Efficient Provably-Secure 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 *hierarchical key assignment scheme* is a method to assign some private information and encryption keys to a set of classes in a partially ordered hierarchy, in such a way that the private information of a higher class can be used to derive the keys of all classes lower down in the hierarchy.

In this paper we design and analyze hierarchical key assignment schemes which are provably-secure and support dynamic updates to the hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

- We first consider the problem of constructing a hierarchical key assignment scheme by using as a building block a symmetric encryption scheme. We propose a new construction which is provably secure with respect to key indistinguishability, requires a single computational assumption, and improves on previous proposals.
- Then, we show how to reduce key derivation time at the expense of an increment of the amount of public information, by improving a previous result.
- Finally, we show how to construct a hierarchical key assignment scheme by using as a building block a public-key broadcast encryption scheme. In particular, one of our constructions provides constant private information and public information linear in the number of classes in the hierarchy.

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

1 Introduction

The *hierarchical access control problem* is defined in a scenario where the users of a computer system are organized in a hierarchy formed by a certain number of disjoint classes, called *security classes*. A hierarchy arises from the fact that some users have more access rights than others. For example, there are several situations where supervisors have all the privileges to control the tasks of their subordinates, while the subordinates have no privileges at all to access the supervisors' tasks. Similar situations 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 system. 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 Trusted Authority (TA), which is active only at the distribution phase. Akl and Taylor [2] first proposed an elegant hierarchical key assignment scheme in which each class is assigned a key that can be used, along with some public information generated by the TA, to compute the key assigned to any class lower down in the hierarchy. Subsequently, many researchers have proposed different schemes that either have better performances or allow insertions and deletions of classes in the hierarchy (e.g., [4, 21, 23, 27, 28, 29, 31]). A detailed classification of many schemes in the literature has been recently provided by Crampton et al. [13], according to several parameters, such as the memory requirements for public and private information, the complexity of key derivation, the complexity of handling dynamic updates to the hierarchy, and the resistance to collusive attacks. Atallah et al. [4] first addressed the problem of formalizing security requirements for hierarchical key assignment schemes. They proposed a first construction based on pseudorandom functions and a second one requiring the use of a symmetric encryption scheme secure against chosen-ciphertext attacks. Their constructions also manage with dynamic changes to the hierarchy, such as insertion and deletion of classes.

In this paper we design and analyze hierarchical key assignment schemes which are provably-secure and efficient. We consider security with respect to *key indistinguishability*, which corresponds to the requirement that an adversary is not able *to learn any information* about a key that it should not have access to, i.e., it is not able to distinguish it from a random string having the same length. We propose two constructions for hierarchical key assignment schemes. Both constructions support updates to the access hierarchy with local changes to the public information and without requiring any private information to be re-distributed. The first construction, which is based on symmetric encryption schemes, is simpler than the one proposed by Atallah et al. [4], requires a single computational assumption, and offers more efficient procedures for key derivation and key updates. We also focus on improving efficiency of key derivation in hierarchical key assignment schemes. Such a problem has been recently considered by Atallah et al. [4, 5], who proposed two different techniques requiring an increment of public information. We show how to further reduce key derivation time by improving one of their techniques. Finally, we show how to construct a hierarchical key assignment scheme by using only a public-key broadcast encryption scheme. In particular, by plugging in the scheme proposed by Boneh et al. [9] we obtain a hierarchical key assignment scheme offering constant private information and public information linear in the number of classes.

The paper is organized as follows: in Section 2 we review the definition of hierarchical key assignment schemes. In Section 3 we show how to construct a hierarchical key assignment scheme using as a building block a symmetric encryption scheme, whereas, in Section 4 we consider the problem of reducing the number of steps required to perform key derivation in hierarchical key assignment schemes. In Section 5 we show a construction based on public-key broadcast encryption schemes. In Section 6 we conclude the paper by showing a comparison between our constructions and previous work.

2 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. 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 [31, 5], etc.
A hierarchical key assignment scheme for a family Γ of graphs, corresponding to partially ordered hierarchies, is defined as follows.

Definition 2.1 *A* hierarchical key assignment scheme *for Γ is a pair (Gen, Der) of algorithms satisfying the following conditions:*

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

   (a) a private information s_u, for any class $u \in V$;

   (b) a key k_u, for any class $u \in V$;

   (c) a public information pub.

   We denote by (s, k, pub) the output of the algorithm Gen on inputs 1^τ and G, 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, two classes $u \in V$ and $v \in A_u$, the private information s_u assigned to class u and the public information pub, and produces as output the key k_v assigned to class v.*

   We require that for each class $u \in V$, each class $v \in A_u$, each private information s_u, each key k_v, each public information pub which can be computed by Gen on inputs 1^τ and G, it holds that

$$Der(1^\tau, G, u, v, s_u, pub) = k_v.$$

A 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, the resistance to collusive attacks. As regards as the complexity of key derivation, we are interested both in the number and in the 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 u performs key derivation to compute the key of a class $v \in A_u$, it does not need to input the algorithm Der with the whole *pub* and G, but only with those parts of them involved in the computation. 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 classes or edges in the hierarchy, without requiring the TA to re-distribute any private information.

However, the most fundamental feature that a good scheme should have is the resistance to collusive attacks. More precisely, for each class $u \in V$, the key k_u should be protected against a coalition of all users in the set $F_u = \{v \in V : u \notin A_v\}$, corresponding to all users which are not allowed to compute the key k_u. We consider security with respect to *key indistinguishability*. Such a requirement, first introduced by Atallah et al. [4], 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* $\texttt{STAT}_u$ which wants to attack a class $u \in V$ and which is able to corrupt *all* users in F_u. We define an algorithm $Corrupt_u$ which, on input the private information s generated by the algorithm Gen, extracts the secret values s_v associated to all classes $v \in F_u$. We denote by $corr$ the sequence output by $Corrupt_u(s)$. Two experiments are considered. In the first one, the adversary is given the key k_u, whereas, in the second one, it is given a random string ρ having the same length as k_u. It is the adversary's job to determine whether the received challenge corresponds to k_u 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 $\epsilon : N \to R$ is *negligible* if for every constant $c > 0$ there exists an integer n_c such that $\epsilon(n) < n^{-c}$ for all $n \geq n_c$.

Definition 2.2 `[IND-ST]` *Let Γ be a family of graphs corresponding to partially ordered hierarchies, let $G = (V, E)$ be a graph in Γ, let (Gen, Der) be a hierarchical key assignment scheme for Γ and let $\texttt{STAT}_u$ be a static adversary which attacks a class u. Consider the following two experiments:*

Experiment $\mathbf{Exp}^{\texttt{IND-1}}_{\texttt{STAT}_u}(1^\tau, G)$	Experiment $\mathbf{Exp}^{\texttt{IND-0}}_{\texttt{STAT}_u}(1^\tau, G)$
$(s, k, pub) \leftarrow Gen(1^\tau, G)$	$(s, k, pub) \leftarrow Gen(1^\tau, G)$
$corr \leftarrow Corrupt_u(s)$	$corr \leftarrow Corrupt_u(s)$
$d \leftarrow \texttt{STAT}_u(1^\tau, G, pub, corr, k_u)$	$\rho \leftarrow \{0,1\}^{length(k_u)}$
return d	$d \leftarrow \texttt{STAT}_v(1^\tau, G, pub, corr, \rho)$
	return d

The advantage of $\texttt{STAT}_u$ *is defined as*

$$\mathbf{Adv}^{\texttt{IND}}_{\texttt{STAT}_u}(1^\tau, G) = |Pr[\mathbf{Exp}^{\texttt{IND-1}}_{\texttt{STAT}_u}(1^\tau, G) = 1] - Pr[\mathbf{Exp}^{\texttt{IND-0}}_{\texttt{STAT}_u}(1^\tau, G) = 1]|.$$

The scheme is said to be secure in the sense of `IND-ST` *if, for each graph $G = (V, E)$ in Γ and each $u \in V$, the function $\mathbf{Adv}^{\texttt{IND}}_{\texttt{STAT}_u}(1^\tau, G)$ is negligible, for each static adversary $\texttt{STAT}_u$ 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. 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.

3 A Construction based on Symmetric Encryption Schemes

In this section we consider the problem of constructing a hierarchical key assignment scheme by using as a building block a symmetric encryption scheme. A simple way to realize an encryption based scheme would be to assign a key k_u to each class $u \in V$ and a public information $p_{(u,v)}$, for each edge $(u, v) \in E$, corresponding to the encryption of k_v with the key k_u. This would allow any user in a class u to compute the key k_v held by any class v lower down in the hierarchy, by performing $dist_G(u, v)$ decryptions, where $dist_G(u, v)$ denotes the length of the shortest path between u and v in G. Such a scheme belongs to the family of *iterative key encrypting key assignment schemes (IKEKAS)*, defined by Crampton et al. [13], where each user is required to store a single secret value, corresponding to its key, and $|E|$ values are made public. As regards as the security of the schemes, it depends on the security properties of the underlying encryption scheme. It is not difficult to show that, by using an encryption scheme which is secure with respect to a non-adaptive chosen plaintext attack, we obtain a key assignment scheme which guarantees security against *key recovery*. Such a security requirement corresponds to the fact that an adversary is not able to *compute* a key that it should not have access to (see [6] for a formal definition).

Unfortunately, the simple solution described above is not secure with respect to *key indistinguishability*. Indeed, consider an adversary attacking a class u and corrupting a class v such that $(u, v) \in E$. The adversary, on input a challenge ρ, corresponding either to the key k_u or to a random value, is able to tell if ρ corresponds to the encryption key k_u simply by checking whether the decryption of the public value $p_{(u,v)}$ with key ρ corresponds to the key k_v held by class v. In the following we show how to construct a hierarchical key assignment scheme which is provably-secure with respect to key indistinguishability and allows dynamic updates to the hierarchy with local changes to the public information only.

The idea behind our construction, referred in the following as the *Encryption Based Construction (EBC)*, is to avoid the attack described above by never using the key assigned to a class to encrypt the keys assigned to other classes. In particular, in the EBC each class $u \in V$ is assigned a private information s_u, an encryption key k_u, and a public information $\pi_{(u,u)}$, 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 $p_{(u,v)}$, which allows class u to compute the private information s_v held by class v. Indeed, $p_{(u,v)}$, consists of the encryption of the private information s_v with the private information s_u. This allows any user in a class u to compute the key k_v held by any class v lower down in the hierarchy, by performing $dist_G(u, v) + 1$ decryptions. We will show that an adversary attacking a class u is not able to distinguish the key k_u from a random string of the same length unless it is able to break the underlying encryption scheme. Before describing our construction, we first recall the definition of a symmetric encryption scheme.

Definition 3.1 *A* symmetric encryption scheme *is a triple* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *of algorithms satisfying the following conditions:*

1. *The* key-generation algorithm $\mathcal{K}$ *is probabilistic polynomial-time. It takes as input the security parameter* 1^τ *and produces as output a string key.*

2. *The* encryption algorithm $\mathcal{E}$ *is probabilistic polynomial-time. It takes as inputs* 1^τ*, a string key produced by* $\mathcal{K}(1^\tau)$*, and a message* $m \in \{0, 1\}^*$*, and produces as output the ciphertext* y*.*

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

Let Γ be a family of graphs corresponding to partially ordered hierarchies. Let $G = (V, E) \in \Gamma$ and let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme.

