Simple, Efficient and Strongly KI-Secure Hierarchical Key Assignment Schemes

Eduarda S. V. Freire*, Kenneth G. Paterson**, and Bertram Poettering**

Information Security Group,
Royal Holloway, University of London, U.K.

Abstract Hierarchical Key Assignment Schemes can be used to enforce access control policies by cryptographic means. In this paper, we present a new, enhanced security model for such schemes. We also give simple, efficient, and strongly-secure constructions for Hierarchical Key Assignment Schemes for arbitrary hierarchies using pseudorandom functions and forward-secure pseudorandom generators. We compare instantiations of our constructions with state-of-the-art Hierarchical Key Assignment Schemes, demonstrating that our new schemes possess an attractive trade-off between storage requirements and efficiency of key derivation.

Keywords: key assignment scheme, general poset, pseudorandom function, pseudorandom generator, forward security.

1 Introduction

Access control: There are numerous examples where it is desirable to provide differentiated access to data according to an *access control policy*. As an illustration, consider a hospital where doctors are assigned access permission to a set of files containing some personal information in a patient's medical record, depending on their seniority, while nurses, being at a lower level in the hierarchy, have more restricted access to that information. As another example, consider a building management scenario where sensors are installed to capture temperature, humidity, light, motion, sound, or other data. These data have different levels of sensitivity, and access to information of different types might be restricted to different personnel, depending on their roles in the organization. Normal employees would only be able, for example, to access temperature, humidity and light of the floor where they work, while managers of that floor would be able to have access to information related to presence in rooms on that floor, like motion and sound data. The manager of the building would, however, have access to all information for the different floors of the building. As a third example, broadcasters wish to control access to broadcast services in such a way that only paying customers can access the programmes included in the package to which they have subscribed, and nothing else. Other application domains include management of databases containing sensitive information, military and government communication, and protection of industrial secrets. Indeed the field of access control is a healthy sub-discipline of Information Security in its own right.

Cryptographic enforcement: The use of cryptographic techniques to enforce access control policies for hierarchical structures was first proposed in 1983, by Akl and Taylor [1], who put forward the concept of a *(hierarchical) key assignment scheme (KAS)*. Such a scheme is a method to assign some private information and encryption keys to each

* This author supported by CAPES Foundation/Brazil on grant 0560/09-0 and Royal Holloway, University of London.
** This author supported by EPSRC Leadership Fellowship EP/H005455/1.

class in a hierarchy in such a way that the private information assigned to a class, along with some public information, can be used to derive symmetric encryption keys assigned to all classes lower down in the hierarchy. Formally, the hierarchy is modelled as a *partially ordered set (poset)*, each data item is labelled by a class u in the hierarchy, and is encrypted using the encryption key k_u corresponding to that class. Now a user, given access to the private information S_u, can derive the relevant encryption key k_v for any descendant class v, and hence gain access to the data of class v. Since the original paper by Akl and Taylor, a large number of different schemes have been proposed, offering different trade-offs in terms of the amount of public and private storage required and the complexity of key derivation – see for example [2–17]. Many additional issues are addressed in these works: time-dependent constraints, dynamic addition and removal of classes, and revocation, for example. A recent survey of this area by Crampton *et al.* [18] provides a detailed classification and analyses of many of the schemes proposed in the last decades.

Many of the early schemes lacked any formal security analysis, but this shortcoming has been gradually addressed beginning with the work of Atallah *et al.* [8], who proposed two different security notions: security against key recovery attacks (KR-security) and security with respect to key indistinguishability (KI-security). Informally, KR-security captures the notion that an adversary should not be able to compute a key to which it should not have access; whereas in the notion of KI-security, the adversary should not even be able to distinguish between the real key and a random string of the same length. The stronger KI-security notion is important in enabling secure composability for hierarchical key assignment schemes, that is, in achieving the property that any secure key assignment scheme can be safely used alongside any suitably secure encryption scheme.

