New Constructions for Provably-Secure Time-Bound Hierarchical Key Assignment Schemes

Alfredo De Santis, Anna Lisa Ferrara, and Barbara Masucci

Dipartimento di Informatica ed Applicazioni, Università di Salerno, 84084 Fisciano (SA), Italy
{ads, ferrara, masucci}@dia.unisa.it

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 in the hierarchy can compute the keys of all classes lower down in the hierarchy, according to temporal constraints.

In this paper we propose new constructions for time-bound hierarchical key assignment schemes which are provably secure with respect to key indistinguishability. Our constructions use as a building block any provably-secure hierarchical key assignment scheme without temporal constraints and exhibit a tradeoff among the amount of private information held by each class, the amount of public data, the complexity of key derivation, and the computational assumption on which their security is based. Moreover, the proposed schemes support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

Keywords: Access control, key assignment, provable security, efficient key derivation.

1 Introduction

Users of a computer system could be organized in a hierarchy formed by a certain number of disjoint classes. These classes, called *security classes*, are positioned and ordered within the hierarchy based on the fact that some users have more access rights than others. For example, within a hospital system, doctors can access all data concerning their patients, 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 insertions and deletions of classes in the hierarchy (e.g., [4, 17, 19, 22, 23, 24, 25]). A recent work by Crampton et al. [13] provides a detailed classification of many schemes in the literature and evaluates their merits according to different parameters, such as the amount of private information distributed and stored by users, the amount of public information, the complexity of key derivation, the complexity of handling dynamic updates to the hierarchy, and the resistance to collusive attacks.

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). A *time-bound hierarchical key assignment scheme* is a method to assign time-dependent encryption keys and private information to each class in the hierarchy, in such a way that key derivation also depends on temporal constraints. 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. Several proposals for time-bound hierarchical key assignment schemes [30, 12, 18, 34] have 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 [36, 14, 35, 27, 6]. Recently, Wang and Laih [37] and Tzeng [31] showed how to construct a time-bound hierarchical key assignment scheme starting from the Akl-Taylor scheme [2]. 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 to be secure.

Ateniese et al. [6] first addressed the problem of formalizing security requirements for time-bound hierarchical key assignment schemes. They distinguished between two different goals, i.e., security with respect to *key indistinguishability* and against *key recovery*, and two different kinds of adversarial behaviors, i.e., the *static* and *adaptive* adversarial behaviors. In particular, they proved that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries. They also proposed two different constructions for time-bound key assignment schemes. Both constructions support dynamic updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

In this paper we propose new constructions for time-bound hierarchical key assignment schemes which are provably-secure with respect to key indistinguishability. Our constructions use as a building block any provably-secure hierarchical key assignment scheme without temporal constraints and allow to establish a tradeoff among the amount of private information held by users, the amount of public data, the complexity of key derivation, and the computational assumption on which the security of the schemes is based. The paper is organized as follows: in Section 2 we first review the definition of time-bound hierarchical key assignment schemes and then analyze the constructions proposed in [6]. In Section 3 we show how to construct a provably secure time-bound hierarchical key assignment scheme using as a building block *any* provably-secure hierarchical key assignment scheme without temporal constraints. To this aim, we show how to merge the hierarchical and temporal constraints into a new hierarchy, by using the so called *Interval-Hierarchy Based Transformation (IHBT)*. Afterwards, we provide a tradeoff between the amount of public data and the complexity of key derivation. In Section 4 we show how to obtain a further tradeoff, also involving the amount of secret information held by each user, by using the so called *Covering Set Based Transformation (CSBT)*. In Section 5 we conclude the paper by showing a comparison between our constructions and previous work.

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 hierarchies [25, 5, 15], etc.

As done in [6], in this paper we consider the case where the membership of a user to a certain class also depends on temporal constraints. Let $T = (t_1, \ldots, t_n)$ be a sequence of distinct time periods. 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. We recall the definition of a time-bound hierarchical key assignment scheme given in [6].

Definition 2.1 ([6]) *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*

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

A time-bound hierarchical key assignment scheme is 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 public data, the complexity of key derivation, the complexity of key updates due to dynamic changes to the hierarchy, and the resistance to collusive attacks.

As regards as the complexity of key derivation, we are interested both in the number and of type of operations needed to derive a key. Moreover, notice that in Definition 2.1 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 v performs key derivation to compute the key of a class $u \in A_v$, 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. As regards as the complexity of key updates, due to dynamic changes to the hierarchy, we would like to allow the insertion and deletion of users, classes, or edges in the hierarchy at any time period, without requiring the TA to re-distribute the private information to the other users.

However, the most fundamental feature that a good scheme should have is the resistance to collusive attacks. More precisely, for each class $u \in V$ and each time period $t \in T$, the key $k_{u,t}$ should be protected against a coalition of users belonging to each class v such that $u \notin A_v$ in all time periods, and users belonging to each class w such that $u \in A_w$ in all time periods but t. We denote by $F_{u,t}$ the set $\{(v, \lambda) \in V \times \mathcal{P} : u \notin A_v \text{ or } t \notin \lambda\}$, corresponding to all users which are not allowed to compute the key $k_{u,t}$. In this paper we consider security with respect to *key indistinguishability*. Such a requirement, introduced by Ateniese et al. [6], formalizes the fact that the adversarial coalition is not able to *distinguish* a key, that should not be accessible by any user of the coalition, from a random string of the same length. We consider a *static adversary* $\mathtt{STAT}_{u,t}$, which wants to attack 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 $Corrupt_{u,t}(s)$. 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 $1/2$.

If $A(\cdot, \cdot, \ldots)$ is any probabilistic algorithm then $a \leftarrow A(x, y, \ldots)$ denotes the experiment of running A on inputs $x, y, \ldots$ and letting a be the outcome, the probability being over the coins of A. Similarly, if X is a set then $x \leftarrow X$ denotes the experiment of selecting an element uniformly from X and assigning x this value. If w is neither an algorithm nor a set then $x \leftarrow w$ is a simple assignment statement. A function $\epsilon : N \rightarrow R$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

Definition 2.2 ([6]) *Let* Γ *be a family of graphs corresponding to partially ordered hierarchies, 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* (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*.*

In Definition 2.2 we have considered a static adversary attacking a class. A different kind of adversary, the *adaptive* one, could also be considered. Such an adversary is first allowed to access all public information as well as all private information of a number of classes 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. In [6] it has been proven that security against adaptive adversaries is (polynomially) equivalent to security against static adversaries. Hence, in this paper we will only consider static adversaries.

An Extension of the Model. In Definition 2.1 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. However, as noticed in [6], the model could be generalized to the cases 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.

In order to describe a time-bound hierarchical key assignment scheme for the above general setting, we need to slightly extend Definition 2.1 to allow the algorithms *Gen* and *Der* to take as inputs more than one graph, instead that a single one. In particular, the algorithm *Gen* should take as inputs all graphs $G_1, \ldots, G_{|T|}$, whereas, the algorithm *Der* should take as inputs the graph G_i, for any i such that $t_i \in \lambda$. Throughout the paper, we will consider the case where the access control policy is the same for any time period. However, all the constructions presented in this paper, as well as those proposed in [6] and described in the following, could also be used for the more general setting.

2.1 Previous Solutions

In this section we recall some constructions of time-bound hierarchical key assignment schemes and evaluate their merits. Since we are mainly interested in schemes which are provably secure, we do not consider those which have been shown to be insecure against collusion attacks [30, 12, 18, 34].

Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E)$ be a graph in Γ, and let $T = (t_1, \ldots, t_n)$ be a sequence of distinct time periods. The constructions proposed by Ateniese et al. [6] both involve a transformation of the graph G resulting in a two-level partially ordered hierarchy. The first construction, which we refer to as the *Two-Level Encryption Based Construction (TLEBC)* makes use of a symmetric encryption scheme secure against a non-adaptive chosen plaintext attack, denoted by `IND-P1-C0` in [20]. The TLEBC requires $O(|E^*| \cdot |T|^3)$ values to be made public, whereas, 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 operation in order to derive a key (no other operations are involved in key derivation). The parameters of the TLEBC are summarized in row 1 of Figure 9. The second solution, which we refer to as the *Two-Level Pairing Based Construction (TLPBC)*, makes use of pairings and assumes the intractability of the *Bilinear Decisional Diffie-Hellman* (`BDDH`) problem [6]. The TLPBC offers a smaller amount of public information compared to the TLEBC, since $O(|E^*|)$ values are made public, but requires each user to hold at most $|T|$ private values. However, key derivation requires one group operation. Both the TLEBC and the TLPBC are provably secure with respect to key indistinguishability and support dynamic changes to the hierarchy, without requiring the re-distribution of any private information. The parameters of the TLPBC are summarized in row 1 of Figure 10.

The schemes proposed by Wang and Laih [37] and Tzeng [31] both require $|V| \cdot |T|$ public values, whereas, each user has to store a single secret value and key derivation involves one modular exponentiation and $O(|V| \cdot |T|)$ operations. However, since the authors did not formalize the definition of security and the adversarial model, it is not clear under which assumption their schemes can be considered to be provably secure.

3 A General Construction

In this section we consider the problem of constructing a provably secure time-bound hierarchical key assignment scheme for a family of graphs Γ, using as a building block *any* provably secure hierarchical key assignment scheme Σ, without temporal constraints, for Γ.

The naive solution to construct a time-bound hierarchical key assignment scheme for Γ involves replicating Σ once for each time period. Let $G = (V, E)$ be a graph in Γ, let pub_{t_i} and s_{u,t_i} denote the public information and the private information held by users in class $u \in V$, respectively, in the i-th replicate of Σ. The public information of the time-bound scheme obtained by the naive solution is $pub = (pub_1, \ldots, pub_n)$, whereas, the private information held by users belonging to a class $u \in V$ for a time sequence $\lambda = (t_i, \ldots, t_j) \in \mathcal{P}$ is $s_{u,\lambda} = (s_{u,t_i}, \ldots, s_{u,t_j})$. The drawback of such a solution is that it requires a large amount of public and private information.

In order to obtain schemes offering better performances, we should avoid a simple replication of Σ. For example, we could use Σ just once, to assign keys and private information to the classes of a new hierarchy, obtained by merging the hierarchical and temporal constraints in such a way that the users belonging to the same class share the same access rights (this does not happen in time-bound hierarchical key assignment schemes, since users in the same class may have different access rights, according to the time periods in which they are able to access the data). The new hierarchy could be obtained by performing a graph transformation, starting from the graph G and the interval-set $\mathcal{P}$ over T. The transformation we propose makes use of the partially ordered hierarchy whose classes are the contiguous time subsequences of T. Indeed, such subsequences, corresponding to the elements of the interval-set $\mathcal{P}$ over T, naturally induce a partially ordered hierarchy, called the *interval-hierarchy over* T, with respect to the binary relation of inclusion between contiguous subsequences. We denote by $H_T = (\mathcal{P}, E_\mathcal{P})$ the *minimal representation* of the directed acyclic graph representing such a hierarchy. It is easy to see that $|E_\mathcal{P}| = 2(|\mathcal{P}| - |T|) = \Theta(|T|^2)$, since any node in $\mathcal{P}$, with the exception of sink nodes, has exactly two outgoing edges. Figure 1 shows the minimal representation of the interval-hierarchy over $T = (t_1, t_2, t_3, t_4)$, where the node corresponding to each sequence $(t_i, \ldots, t_j)$ is denoted by $[i, j]$.