Algorithm $Gen(1^\tau, G)$

1. For any class $u \in V$, let $s_u \leftarrow \mathcal{K}(1^\tau)$ and $k_u \leftarrow \{0,1\}^\tau$;
2. Let s and k be the sequences of private information and keys, respectively, computed in the previous step;
3. For any two classes $u, v \in V$ such that $(u, v) \in E$, compute the public information $p_{(u,v)} = \mathcal{E}_{s_u}(s_v)$;
4. For any class u in V, compute the public information $\pi_{(u,u)} = \mathcal{E}_{s_u}(k_u)$;
5. Let pub be the sequence of public information computed in the previous two steps;
6. Output (s, k, pub).

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Consider a path $(w_0, w_1), \ldots, (w_{m-1}, w_m) \in E$, from $u = w_0$ to $v = w_m$.
   For any $i = 1, \ldots, m$, extract the public value $p_{(w_{i-1}, w_i)}$ from pub and compute the private information $s_{w_i} = \mathcal{D}_{s_{w_{i-1}}}(p_{(w_{i-1}, w_i)})$;
2. Extract the public value $\pi_{(v,v)}$ from pub and output the key $k_v = \mathcal{D}_{s_v}(\pi_{(v,v)})$.

Figure 1: The Encryption Based Construction (EBC).

The Encryption Based Construction is described in Figure 1.

The EBC associates a public value $p_{(u,v)}$ to each edge $(u, v) \in E$, as well as a public value $\pi_{(u,u)}$ to each class $u \in V$. In order to simplify the analysis of the scheme, in the following we consider the public value $\pi_{(u,u)}$ as it were associated to an additional edge connecting the class u to a dummy class u'. This will enable us to consider all public information as values associated to the edges of a hierarchy. Figure 2 illustrates a partially ordered hierarchy along with the public information associated by the EBC to the edges of the hierarchy, as well as those associated to additional edges, which are represented by dashed lines.

3.1 Analysis of the Scheme

In this section we first show that the security property of the EBC depends on the security property of the underlying encryption scheme. Afterwards, we evaluate the performances of the EBC with respect to several parameters, such as space requirements for public and private information storage and computational requirements for key derivation and key updates.

Before analyzing the security of the EBC we first need to define what we mean by a *secure* symmetric encryption scheme. We formalize security with respect to *plaintext indistinguishability*, which is an adaption of the notion of *polynomial security* as given in [20]. We consider an adversary $A = (A_1, A_2)$ running in two stages. In advance of the adversary's execution, a random key key is chosen and kept hidden from the adversary. During the first stage, the adversary A_1 outputs a triple $(x_0, x_1, state)$, where x_0 and x_1 are two messages of the same length, and $state$ is some state information which could be useful later. One message between x_0 and x_1 is chosen at random and encrypted to give the challenge ciphertext y. In the second stage, the adversary A_2 is given y and

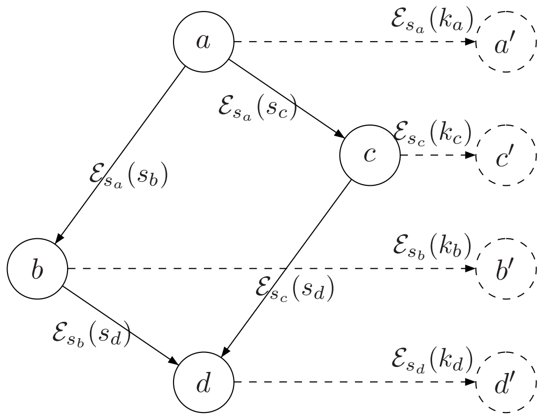

Figure 2: A partially ordered hierarchy along with the public information associated by the EBC.

state and has to determine whether y is the encryption of x_0 or x_1. Informally, the encryption scheme is said to be secure with respect to a non-adaptive chosen plaintext attack, denoted by `IND-P1-C0` in [25], if every polynomial-time adversary A, which has access to the encryption oracle only during the first stage of the attack and has never access to the decryption oracle, succeeds in determining whether y is the encryption of x_0 or x_1 with probability only negligibly different from 1/2.

Definition 3.2 `[IND-P1-C0]` *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme and let τ be a security parameter. Let $A = (A_1, A_2)$ be an adversary that has access to the encryption oracle only during the first stage of the attack and has never access to the decryption oracle. Consider the following two experiments:*

$$\begin{array}{l|l}
\text{Experiment } \mathbf{Exp}^{\texttt{IND-P1-C0-1}}_{\Pi,A}(1^\tau) & \text{Experiment } \mathbf{Exp}^{\texttt{IND-P1-C0-0}}_{\Pi,A}(1^\tau) \\
\quad key \leftarrow \mathcal{K}(1^\tau) & \quad key \leftarrow \mathcal{K}(1^\tau) \\
\quad (x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{key}(\cdot)}(1^\tau) & \quad (x_0, x_1, state) \leftarrow A_1^{\mathcal{E}_{key}(\cdot)}(1^\tau) \\
\quad y \leftarrow \mathcal{E}_{key}(x_1) & \quad y \leftarrow \mathcal{E}_{key}(x_0) \\
\quad d \leftarrow A_2(1^\tau, y, state) & \quad d \leftarrow A_2(1^\tau, y, state) \\
\quad \textbf{return } d & \quad \textbf{return } d
\end{array}$$

The advantage of A is defined as

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

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

Theorem 3.3 *If the encryption scheme $\Pi = (\mathcal{K}, \mathcal{D}, \mathcal{E})$ is secure in the sense of* `IND-P1-C0`*, then the EBC is secure in the sense of* `IND-ST`*.*

Proof. Let $G = (V, E)$ be a graph in Γ, let $u \in V$ and let $G_u = (I_u, E_u)$ be the subgraph of G induced by the set of vertices $I_u = \{v \in V : \text{ there is a path from } v \text{ to } u \text{ in } G\}$. W.l.o.g., let $(u_1, \ldots, u_m)$, where $u_m \equiv u$, be any topological ordering of the vertices in I_u and let $(e_1, \ldots, e_{h-1})$ be the sequence of edges in E_u such that $e_i = (u_\alpha, u_\beta)$ precedes $e_j = (u_\gamma, u_\delta)$ if and only if either $\alpha < \gamma$ or $\alpha = \gamma$ and $\beta < \delta$. Moreover, let $e_h = (u, u')$ (see Figure 3).

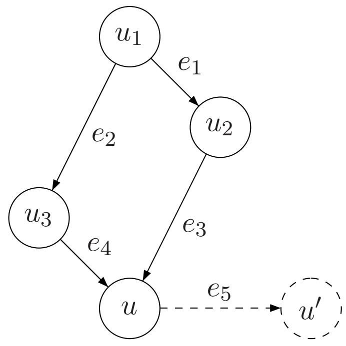

Figure 3: A topological sorting of the classes in I_u and corresponding sequence of edges.

Let $\mathtt{STAT}_u$ be a static adversary attacking class u. In order to prove the theorem, we need to show that the adversary's views in experiments $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}$ are indistinguishable. In particular, we prove that such views are both indistinguishable from the adversary's view in the experiment $\mathbf{Exp}^{*}_{u}$, defined as follows:

Experiment $\mathbf{Exp}^{*}_{u}(1^\tau, G)$
$(s, k, pub^*) \leftarrow Gen^*(1^\tau, G)$
$corr \leftarrow Corrupt_u(s)$
$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub^*, corr, k_u)$
return d

The algorithm Gen^* differs from Gen for the way part of the public information pub^* is computed. Indeed, the public value $\pi_{(u,u)}$ associated to class u is computed as the encryption $\mathcal{E}_{s_u}(\rho)$, where ρ is randomly chosen in $\{0,1\}^\tau$, whereas, the public value associated to each edge $e_i = (u_\alpha, u_\beta) \in E_u$ is computed as the encryption $\mathcal{E}_{s_{u_\alpha}}(r_i)$, where $r_1, \ldots, r_h$ are randomly and independently chosen values in $\{0,1\}^\tau$.

We will first show that the adversary's views in experiments $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{*}_{u}$ are indistinguishable. We construct a sequence of $h+1$ experiments $\mathbf{Exp1}^{1}_{u}, \ldots, \mathbf{Exp1}^{h+1}_{u}$, all defined over the same probability space, where the first and the last experiments of the sequence correspond to $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}$ and $\mathbf{Exp}^{*}_{u}$. In each experiment we modify how the view input to $\mathtt{STAT}_u$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $q = 2, \ldots, h$, experiment $\mathbf{Exp1}^{q}_{u}$ is defined as follows:

Experiment $\mathbf{Exp1}^{q}_{u}(1^\tau, G)$
$(s, k, pub^q) \leftarrow Gen^q(1^\tau, G)$
$corr \leftarrow Corrupt_u(s)$
$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub^q, corr, k_u)$
return d

The algorithm Gen^q used in $\mathbf{Exp1}^{q}_{u}$ differs from Gen for the way part of the public information pub^q is computed. Indeed, for any $i = 1, \ldots, q-1$, the public values associated to the edge $e_i = (u_\alpha, u_\beta)$ is computed as the encryption $\mathcal{E}_{s_{u_\alpha}}(r_i)$, where $r_1, \ldots, r_{q-1}$ are randomly and independently chosen values in $\{0,1\}^\tau$. Figure 4 illustrates two consecutive experiments on the hierarchy of Figure 3.

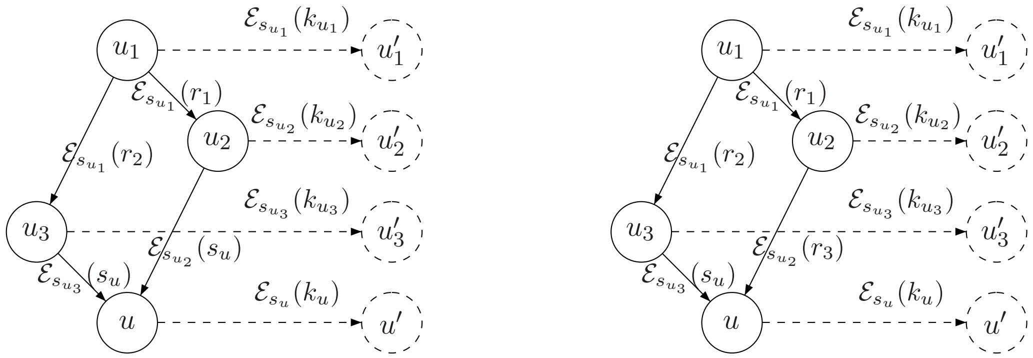

Figure 4: The left hand side illustrates $\mathbf{Exp1}_u^3$ while the right hand side illustrates $\mathbf{Exp1}_u^4$.

In the following we show that, for any $q = 1, \ldots, h$, the adversary's view in the q-th experiment is indistinguishable from the adversary's view in the $(q+1)$-th one.

