathbf{X}|\mathbf{Z}\mathbf{Y}) \tag{11}$$

and satisfies the following properties:

$$I(\mathbf{X};\mathbf{Y}|\mathbf{Z}) = I(\mathbf{Y};\mathbf{X}|\mathbf{Z})$$

and $I(\mathbf{X};\mathbf{Y}|\mathbf{Z}) \geq 0$. Since the conditional mutual information is always non negative we get

$$H(\mathbf{X}|\mathbf{Z}) \geq H(\mathbf{X}|\mathbf{Z}\mathbf{Y}). \tag{12}$$

From (8) and (12) one easily gets that for any sets $Y, X_1, \ldots, X_n$ and a joint probability distribution on their cartesian product it holds that

$$\sum_{i=1}^{n} H(\mathbf{X}_i|\mathbf{Y}) \geq H(\mathbf{X}_1\mathbf{X}_2\ldots\mathbf{X}_n|\mathbf{Y}). \tag{13}$$