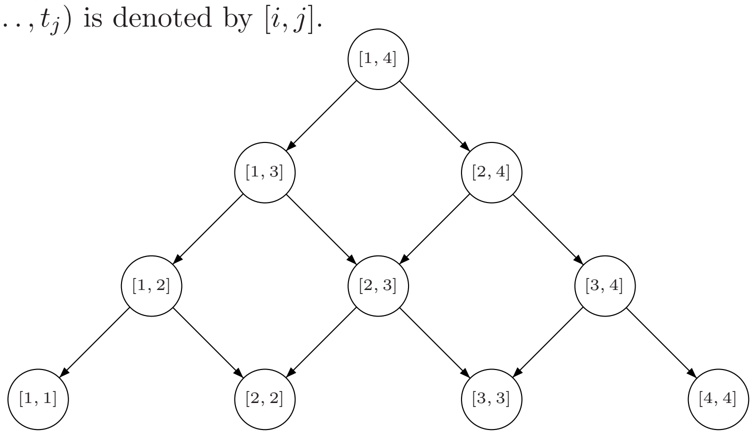

Figure 1: The minimal representation of the interval-hierarchy over $T = (t_1, t_2, t_3, t_4)$.

Starting from the two graphs $G = (V, E)$ and $H_T = (\mathcal{P}, E_\mathcal{P})$, we construct a new graph $G' = (V', E')$, by using the *Interval-Hierarchy Based Transformation (IHBT)*, defined as follows:

- For each class $u \in V$ and each time sequence $\lambda \in \mathcal{P}$, we place a class $u_\lambda \in V'$.
- For each class $u \in V$ and each pair of time sequences $\lambda, \gamma \in \mathcal{P}$ connected by an edge in $E_\mathcal{P}$, we place an edge between u_λ and u_γ in G', i.e., $(u_\lambda, u_\gamma) \in E'$. Such edges are called *time edges*.
- For each pair of classes u and v connected by an edge in E and each time period $t \in T$, we place an edge between u_t and v_t in G', i.e., $(u_t, v_t) \in E'$. Such edges are called *policy edges*.

It is easy to see that $|V'| = |V| \cdot |\mathcal{P}| = O(|V| \cdot |T|^2)$. Moreover, $|E'| = |V| \cdot |E_\mathcal{P}| + |E| \cdot |T| = O(|V| \cdot |T|^2 + |E| \cdot |T|)$. Indeed, E' contains $|V| \cdot |E_\mathcal{P}|$ time edges and $|E| \cdot |T|$ policy edges. The right hand side of Figure 2 shows the graph G' obtained by the IHBT starting from the graph G shown on the left hand side of Figure 2 and from the graph H_T representing the interval-hierarchy over T shown in Figure 1. Time edges and policy edges are represented by dashed and solid lines, respectively.

Once the graph $G' = (V', E')$ has been obtained, we can use any hierarchical key assignment scheme without temporal constraints. In the following we recall two constructions recently proposed in [15] which will be used as building blocks of our constructions.

The Broadcast Encryption Based Construction (BEBC). A provably-secure hierarchical key assignment scheme can be obtained by using as a building block a secure public-key broadcast encryption scheme. In particular, if the underlying public-key broadcast encryption scheme is semantically secure, then the resulting hierarchical key assignment scheme is secure in the sense of IND-ST. Boneh et al. [9] showed how to construct a semantically secure broadcast encryption scheme

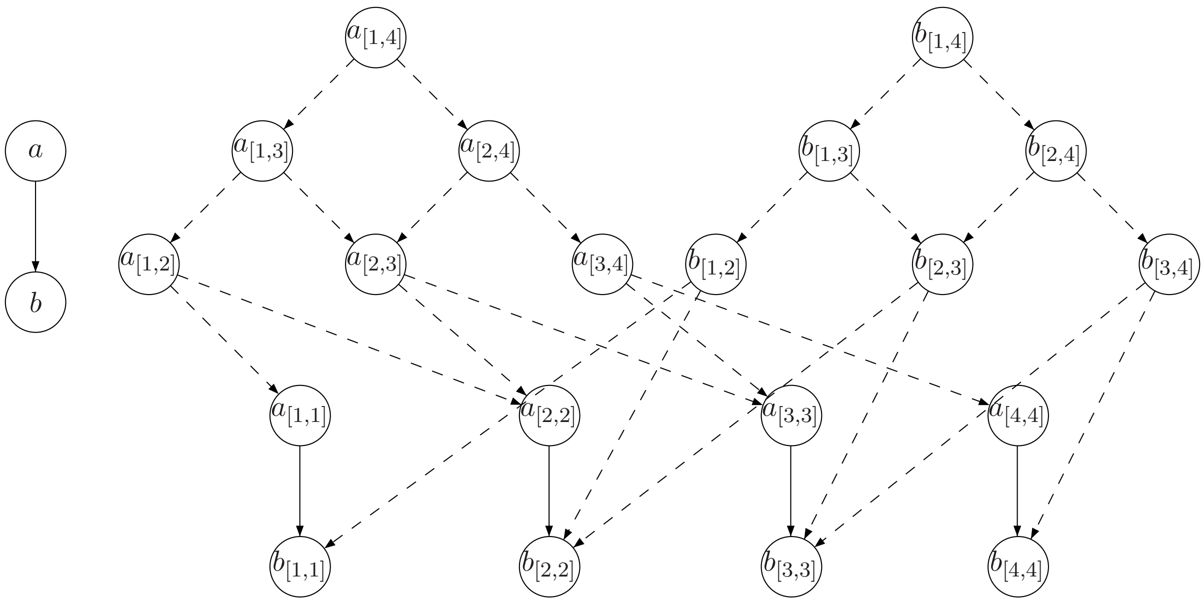

Figure 2: An example of the graph G' (on the right) corresponding to the graph G (on the left) and the graph H_T depicted in Figure 1. Time edges and policy edges are represented by dashed and solid lines, respectively.

for a set of n users, assuming the intractability of the *n-Bilinear Decisional Diffie-Hellman Exponent* (n-BDDHE) problem. The use of such a broadcast encryption scheme, based on pairings, allows to obtain a hierarchical key assignment scheme where the public information consists of $4|V|+1$ group elements, whereas, the private information has constant size. Moreover, key derivation requires a single (complex) decryption operation, which involves at most $|V|-2$ group operations. The BEBC supports dynamic changes to the hierarchy without requiring re-distribution of private information to the classes affected by such changes.

By using the BEBC based on pairings we obtain a time-bound hierarchical key assignment scheme where the number of data to be made public is $4|V'|+1 = O(|V| \cdot |T|^2)$, whereas, key derivation requires a single (complex) decryption operation, involving at most $|V'|-2$ group operations. Moreover, each user belonging to a certain class for a time sequence has to store a single secret value. The parameters of the scheme described above are summarized in row 3 of Figure 10.

The Encryption Based Construction (EBC). A different construction to obtain a provably-secure hierarchical key assignment scheme uses as a building block a secure symmetric encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. In particular, if the underlying encryption scheme is secure in the sense of `IND-P1-C0`, i.e., with respect to a non-adaptive chosen plaintext attack, then the resulting hierarchical key assignment scheme is secure in the sense of `IND-ST`. The idea is the following: each class $u \in V$ is assigned a private information s_u, an encryption key k_u, and a public information $p_{(u,u)} = \mathcal{E}_{s_u}(k_u)$, which is the encryption of the key k_u with the private information s_u; moreover, for each edge $(u,v) \in E$, there is a public value $p_{(u,v)} = \mathcal{E}_{s_u}(s_v)$, which allows class u to compute the private information s_v held by class v. The scheme requires $|E| + |V|$ public values; on the other hand, each class has to store a single secret value, corresponding to its private information. As for key derivation, in the EBC a class $u \in V$ which wants to compute the key held by a class $v \in A_u$ is required to perform $dist_G(u,v)+1$ decryptions, where $dist_G(u,v)$ denotes the length of the shortest path between u and v in G. The EBC as described above supports insertions, but not deletions, of classes and edges in the hierarchy without re-distributing private information to the classes affected by such changes. However, in [15] a simple modification of the scheme, which avoids the re-distribution of private information when deletions of classes or edges occur, has been proposed. The modified scheme requires $|E|+2|V|$ public values, whereas, each class has to store a single secret value. Moreover, the number of decryptions needed by class u to compute the key of class $v \in A_u$ is $dist_G(u,v)+2$.

A scheme also based on encryption schemes and offering similar features has been proposed by Atallah et al. [4]. Their scheme requires $2|E|+|V|$ public values, whereas, each class has to store a single secret value. In order to compute the key held by a class $v \in A_u$, users in class u have to perform $dist_G(u,v)$ decryptions and $dist_G(u,v)+1$ pseudorandom function evaluations. However, the security of their scheme is based on two different computational assumptions (the existence of pseudorandom function families and of symmetric encryption schemes secure against chosen ciphertext attacks). Thus, compared to the scheme in [4], the EBC is simpler and more efficient.

In the following we evaluate the time-bound hierarchical key assignment scheme obtained by using as a building block the first version of the EBC, i.e., the one which does not support deletions of classes or edges in the hierarchy. By using such a scheme, we obtain a time-bound hierarchical key assignment scheme where the number of data to be made public is $|V'|+|E'| = O(|E|\cdot|T|+|V|\cdot|T|^2)$. However, the number of steps required to perform key derivation can be quite large. Indeed, consider a user assigned to a class $v \in V$ for a time sequence λ. In order to compute the key held by class $u \in A_v$ at time period $t \in \lambda$, the user is required to perform $dist_{G'}(u_\lambda, v_t)+1$ decryptions, where $dist_{G'}(u_\lambda, v_t) = dist_G(u,v)+dist_{H_T}(\lambda,t)$. Therefore, the number of decryption operations required to perform key derivation is upper bounded by the sum of the diameters of the two graphs H_T and G, where the *diameter* of a directed graph is defined as the maximum distance between a pair of vertices in the graph connected by a path. In Section 3.1 we consider the problem of improving the efficiency of key derivation by using some techniques, described in [15], which allow to reduce the diameter of a directed graph. In particular, we restrict our attention to the graph H_T, because the diameter of the graph G, denoted in the following by $diam(G)$, could have already been reduced by employing similar techniques.

3.1 A Tradeoff between Public Storage and Key Derivation Time

In this section we show how to use two techniques described in [15], which we refer to as the *Shortcutting Technique* and the *Improved Shortcutting and Point-Inserting Technique*, respectively, in order to reduce key derivation time in the schemes obtained by the EBC. Such techniques, applied to a directed graph, allow to obtain a new graph, having a smaller diameter than the previous graph, such that there exists a path between two vertices in the previous graph if and only if there exists a path between them in the new one.

The Shortcutting Technique (ST). The *Shortcutting Technique* applied to a directed graph $H=(V_H, E_H)$ consists into adding to E_H additional edges, called *shortcut edges* and belonging to the transitive closure of E_H. The goal is to obtain another directed graph, called a *shortcut graph*, and denoted by $H^+ = (V_H, E_H^+)$, having a smaller diameter than H. Thorup [29] showed that, given a rooted acyclic planar directed graph $H = (V_H, E_H)$, by adding at most $|E_H|$ shortcut edges we obtain a shortcut graph having diameter $O((\log |V_H|) \cdot \alpha(|V_H|, |V_H|))$, by adding at most $|E_H|$ shortcut edges, where α denotes the inverse of the Ackermann's function defined by Tarjan [28]. The function $\alpha(p, p)$ is very slowly-growing; in particular, it grows even slower than the *iterated logarithmic function* $\log^* p$, which, for all values of p less than $2^{65,536}$, corresponding to much more than the number of atoms in the universe, does not exceed 5. The iterated logarithm function $\log^* p$ is defined to be the number of times the logarithm function must be applied in succession, starting with argument p, before the result is less than or equal to 1, i.e., $\log^* p = \min\{i \geq 0 : \log^{(i)} p \leq 1\}$, where

$$\log^{(i)} p = \begin{cases} p & \text{if } i = 0, \\ \log(\log^{(i-1)} p) & \text{if } i > 0 \text{ and } \log^{(i-1)} p > 0, \\ \text{undefined} & \text{if } i > 0 \text{ and } \log^{(i-1)} p \leq 0 \text{ or } \log^{(i-1)} p \text{ is undefined.} \end{cases}$$

Since the graph $H_T = (\mathcal{P}, E_{\mathcal{P}})$, representing the interval-hierarchy over T, is a rooted acyclic planar graph, we can apply Thorup's technique in order to obtain a shortcut graph $H_T^+ = (\mathcal{P}, E_{\mathcal{P}}^+)$, having diameter $O(\log |T| \cdot \alpha(|T|^2, |T|^2))$, by adding at most $|E_{\mathcal{P}}| = O(|T|^2)$ shortcut edges. Thus, by using the IHBT starting on the graphs $G = (V, E)$ and $H_T^+ = (\mathcal{P}, E_{\mathcal{P}}^+)$ and by applying the EBC on the output of the IHBT, we obtain a time-bound hierarchical key assignment scheme where the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T|^2)$, whereas, the number of decryption operations needed for key derivation is $O(\log |T| \cdot \log^* |T| + diam(G))$. The parameters of the scheme described above are summarized in row 3 of Figure 9.