Our contributions: We first argue that the KI-security notion introduced in [8] needs to be strengthened in order to capture the widest possible range of realistic attacks. In particular, the current model does not allow an adversary to gain access to encryption keys k_v for classes above the target class u, even though these encryption keys might leak through usage and their compromise need not directly lead to a compromise of the private information S_u or encryption key k_u for the target class. We then define a model that provides this additional compromise capability to the adversary, and show that our new model is strictly stronger than the existing KI-security notion. Section 2 contains the details.

We next propose two very simple and efficient hierarchical key assignment schemes for arbitrary posets, and prove them to be secure in the sense of our strengthened security notion. Both of our schemes exploit the chain partition idea recently introduced by Crampton *et al.* [15]. This gives a method of constructing a KAS for an arbitrary access structure (modelled as a poset), represented by a directed acyclic graph $P = (V, E)$, from a KAS for a simple chain (i.e. a KAS for a totally ordered set) by partitioning the poset into chains and building the keys for the more complex scheme for P in a particular way from the keys of the simpler chain KAS. This approach has the nice property that the amount of private storage needed per class is bounded by the *width* of the poset P. This approach was proposed without any formal security analysis in [15], and analysed in some specific cases in [16]. We provide in Section 3 a generic security analysis of this approach, showing that the security of the resulting scheme for $P = (V, E)$ in our strengthened model is equivalent to the security (also in our strengthened model) for the chain scheme. It is worth noting that this construction can support different levels

of security or efficiency of key derivation for different subgroups in a hierarchy, by using different schemes in each chain.

This construction enables us to focus on constructing efficient KAS for chain posets in our strengthened model. Our first construction in Section 4 is based only on pseudorandom functions (PRFs), which can be efficiently implemented using, for example, HMAC [19] built using only a cryptographic hash function. Our second construction, in Section 5, is based on any forward-secure pseudorandom generator (FS-PRG). This construction is a generalization and a strengthening of the construction for chains given in [16], which implicitly makes use of the known forward-security of the BBS PRG [20] in order to achieve KI-security. Note that the BBS generator was not originally presented as a stateful generator and its forward-security property was first used in the Blum-Goldwasser cryptosystem [21] and later by Bellare and Yee [22]. An FS-PRG can be obtained cheaply and generically from any PRG using the constructions of Bellare and Yee [22]; moreover a PRG can be easily obtained from a PRF. Thus our second scheme can be instantiated in a variety of ways.

We also provide a comparison of instantations of our new constructions with a variety of proven-secure KAS from the literature. Details can be found in Section 6.

2 Hierarchical Key Assignment Schemes

2.1 Basic Definitions

A partially ordered set (poset) is a pair $(V, \leq)$ where V is a finite set of pairwise disjoint classes, called *security classes*, and '$\leq$' is a partial order on V, i.e. is a reflexive, antisymmetric, and transitive binary relation. A security class can represent a person, a department, or a user group in an organisation. Relation $\leq$ is defined in accordance with authority for each class in V: for any two classes $u, v \in V$ we write $v \leq u$ or $u \geq v$ to indicate that users in class u can access the data of users in class v. We say that u *covers* v, denoted $v \lessdot u$ or $u \gtrdot v$, if $v < u$ and there does not exist $c \in V$ such that $v < c < u$. $(V, \leq)$ is a *totally ordered set* (or *chain*) if for all $u, v \in V$, either $v < u$ or $u > v$ or $u = v$. We say that $A \subseteq V$ is an *antichain* in V if for all $u, v \in A, u \neq v$, we have $v \not\leq u$ and $v \not\geq u$. Any poset $(V, \leq)$ can be represented by a specific directed acyclic graph $G = (V, E)$, called *access graph*, where the vertices coincide with the security classes and there is an edge from class u to class v if and only if $u > v$. A *partition* of set V is a collection of sets $\{V_1, \ldots, V_s\}$ such that (i) $V_i \subseteq V \, \forall i$, (ii) $V_1 \cup \ldots \cup V_s = V$, and (iii) $i \neq j \Rightarrow V_i \cap V_j = \emptyset$.