Assume by contradiction that there exists a polynomial-time distinguisher B_q which is able to distinguish between the adversary $\texttt{STAT}_u$'s views in experiments $\mathbf{Exp1}_u^q$ and $\mathbf{Exp1}_u^{q+1}$ with non-negligible advantage. Notice that such views differ only for the way the public information associated to the edge $e_q = (a, b)$ is computed. We show how to construct a polynomial-time adversary $A = (A_1, A_2)$ which uses B_q to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of IND-P1-C0. In particular, the algorithm A_1, on input 1^τ, randomly chooses two messages $x_0, x_1 \in \{0,1\}^\tau$ and associates x_1 either to the key k_u, if $(a, b) = (u, u')$, or to the private information s_b, otherwise. All other keys and private information are chosen at random. Moreover, A_1 constructs all public values associated to the outgoing edges of class a, with the exception of the edge (a, b), making queries to the encryption oracle $\mathcal{E}_{s_a}(\cdot)$. The sequences s, k, and pub' of all private information, keys, and public values constructed by A_1, along with the values x_0 and x_1 are saved in the state information *state*. Formally, the algorithm A_1 is defined as follows:

Algorithm $A_1^{\mathcal{E}_{s_a}(\cdot)}(1^\tau)$
  $x_0, x_1, k_a \leftarrow \{0,1\}^\tau$
  for each class $v \in V \setminus \{a\}$
    $s_v \leftarrow \mathcal{K}(1^\tau)$, $k_v \leftarrow \{0,1\}^\tau$
  if $(a, b) = (u, u')$ **then** $k_u \leftarrow x_1$
    else $s_b \leftarrow x_1$
  for each edge $(a, v) \neq (a, b)$
    $p_{(a,v)} \leftarrow \mathcal{E}_{s_a}(s_v)$
  $pub' \leftarrow$ public values constructed above
  $state \leftarrow (s, k, pub', x_0, x_1)$
  return $(x_0, x_1, state)$

Let y be the challenge for the algorithm A, corresponding to the encryption of either x_0 or x_1 with the unknown key s_a. The algorithm A_2, on input $1^\tau, y$, and *state*, constructs the view for the distinguisher B_q as follows: it first extracts from s the private information *corr* held by corrupted users, by using the algorithm $Corrupt_u(s)$. Then, it computes the public values not included in pub', in order to obtain the sequence pub. In particular, the public value associated to the edge $e_q = (a, b)$ is set equal to the challenge y. Finally, A_2 outputs the same output as $B_q(1^\tau, G, pub, corr, x_1)$. More formally,

Algorithm $A_2(1^\tau, y, state)$

let $state = (s, k, pub', x_0, x_1)$

$corr \leftarrow Corrupt_u(s)$

//construction of missing public values

if $(a,b) = (u,u')$ **then** $\pi_{(u,u)} \leftarrow y$

else $p_{(a,b)} \leftarrow y$

for each edge $(v,z) \notin \{e_1, \ldots, e_q\}$

$p_{(v,z)} \leftarrow \mathcal{E}_{s_v}(s_z)$

for $i = 1, \ldots, q-1$

$r_i \leftarrow \{0,1\}^\tau$

let $e_i = (v,z)$

$p_{(v,z)} \leftarrow \mathcal{E}_{s_v}(r_i)$

for each edge $(v,v') \neq (a,b)$

$\pi_{(v,v)} \leftarrow \mathcal{E}_{s_v}(k_v)$

$d \leftarrow B_q(1^\tau, G, pub, corr, x_1)$

return d

Notice that if y corresponds to the encryption of x_1, then the random variable associated to the adversary's view is exactly the same as the one associated to the adversary view in experiment $\mathbf{Exp1}_u^q$, whereas, if y corresponds to the encryption of x_0, it has the same distribution as the one associated to the adversary's view in experiment $\mathbf{Exp1}_u^{q+1}$. Therefore, if the algorithm B_q is able to distinguish between such views with non negligible advantage, it follows that algorithm A is able to break the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of IND-P1-C0. Contradiction.

By using similar arguments we can also show that the adversary's views in experiments $\mathbf{Exp}_{\mathtt{STAT}_u}^{\mathtt{IND-0}}$ and $\mathbf{Exp}_u^*$ are indistinguishable. In particular, we construct a sequence of h experiments $\mathbf{Exp0}_u^1, \ldots, \mathbf{Exp0}_u^h$, all defined over the same probability space, where the first and the last experiments of the sequence correspond to $\mathbf{Exp}_{\mathtt{STAT}_u}^{\mathtt{IND-0}}$ and $\mathbf{Exp}_u^*$. In each experiment we modify the way the view of $\mathtt{STAT}_u$ is computed, while maintaining the view's distributions indistinguishable among any two consecutive experiments. For any $q = 2, \ldots, h-1$, experiment $\mathbf{Exp0}_u^q$ is defined as follows:

Experiment $\mathbf{Exp0}_u^q(1^\tau, G)$

$(s, k, pub^q) \leftarrow Gen^q(1^\tau, G)$

$corr \leftarrow Corrupt_u(s)$

$\rho \leftarrow \{0,1\}^\tau$

$d \leftarrow \mathtt{STAT}_u(1^\tau, G, pub^q, corr, \rho)$

return d

Notice that, for any $q = 2, \ldots, h$, in experiment $\mathbf{Exp1}_u^q$, the last input of $\mathtt{STAT}_u$ coincides with the value k_u used by Gen^q to compute the public information $\pi(u,u) = \mathcal{E}_{s_u}(k_u)$, whereas, in experiment $\mathbf{Exp0}_u^q$, the last input of $\mathtt{STAT}_u$ corresponds to value ρ randomly chosen in $\{0,1\}^\tau$, which is different from the decryption of $\pi(u,u)$ with the private information s_u.

The proof that the existence of a polynomial-time distinguisher which is able to distinguish between the adversary's views in two consecutive experiments implies the existence of a polynomial-time adversary which breaks the security of the encryption scheme $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ in the sense of IND-P1-C0 is the same as the one shown before. This concludes the proof.

□

The EBC 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, a class $u \in V$ which wants to compute the key held by a class $v \in A_u$ is required to perform $dist_G(u, v) + 1$ decryption operations.

Notice that the number of decryption operations needed for key derivation could be reduced at the expense of an increment of the amount of public information. The idea behind the new construction, referred in the following as the *modified EBC*, is to add the value $\pi_{(z,v)} = \mathcal{E}_{s_z}(k_v)$ to the public information *pub*, for any edge $(z, v) \in E$. This allows any user in a class $u \in V$ to compute the key held by a class $v \in A_u$ by performing $dist_G(u, v)$ decryption operations. Indeed, consider the shortest path from u to v and let z be the direct predecessor of v on such a path; it follows that $dist_G(u, z) = dist_G(u, v) - 1$ decryptions are needed to compute the private information held by z, whereas, one decryption is needed to obtain k_v from the public value $\pi_{(z,v)}$. However, the modified EBC requires $2|E| + |V|$ public values. Regarding as the security of the scheme, it is easy to see that the same technique used in Theorem 3.3 can be used to prove that the modified EBC is secure in the sense of `IND-ST`.

Finally, we notice that the technique we have used to turn the simple encryption based scheme described at the beginning of Section 3 into the EBC could be used as well to turn some other schemes offering security against key recovery in schemes which guarantee security in the sense of `IND-ST`. For example, one could use such a technique starting from the pseudorandom based construction proposed by Atallah et al. [4].

3.2 Handling Dynamic Changes

In the following we show how to manage changes to the hierarchy, such as additions and deletions of classes and edges, in the EBC. Unfortunately, the EBC supports insertions, but not deletions, of classes and edges in the hierarchy without re-distributing private information to the classes affected by such changes. Indeed, whenever an edge (u, v) is removed from E, the private information held by any class lower down u needs to be changed, since users in class u already knows it. However, it is possible to extend the EBC in order to allow additions and deletions of classes and edges, with local changes to the public information only.

The idea behind the extended construction, referred in the following as the *Dynamic Encryption Based Construction (DEBC)*, is to avoid the re-distribution of private information, whenever deletions of classes or edges occur, by never allowing a class to compute the private information of another class lower down in the hierarchy. This can be done by assigning each class u an additional value η_u, which plays the role of an *intermediate key*. Such a value is encrypted with the private information s_u and the resulting encryption, denoted by $\omega_{(u,u)}$ is made public and allows any user in class u to compute its key k_u, as well as the intermediate key η_v associated to any class v lower down in the hierarchy. The scheme requires $|V|$ additional public values and allows any user in a class u to compute the key k_v held by any class $v \in A_u$, by performing $dist_G(u, v) + 2$ decryptions. The Dynamic Encryption Based Construction is described in Figure 5.

The additional public value $\omega_{(u,u)}$ associated by the DEBC to each class $u \in V$ could be considered as it were associated to an additional edge connecting a dummy node u'' to the class u. Figure 6 illustrates a partially ordered hierarchy along with the public information associated to the edges of the hierarchy, as well as those associated to additional edges, which are represented by dashed lines.

Regarding as the security of the scheme, it is easy to see that the same technique used in Theorem 3.3 can be used to prove that the DEBC is secure in the sense of `IND-ST`.

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

Let Γ be a family of graphs corresponding to partially ordered hierarchies. Let $G = (V, E) \in \Gamma$ and let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be a symmetric encryption scheme.

Algorithm $Gen(1^\tau, G)$

1. For any class $u \in V$, let $s_u \leftarrow \mathcal{K}(1^\tau)$ and $k_u \leftarrow \{0,1\}^\tau$;
2. Let s and k be the sequences of private information and keys, respectively, computed in the previous step;
3. For any class $u \in V$, let $\eta_u \leftarrow \mathcal{K}(1^\tau)$;
4. For any two classes $u, v \in V$ such that $(u, v) \in E$, compute the public information $p_{(u,v)} = \mathcal{E}_{\eta_u}(\eta_v)$;
5. For any class u in V, compute the public information $\omega_{(u,u)} = \mathcal{E}_{s_u}(\eta_u)$ and $\pi_{(u,u)} = \mathcal{E}_{\eta_u}(k_u)$;
6. Let pub be the sequence of public information computed in the previous two steps;
7. Output (s, k, pub).

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Extract the public information $\omega_{(u,u)}$ from pub and compute the intermediate key $\eta_u = \mathcal{D}_{s_u}(\omega_{(u,u)})$.
2. Let $(w_0, w_1), \ldots, (w_{m-1}, w_m)$, where $w_0 = u$ and $w_m = u$, be a path from u to v.
   For any $i = 1, \ldots, m$, extract the public value $p_{(w_{i-1}, w_i)}$ from pub and compute the intermediate key $\eta_{w_i} = \mathcal{D}_{\eta_{w_{i-1}}}(p_{(w_{i-1}, w_i)})$;
3. Extract the public value $\pi_{(v,v)}$ from pub and output the key $k_v = \mathcal{D}_{\eta_v}(\pi_{(v,v)})$.

Figure 5: The Dynamic Encryption Based Construction.