The Improved Shortcutting and Point Inserting Technique (ISPIT). The *Improved Shortcutting and Point Inserting Technique* applied to the directed graph $H = (V_H, E_H)$, representing a partially ordered hierarchy, consists into inserting additional edges in E_H, as well as new vertices in V_H, in order to obtain a graph H^{++} having a smaller diameter than H. Such a technique, described in [15], makes use of the concept of *dimension* of a poset. Let $\preceq$ and $\diamond$ be two binary relations that partially order the elements of a finite set X. The poset $(X, \diamond)$ is called an *extension* of $(X, \preceq)$ if $x \preceq y$ implies $x \diamond y$, for any $x, y \in X$. If $(X, \diamond)$ is a *totally ordered set*, i.e., if for any $x, y \in X$ either $x \diamond y$ or $y \diamond x$, then $(X, \diamond)$ is also said to be a *linear extension* of $(X, \preceq)$. The problem of counting the number of linear extensions of a poset has been shown to be #P-complete [10]. The *dimension* of a poset $(X, \preceq)$, originally defined by Dushnik and Miller [16], is the minimum number of linear extensions of $(X, \preceq)$ whose intersection is $(X, \preceq)$. It can also be seen as the smallest nonnegative integer d for which each $x \in X$ can be represented by a d-vector $(x_1, \ldots, x_d)$ of integers such that $x \preceq y$ if and only if $x_i \leq y_i$, for any $i = 1, \ldots, d$, and any $x, y \in X$. There are efficient algorithms to test if a poset has dimension 1 or 2, but the problem of determining if a poset has dimension 3 is NP-complete [32]. A poset has dimension one if and only if it is a total order.

It is easy to see that, given the interval set $\mathcal{P}$ over a sequence T of time periods, the interval-hierarchy represented by the graph $H_T = (\mathcal{P}, E_{\mathcal{P}})$ has dimension two. Indeed, the two linear extensions can be constructed as follows: given two sequences $\lambda, \gamma \in \mathcal{P}$, where $\lambda = (t_a, \ldots, t_b)$ and $\gamma = (t_c, \ldots, t_d)$, in the first linear extension λ follows γ if either $a > c$ or $a = c$ and $b \leq d$; whereas, in the second linear extension λ follows γ if either $b > d$ or $b = d$ and $c \leq a$. Figure 3 shows the two linear extensions of the interval-hierarchy represented by the graph H_T in Figure 1.

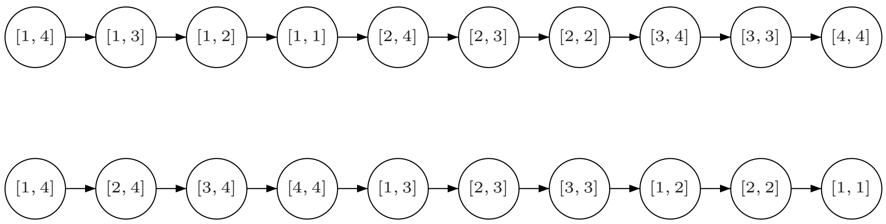

Figure 3: The two linear extensions of the hierarchy of time sequences in Figure 1.

Given a partially ordered hierarchy with dimension d on a set of p vertices, the ISPIT proposed in [15] allows to reduce the diameter of the graph representing it to $\log^* p$, by adding $O(p \cdot d \cdot (3 \log p)^{d-1})$ new edges and vertices. Therefore, by using such a technique on the graph H_T, which as said above has dimension two, we obtain a graph H_T^{++} having diameter $O(\log^* |T|)$, by adding $O(|T|^2 \cdot \log |T|)$ new edges and vertices. Thus, by using the IHBT starting on the graphs G and H_T^{++} and by applying the EBC on the resulting graph, we obtain a time-bound hierarchical key assignment scheme where the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T|^2 \cdot \log |T|)$, whereas, the number of decryption operations needed for key derivation is $O(\log^* |T| + diam(G))$. The parameters of the scheme described above are summarized in row 4 of Figure 9.

On the other hand, the ISPIT in [15] can be also used to reduce the diameter of the graph representing the partially ordered hierarchy to three, by adding $O(p \cdot d \cdot (3 \log p)^{d-1} \cdot \log \log p)$ new edges and vertices. Therefore, by using such a technique on the graph H_T we obtain a graph H_T^{++} having diameter three, by adding $O(|T|^2 \cdot \log |T| \cdot \log \log |T|)$ new edges and vertices. Thus, by using the IHBT starting on the graphs G and H_T^{++} and by applying the EBC on the resulting graph, we obtain a time-bound hierarchical key assignment scheme where the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T|^2 \cdot \log |T| \cdot \log \log |T|)$, whereas, the number of decryption operations needed for key derivation is $O(diam(G))$. The parameters of the scheme described above are summarized in row 5 of Figure 9.

4 A Tradeoff among Key Derivation Time, Public, and Private Storage

The techniques described in Section 3.1 allow us to obtain a tradeoff between the amount of public information and the complexity of key derivation. In this section we show how to obtain a further tradeoff, also involving the amount of secret information held by each user. Let $\ell \geq 1$ and assume we have a method to decompose the sequence T of time periods into a set of subsequences $\mathcal{P}_\ell \subseteq \mathcal{P}$, called an *$\ell$-covering set for T*, in such a way that any contiguous subsequence of T is the union of at most ℓ elements of $\mathcal{P}_\ell$. This would allows us to construct a time-bound hierarchical key assignment scheme Σ', on input a graph G and the interval-set $\mathcal{P}$, using as a building block any time-bound hierarchical key assignment scheme Σ, on input G and $\mathcal{P}_\ell$. For each class $u \in V$, the private information assigned by Σ' to u for any time sequence $\lambda \in \mathcal{P}_\ell$ is the same as the one assigned by Σ; whereas, the private information assigned to u for any time sequence $\gamma \in \mathcal{P} \setminus \mathcal{P}_\ell$ consists of a sequence of at most ℓ private information assigned by Σ to u. More precisely, let $\gamma_1, \ldots, \gamma_h \in \mathcal{P}_\ell$ be

$h \leq \ell$ sequences whose union gives the sequence γ. Then, the private information assigned by Σ' to u for time sequence γ consists of the private information assigned by Σ to u for the time sequences $\gamma_1, \ldots, \gamma_h$. As regards as the amount of public information required by the scheme Σ', it depends on the number of elements of the ℓ-covering set $\mathcal{P}_\ell$. In the following we show that the larger the cardinality of $\mathcal{P}_\ell$, the smaller the value of ℓ, thus establishing a tradeoff between the size of the private information held by each user, the amount of public information required by the scheme, and the complexity of key derivation.

Given a sequence T of n elements and an integer $\ell \geq 1$, Alon and Schieber [3] in 1987 established upper and lower bounds on the minimum possible cardinality of an ℓ-covering set for T. In particular, they considered the problem in a quite different context, where the n elements of T belong to a given semigroup $(S, \circ)$ and one is interested in answering queries of the form *"what is the value of $t_i \circ t_{i+1} \circ \cdots \circ t_{j-1} \circ t_j$?"* for any $1 \leq i \leq j \leq n$. Their construction is essentially the same proposed by Yao [33] in 1982 and further analyzed by Bodlaender et al. [7] in 1994. Translating the results in [3] to our scenario, we can construct an ℓ-covering set $\mathcal{P}_\ell$, for a sequence T of n elements, as described in the following. We first show a construction which is optimal for constant ℓ, i.e., resulting in an ℓ-covering set with cardinality $C_{min}(\ell, n)$. Afterwards, we consider the cases $\ell = O(\log n)$ and $\ell = O(\log^* n)$.

4.1 A Construction Optimal for Constant ℓ

The following construction results in an ℓ-covering set whose cardinality is denoted by $C(\ell, n)$. We will see later that such a construction is the best possible when ℓ is a constant.

- **Case** $\ell = 1$: Set $\mathcal{P}_1 = \mathcal{P}$. Clearly, $C(1, n) = O(n^2)$.
- **Case** $\ell = 2$: The algorithm to construct $\mathcal{P}_2$ works as follows:
  1. For $i = 1, \ldots, n$, insert the subsequence (t_i) into $\mathcal{P}_2$;
  2. If $n \geq 3$, then
     (a) Partition T in two consecutive sequences $T_1 = (t_1, \ldots, t_{\lceil n/2 \rceil})$ and $T_2 = (t_{\lceil n/2 \rceil + 1}, \ldots, t_n)$;
     (b) For $j = 1, \ldots, \lceil n/2 \rceil - 1$, insert the subsequence $(t_j, \ldots, t_{\lceil n/2 \rceil})$ into $\mathcal{P}_2$;
     (c) For $h = \lceil n/2 \rceil + 2, \ldots n$, insert the subsequence $(t_{\lceil n/2 \rceil + 1}, \ldots, t_h)$ into $\mathcal{P}_2$;
     (d) Apply the construction recursively to the subsequences T_1 and T_2.

  The construction results in a set $\mathcal{P}_2$ whose cardinality is given by the recurrence

$$C(2, n) = \begin{cases} n & \text{if } n \leq 2; \\ C(2, \lceil n/2 \rceil) + C(2, \lfloor n/2 \rfloor) + O(n) & \text{otherwise,} \end{cases}$$

  whose solution is $C(2, n) = O(n \cdot \log n)$.

Example 4.1 *For $n = 16$ and $\ell = 2$, by using the above construction, we obtain the covering set $\mathcal{P}_2$ for $T = (t_1, \ldots, t_{16})$, whose elements are the following:*

$$
\begin{array}{llll}
(t_1), & (t_2), & (t_3), & (t_4), \\
(t_5), & (t_6), & (t_7), & (t_8), \\
(t_9), & (t_{10}), & (t_{11}), & (t_{12}), \\
(t_{13}), & (t_{14}), & (t_{15}), & (t_{16}), \\
(t_1, \ldots, t_8), & (t_2, \ldots, t_8), & (t_3, \ldots, t_8), & (t_4, \ldots, t_8), \\
(t_5, \ldots, t_8), & (t_6, t_7, t_8), & (t_7, t_8), & (t_9, t_{10}), \\
(t_9, t_{10}, t_{11}), & (t_9, \ldots, t_{12}), & (t_9, \ldots, t_{13}), & (t_9, \ldots, t_{14}), \\
(t_9, \ldots, t_{15}), & (t_9, \ldots, t_{16}), & (t_1, \ldots, t_4), & (t_2, t_3, t_4), \\
(t_3, t_4) & (t_5, t_6), & (t_5, t_6, t_7), & (t_1, t_2), \\
(t_{10}, t_{11}, t_{12}), & (t_{11}, t_{12}), & (t_{13}, t_{14}), & (t_{13}, t_{14}, t_{15}), \\
(t_{13}, \ldots, t_{16}) & (t_{15}, t_{16}). & &
\end{array}
$$

- **Case** $\ell \geq 3$: Before describing the construction, we need to define two very rapidly growing functions $A(i, j)$ and $B(i, j)$, related to the Ackermann's function (see [28]):

$$
\begin{array}{rcl}
A(0, j) & = & 2j, \text{ for } j \geq 1, \\
A(i, 0) & = & 1, \text{ for } i \geq 1, \\
A(i, j) & = & A(i-1, A(i, j-1)) \text{ for } i, j \geq 1,
\end{array}
$$

and

$$
\begin{array}{rcl}
B(0, j) & = & j^2, \text{ for } j \geq 1, \\
B(i, 0) & = & 2, \text{ for } i \geq 1, \\
B(i, j) & = & B(i-1, B(i, j-1)) \text{ for } i, j \geq 1.
\end{array}
$$