Insertion of an edge. Let (u, v) be an edge to be inserted in E. Such an update can be managed by the TA by adding the public value $p_{(u,v)} = \mathcal{E}_{\eta_u}(\eta_v)$ to the public information pub.

Deletion of an edge. Let (u, v) be an edge to be deleted from E. In order to forbid users belonging to class u from computing any key which can be computed by class v, for any $z \in A_v$, the TA has to choose a new key $k_z \in \{0,1\}^\tau$ and a new intermediate key $\eta_z \leftarrow \mathcal{K}(1^\tau)$, whereas, there is no need to change the private information s_z. On the other hand, in order to allow authorized users to compute such new values, the TA has to update the public information pub, as follows: for any $z \in A_v$, it first recomputes the public values $\omega_{(z,z)} = \mathcal{E}_{s_z}(\eta_z)$ and $\pi_{(z,z)} = \mathcal{E}_{\eta_z}(k_z)$; then, for any edge $(w, z) \in E$, it recomputes the public value $p_{(w,z)} = \mathcal{E}_{\eta_w}(\eta_z)$.

Insertion of a class. Let u be a class to be inserted in V along with new incoming and outgoing edges. First, the TA computes a private information s_u, a key k_u and an intermediate key η_u for class u. Afterwards, it computes $\omega_{(u,u)} = \mathcal{E}_{s_u}(\eta_u)$ and $\pi_{(u,u)} = \mathcal{E}_{\eta_u}(k_u)$ and adds them to the public information pub. Finally, the TA adds the edges by using the above procedure for edge insertions.

Deletion of a class. Let u be a class to be deleted from V. For each edge outgoing from v, the TA uses the above procedure for edge deletions. Afterwards, the TA deletes from pub the public information associated with all incoming edges of u.

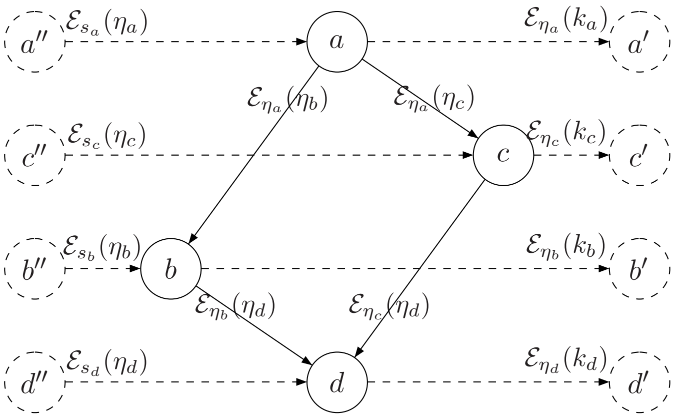

Figure 6: A partially ordered hierarchy along with the public information associated by the DEBC.

The DEBC requires $|E| + 2|V|$ public values; on the other hand, each class has to store a single secret value, corresponding to its private information. As for key derivation, the number of decryption operations required by a class $u \in V$ to compute the key held by a class $v \in A_u$ is $dist_G(u, v) + 2$.

Notice that the number of decryption operations needed for key derivation could be reduced at the expense of an increment of the amount of public information. The idea behind the new construction, referred in the following as the *modified DEBC*, is to add the values $\omega_{(u,w)} = \mathcal{E}_{s_u}(\eta_w)$ and $\pi_{(z,v)} = \mathcal{E}_{\eta_z}(k_v)$ to *pub*, for any $(u, w) \in E$ and $(z, v) \in E$. This allows any user in a class $u \in V$ to compute the key held by a class $v \in A_u$ by performing $dist_G(u, v)$ decryption operations. However, the modified DEBC requires $3|E| + 2|V|$ public values. Regarding as the security of the scheme, it is easy to see that the same technique used in Theorem 3.3 can be used to prove that the modified DEBC is secure in the sense of `IND-ST`.

In the following we compare the DEBC with the scheme proposed by Atallah et al. [4] with respect to the number of operations needed to perform dynamic updates to the hierarchy. In their scheme, the insertion of an edge requires the TA to perform one pseudorandom evaluation and two encryption operations, whereas only one encryption operation is required in the DEBC. Regarding as the deletion of an edge $(u, v) \in E$, the scheme in [4] requires the TA to choose a new label and to perform $2 \cdot |A_v|$ pseudorandom function evaluations, as well $|X_z|$ pseudorandom function evaluations and $2 \cdot |X_z|$ encryption operations, where $X_z = \{w \in V : (w, z) \in E\}$. On the other hand, in the DEBC the TA has to choose two new values and to perform $2 \cdot |A_v|$ encryption operations, as well as $|X_z|$ encryption operations. For the insertion of a class u, along with its incoming and outgoing edges, the scheme in [4] requires the TA to generate two random values and to perform two pseudorandom function evaluations, whereas, in the DEBC the TA has to choose three new values and to perform two encryption operations. Moreover, both schemes also use the procedure for edge insertions, for each incoming and outgoing edge. Finally, in both schemes the deletion of a class reduces to the execution of the procedure for the deletion of each outgoing edge from the deleted class.

To summarize, the DEBC improves on the scheme proposed by Atallah et al. [4] because its security is based on a single computational assumption and offers more efficient procedures for key derivation and key updates, since such procedures involve essentially half the number of operations required by the scheme in [4].

Finally, we notice that the technique we have used to turn the EBC into the DEBC could be used as well to turn some other schemes, which do not allow dynamic updates to the hierarchy without re-distributing any private information, in schemes which allow such updates with local changes to the public information only. For example, one could use such a technique staring from the pseudorandom based construction proposed by Atallah et al. [4].

4 Improving Key Derivation Time

In both schemes described in Section 3, as well as in those proposed by Atallah et al. [4], the number of steps that a class u has to perform, in order to compute the key of another class v lower down in the hierarchy, is proportional to the length of the shortest path from u to v. Atallah et al. [4, 5] analyzed the problem of reducing key derivation time by modifying the graph representing the hierarchy, in order to reduce its diameter. To this aim in [4] they proposed some constructions to add additional edges, called *shortcut edges*, to the hierarchy. Such a technique, referred to as the *shortcutting technique*, has already been used in frameworks quite different from access control.

In Section 4.1 we translate to the access control framework existing results concerning the shortcutting technique and outline the limits of such a technique. An immediate consequence is that for a totally ordered hierarchy of n classes, the number of steps needed to perform key derivation cannot be reduced to three with the addition of $O(n)$ shortcut edges, as claimed in [5]. Indeed, the number of edges to be added in order to reach diameter three is $\Theta(n \cdot \log\log n)$, as we will show in Section 4.1.

In Section 4.2 we review a different technique proposed in [5] to reduce the diameter of a hierarchy. Such a technique, referred in the following as the *shortcutting and point-inserting technique*, makes use of the concept of *dimension* of a poset and consists of the addition of dummy vertices, as well as new edges, to the hierarchy. The idea is to obtain a new hierarchy such that there exists a path between two classes u and v in the old hierarchy if and only if there exists a path between u and v in the new one. The addition of dummy vertices results in a smaller number of new edges to be added to the hierarchy. However, dummy vertices are used only for performance reasons and there are no real classes corresponding to them. The technique is recursive and uses the one-dimensional case, corresponding to a totally ordered hierarchy, as the basis of the recursion. We point out that the number of dummy classes and new edges added by such a technique in a hierarchy with n classes and dimension d, in order to reduce key derivation time to $2d+1$, is $O(n \cdot d \cdot (\log n)^{d-1} \cdot \log\log n)$ and not $O(n \cdot d \cdot (\log n)^{d-1})$ as claimed in [5].

Finally, in Section 4.3 we show how to further reduce key derivation time by improving the shortcutting and point-inserting technique. Our technique performs a further shortcutting of the graph obtained by the technique in [5] and allows key derivation time to be independent on d.

4.1 The Shortcutting Technique

The *shortcutting* of a directed graph $G = (V, E)$ consists into inserting additional edges, called *shortcut edges*, in E without changing the transitive closure of G. The goal is to obtain another directed graph, called a *shortcut graph*, having a smaller diameter than G. Atallah et al. [4] showed two constructions to add shortcut edges to tree hierarchies with n classes. The former allows the diameter of the shortcut graphs to be reduced to $O(\log\log n)$ by adding $O(n)$ shortcut edges, whereas, the latter allows the diameter to be reduced to three by adding $O(n \log\log n)$ shortcut edges.

The shortcutting technique is quite old, indeed it has been first considered in 1982 by Yao [39]. In particular, he considered the problem in a quite different context, where the n elements of V belong to a given semigroup $(S, \circ)$ and one is interested in answering queries of the form *"what is the value of $v_i \circ v_{i+1} \circ \cdots \circ v_{j-1} \circ v_j$?"* for any $1 \leq i \leq j \leq n$. As noticed by Thorup [36], the shortcutting technique has been later rediscovered by other authors [3, 7]. In the following we translate to our scenario the main existing results concerning the shortcutting technique when applied to particular kinds of graphs. We start discussing chains, then we analyze trees and finally general graphs.

CHAINS. By using the techniques proposed by Yao [39] in 1982 we can add shortcut edges to a chain $(v_1, \ldots, v_n)$ of n vertices. The techniques proposed by Alon and Schieber [3] in 1987 and Bodlaender et al. [7] in 1994 are essentially the same as the ones proposed by Yao, but their description is easier to illustrate and treats the case of constant diameter in a more detailed way. Given a parameter $\ell \geq 1$, Alon and Schieber established both upper and lower bounds on the minimum number of shortcut edges to be added to the chain in order to obtain a shortcut graph with diameter at most ℓ. Their construction, on input a chain $(v_1, \ldots, v_n)$, results in a shortcut graph having diameter at most ℓ, with the addition of $C(\ell, n)$ shortcut edges to the chain. We will see later that such a construction is the best possible when ℓ is a constant.

- **Case** $\ell = 1$: For any $i = 1, \ldots, n-1$ and any $j \geq i$, add the shortcut edge (v_i, v_j). Clearly, $C(1, n) = O(n^2)$.

- **Case** $\ell = 2$: The algorithm works as follows:

  1. If $n \geq 4$, then
     (a) Partition the chain $(v_1, \ldots, v_n)$ in two consecutive subchains $T_1 = (v_1, \ldots, v_{\lceil n/2 \rceil})$ and $T_2 = (v_{\lceil n/2 \rceil}, \ldots, v_n)$ having one vertex in common;
     (b) For $j = 1, \ldots, \lceil n/2 \rceil - 1$, add the shortcut edge $(v_j, v_{\lceil n/2 \rceil})$;
     (c) For $h = \lceil n/2 \rceil + 1, \ldots n$, add the shortcut edge $(v_{\lceil n/2 \rceil}, v_j)$;
     (d) Apply the construction recursively to the subchains T_1 and T_2.

  The number of shortcut edges added by the algorithm is given by the recurrence

  $$C(2, n) = \begin{cases} 0 & \text{if } n \leq 3; \\ C(2, \lceil n/2 \rceil) + C(2, \lfloor n/2 \rfloor) + O(n) & \text{otherwise;} \end{cases}$$

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

- **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 [33]):

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

  and

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

  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 function $w(\cdot, n)$ is very slowly-growing; in particular, it grows even slower than the *iterated* *logarithmic function* $\log^* n$, which, for all values of n 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 $\log^* n$ is defined to be the number of times the logarithm function must be applied in succession, starting with argument n, before the result is less than or equal to 1, i.e., $\log^* n = \min\{i \geq 0 : \log^{(i)} n \leq 1\}$, where

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

The first five values of $w(\cdot, n)$ are the following: $w(0, n) = \lceil n/2 \rceil$, $w(1, n) = \lceil \sqrt{n} \rceil$, $w(2, n) = \lceil \log n \rceil$, $w(3, n) = \lceil \log \lceil \log n \rceil \rceil$, and $w(4, n) = \log^* n$. Next values of $w(\cdot, n)$ can be computed as follows: $w(i, n) = \min\{j : w^{(j)}(i-2, n) \leq 1\}$, 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 construction for $\ell \geq 3$ works as follows:

1. If $n \geq \ell + 2$, then
   (a) Let $k = w(\ell - 2, n)$ and partition the chain $(v_1, \ldots, v_n)$ into $f = \lceil n/k \rceil \geq 2$ subchains $T_1, \ldots, T_f$ of k vertices each, where any two consecutive subchains have one vertex in common and the last subchain can contain less than k vertices. For $i = 1, \ldots, f-1$, let $T_i = (v_{(i-1)k-i+2}, \ldots, v_{ik-i+1})$ and let $T_f = (v_{(f-1)k-f+2}, \ldots, v_n)$;
   (b) For any $i = 1, \ldots, f-1$, and any $j = (i-1)k - i + 3, \ldots, ik - 1 + i$, add the shortcut edges $(v_{(i-1)k-i+2}, v_j)$ and (v_j, v_{ik-i+1}). For any $j = (f-1)k - f + 3, \ldots, n$, add the shortcut edges $(v_{(f-1)k-f+2}, v_j)$ and (v_j, v_n);
   (c) Apply the construction recursively to the subchains $T_1, \ldots, T_f$.
   (d) Let EP be the set containing the endpoints of each subchain. Apply the construction for $\ell - 2$ to the chain whose vertices are the elements of EP.

The number of shortcut edges added by the algorithm is given by the recurrence

$$C(\ell, n) = \begin{cases} 0 & \text{if } n \leq \ell + 1; \\ \lceil n/k \rceil \cdot C(\ell, k) + C(\ell - 2, \lceil n/k \rceil) + O(n) & \text{otherwise;} \end{cases}$$

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 $C(4, n) = O(n \cdot \log^* n)$.

For constant $\ell \geq 2$, the above construction is optimal. Indeed, Alon and Schieber [3] showed that the minimum number of shortcut edges to be added to a chain of n vertices in order to obtain a shortcut graph with diameter at most ℓ is

$$C_{min}(\ell, n) = \Omega(n \cdot w(\ell, n)). \tag{1}$$

Moreover, they showed that, by adding $O(n)$ shortcut edges, the diameter of the resulting shortcut graph is $\Omega(\log^* n)$. The parameters of known constructions for a chain of n vertices are summarized in Figure 7, where $\log^* n$, $\log^{**} n$, etc., correspond to the so called *milky way functions* in [7].

Trees. In 1987 Chazelle [10], as well as Alon and Schieber [3], considered free trees, i.e., indirected connected acyclic graphs, and showed that the minimum diameter ℓ which can be achieved with the addition of $m \geq n$ shortcut edges is $\Theta(\alpha(m,n) + \frac{n}{m-n+1})$, where α denotes the inverse of the Ackermann's function defined by Tarjan [33]. An equivalent result that lends itself well to parallelization and allows an algorithmically simpler proof has been achieved by Thorup [36]. The idea behind Chazelle's construction for a free tree T is the following: Choose an integer $1 \leq k \leq n$ and partition the tree into p subtrees $T_1, \dots, T_p$, such that each T_i contains a number n_i of nodes, where $k/3 < n_i \leq k$. Such a partition can be easily computed by using a well known result also appearing in [26]. Then, the subtrees $T_1, \dots, T_p$ are considered as the *super nodes* of a free tree. This allows the classification of any path in T as either falling entirely within a super node or stretching over several super nodes. The algorithm is recursively called on each super node; moreover, the super nodes become connected with the addition of shortcuts. Chazelle's result was also shown to hold for directed trees [35]. Given a forest with n vertices and height h, Thorup [36] showed how to obtain a shortcut graph having diameter three by adding $O(n \cdot \log\log h)$ shortcut edges. The problem of adding shortcut edges to chains and trees was also considered in [7], where a detailed treatment of the case of constant diameter was given. The parameters of known constructions are summarized in Figure 7.

Diameter ℓ	Minimal number of shortcut edges
1	$\Theta(n^2)$
2	$\Theta(n \cdot \log n)$
3	$\Theta(n \cdot \log\log n)$
4	$\Theta(n \cdot \log^* n)$
5	$\Theta(n \cdot \log^* n)$
6	$\Theta(n \cdot \log^{**} n)$
7	$\Theta(n \cdot \log^{**} n)$
8	$\Theta(n \cdot \log^{***} n)$
etc.	etc.
$O(\log^* n)$	$\Theta(n)$

Figure 7: Minimal number of shortcut edges to be added to chains and trees with n vertices in order to obtain a shortcut graph with diameter ℓ.

General Graphs. Thorup [34] conjectured that for any directed graph $G = (V, E)$ one can obtain a shortcut graph with diameter polylogarithmic in $|V|$, i.e., $(\log |V|)^{O(1)}$, by adding at most $|E|$ shortcut edges. He also showed his conjecture to be true for planar directed graphs [35]. However, Hesse [22] gave a counterexample to Thorup's conjecture. He showed how to construct a direct graph requiring the addition of $\Omega(|E| \cdot |V|^{1/17})$ shortcut edges to reduce its diameter below $\Theta(|V|^{1/17})$. By extending his construction to higher dimensions, it is possible to obtain graphs with $|V|^{1+\epsilon}$ edges that require the addition of $\Omega(|V|^{2-\epsilon})$ shortcut edges to reduce their diameter.

All constructions described in this section can be used to reduce key derivation time in hierarchical key assignment schemes. However, the result by Hesse [22] implies that key derivation time cannot be reduced *essentially* below $\Omega(|V|^2)$ for some kinds of graphs by adding only shortcut edges.

4.2 The Shortcutting and Point-Inserting Technique

Atallah et al.[5] also proposed a different technique to reduce the diameter of an access hierarchy. Such a technique makes use of the concept of dimension of a poset. The *dimension* of a poset $(V, \preceq)$, originally defined by Dushnik and Miller [19], is the minimum number of total orders on

V whose intersection is $(V, \preceq)$. It can also be seen as the smallest nonnegative integer d for which each $u \in V$ can be represented by a d-vector $(x_{u,1}, \ldots, x_{u,d})$ of integers such that $u \preceq v$ if and only if $x_{u,i} \leq x_{v,i}$, for any $i = 1, \ldots, d$, and any $u, v \in V$. 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 [38]. A poset has dimension one if and only if it is a total order.

Given a poset $(V, \preceq)$ with dimension d and n vertices, Atallah et al. [5] proposed an algorithm to add new edges, as well as dummy vertices, to the corresponding hierarchy. Their construction is recursive and for the base case $d = 1$, corresponding to a totally ordered set, i.e., a chain of n vertices, uses the following algorithm: Partition the vertices of the chain into $\lceil \sqrt{n} \rceil$ regions, where each region contains $\lceil \sqrt{n} \rceil$ vertices, with the exception of the last region, which can contain less vertices; let S be the set consisting of each $\lceil \sqrt{n} \rceil$-th vertex of the chain, including the last one. Any two vertices $v_i, v_j \in S$ such that $i < j$ are connected by a shortcut edge. Moreover, a shortcut edge between any two vertices $v_i \notin S$ and $v_j \in S$ ($v_i \in S$ and $v_j \notin S$, respectively), whose distance is shorter than $\lceil \sqrt{n} \rceil$, is added. Atallah et al. [5] claimed their algorithm results in a shortcut graph having diameter equal to three with the addition of at most $O(n)$ shortcut edges. However, since no shortcut edge is added between any two vertices $v_i, v_j \notin S$ inside the same region, the diameter of the resulting graph is not three, but essentially $\lceil \sqrt{n} \rceil$. If we want the diameter of the shortcut graph to be equal to three, we may recursively call the algorithm on each region, as done by the algorithm for $\ell = 3$ described in Section 4.1; the recursion bottoms up when each region contains at most four vertices, since in this case the diameter is already equal to three and no more shortcut edges need to be added. It is easy to see that the above algorithm results in the addition of $\Theta(n \cdot \log \log n)$ shortcut edges. Moreover, if we want the diameter of the shortcut graph to be at most three, we need to add at least $\Omega(n \cdot \log \log n)$ shortcut edges, because of (1).

For the case $d \geq 2$, the input consists of a set of n points in the d-dimensional space. Such points correspond to the d-vectors associated to the vertices of the poset $(V, \preceq)$ and are not connected by edges. Let M be a $(d-1)$-dimensional hyperplane perpendicular to the d-th dimension; M partitions the set of vertices V into two sets V_1 and V_2, where V_1 is the set of points that are on the smaller side of the hyperplane, according to the d-th coordinate. Let V_1' and V_2' be the projections of the points in V_1 and V_2 on M. A new edge from every point of V_1' (resp., V_2) to its corresponding point of V_1 (resp., V_2') is added. Then the algorithm is recursively executed on the two sets of points V_1 and V_2. Finally, the $(d-1)$-dimensional problem on the set of points $V_1' \cup V_2'$ is solved by using the algorithm for dimension $d-1$. According to Atallah et al. [5], the above construction results in a graph having diameter equal to $2d+1$, with the addition of at most $O(n \cdot d \cdot (\log n)^{d-1})$ new edges and dummy vertices. However, their computations take in account the number of shortcut edges added to a chain of n vertices, which, as noticed before, is not $O(n)$, but $\Theta(n \cdot \log \log n)$. According to the correct result for the chain, the number of new edges and dummy vertices added by their algorithm is thus $O(n \cdot d \cdot (\log n)^{d-1} \cdot \log \log n)$. In the following we show how to further reduce key derivation time. Our technique performs a further shortcutting of the graph obtained by Atallah et al.'s technique and allows key derivation time to be independent on d.

4.3 Reducing the Diameter in the Shortcutting and Point-Inserting Technique

In this section we consider the problem of reducing the diameter of the graph obtained by the shortcutting and point-inserting technique, on input a poset $(V, \preceq)$ with dimension d. Our construction, which we refer in the following as the *Improved Shortcutting and Point-Inserting Technique (ISPIT)* is recursive, and for the base case $d = 1$ reduces to the construction proposed by

Yao [39] and described in Section 4.1. The construction for the case $d \geq 2$ is described in Figure 8. The input is a set of n d-dimensional points corresponding to the vertices in V; for each vertex $v \in V$, let $P_v^{(d)}$ be the corresponding point and let $V^{(d)} = \{P_v^{(d)} : v \in V\}$.