For $i \geq 0$, let $w(2i, n) = \min\{j : A(i, j) \geq n\}$ and $w(2i+1, n) = \min\{j : B(i, j) \geq n\}$. The first five values of $w(\cdot, n)$ are the following:

$$
\begin{array}{ll}
w(0, n) = \lceil n/2 \rceil, & w(1, n) = \lceil \sqrt{n} \rceil, \\
w(2, n) = \lceil \log n \rceil, & w(3, n) = \lceil \log \lceil \log n \rceil \rceil, \\
w(4, n) = \log^* n, &
\end{array}
$$

whereas, $w(i, n) = \min\{j : w^{(j)}(i-2, n) \leq 1\}$, for $i \geq 2$, where $w^{(1)}(i, n) = w(i, n)$ and $w^{(j)}(i, n) = w(i, w^{(j-1)}(i, n))$, for $j \geq 2$. The function $w(\cdot, n)$ satisfies the following property (see [21]):

$$
w(2i, n) \leq w(2j, m), \text{ for any } i \geq j \text{ and any } m \geq n. \tag{1}
$$

The construction for $\ell \geq 3$ works as follows:

1. For $i = 1, \ldots, n$, insert the subsequence (t_i) into $\mathcal{P}_\ell$;
2. If $n \geq \ell + 1$, then
   (a) Let $k = w(\ell - 2, n)$ and partition T into $f = \lceil n/k \rceil \geq 2$ consecutive subsequences $T_1, \ldots, T_f$ of k elements each, where the last subsequence can contain less than k elements. For $i = 1, \ldots, f-1$, let $T_i = (t_{(i-1)k+1}, \ldots, t_{ik})$ and let $T_f = (t_{(f-1)k+1}, \ldots, t_n)$;

(b) For any $i = 1, \ldots, f-1$, and any $t \in T_i$, insert in $\mathcal{P}_\ell$ the subsequences $(t_{(i-1)k+1}, \ldots, t)$ and $(t, \ldots, t_{ik})$. For any $t \in T_f$, insert in $\mathcal{P}_\ell$ the subsequences $(t_{(f-1)k+1}, \ldots, t)$ and $(t, \ldots, t_n)$;

(c) Apply the construction recursively to each subsequence $T_1, \ldots, T_f$.

(d) Apply the construction for $\ell - 2$ to the sequence T' whose elements are $T_1, \ldots, T_f$. Let $\mathcal{P}'_{\ell-2}$ be the $(\ell-2)$-covering set constructed by the recursive call on T'. For each element $(T_i, \ldots, T_j) \in \mathcal{P}'_{\ell-2}$, insert $(t_{(i-1)k+1}, \ldots, t_{jk})$, i.e., the sequence constituted by the elements of $T_i, \ldots, T_j$, in $\mathcal{P}_\ell$.

The above construction results in a set $\mathcal{P}_\ell$ whose cardinality is given by the recurrence

$$C(\ell, n) = \begin{cases} n & \text{if } n \leq \ell; \\ \lceil n/k \rceil \cdot C(\ell, k) + C(\ell - 2, \lceil n/k \rceil) + O(n) & \text{otherwise,} \end{cases}$$

whose solution is $C(\ell, n) = O(n \cdot \ell \cdot w(\ell, n))$. For example, $C(3, n) = O(n \cdot \log\log n)$ and $C(4, n) = O(n \cdot \log^* n)$.

Example 4.2 *For $n = 16$ and $\ell = 4$, by using the above construction, we obtain the covering set $\mathcal{P}_4$ for $T = (t_1, \ldots, t_{16})$, whose elements are the following:*

$$\begin{array}{llll}
(t_1), & (t_2), & (t_3), & (t_4), \\
(t_5), & (t_6), & (t_7), & (t_8), \\
(t_9), & (t_{10}), & (t_{11}), & (t_{12}), \\
(t_{13}), & (t_{14}), & (t_{15}), & (t_{16}), \\
(t_1, t_2), & (t_1, t_2, t_3), & (t_1, \ldots, t_4), & (t_2, t_3, t_4), \\
(t_3, t_4), & (t_5, t_6), & (t_5, t_6, t_7), & (t_5, \ldots, t_8), \\
(t_6, t_7, t_8), & (t_7, t_8), & (t_9, t_{10}), & (t_9, t_{10}, t_{11}), \\
(t_9, \ldots, t_{12}), & (t_{10}, t_{11}, t_{12}), & (t_{11}, t_{12}), & (t_{13}, t_{14}), \\
(t_{13}, t_{14}, t_{15}), & (t_{13}, \ldots, t_{16}), & (t_{14}, t_{15}, t_{16}), & (t_{15}, t_{16}), \\
(t_1, \ldots, t_8), & (t_9, \ldots, t_{16}). & &
\end{array}$$

Notice that the first sixteen elements are inserted in step 1., whereas, the last two are inserted in step 2.(d). All other elements are inserted in step 2.(b) (including recursive calls in step 2.(c)).

For constant $\ell \geq 2$, the above construction is optimal. Indeed, Alon and Schieber [3] showed that the minimum possible cardinality of an ℓ-covering set for T, denoted by $C_{min}(\ell, n)$, is $\Omega(n \cdot w(\ell, n))$. Moreover, they showed that, in order to obtain an ℓ-covering set for T containing $O(n)$ elements, the value of ℓ is $\Omega(\log^* n)$.

4.2 An Optimal Construction for $\ell = O(\log^* n)$

Alon and Schieber [3] also showed an optimal construction for the case $\ell = 2\log^* n + 2$. Their construction uses as a building block the following construction for $\ell = 2\lceil \log n \rceil$.

A Tree Construction for $\ell = 2\lceil \log n \rceil$. Construct a balanced binary tree whose leaves correspond to the n elements of T and such that each internal node corresponds to the sequence whose elements are associated to the leaves in the subtree rooted at that node. An ℓ-covering set $\mathcal{P}_\ell$ for T can be obtained by considering the sequences corresponding to the nodes of the tree. Indeed, any contiguous subsequence of T can be obtained as the union of at most $2\lceil \log n \rceil$ elements of $\mathcal{P}_\ell$. Since the number of nodes in the tree is $2n - 1$, it follows that the cardinality of $\mathcal{P}_\ell$ is $O(n)$.

A Construction for $\ell = 2\log^* n + 2$. The tree construction, as well as the one of Section 4.1, can be used to construct an ℓ-covering set for T, where $\ell = 2\log^* n + 2$, containing $O(n)$ elements. The new construction works as follows:

1. Let $h = 2(\log^* n)^2$ and partition T into $f = \lceil n/h \rceil$ consecutive subsequences $T_1, \ldots, T_f$ of h elements each, where the last sequence can contain less elements;
2. For any $i = 1, \ldots, f$ and any sequence T_i, apply the tree construction for $\ell' = 2\lceil \log h \rceil$ to the sequence T_i. For any $i = 1, \ldots, f$, insert into $\mathcal{P}_\ell$ all the elements of the resulting ℓ'-covering set for T_i;
3. Apply the construction of Section 4.1 for $\ell - 2 = 2\log^* n$ to the sequence T' whose elements are $T_1, \ldots, T_f$. Let $\mathcal{P}'_{\ell-2}$ be the $(\ell - 2)$-covering set constructed by the recursive call on T'. For each element $(T_i, \ldots, T_j) \in \mathcal{P}'_{\ell-2}$, insert $(t_{(i-1)h+1}, \ldots, t_{jh})$, i.e., the sequence constituted by the elements of $T_i, \ldots, T_j$, in $\mathcal{P}_\ell$.

The above construction results in a set $\mathcal{P}_\ell$ containing $O(n)$ elements. Indeed, step 2. inserts $2h - 1$ elements in $\mathcal{P}_\ell$ for each sequence T_i, for a total of $\lceil n/h \rceil \cdot (2h - 1)$ elements, whereas, step 3. inserts $C(2\log^* n, \lceil n/h \rceil) = O(\lceil n/h \rceil \cdot 2\log^* n \cdot w(2\log^* n, \lceil n/h \rceil))$ elements. Therefore

$$|\mathcal{P}_\ell| = \lceil n/h \rceil \cdot (2h - 1) + O(\lceil n/h \rceil \cdot 2\log^* n \cdot w(2\log^* n, \lceil n/h \rceil)).$$

Since from inequality (1) it holds that $w(2\log^* n, \lceil n/h \rceil) \leq w(2\log^* n, n)$ and $w(2\log^* n, n) \leq w(4, n) = \log^* n$ for any $n \geq 4$, and since $h = 2(\log^* n)^2$, it follows that $|\mathcal{P}_\ell| = O(n)$.

The above construction is optimal. Indeed, any ℓ-covering set for T has to contain at least the n elements of T.

4.3 A Graph Transformation using an ℓ-Covering Set

In this section we show how to use an ℓ-covering set in order to construct a time-bound hierarchical key assignment scheme. We first need to represent the ℓ-covering set as a directed acyclic graph. We consider the following two representations:

- The *minimal edge representation* $H^{(1)}_{T,\ell} = (\mathcal{P}_\ell, E^{(1)}_{\mathcal{P}_\ell})$ is the minimal representation of the subgraph of $H^*_T = (\mathcal{P}, E^*_{\mathcal{P}})$ induced by $\mathcal{P}_\ell$. Figures 4 and 5 show the graphs $H^{(1)}_{T,2}$ and $H^{(1)}_{T,4}$ representing the 2-covering set $\mathcal{P}_2$ and the 4-covering set $\mathcal{P}_4$, respectively, where the node corresponding to each sequence $(t_i, \ldots, t_j)$ is denoted by $[i, j]$.
- The *minimal diameter representation* $H^{(2)}_{T,\ell} = (\mathcal{P}_\ell, E^{(2)}_{\mathcal{P}_\ell})$ is defined as follows: for any $\lambda \in \mathcal{P}_\ell$ containing at least two elements and any $t \in \lambda$, there is an edge between λ and (t) in $H^{(2)}_{T,\ell}$. Figure 6 shows the graph $H^{(2)}_{T,4}$ representing the 4-covering set $\mathcal{P}_4$, where the node corresponding to each sequence $(t_i, \ldots, t_j)$ is denoted by $[i, j]$.

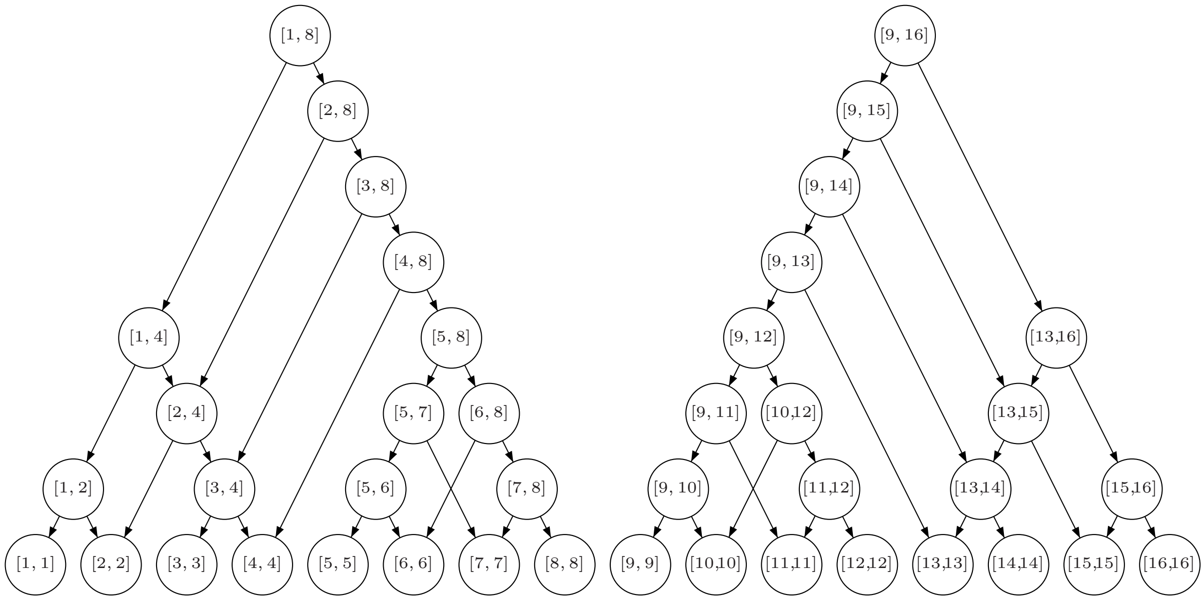

Figure 4: The graph $H^{(1)}_{T,2}$ representing the 2-covering set $\mathcal{P}_2$ for $T = (t_1, \ldots, t_{16})$.