Let $(V, \preceq)$ be a poset with dimension $d \geq 2$, let $V^{(d)}$ be the set of points in the vectorial representation of the Hasse diagram associated to $(V, \preceq)$ and based on its d total orders, and let $\ell \geq 1$.

1. If $|V^{(d)}| = 1$, then output $V^{(d)}$.
2. If $|V^{(d)}| \geq 2$, compute a $(d-1)$-dimensional hyperplane M perpendicular to the d-th dimension that partitions the set of points in $V^{(d)}$ into two sets $V_1^{(d)}$ and $V_2^{(d)}$ of $\lfloor n/2 \rfloor$ and $\lceil n/2 \rceil$ points, respectively, where $V_1^{(d)}$ is the set on the smaller side of the hyperplane (according to the d-th coordinate). Such points are projected on M. Denote by $P_v^{(d-1)}$ the projection of $P_v^{(d)}$ on M. Let $V_1^{(d-1)}$ and $V_2^{(d-1)}$ be the projections of $V_1^{(d)}$ and $V_2^{(d)}$.
3. If $d = 2$, use the construction of Section 4.1 on the chain whose vertices are the points in the set $V^{(1)}$, in order to obtain a shortcut graph $G^{(1)} = (V^{(1)}, E^{(1)})$, having diameter at most ℓ. The set of dummy points added by the algorithm is $D^{(1)} = \emptyset$ (no dummy points are added).
4. If $d \geq 3$, recursively call the algorithm on the set of points in $V^{(d-1)} = V_1^{(d-1)} \cup V_2^{(d-1)}$, corresponding to a $(d-1)$-dimensional hyperplane; let $G^{(d-1)} = (V^{(d-1)} \cup D^{(d-1)}, E^{(d-1)})$ be the corresponding output.
5. Let $D^{(d)} = V^{(d-1)} \cup D^{(d-1)}$.
6. Let $E^{(d)} = E^{(d-1)}$.
7. Add edges between points in $V^{(d)}$ and corresponding projections:
   (a) For each point $P_v^{(d)} \in V_1^{(d)}$, add an edge $(P_v^{(d-1)}, P_v^{(d)})$ to $E^{(d)}$.
   (b) For each point $P_v^{(d)} \in V_2^{(d)}$, add an edge $(P_v^{(d)}, P_v^{(d-1)})$ to $E^{(d)}$.
8. Add shortcut edges between points in $V^{(d)}$ and dummy points:
   (a) For each edge $(P_u^{(d-1)}, P_v^{(j)}) \in E^{(d-1)}$, add an edge $(P_u^{(d)}, P_v^{(j)})$ to $E^{(d)}$.
   (b) For each edge $(P_u^{(j)}, P_v^{(d-1)}) \in E^{(d-1)}$, add an edge $(P_u^{(j)}, P_v^{(d)})$ to $E^{(d)}$.
9. Recursively call the algorithm on the two sets of points in $V_1^{(d)}$ and $V_2^{(d)}$.
10. Output the graph $G^{(d)} = (V^{(d)} \cup D^{(d)}, E^{(d)})$.

Figure 8: The Improved Shortcutting and Point-Inserting Technique.

The number $DP(n, d)$ of dummy points added by the ISPIT is $DP(n, d) = 2 \cdot DP(\lceil n/2 \rceil, d) + DP(n, d-1) + \Theta(n)$, where $DP(n, 1) = 0$ and $DP(1, d) = 0$. Indeed, in order to construct $G^{(d)}$, the algorithm adds n dummy points, corresponding to the projections of the points in $V^{(d)}$ on the $(d-1)$-dimensional hyperplane M, plus $DP(n, d-1)$ dummy points for the construction of $G^{(d-1)}$, and then is recursively called on the two sets $V_1^{(d)}$ and $V_2^{(d)}$. The solution of the above recurrence is $DP(n) = \Theta(n \cdot d \cdot (\log n)^{d-1})$. On the other hand, the number $T(n)$ of new edges added by the ISPIT is $T(n, d) \leq 2 \cdot T(\lceil n/2 \rceil, d) + 3 \cdot T(n, d-1) + \Theta(n)$, where $T(n, 1)$ denotes the number of shortcut edges added by the construction of Section 4.1 for the case $d = 1$ in order to obtain a shortcut graph having a certain diameter, whereas, $T(1, d) = 0$. Indeed, at most $3 \cdot |E^{(d-1)}| + n$ new edges are added in steps 7. and 8. and then the algorithm is recursively called on the two sets $V_1^{(d)}$ and $V_2^{(d)}$. Clearly, the solution of $T(n, d)$, as well as the diameter of the graph $G^{(d)}$, depends on the the number $T(n, 1)$ of shortcut edges added by the construction of Section 4.1. If $T(n, 1) = \Theta(n)$, then $T(n, d) = O(n \cdot d \cdot (3 \log n)^{d-1})$. On the other hand, if $T(n, 1) = \Theta(n \cdot \log \log n)$, then $T(n, d) = O(n \cdot d \cdot (3 \log n)^{d-1} \cdot \log \log n)$ and the diameter of the graph $G^{(d)}$ is three, i.e., it is independent on d. It is easy to see that, for any two vertices $u, v \in V$ such that $u \preceq v$, there exists a path from $P_v^{(d)}$ to $P_u^{(d)}$ in $G^{(d)}$ which has length at most the diameter of the graph $G^{(1)}$ obtained by solving the 1-dimensional problem on $V^{(1)}$.

Compared to the technique in [5], the ISPIT allows a further reduction of the diameter, but in each recursive call, it adds at most three times the number of new edges added by that algorithm. In the following we show a trade-off between the number of edges added by the ISPIT and the diameter of the resulting graph. The idea behind the construction is the following: Assume the 1-dimensional problem is solved by adding $\Theta(n \log \log n)$ shortcut edges. For each $j = 2, \ldots, d$, the j-dimensional problem could be solved either with the technique in [5] or with ours. Let $1 \leq t \leq d$ and assume, for example, that for $j = 2, \ldots, t$, the technique in [5] is used to solve the j-dimensional problem, whereas, our technique is used to solve the problems with dimensions from $t + 1$ to d. It is easy to see that the graph resulting by the above construction has diameter $2t + 1$. Moreover, the number of new edges added by the modified algorithm is $O(3^{d-t} \cdot n \cdot t \cdot (\log n)^{d-1} \cdot \log \log n)$.

An Example of the New Technique. For the reader's convenience, in the following we illustrate how the ISPIT works for the case $d = 2$. Consider the hierarchy represented on the left hand side of Figure 9. The vectorial representation of the hierarchy is represented on the right hand side of Figure 9.

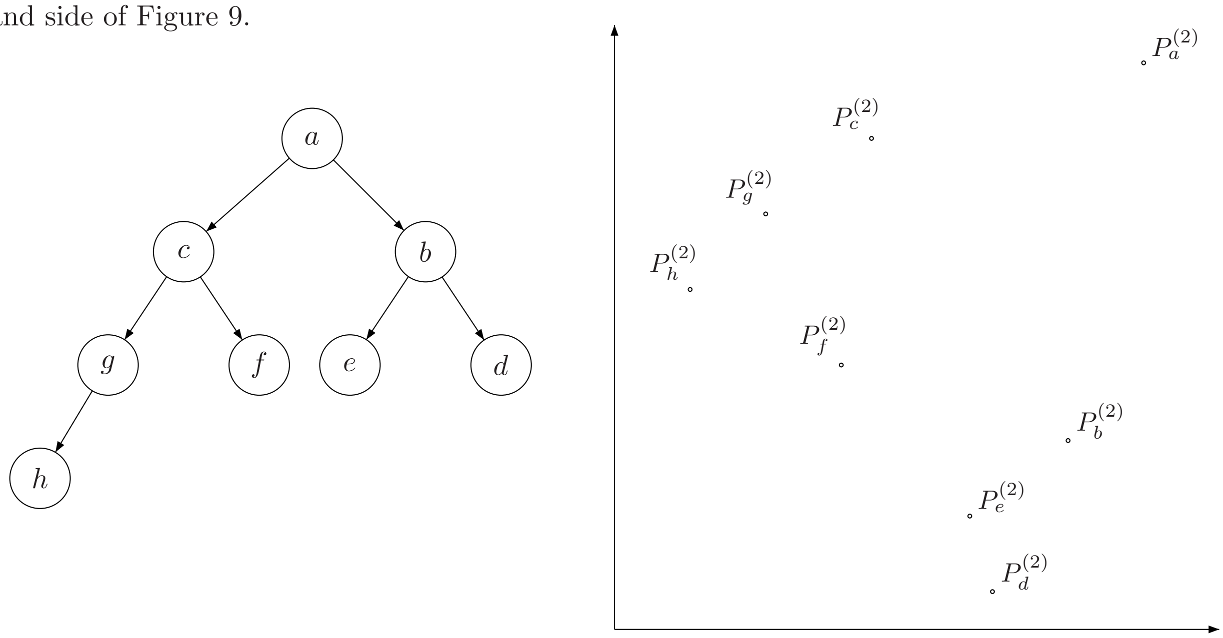

Figure 9: A hierarchy with dimension $d = 2$ and its vectorial representation.

Notice that if a point $P_v^{(2)} = (x_v, y_v)$ dominates another point $P_u^{(2)} = (x_u, y_u)$ in the vectorial representation, i.e., if $x_u < x_v$ and $y_u < y_v$, then there is a path from vertex v to vertex u in the hierarchy. The left hand side of Figure 10 shows the dummy points, represented by open circles, corresponding to the projections of the 2-dimensional points on the median line. The right hand side of Figure 10 shows the shortcut graph obtained by the construction of Section 4.1, for $\ell = 2$, on the chain of dummy points.

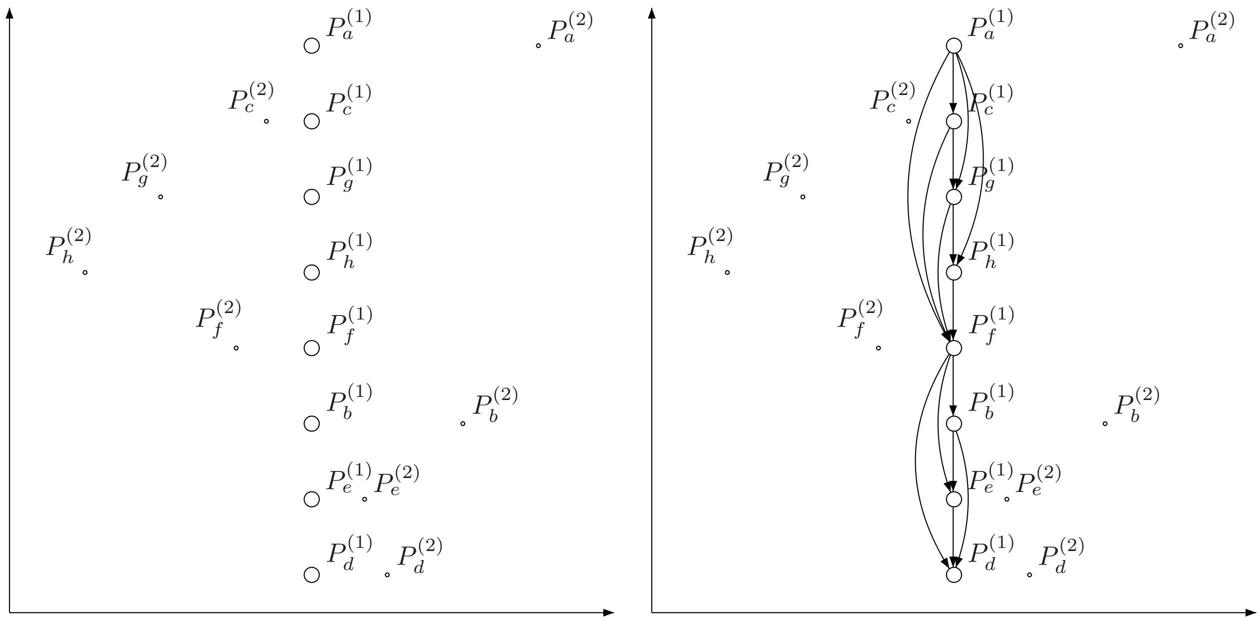

Figure 10: Dummy points corresponding to the projections of two-dimensional points and shortcut graph obtained by the construction of Section 4.1, for $\ell = 2$, on the chain of dummy points.

The left hand side of Figure 11 shows the new edges, represented by dashed lines, added by the ISPIT in step 7. Finally, the right hand side of Figure 11 shows the shortcut edges added by the ISPIT in step 8. New edges and points added in recursive calls of the algorithm are not represented.

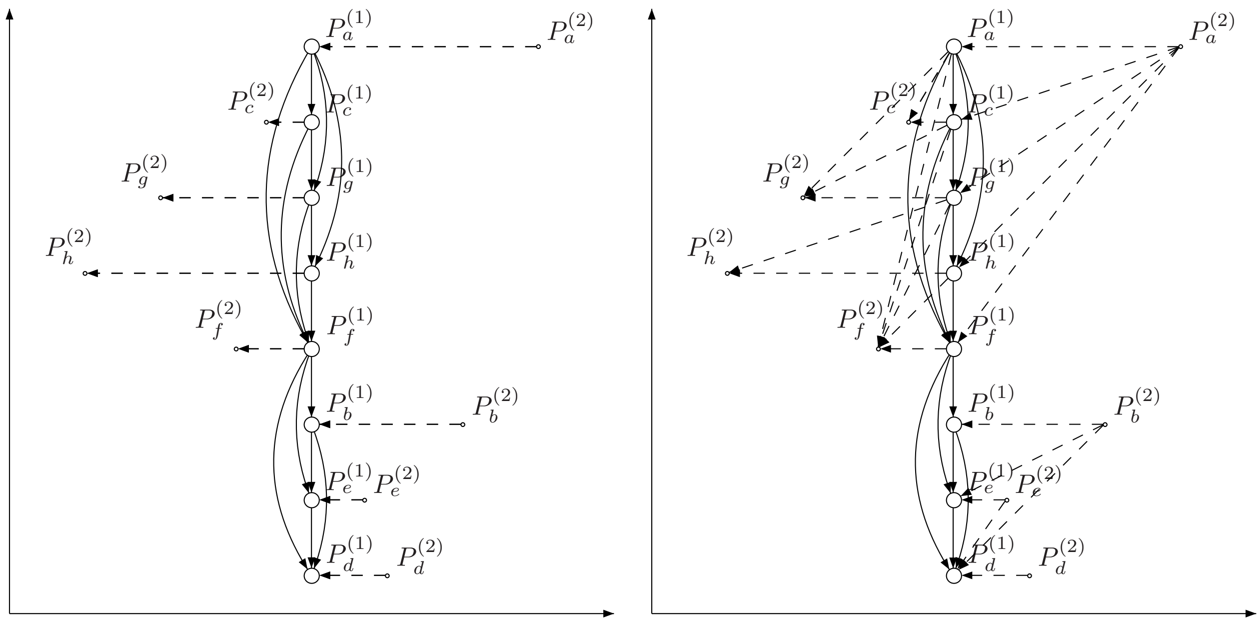

Figure 11: New edges added by our algorithm in steps 7. and 8.

5 A Construction based on Broadcast Encryption Schemes

In this section we show how to construct a hierarchical key assignment scheme using as a building block a broadcast encryption scheme. A broadcast encryption scheme allows a sender to broadcast an encrypted message to a set of users in such a way that only legitimate users can decrypt it. Broadcast encryption schemes can be either public key or symmetric key based. In the symmetric key setting, only a trusted authority can broadcast data to the receivers. In contrast, in the public key setting a public key published by a trusted authority allows anybody to broadcast a message. We first recall the definition of a public-key broadcast encryption scheme [9].

Definition 5.1 *A* public-key broadcast encryption scheme *for a set $\mathcal{U}$ of users is a triple of algorithms (Set, Enc, Dec) satisfying the following conditions:*

1. *The* setup algorithm *Set is probabilistic polynomial-time. It takes as input a security parameter 1^τ and the set of users $\mathcal{U}$ and produces as output a private key sk_u, for each user $u \in \mathcal{U}$, and a public key pk.*

2. *The* encryption algorithm *Enc is probabilistic polynomial-time. It takes as inputs 1^τ, a subset $X \subseteq \mathcal{U}$, and the public key pk, and produces as output the a pair (Hdr, k), where Hdr is called the broadcast header and k is a encryption key. Let m be a message to be broadcast in such a way that only users in X are allowed to obtain m and let y be the encryption of m under the symmetric key k. The broadcast message consists of (X, Hdr, y), where the pair (X, Hdr) is called the full broadcast header and y is called the broadcast body.*

3. *The* decryption algorithm *Dec is deterministic polynomial-time. It takes as inputs 1^τ, a subset $X \subseteq \mathcal{U}$, a user $u \in X$ and its private key sk_u, a broadcast header Hdr, and the public key pk, and produces as output the key k. Such a key can be used to decrypt the broadcast body y in order to obtain m.*

We require that for all subsets $X \subseteq \mathcal{U}$, all users $u \in X$, all public keys and private keys which can be output by $Set(1^\tau, \mathcal{U})$, all pairs (Hdr, k), which can be output by $Enc(1^\tau, X, pk)$, we have that $Dec(1^\tau, X, u, sk_u, Hdr, pk) = k$.

In the following we show how to construct a hierarchical key assignment scheme using as a building block a public-key broadcast encryption scheme. The idea behind our construction, referred in the following as the *Broadcast Encryption Based Construction (BEBC)*, is to compute the private and public information by using the broadcast encryption scheme; more precisely, the public information will contain a broadcast header Hdr_u, which corresponds to an encryption of the key k_u, for each class $u \in V$. Such a broadcast header can be decrypted by all classes in the set $I_u = \{v \in V : \text{ there is a path from } v \text{ to } u \text{ in } G\}$, allowing them to compute the key k_u. The Broadcast Encryption Based Construction is described in Figure 12.

5.1 Analysis of the Scheme

In this section we first show that the security property of the BEBC depends on the security property of the underlying broadcast encryption scheme. Afterwards, we evaluate the performances of the BEBC with respect to several parameters, such as space requirements for public and private information storage and computational requirements for key derivation and key updates.

Before analyzing the security of the BEBC we first need to define what we mean by a *secure* public-key broadcast encryption scheme. The security of a public-key broadcast encryption scheme

Let Γ be a family of graphs corresponding to partially ordered hierarchies. Let $G = (V, E) \in \Gamma$ and let (Set, Enc, Dec) be a public-key broadcast encryption scheme for users in V.

Algorithm $Gen(1^\tau, G,)$

1. Run $Set(1^\tau, V)$ to generate a public key pk and a secret key sk_u for any $u \in V$;
2. For each class $u \in V$, let $s_u = sk_u$;
3. For each class $u \in V$, run $Enc(1^\tau, I_u, pk)$ to obtain the pair (Hdr_u, k_u);
4. Let s and k be the sequences of private information and keys computed in the previous two steps;
5. Let pub be the sequence constituted by the public key pk along with the header Hdr_u, for any $u \in V$;
6. Output (s, k, pub).

Algorithm $Der(1^\tau, G, u, v, s_u, pub)$

1. Extract the public key pk and the header Hdr_v from pub.
2. Output $k_v = Dec(1^\tau, I_v, u, s_u, Hdr_v, pk)$.

Figure 12: The Broadcast Encryption Based Construction.

is defined through a game between an adversary A and a challenger. According to the capabilities of the adversary and the security goal, several types of security notions for public-key broadcast can be defined. We consider the definition of *semantic security* given by Boneh et al. [9], where the adversary is not allowed to issue decryption queries to the challenger. Different notions of security, such as *chosen ciphertext security* can be found in [18, 9, 24]. We consider the following game:

- **Initialization.** Algorithm A outputs a set $X \subseteq \mathcal{U}$ of receivers that it wants to attack.
- **Setup.** The challenger first runs $Set(1^\tau, \mathcal{U})$ to obtain a private key sk_u for each user $u \in \mathcal{U}$ and a public key pk. Afterwards, it gives the public key pk and all private keys sk_v for which $v \notin X$ to A.
- **Challenge.** The challenger runs $Enc(1^\tau, X, pk)$ to obtain (Hdr, k). Then, it picks a random bit $b \in \{0, 1\}$, sets $k_b = k$ and chooses k_{1-b} as a random key. The challenge (Hdr, k_0, k_1) is given to A.
- **Guess.** Algorithm A outputs its guess $b' \in \{0, 1\}$ for b and wins the game if $b = b'$.

The advantage of the adversary A is defined as

$$\mathbf{Adv}_{A,\mathcal{U}}(1^\tau) = |Pr[b' = b] - 1/2|.$$

Definition 5.2 *Let (Set, Enc, Dec) be a public-key broadcast encryption scheme for a set $\mathcal{U}$ of users. The scheme is said to be* semantically secure *if the function $\mathbf{Adv}_{A,\mathcal{U}}(1^\tau)$ is negligible, for any adversary A whose time complexity is polynomial in τ.*

Now we are ready to show that if the public-key broadcast encryption scheme (Set, Enc, Dec) is semantically secure, then the BEBC is secure in the sense of `IND-ST`.

Theorem 5.3 *If the public-key broadcast encryption scheme* (Set, Enc, Dec) *is semantically secure, then the BEBC is secure in the sense of* `IND-ST`.

Proof. Assume by contradiction that the BEBC is not secure in the sense of `IND-ST`. Thus, there exists a graph $G = (V, E)$ in Γ and a class $u \in V$ for which there exists a polynomial time adversary $\mathtt{STAT}_u$ whose advantage $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G)$ is non negligible. We show how to construct a polynomial time adversary A which, by using $\mathtt{STAT}_u$, is able to break the semantic security of the broadcast encryption scheme used as a building block of the BEBC.