In order to obtain a time-bound hierarchical key assignment scheme, we define a graph transformation, called the (ℓ, j)-*Covering Set Based Transformation (CSBT)*, making use of an ℓ-covering set for T. Such a transformation, on input the graphs $G = (V, E)$ and $H^{(j)}_{T,\ell} = (\mathcal{P}_\ell, E^{(j)}_{\mathcal{P}_\ell})$, where $j \in \{1, 2\}$, constructs a new graph $G'' = (V'', E'')$, as follows:

- For each class $u \in V$ and each time sequence $\lambda \in \mathcal{P}_\ell$, we place a class $u_\lambda \in V''$.
- For each class $u \in V$ and each pair of time sequences $\lambda, \gamma \in \mathcal{P}_\ell$ connected by an edge in $E^{(j)}_{\mathcal{P}_\ell}$, we place and edge between u_λ and u_γ in G'', i.e., $(u_\lambda, u_\gamma) \in E''$. Such edges are called *decomposition time edges*.
- For each pair of classes u and v connected by an edge in E and each time period $t \in T$, we

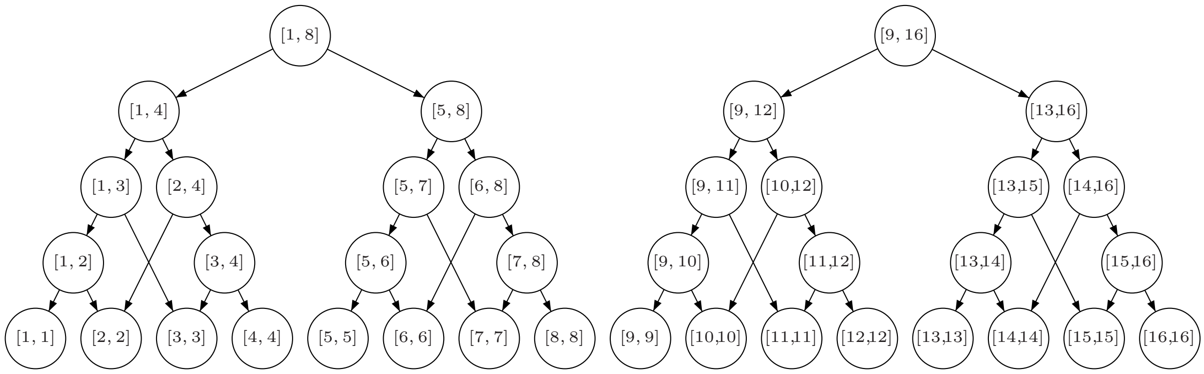

Figure 5: The graph $H^{(1)}_{T,4}$ representing the 4-covering set $\mathcal{P}_4$ for $T = (t_1, \ldots, t_{16})$.

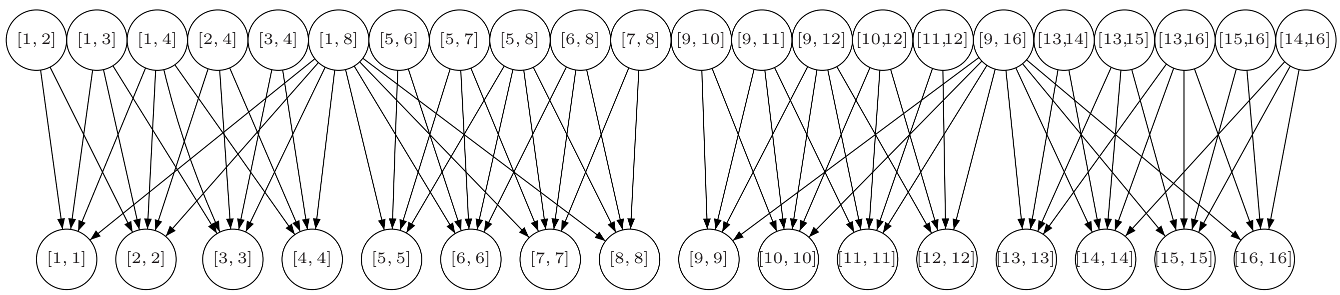

Figure 6: The graph $H^{(2)}_{T,4}$ representing the 4-covering set $\mathcal{P}_4$ for $T = (t_1, \ldots, t_{16})$.

place an edge between u_t and v_t in G'', i.e., $(u_t, v_t) \in E''$. Such edges are called *policy edges.*

The right hand side of Figure 7 shows the graph G'' obtained by the $(2, 1)$-CSBT starting from the graph G shown on the left hand side of Figure 7 and from the graph $H^{(1)}_{T,2}$ representing the 2-covering set $\mathcal{P}_2 = \{(t_1), (t_2), (t_3), (t_4), (t_1, t_2), (t_3, t_4)\}$ for $T = (t_1, \ldots, t_4)$. Decomposition time edges and policy edges are represented by dashed and solid lines, respectively.

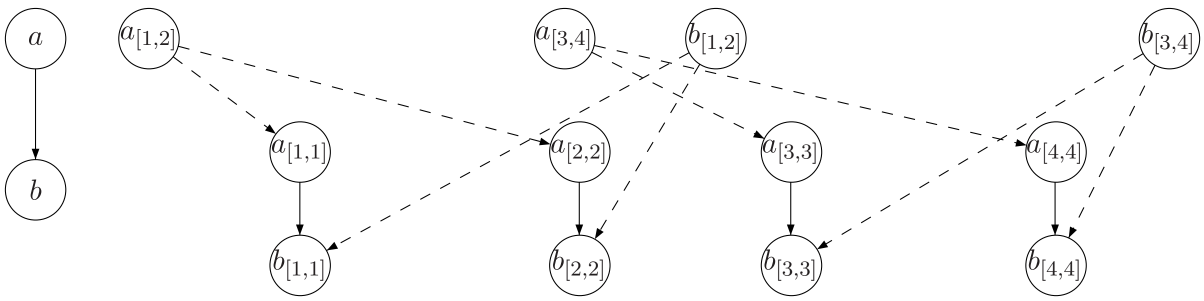

Figure 7: The graph G'' (on the right) corresponding to the graph G (on the left) and the graph $H^{(1)}_{T,2}$.

It is easy to see that $|V''| = |V| \cdot |\mathcal{P}_\ell|$ and $|E''| = |V| \cdot |E^{(j)}_{\mathcal{P}_\ell}| + |E| \cdot |T|$. Indeed, E'' contains $|V| \cdot |E^{(j)}_{\mathcal{P}_\ell}|$ decomposition time edges and $|E| \cdot |T|$ policy edges.

Once the graph $G'' = (V'', E'')$ has been obtained, we can use any hierarchical key assignment scheme. For example, the BEBC based on pairings requires $4|V''| + 1 = O(|V| \cdot |\mathcal{P}_\ell|)$ public values, whereas, each user is required to hold at most ℓ secret values. In particular, the amount of public information is

- $O(|V| \cdot |T| \cdot \log |T|)$ if $\ell = 2$;
- $O(|V| \cdot |T| \cdot \log \log |T|)$ if $\ell = 3$;
- $O(|V| \cdot |T| \cdot \log^* |T|)$ if $\ell = 4$;
- $O(|V| \cdot |T|)$ if $\ell = 2 \log^* |T| + 2$.

The parameters of the above schemes are summarized in rows 4, 5, 6, and 7 of Figure 10. Notice that since for all values of $|T|$ less than $2^{65,536}$, corresponding to much more than the number of atoms in the universe, $\log^* |T|$ does not exceed 5, the number of private information held by each user when $\ell = 2\log^* |T| + 2$ may be considered to be a constant.

On the other hand, the EBC requires $|V''| + |E''| = O(|E| \cdot |T| + |V| \cdot |E^{(j)}_{\mathcal{P}_\ell}|)$ public values, whereas, each user holds at most ℓ secret values. Moreover, the number of decryption operations needed to perform key derivation is $O(diam(H^{(j)}_{T,\ell}) + diam(G))$. Therefore, in order to evaluate the resulting scheme, we need to compute the number of edges and the diameter of both graphs $H^{(1)}_{T,\ell}$ and $H^{(2)}_{T,\ell}$.

The Minimal Edge Representation. In the following we evaluate the schemes obtained by the EBC when the minimal edge representation of the ℓ-covering set is considered. We first show that the minimal edge representation of an ℓ-covering set for T satisfies an interesting property, called the *Graph Composition Property*, then we compute both the number of edges and the diameter of such a graph.

GRAPH COMPOSITION PROPERTY: For any $\ell \geq 3$, the minimal edge representation of the ℓ-covering set for T, obtained by the constructions in Section 4.1 and 4.2, can be seen as the composition of two graphs, called the bottom and the top graph, respectively.

Indeed, consider the construction of Section 4.1 and let $T_1, \ldots T_{\lceil n/k \rceil}$ be the subsequences of T obtained in step 2.(a) of the algorithm. The nodes of the bottom graph are the successors of the nodes corresponding to the sequences $T_1, \ldots T_{\lceil n/k \rceil}$, whereas, those in the top graph are the predecessor of such nodes. In particular, the nodes of the bottom graph correspond to sequences inserted in $\mathcal{P}_\ell$ in step 2.(b), including recursive calls in step 2.(c), whereas, the nodes of the top graph are those corresponding to the sequences inserted in step 2.(d).

As regards as the construction of Section 4.2 for $\ell = 2\log^* n + 2$, let $T_1, \ldots T_{\lceil n/h \rceil}$ be the subsequences of T obtained in step 1. of the algorithm. The nodes of the bottom graph are the successors of the nodes corresponding to the sequences $T_1, \ldots T_{\lceil n/h \rceil}$, whereas, those in the top graph are the predecessors of such nodes. In particular, the nodes of the bottom graph correspond to sequences inserted in $\mathcal{P}_\ell$ in step 2., whereas, the nodes of the top graph are those corresponding to the sequences inserted in step 3.

Now we are ready to compute the number of edges of the minimal edge representation of the ℓ-covering set obtained by the construction of Section 4.1.

Lemma 4.3 *Let $\ell \geq 1$ and let $\mathcal{P}_\ell$ be the ℓ-covering set for a sequence T of n elements obtained by the construction in Section 4.1. The number of edges in the minimal edge representation of $\mathcal{P}_\ell$ is $2(|\mathcal{P}_\ell| - |T|)$.*

Proof. We prove that each node in $\mathcal{P}_\ell$, with the exception of sink nodes, has exactly two outgoing edges in $H^{(1)}_{T,\ell}$. The proof is by induction on ℓ. Clearly, the lemma holds for $\ell = 1$, since $\mathcal{P}_1 = \mathcal{P}$ and $|E^{(1)}_{\mathcal{P}_1}| = |E_{\mathcal{P}}| = 2(|\mathcal{P}| - |T|)$.