The adversary A first chooses $I_u \subseteq V$ as the set of receivers it wants to attack. Then, it interacts with the challenger, obtaining the public key pk and the secret keys for all users in $V \setminus I_u$. Afterwards, it gets the challenge (Hdr, k_0, k_1) computed by the challenger. Then, A constructs the inputs for $\mathtt{STAT}_u$ as follows:

- For each class $v \in V \setminus \{u\}$, A runs $Enc(1^\tau, I_v, pk)$ to obtain the pair (Hdr_v, k_v);
- The public information pub consists of the public key pk along with the header Hdr_v, for any $v \in V$, with Hdr_u set equal to the challenge Hdr;
- The private information $corr$ held by corrupted users consists of the secret keys for all users in $V \setminus I_u$;
- The last input for $\mathtt{STAT}_u$ consists of the key k_1 contained in the challenge (Hdr, k_0, k_1).

Notice that if the last input for $\mathtt{STAT}_u$ is equal to the key k hidden into the header Hdr, then the random variable associated to $\mathtt{STAT}_u$'s view is exactly the same as in experiment $\mathbf{Exp}^{\mathtt{IND-1}}_{\mathtt{STAT}_u}(1^\tau, G)$, whereas, if it is a random string, such a variable has the same distribution as the one associated to $\mathtt{STAT}_u$'s view in experiment $\mathbf{Exp}^{\mathtt{IND-0}}_{\mathtt{STAT}_u}(1^\tau, G)$.

Finally, A outputs the same output as $\mathtt{STAT}_u(1^\tau, G, pub, corr, k_1)$. It is easy to see that

$$\mathbf{Adv}_{A,V}(1^\tau) = \mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G).$$

Since $\mathbf{Adv}^{\mathtt{IND}}_{\mathtt{STAT}_u}(1^\tau, G)$ is non negligible, it follows that adversary A is able to break the semantic security of the public-key broadcast encryption scheme. Contradiction. $\square$

Regarding space requirements, the public information pub in the BEBC consists of the public key pk, as well as of a public header Hdr_u for each class $u \in V$. Hence, the size of the public information depends on the size of the public key and of the header in the underlying public-key broadcast encryption scheme. On the other hand, each class has to store a single secret value, corresponding to the secret key in the underlying scheme. Moreover, users are required to perform a single decryption in order to derive a key.

5.2 An Efficient Construction using Bilinear Maps

In this section we show how to obtain a broadcast encryption based hierarchical key assignment scheme where the amount of public information is linear in the number of classes and the private information assigned to each class has constant size. The idea is to use the BEBC by plugging in the public-key broadcast encryption scheme proposed by Boneh et al. [9], which is based on a bilinear map between two groups. A function $e : G_1 \times \hat{G}_1 \to G_2$ is said to be a *bilinear map* if: 1) G_1 and $\hat{G}_1$ are two groups of the same prime order q; 2) For each $\alpha, \beta \in Z_q$, each $g \in G_1$, and each $h \in \hat{G}_1$, the value $e(g^\alpha, h^\beta) = e(g, h)^{\alpha\beta}$ is efficiently computable; and 3) The map is non-degenerate (i.e., if g generates G_1 and h generates $\hat{G}_1$, then $e(g, h)$ generates G_2). In the following, for simplicity, we focus on symmetric bilinear maps (i.e., such that $G_1 = \hat{G}_1$).

The *m-Bilinear Decisional Diffie-Hellman Exponent Problem (m-BDDHE)* in $< G_1, G_2, e >$, formally introduced in [8], is as follows: given a tuple $(g, h, g_1, \dots, g_m, g_{m+2}, \dots, g_{2m}, x) \in G_2^{2m+2}$, where $g_i = g^{(\alpha^i)}$ for $i = 1, \dots, m-1, m+1, \dots, 2m$, for a randomly chosen generator g of G_1, randomly chosen $\alpha \in Z_q^*$, $h \in G_1$, and $x \in G_2$, decide whether $x = e(g^{\alpha^{m+1}}, h)$. The *m-Bilinear Decisional Diffie-Hellman Exponent Assumption* is the assumption that the m-BDDHE problem is computationally hard. Such an assumption holds in generic bilinear groups [8].

Boneh et al. [9] showed how to construct a semantically secure broadcast encryption scheme for a set $\mathcal{U}$ of n users, assuming the intractability of the n-BDDHE problem. In their scheme the private key held by each user consists of a single group element, the public key contains $2n+1$ group elements, whereas, each broadcast header consists of two group elements. For a subset $X \subseteq \mathcal{U}$ of receivers, one decryption operation requires at most $|X|-2$ group operations. However, if a receiver in X has already decrypted a broadcast message for a set of receivers X' which is similar to X, then only $|X|-|X'|$ group operations are needed. It follows that if we use such a public-key broadcast encryption scheme in the BEBC, we obtain a hierarchical key assignment scheme where the public information consists of $4|V|+1$ group elements, whereas, the private information has constant size. Finally, key derivation requires a single (complex) decryption operation (as already discussed before, such a decryption can require at most $|V|-2$ group operations).

5.2.1 Handling Dynamic Changes

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

Before describing how the updates can be managed by the TA, we notice that the scheme proposed by Boneh et al. [9] also handles the *incremental addition* of new users, without restricting a-priori the total number of users which can be managed. However, this involves the distribution of one more private value to each user. Moreover, the scheme also supports the *incremental sharing* operation, i.e., the broadcaster may enable new users to decrypt the broadcast message. Such an addition requires the broadcaster to compute a new header and to remember a secret value associated to such a header. Now we are ready to discuss how the TA can manage any change to the hierarchy in our construction.

Insertion of an edge. Let (u, v) be an edge to be inserted in E. Such an update, involving the insertion of the class u in the set I_v, can be managed by the TA by performing the incremental sharing operation offered by the scheme in [9].

Deletion of an edge. Let (u, v) be an edge to be deleted from E. The TA first updates I_v to be the set $I_v \setminus \{u\}$ and then substitutes the old header Hdr_v, contained in the public information *pub*, with a new one corresponding to a new key k_v. Such a substitution is necessary to forbid users belonging to class u from computing the key of class v. The new pair (Hdr_v, k_v) is obtained by running $Enc(1^\tau, I_v, pk)$.

Insertion of a class. Let u be a class to be inserted in V along with new incoming and outgoing edges. Such an update can be managed by the TA by first performing the incremental addition of a new user offered by the scheme in [9], and then by adding the edges using the above procedure for edge insertions.

Deletion of a class. Let u be a class to be deleted from V. For each edge outgoing from u, the TA uses the above procedure for edge deletions. Afterwards, the TA deletes the header Hdr_u from *pub*.

6 Summary of the Results

In this paper we have designed and analyzed hierarchical key assignment schemes which are provably-secure and efficient. We have proposed a first construction based on symmetric encryption schemes and a second one using as a building block a public-key broadcast encryption scheme. Both constructions are provably secure with respect to key indistinguishability, require a single computational assumption and improve on previous proposals. In particular, one of our constructions provides constant private information and public information linear in the number of the classes. Moreover, both schemes support dynamic updates to the hierarchy with local changes to the public information and without requiring any private information to be re-distributed.

Figure 13 shows a comparison between our constructions and the one proposed by Atallah et al. [4] for a partially ordered hierarchy $G = (V, E)$. The comparison takes into account several parameters, such as the public and private information, the number and the type of operations required by a class $u \in V$ to compute the key of a class v lower down in the hierarchy, and the computational assumption.

Scheme	Public information	Private information	Key derivation	Computational assumption
Atallah et al. [4]	$2\|E\| + \|V\|$	One	$2 \cdot dist_G(u,v) + 1$ operations: • $dist_G(u,v)$ decryptions • $dist_G(u,v) + 1$ PRF eval.	CCA-secure encryption + PRF
EBC	$\|E\| + \|V\|$	One	$dist_G(u,v) + 1$ decryptions	IND-P1-C0-secure encryption
Modified EBC	$2\|E\| + \|V\|$	One	$dist_G(u,v)$ decryptions	IND-P1-C0-secure encryption
DEBC	$\|E\| + 2\|V\|$	One	$dist_G(u,v) + 2$ decryptions	IND-P1-C0-secure encryption
Modified DEBC	$3\|E\| + 2\|V\|$	One	$dist_G(u,v)$ decryptions	IND-P1-C0-secure encryption
BEBC	$4\|V\| + 1$	One	One (complex) decryption	$\|V\|$-BDDHE

Figure 13: Comparison between hierarchical key assignment schemes which are provably secure in the sense of IND-ST.

Using the constructions proposed in this work, new constructions for hierarchical key assignment schemes with temporal constraints have been recently proposed [17]. Such schemes exhibit a tradeoff among the amount of secret data that needs to be distributed and stored by the users, the amount of public data, the complexity of key derivation, and the computational assumption on which the security of the scheme is based.

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.

[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] B. Chazelle, *Computing on a Free Tree via Complexity-Preserving Mappings*, Algorithmica, 2, 337–361, 1987.

[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, *Cryptographic Key Assignment Schemes for any Access Control Policy*, Information Processing Letters, 92(4), 199–205, 2004.

[15] 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.

[16] A. De Santis, A. L. Ferrara, and B. Masucci, *Unconditionally Secure Key Assignment Schemes*, Discrete Applied Mathematics, 154(2) , 234–252, 2006.

[17] A. De Santis, A. L. Ferrara, and B. Masucci, *New Constructions for Provably-Secure Time-Bound Hierarchical Key Assignment Schemes*, available as Report 2006/483 at the IACR Cryptology ePrint Archive.

[18] Y. Dodis and N. Fazio, *Public Key Broadcast Encryption Secure against Adaptive Chosen Ciphertext Attacks*, in Proc. of the 6th International Workshop on Theory and Practice in Public Key Cryptography - PKC 2003, Miami, Florida, January 2003, Lecture Notes in Computer Science, 2567, 100–115.

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

[20] S. Goldwasser and S. Micali, *Probabilistic Encryption*, Journal of Computer and System Sciences, 28, 270–299, 1984.

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

[22] W. Hesse, *Directed Graphs Requiring Large Number of Shortcuts*, in Proc. of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms - SODA 2003, Baltimore, Maryland, USA, January 2003, 665–669.

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

[24] J. Y. Hwang, D. H. Lee, and J. Lim, *Generic Transformation for Scalable Broadcast Encryption Schemes*, in Proc. of Advances in Cryptology - Crypto 2005, Santa Barbara, California, USA, August 2005, Lecture Notes in Computer Science, 3621, 276–292.

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

[26] D. Knuth, *The Art of Computer Programming*,Vol. 1, Fundamental Algorithms, Addison-Wesley, 1973.

[27] 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.

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

[29] 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.

[30] 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.

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

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

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

[34] M. Thorup, *On Shortcutting Digraphs*, in Proc. of the 18th International Workshop on Graph-Theoretic Concepts in Computer Science - WG '92, Wiesbaden-Naurod, Germany, June 1992, Lecture Notes in Computer Science, 657, 205–211.

[35] M. Thorup, *Shortcutting Planar Digraphs*, Combinatorics, Probability & Computing, 4, 287 – 315, 1995.

[36] M. Thorup, *Parallel Shortcutting of Rooted Trees*, Journal of Algorithms, 23, 139–159, 1997.

[37] 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.

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

[39] 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.

[40] 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.