Let $\ell = 2$ and let $(t_i, \ldots, t_j)$ be a sequence inserted in $\mathcal{P}_2$ in step 2.(b) of the algorithm. Notice that also $(t_{i+1}, \ldots, t_j)$ has been inserted in $\mathcal{P}_2$ in step 2.(b). Let $h_{max} = \max\{i \leq h < j : (t_i, \ldots, t_h) \in \mathcal{P}_2\}$. Hence, in the minimal edge representation of $\mathcal{P}_2$, the node $(t_i, \ldots, t_j)$ is connected both to $(t_i, \ldots, t_{h_{max}})$ and to $(t_{i+1}, \ldots, t_j)$. It is easy to see that all other subsequences of $(t_i, \ldots, t_j)$ inserted in $\mathcal{P}_2$ are subsequences of either $(t_i, \ldots, t_{h_{max}})$ or $(t_{i+1}, \ldots, t_j)$, thus the node $(t_i, \ldots, t_j)$ has only two outgoing edges in the minimal edge representation of $\mathcal{P}_2$. Similarly, we can show that each node corresponding to a sequence inserted in $\mathcal{P}_2$ in step 2.(c) of the algorithm has exactly two outgoing edges.

Assume by inductive hypothesis that for any $1 \leq \ell' < \ell$, each node in $\mathcal{P}_{\ell'}$, with the exception of sink nodes, has exactly two outgoing edges in the minimal edge representation of $\mathcal{P}_{\ell'}$.

By the *Graph Composition Property*, the minimal edge representation of the ℓ-covering set $\mathcal{P}_\ell$ can be seen as the composition of the bottom graph and the top graph. Since the top graph corresponds to the minimal edge representation of the $(\ell-2)$-covering set constructed in step 2.(d) of the algorithm, by the inductive hypothesis it follows that the nodes in the top graph have exactly two outgoing edges. Thus, we only have to consider the nodes in the bottom graph, corresponding to sequences inserted in $\mathcal{P}_\ell$ in step 2.(b). Let $(t_i, \ldots, t_j)$ be a sequence inserted in $\mathcal{P}_\ell$ in step 2.(b) of the algorithm. It is easy to see that at least one sequence between $(t_i, \ldots, t_{j-1})$ and $(t_{i+1}, \ldots, t_j)$ has been inserted in $\mathcal{P}_\ell$. W.l.o.g., assume that $(t_{i+1}, \ldots, t_j)$ has been inserted in $\mathcal{P}_\ell$. Let $h_{max} = \max\{i \leq h < j : (t_i, \ldots, t_h) \in \mathcal{P}_2\}$. Hence, in the minimal edge representation of $\mathcal{P}_\ell$, the node $(t_i, \ldots, t_j)$ is connected both to $(t_i, \ldots, t_{h_{max}})$ and to $(t_{i+1}, \ldots, t_j)$. It is easy to see that all other subsequences of $(t_i, \ldots, t_j)$ inserted in $\mathcal{P}_\ell$ are subsequences of either $(t_i, \ldots, t_{h_{max}})$ or $(t_{i+1}, \ldots, t_j)$, thus the node $(t_i, \ldots, t_j)$ has only two outgoing edges in the minimal edge representation of $\mathcal{P}_\ell$.

$\square$

In the following we compute the diameter of the minimal edge representation of the ℓ-covering set $\mathcal{P}_\ell$ obtained by the construction in Section 4.1.

Lemma 4.4 *Let $\ell \geq 1$ and let $\mathcal{P}_\ell$ be the ℓ-covering set for a sequence T of n elements obtained by the construction in Section 4.1. If $n \geq \ell+1$, the diameter $D(\ell, n)$ of the minimal edge representation of $\mathcal{P}_\ell$ satisfies*

$$D(\ell, n) = \begin{cases} n-1 & \text{if } \ell = 1; \\ \lceil n/2 \rceil - 1 & \text{if } \ell = 2; \\ k - 1 + D(\ell-2, \lceil n/k \rceil) & \text{if } \ell \geq 3, \end{cases}$$

where $k = w(\ell-2, n)$, whereas, $D(\ell, n) = 0$ if $n \leq \ell$.

Proof. The case $\ell = 1$ is trivial. For the case $\ell = 2$, notice that the largest sequence in $\mathcal{P}_2$ is $(t_1, \ldots, t_{\lceil n/2 \rceil})$, which has been inserted in step 2.(b) of the algorithm. Since, for any $j = 2, \ldots, \lceil n/2 \rceil - 1$, the sequence $(t_j, \ldots, t_{\lceil n/2 \rceil})$ has also been inserted in $\mathcal{P}_2$ in step 2.(b), there is an edge connecting the nodes $(t_{j-1}, \ldots, t_{\lceil n/2 \rceil})$ and $(t_j, \ldots, t_{\lceil n/2 \rceil})$ in the minimal edge representation of $\mathcal{P}_2$. Similarly, since the sequence $(t_{\lceil n/2 \rceil})$ has been inserted in $\mathcal{P}_2$ in step 1., there is an edge connecting $(t_{\lceil n/2 \rceil - 1}, t_{\lceil n/2 \rceil})$ and $(t_{\lceil n/2 \rceil})$. By the above discussion it follows that there is a path of length $\lceil n/2 \rceil - 1$ between $(t_1, \ldots, t_{\lceil n/2 \rceil})$ and $(t_{\lceil n/2 \rceil})$, thus the diameter of the minimal edge representation of $\mathcal{P}_2$ is $\lceil n/2 \rceil - 1$.

For the case $\ell \geq 3$, recall that by the *Graph Composition Property*, the minimal edge representation of $\mathcal{P}_\ell$ can be seen as the composition of the bottom graph and the top graph. Therefore, the diameter $D(\ell, n)$ of the minimal edge representation of $\mathcal{P}_\ell$ is the sum of the diameters of such two graphs. Since the top graph corresponds to the minimal edge representation of the $(\ell-2)$-covering set for a sequence of $\lceil n/k \rceil$ elements, constructed in step 2.(d) of the algorithm, its diameter is $D(\ell-2, \lceil n/k \rceil)$. Thus, we only need to compute the diameter of the bottom graph, whose nodes correspond to the sequences inserted in $\mathcal{P}_\ell$ in step 2.(b) of the algorithm. To this aim, notice that the largest sequence inserted in $\mathcal{P}_\ell$ in step 2.(b) contains $k = w(\ell-2, n)$ elements. Without loss of generality, let $(t_1, \ldots, t_k)$ be one of the largest sequences. Since, for any $j = 2, \ldots, k-1$, the sequence $(t_j, \ldots, t_k)$ has also been inserted in $\mathcal{P}_\ell$ in step 2.(b), there is an edge connecting the nodes $(t_{j-1}, \ldots, t_k)$ and $(t_j, \ldots, t_k)$ in the minimal edge representation of $\mathcal{P}_\ell$. Similarly, since the sequence (t_k) has been inserted in $\mathcal{P}_\ell$ in step 1., there is an edge connecting (t_{k-1}, t_k) and (t_k). By the above discussion it follows that there is a path of length $k-1$ between $(t_1, \ldots, t_k)$ and (t_k), therefore, the diameter of the bottom graph is $k-1$. Thus, the lemma follows. □

By the above lemma it follows that

$$D(3, n) = 2\lceil \sqrt{n} \rceil - 2,$$

$$D(4, n) = \lceil \log n \rceil + \lceil n / \lceil 2 \log n \rceil \rceil - 2,$$

and

$$D(6, n) = \log^* n + \lceil \log \lceil n / \log^* n \rceil \rceil + \left\lceil \frac{n}{2 \log^* n \cdot \lceil \log \lceil \frac{n}{\log^* n} \rceil \rceil} \right\rceil - 3.$$

By Lemmas 4.3 and 4.4 it follows that the schemes obtained by the EBC on the graph $G'' = (V'', E'')$ resulting from the $(\ell, 1)$-CSBT have the following parameters:

- If $\ell = 2$, then the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T| \log |T|)$, whereas, key derivation requires $O(|T| + diam(G))$ decryptions;
- If $\ell = 3$, then the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T| \cdot \log \log |T|)$, whereas, key derivation requires $O(\sqrt{|}T| + diam(G))$ decryptions;
- If $\ell = 4$, then the amount of public information is $O(|E| \cdot |T| + |V| \cdot |T| \cdot \log^* |T|)$, whereas, key derivation requires $O(|T| / \log |T| + diam(G))$ decryptions.

The parameters of the above schemes are summarized in rows 6, 7, and 8 of Figure 9.

In order to evaluate the schemes obtained by the EBC on the graph G'' resulting from the $(\ell, 1)$-CSBT when $\ell = 2 \log^* |T| + 2$, we need to consider the construction of Section 4.2.

Lemma 4.5 *Let T be a sequence of n elements, let $\ell = 2 \log^* n + 2$ and let $\mathcal{P}_\ell$ be the ℓ-covering set for T obtained by the construction in Section 4.2. The number of edges in the minimal edge representation of $\mathcal{P}_\ell$ is $2(|\mathcal{P}_\ell| - |T|)$.*

Proof. We prove that each node in $\mathcal{P}_\ell$, with the exception of sink nodes, has exactly two outgoing edges in $H^{(1)}_{T,\ell}$.

Let $(t_i, \ldots, t_j)$ be a sequence inserted in $\mathcal{P}_\ell$ in step 2. of the algorithm. Thus, it corresponds to either an internal node or a leaf of a balanced binary tree. In the first case, the node has exactly two outgoing edges in $H^{(1)}_{T,\ell}$, whereas, in the second case, it corresponds to a sink node.

Let $(t_i, \ldots, t_j)$ be a sequence inserted in $\mathcal{P}_\ell$ in step 3. of the algorithm. By Lemma 4.3 it follows that the corresponding node in $\mathcal{P}_\ell$ has exactly two outgoing edges in $H^{(1)}_{T,\ell}$. □

Lemma 4.6 *Let T be a sequence of n elements, let $\ell = 2\log^* n + 2$ and let $\mathcal{P}_\ell$ be the ℓ-covering set for T obtained by the construction in Section 4.2. The diameter of the minimal edge representation of $\mathcal{P}_\ell$ is*

$$diam(H^{(1)}_{T,\ell}) = O\left(\frac{n}{(\log^* n)^3 \cdot \log n}\right).$$

Proof. Recall that by the *Graph Composition Property*, the minimal edge representation of $\mathcal{P}_\ell$ can be seen as the composition of the bottom graph and the top graph. Therefore, the diameter of the minimal edge representation of $\mathcal{P}_\ell$ is the sum of the diameters of such two graphs. Since the bottom graph is the binary tree obtained by the tree construction for $\ell' = 2\lceil \log h\rceil$, where $h = 2(\log^* n)^2$, its diameter is $\lceil \log h\rceil = O(\log^* n)$. On the other hand, the diameter of the bottom graph, which corresponds to the minimal representation of the $(2\log^* n)$-covering set for the sequence of $\lceil n/h\rceil$ elements constructed in step 3. of the algorithm, is $D\left(2\log^* n, \lceil \frac{n}{h}\rceil\right)$. By iterative applications of Lemma 4.4 and from inequality (1), it follows that

$$D\left(2\log^* n, \left\lceil\frac{n}{h}\right\rceil\right) \leq \sum_{i=3}^{\log^* n - 1} w(2i, \lceil n/h\rceil) + D(6, \lceil n/h\rceil).$$

The last term can be bounded as follows

$$\begin{aligned} D(6,\lceil n/h\rceil) &\leq \log^*\lceil n/h\rceil + \lceil \log\lceil n/(h\cdot \log^*\lceil n/h\rceil)\rceil\rceil + \left\lceil \frac{n}{2h\cdot \log^*\lceil n/h\rceil \cdot \lceil \log\lceil \frac{n}{h\cdot\log^*\lceil n/h\rceil}\rceil\rceil}\right\rceil \\ &\leq \log^* n + \lceil \log n\rceil + \left\lceil \frac{n}{2(\log^* n)^3\cdot \lceil \log n\rceil}\right\rceil. \end{aligned}$$

Since $w(2i, \lceil n/h\rceil) \leq w(2i, n)$ and $w(2i, n) \leq \log^* n$, for any $i \geq 2$, it follows that

$$D\left(2\log^* n, \left\lceil\frac{n}{h}\right\rceil\right) \leq (\log^* n)^2 + \log^* n + \lceil \log n\rceil + \left\lceil \frac{n}{2(\log^* n)^3\cdot \lceil \log n\rceil}\right\rceil.$$

Therefore, the diameter of the minimal edge representation of $\mathcal{P}_\ell$ is $O\left(\frac{n}{(\log^* n)^3\cdot \log n}\right)$. □

By Lemmas 4.5 and 4.6 it follows that the amount of public information in the scheme obtained by the EBC on the graph $G'' = (V'', E'')$, resulting from the $(\ell, 1)$-CSBT when $\ell = 2\log^* |T| + 2$, is $O(|E|\cdot|T|)$, whereas, key derivation requires $O\left(\frac{|T|}{(\log^*|T|)^3\cdot \log|T|} + diam(G)\right)$ decryptions. The parameters of the above scheme are summarized in row 9 of Figure 9.

Shortcutting the Minimal Edge Representation. The number of decryption operations required for key derivation in the above schemes could be reduced by using the techniques of Section 3.2.1. on the minimal edge representation of the ℓ-covering set for T. First, notice that such a graph is a rooted acyclic planar graph. Thus, by applying Thorup's shortcutting technique we obtain a shortcut graph having diameter $O(\log|T|\cdot\log^*|T|)$, by adding at most $O(|T|\cdot w(\ell, |T|))$ shortcut edges. The parameters of the resulting schemes are summarized in rows 10, 11, and 12 of Figure 9. On the other hand, by using the ISPIT on the minimal edge representation of $\mathcal{P}_\ell$, which has dimension two, we obtain either a graph having diameter $O(\log^*|T|)$, by adding $O(|T|\cdot\log|T|\cdot w(\ell,|T|))$ new edges and vertices, or a graph having diameter three by adding at most $O(|T|\cdot\log|T|\cdot\log\log|T|\cdot w(\ell,|T|))$. The parameters of the resulting schemes are summarized in rows from 13 to 18 of Figure 9.

The Minimal Diameter Representation. In the following we compute the number of edges $F(\ell, n)$ in the minimal diameter representation of the ℓ-covering set for a sequence T of n elements obtained by the construction in Section 3.3.1. for $\ell \geq 1$. Notice that for each node in $\mathcal{P}_\ell$ corresponding to a sequence λ, with the exception of sink nodes, there are $|\lambda|$ edges in the minimal diameter representation of $\mathcal{P}_\ell$, thus $F(\ell, n) = \sum_{\lambda \in \mathcal{P}_\ell} |\lambda|$.

First, consider the case $\ell = 1$. It is easy to see that $F(1, n) = \sum_{i=1}^{n} i = O(n^2)$. As regards as the case $\ell = 2$ and $n \geq 3$, since there are $2(\lceil n/2 \rceil - 1)$ sequences inserted in $\mathcal{P}_2$ in steps 2.(b) and 2.(c), each having length at most $\lceil n/2 \rceil$, it follows that the sum of the lengths of such sequences is $n \cdot (\lceil n/2 \rceil - 1) = O(n^2)$. Therefore,

$$F(2, n) \leq 2 \cdot F(2, \lceil n/2 \rceil) + O(n^2),$$

whereas, $F(2, n) = 0$ if $n \leq 2$. It is easy to see that $F(2, n) = O(n^2 \cdot \log n)$.

Now, consider the case $\ell \geq 3$ and $n \geq \ell + 1$. Since there are $2(k-1) \cdot \lceil n/k \rceil$ sequences inserted in $\mathcal{P}_\ell$ in step 2.(b), each having length at most $k = w(\ell - 2, n)$, it follows that the sum of the lengths of such sequences is $2(k-1) \cdot \lceil n/k \rceil \cdot k = O(n \cdot k)$. Therefore,

$$F(\ell, n) \leq \lceil n/k \rceil \cdot F(\ell, k) + F(\ell - 2, \lceil n/k \rceil) + O(n \cdot k),$$

whereas, $F(\ell, n) = 0$ if $n \leq \ell$. It is easy to compute $F(\ell, n)$ for $\ell = 3, 4, 5$. Indeed we have that

- $F(3, n) \leq \lceil \sqrt{n} \rceil \cdot F(3, \lceil \sqrt{n} \rceil) + F(1, \lceil \sqrt{n} \rceil) + O(n \cdot \sqrt{n})$, whose solution is $F(3, n) = O(n \cdot \sqrt{n} \cdot \log \log n)$;

- $F(4, n) \leq \lceil n / \lceil \log n \rceil \rceil \cdot F(4, \lceil \log n \rceil) + F\left(2, \lceil n / \lceil \log n \rceil \rceil\right) + O(n \cdot \log n)$, whose solution is $F(4, n) = O(n^2 / \log n)$;

- $F(5, n) \leq n / \lceil \log \lceil \log n \rceil \rceil \cdot F(5, \lceil \log \lceil \log n \rceil \rceil) + F(3, n / \lceil \log \lceil \log n \rceil \rceil) + O(n \lceil \log \lceil \log n \rceil \rceil)$, whose solution is $F(5, n) = O(n \cdot \sqrt{n} / \sqrt{\log \log n})$.

For the general case, where $\ell \geq 6$, we can easily prove that

$$F(\ell, n) = O(n \cdot \ell \cdot w(\ell, n) \cdot \log^* n).$$

As regards as the diameter of $H^{(2)}_{T,\ell}$, clearly, $diam(H^{(2)}_{T,\ell}) = 1$.

Therefore, it follows that in the schemes obtained by the EBC on the graph $G'' = (V'', E'')$ resulting from the $(\ell, 2)$-CSBT the amount of public information is

- $O(|E| \cdot |T| + |V| \cdot |T|^2 \log |T|)$ if $\ell = 2$;
- $O(|E| \cdot |T| + |V| \cdot |T| \cdot \sqrt{|T|} \cdot \log \log |T|)$ if $\ell = 3$;
- $O(|E| \cdot |T| + |V| \cdot |T|^2 / \log |T|)$ if $\ell = 4$.

On the other hand, all the above schemes require $O(diam(G))$ decryptions for key derivation. The parameters of the above schemes are summarized in rows 19, 20, and 21 of Figure 9.

In order to evaluate the schemes obtained by the EBC on the graph G'' resulting from the $(\ell, 2)$-CSBT when $\ell = 2 \log^* |T| + 2$, we need to compute the number of edges in the minimal diameter representation of the ℓ-covering set obtained by the construction of Section 4.2.

Lemma 4.7 *Let T be a sequence of n elements, let $\ell = 2\log^* n + 2$ and let $\mathcal{P}_\ell$ be the ℓ-covering set for T obtained by the construction in Section 4.2. The number of edges of the minimal diameter representation of $\mathcal{P}_\ell$ is $O(n \cdot (\log^* n)^3)$.*

Proof. First notice that for each node in $\mathcal{P}_\ell$ corresponding to a sequence λ, with the exception of sink nodes, there are $|\lambda|$ edges in the minimal diameter representation of $\mathcal{P}_\ell$. We distinguish between sequences inserted in $\mathcal{P}_\ell$ in steps 2. and 3. of the algorithm. As regards as the former ones, corresponding to nodes of the bottom graph, it is easy to see that there are $n \cdot \lceil \log h \rceil$ edges connecting nodes corresponding to such sequences to leaf nodes. As regards as the latter ones, corresponding to nodes of the top graph, there are $h \cdot F(2\log^* n, \lceil n/h \rceil)$ edges connecting nodes corresponding to such sequences to leaf nodes. Indeed, $F(2\log^* n, \lceil n/h \rceil) = O(n/h \cdot (\log^* n)^2 \cdot w(2\log^* n, \lceil n/h \rceil)) = O(n \cdot w(2\log^* n, \lceil n/h \rceil))$ denotes the number of edges connecting the nodes corresponding to sequences inserted in step 3. to those corresponding to the sequences $T_1, \ldots, T_{\lceil n/h \rceil}$, which have h elements each. It follows that the number of edges of the minimal diameter representation of $\mathcal{P}_\ell$ is

$$n \cdot \lceil \log h \rceil + O(h \cdot n \cdot w(2\log^* n, \lceil n/h \rceil)).$$

¿From inequality (1) we have that $w(2\log^* n, \lceil n/h \rceil) \leq w(2\log^* n, n)$ and $w(2\log^* n, n) \leq w(4, n) = \log^* n$ for any $n \geq 4$. Since $h = 2(\log^* n)^2$, it follows that the number of edges of the minimal diameter representation of $\mathcal{P}_\ell$ is

$$O(n \cdot \log h + h \cdot n \cdot \log^* n) = O(n \cdot (\log^* n)^3).$$

□

By Lemma 4.7 it follows that the amount of public information in the scheme obtained by the EBC on the graph $G'' = (V'', E'')$, resulting from the $(\ell, 2)$-CSBT when $\ell = 2\log^* |T| + 2$, is $O(|E| \cdot |T| + |V| \cdot |T| \cdot (\log^* |T|)^3)$, whereas, key derivation requires $O(diam(G))$ decryptions. The parameters of the above scheme are summarized in row 22 of Figure 9.

5 Summary of the Results

In this paper we have proposed new constructions for time-bound hierarchical key assignment schemes which are provably-secure with respect to key indistinguishability. Our constructions use as a building block *any* provably-secure hierarchical key assignment scheme without temporal constraints and result in time-bound hierarchical key assignment schemes whose parameters depend on those of the underlying schemes. In particular, by using two schemes recently proposed in [15], we obtain different constructions which exhibit a tradeoff among the amount of private information held by each class, the amount of public data, the complexity of key derivation, and the computational assumption on which their security is based. Moreover, the proposed schemes support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed. Clearly, starting from constructions for hierarchical key assignment schemes without time constraints, other than the EBC and the BEBC, our technique allows to obtain time-bound schemes with possible new parameters.

Figure 8 contains a list of the acronyms used in the paper, whereas, Figures 9 and 10 show a comparison between our schemes and previous proposals also offering security with respect to key indistinguishability. In particular, all schemes in Figure 9 are based on a standard computational assumption regarding the existence of IND-P1-C0 secure symmetric encryption schemes, whereas, the schemes in Figure 10 are based on two different assumptions on pairings, the BDDH and the n-BDDHE. It is worth to notice that while the BDDH is considered a standard assumption, the n-BDDHE, first introduced in [8], is quite new.

BEBC	Broadcast Encryption Based Construction [15]
CSBT	Covering Set Based Transformation (Section 4)
EBC	Encryption Based Construction [15]
IHBT	Interval Hierarchy Based Transformation (Section 3)
ISPIT	Improved Shortcutting and Point-Inserting Technique [15]
ST	Shortcutting Technique [29]
TLEBC	Two-Level Encryption Based Construction [6]
TLPBC	Two-Level Pairing Based Construction [6]

Figure 8: List of acronyms used in the paper.

The schemes in both figures are evaluated with respect to several parameters, such as the amount of public information, the amount of private information held by each class, and the number of operations required to perform key derivation. Notice that in the schemes obtained by the ℓ-CSBT the greater the value of ℓ, the smaller the amount of public information required by the resulting scheme. In particular, the amount of public information of the scheme resulting by the $(3,1)$-CSBT using the BEBC as a building block is $O(|V| \cdot |T| \cdot \log\log|T|)$, whereas that of the scheme resulting by the $(4,1)$-CSBT is $O(|V| \cdot |T| \cdot \log^* |T|)$. Since $\log^* |T|$ is a very slowly growing function and we are not interested in functions which grow even slower than $\log^* |T|$, we did not show in Figures 9 and 10 the parameters of the schemes obtained when constants $\ell \geq 5$ are considered. However, in order to obtain a scheme whose public information is $O(|V| \cdot |T|)$, we have used the $(\ell, 2)$-CSBT with $\ell = 2\log^* |T| + 2$.

We remark that no scheme in Figures 9 and 10 is superior to the others with respect to all parameters. An open problem would be to find a time-bound hierarchical key assignment scheme which optimizes all parameters at the same time. However, it is worth to notice that our techniques allow to obtain schemes offering better performances whenever further constraints on the temporal access control policy occur. For example, if the access control policy requires each user to be assigned to a certain class for a sequence of *at least* $\log^* |T|$ contiguous time periods, then by using the $(4,1)$-CSBT along with the EBC, we obtain a scheme whose amount of public information is $O(|E| \cdot |T|)$ instead than $O(|E| \cdot |T| + |V| \cdot |T| \cdot \log^* |T|)$, whereas the amount of private information and the complexity of key derivation stay the same. Similarly, by using the $(4,1)$-CSBT along with the BEBC, we obtain a scheme whose amount of public information is $O(|V| \cdot |T|)$ instead than $O(|V| \cdot |T| \cdot \log^* |T|)$.

The schemes proposed by Wang and Laih [37] and Tzeng [31] are not shown in Figures 9 and 10, since it is not clear under which assumptions such schemes can be considered to be provably-secure. However, we recall that both schemes require $|V| \cdot |T|$ public values, whereas, each user has to store a single secret value. Finally, key derivation involves one modular exponentiation and $O(|V| \cdot |T|)$ operations.

	Scheme	Public information	Private information	Key derivation
1	TLEBC [6]	$O(\lvert E^*\rvert \cdot \lvert T\rvert^3)$	One	One decryption
2	Naive + EBC [15]	$O(\lvert E\rvert \cdot \lvert T\rvert)$	$O(\lvert T\rvert)$	$O(diam(G))$ decryptions
3	ST + IHBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert^2)$	One	$O(\log\lvert T\rvert \cdot \log^*\lvert T\rvert + diam(G))$ decryptions
4	ISPIT + IHBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert^2 \cdot \log\lvert T\rvert)$	One	$O(\log^*\lvert T\rvert + diam(G))$ decryptions
5	ISPIT + IHBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert^2 \cdot \log\lvert T\rvert \cdot \log\log\lvert T\rvert)$	One	$O(diam(G))$ decryptions
6	$(2,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert)$	Two (at most)	$O(\lvert T\rvert + diam(G))$ decryptions
7	$(3,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\log\lvert T\rvert)$	Three (at most)	$O(\sqrt{\lvert T\rvert} + diam(G))$ decryptions
8	$(4,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log^*\lvert T\rvert)$	Four (at most)	$O(\lvert T\rvert/\log\lvert T\rvert + diam(G))$ decryptions
9	$(2\log^*\lvert T\rvert + 2, 1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert)$	$O(\log^*\lvert T\rvert)$	$O\left(\frac{\lvert T\rvert}{(\log^*\lvert T\rvert)^3 \cdot \log\lvert T\rvert} + diam(G)\right)$ decryptions
10	ST + $(2,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert)$	Two (at most)	$O(\log\lvert T\rvert \cdot \log^*\lvert T\rvert + diam(G))$ decryptions
11	ST + $(3,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\log\lvert T\rvert)$	Three (at most)	$O(\log\lvert T\rvert \cdot \log^*\lvert T\rvert + diam(G))$ decryptions
12	ST + $(4,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log^*\lvert T\rvert)$	Four (at most)	$O(\log\lvert T\rvert \cdot \log^*\lvert T\rvert + diam(G))$ decryptions
13	ISPIT + $(2,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot (\log\lvert T\rvert)^2)$	Two (at most)	$O(\log^*\lvert T\rvert + diam(G))$ decryptions
14	ISPIT + $(3,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert \cdot \log\log\lvert T\rvert)$	Three (at most)	$O(\log^*\lvert T\rvert + diam(G))$ decryptions
15	ISPIT + $(4,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert \cdot \log^*\lvert T\rvert)$	Four (at most)	$O(\log^*\lvert T\rvert + diam(G))$ decryptions
16	ISPIT + $(2,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot (\log\lvert T\rvert)^2 \log\log\lvert T\rvert)$	Two (at most)	$O(diam(G))$ decryptions
17	ISPIT + $(3,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert \cdot (\log\log\lvert T\rvert)^2)$	Three (at most)	$O(diam(G))$ decryptions
18	ISPIT + $(4,1)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \log\lvert T\rvert \cdot \log\log\lvert T\rvert \cdot \log^*\lvert T\rvert)$	Four (at most)	$O(diam(G))$ decryptions
19	$(2,2)$-CSBT+ EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert^2 \cdot \log\lvert T\rvert)$	Two (at most)	$O(diam(G))$ decryptions
20	$(3,2)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot \sqrt{\lvert T\rvert} \cdot \log\log\lvert T\rvert)$	Three (at most)	$O(diam(G))$ decryptions
21	$(4,2)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert^2/\log\lvert T\rvert)$	Four (at most)	$O(diam(G))$ decryptions
22	$(2\log^*\lvert T\rvert + 2, 2)$-CSBT + EBC	$O(\lvert E\rvert \cdot \lvert T\rvert + \lvert V\rvert \cdot \lvert T\rvert \cdot (\log^*\lvert T\rvert)^3)$	$O(\log^*\lvert T\rvert)$	$O(diam(G))$ decryptions

Figure 9: Comparison between encryption-based time-bound hierarchical key assignment schemes which are provably secure in the sense of `IND-ST`. The computational assumption for all the schemes is the existence of `IND-P1-C0` symmetric encryption schemes.

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

[3] N. Alon and B. Schieber, *Optimal Preprocessing for Answering On-line Product Queries*, Technical Report TR 71/87, Institute of Computer Science, Tel-Aviv University, 1987.

[4] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken, *Dynamic and Efficient Key Management for Access Hierarchies*, CERIAS Technical Report TR 2006-09, Purdue University. Preliminary version in Proc. of the 12th ACM Conference on Computer and Communications Security - CCS 2005, Alexandria, Virginia, USA, November 2005, 190–201.

	Scheme	Public information	Private information	Key derivation	Computational assumption
1	TLPBC [6]	$O(\|E^*\|)$	$O(\|T\|)$	One decryption	BDDH
2	Naive + BEBC	$O(\|V\| \cdot \|T\|)$	$O(\|T\|)$	One (complex) decryption	$\|V\|$-BDDHE
3	IHBT + BEBC	$O(\|V\| \cdot \|T\|^2)$	One	One (complex) decryption	$\|V\|$-BDDHE
4	$(2,1)$-CSBT + BEBC	$O(\|V\| \cdot \|T\| \cdot \log \|T\|)$	Two (at most)	One (complex) decryption	$\|V\|$-BDDHE
5	$(3,1)$-CSBT + BEBC	$O(\|V\| \cdot \|T\| \cdot \log\log \|T\|)$	Three (at most)	One (complex) decryption	$\|V\|$-BDDHE
6	$(4,1)$-CSBT + BEBC	$O(\|V\| \cdot \|T\| \cdot \log^* \|T\|)$	Four (at most)	One (complex) decryption	$\|V\|$-BDDHE
7	$(2\log^* \|T\| + 2, 1)$-CSBT + BEBC	$O(\|V\| \cdot \|T\|)$	$\log^* \|T\|$ (at most)	One (Complex) decryption	$\|V\|$-BDDHE

Figure 10: Comparison between pairing-based time-bound hierarchical key assignment schemes which are provably secure in the sense of IND-ST. A complex decryption involves at most as many group operations as the number of public information.

[5] M. J. Atallah, M. Blanton, and K. B. Frikken, *Key Management for Non-Tree Access Hierarchies*, in Proc. of the 11th ACM Symposium on Access Control Models and Technologies - SACMAT 2006, Lake Tahoe, California, USA, June 2006, 11–18.

[6] G. Ateniese, A. De Santis, A. L. Ferrara, and B. Masucci, *Provably-Secure Time-Bound Hierarchical Key Assignment Schemes*, in Proc. of the 13th ACM Conference on Computer and Communications Security - CCS 2006, Alexandria, Virginia, USA, November 2006, 288–297. Full version available as Report 2006/225 at the IACR Cryptology ePrint Archive.

[7] H. L. Bodlaender, G. Tel, and N. Santoro, *Trade-offs in Non-reversing Diameter*, Nordic Journal on Computing, 1, 111–134, 1994.

[8] D. Boneh, X. Boyen, and E. Goh, *Hierarchical Identity-based Encryption with Constant Size Ciphertexts*, in Proc. of Advances in Cryptology - Eurocrypt 2005, Aarhus, Denmark, May 2005, Lecture Notes in Computer Science, 3493, 440–456.

[9] D. Boneh, C. Gentry, and B. Waters, *Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys*, in Proc. of Advances in Cryptology - Crypto 2005, Santa Barbara, California, USA, August 2005, Lecture Notes in Computer Science, 3621, 258–275.

[10] G. Brightwell and D. Winkler, *Counting Linear Extensions is #P-complete*, in Proc. of the 23rd Annual ACM Symposium on Theory of Computing - STOC 1991, New Orleans, Louisiana, USA, May 1991, 175–181.

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

[12] H. Y. Chien, *Efficient Time-Bound Hierarchical Key Assignment Scheme*, IEEE Transactions on Knowledge and Data Engineering, 16(10), 1301–1034, 2004.

[13] J. Crampton, K. Martin, and P. Wild, *On Key Assignment for Hierarchical Access Control*, in Proc. of the 19th IEEE Computer Security Foundations Workshop - CSFW 2006, S. Servolo island, Venice, Italy, July 2006, 98–111.

[14] A. De Santis, A. L. Ferrara, and B. Masucci, *Enforcing the Security of a Time-Bound Hierarchical Key Assignment Scheme*, Information Sciences, 176(12), 1684–1694, 2006.

[15] A. De Santis, A. L. Ferrara, and B. Masucci, *Efficient Provably-Secure Hierarchical Key Assignment Schemes*, available as Report 2006/479 at the IACR Cryptology ePrint Archive.

[16] B. Dushnik and E. W. Miller, *Partially Ordered Sets*, American Journal of Mathematics, 63, 600–610, 1941.

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

[18] 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.

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

[20] J. Katz and M. Yung, *Characterization of Security Notions for Probabilistic Private-Key Encryption*, Journal of Cryptology, 19, 67–95, 2006.

[21] J. A. La Poutré, *New Techniques for the Union-Find Problem*, Technical Report RUU-CS-89-19, Department of Computer Science, Utrecht University, The Netherlans, 1989.

[22] 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.

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

[24] S. J. MacKinnon, P. D. Taylor, H. Meijer, and S. G. Akl, *An Optimal Algorithm for Assigning Cryptographic Keys to Control Access in a Hierarchy*, IEEE Transactions on Computers, C-34(9), 797–802, 1985.

[25] R. S. Sandhu, *Cryptographic Implementation of a Tree Hierarchy for Access Control*, Information Processing Letters, 27, 95–98, 1988.

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

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

[28] R. E. Tarjan, *Efficiency of a Good but not Linear Set Union Algorithm*, Journal of the ACM, 22, 215–225, 1975.

[29] M. Thorup, *Shortcutting Planar Digraphs*, DIMACS Technical Report 93-60, August 1993.

[30] 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.

[31] 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 - ASIACCS 2006, Taipei, Taiwan, March 2006, 223–230.

[32] M. Yannakakis, *On the Complexity of Partial Order Dimension Problem*, SIAM J. Alg. Discr. Methods, 3, 351–358, 1982.

[33] A. C. Yao, *Space-Time Tradeoff for Answering Range Queries*, in Proc. of the 14th annual ACM Symposium on the Theory of Computing - STOC 1982, San Francisco, California, USA, May 1982, 128–136.

[34] J. Yeh, *An RSA-Based Time-Bound Hierarchical Key Assignment Scheme for Electronic Article Subscription*, in Proc. of the ACM International Conference on Information and Knowledge Management - CIKM 2005, Bremen, Germany, November 2005, 285–286.

[35] 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.

[36] 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.

[37] 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), 2006.

[38] 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.