The problem that we address consists of assigning keys (e.g., to be used in a symmetric encryption scheme) to each class in a poset in such a way that it should be possible to efficiently derive the keys for any descendant class in the poset. The cryptographic primitive that solves this challenge is called a hierarchical key assignment scheme [1], and is defined as follows.

Definition 1 (Key Assignment Scheme). *Let Γ denote a set of access graphs, i.e. of graphs that correspond to posets. A* hierarchical key assignment scheme (KAS) *for Γ is a pair of algorithms* $(\texttt{Gen}, \texttt{Derive})$ *satisfying the following conditions:*

1. $\texttt{Gen}(1^\rho, G)$ *is a probabilistic polynomial-time algorithm that takes as input a security parameter 1^ρ and a graph $G = (V, E) \in \Gamma$ and outputs*
   (a) *for all classes $u \in V$: private information S_u and key $k_u \in \{0,1\}^{p(\rho)}$, for a fixed polynomial p;*

(b) public information pub.

We denote by (S, k, pub) *the output of* $\texttt{Gen}(1^\rho, G)$*, where* $S = (S_u)_{u \in V}$ *and* $k = (k_u)_{u \in V}$ *are the vectors of private information and keys, respectively.*

2. $\texttt{Derive}(G, u, v, S_u, pub)$ *is a deterministic polynomial-time algorithm that takes as input a graph* G*, classes* $u, v \in V$ *such that* $v \leq u$*, private information* S_u*, and public information pub, and outputs a key* $k \in \{0,1\}^{p(\rho)}$ *assigned to class* v*.*

For correctness we require that for all $\rho \in \mathbb{N}$, all $G \in \Gamma$, all (S, k, pub) output by $\texttt{Gen}(1^\rho, G)$, and all $u, v \in V, v \leq u$, we have $\texttt{Derive}(G, u, v, S_u, pub) = k_v$.

Remark 1. Observe that key assignment schemes are essentially symmetric in nature, i.e. a separation of entities holding secret keys and entities holding public keys is not assumed. As a consequence, Gen's output *pub* does not have to be public in the classical sense, but could also be folded into private information S_u, for all $u \in V$. We point out that we left *pub* in Definition 1 to keep it consistent with prior work. However, the schemes that we propose in this paper will not make use of *pub*, i.e. will assign an empty value to it.

2.2 Security of Key Assignment Schemes

Various informal security models for key assignment schemes have been developed and proposed in the past. Formal security modelling began with [8]. However, as we will argue, all these models – both formal and informal – are inadequate for practical application in the most challenging of security environments. In the following, we first describe our new, strengthened models, and then discuss the differences to the established ones.

We consider variants of the key indistinguishability (KI) security goal proposed by Atallah *et al.* in [8]. We consider models with both static and dynamic adversaries. It will shortly become clear, however, that these two models are polynomially equivalent. We begin with an informal statement of our security models, and then give a formal model in terms of a security experiment involving an adversary.

Static adversaries $\mathcal{A}_{stat}$, upon given an access graph $G = (V, E)$, first choose a security class $u \in V$ to attack. Using Gen algorithm on graph G, the experiment generates (S, k, pub). The adversary is then provided with private information S_v assigned to all classes $v \in V$ that should *not* enable the computation of key k_u, along with the set of all keys k_v associated to classes $v \in V$ such that $v > u$, and the public information *pub*. Precisely, the adversary gets *pub* and the two sets $Corrupt_{G,S,u}$ and $Keys_{G,u}$, where we define

$$Corrupt_{G,S,u} = \{S_v \in S \mid u \not\leq v\} \qquad \text{and} \qquad Keys_{G,u} = \{k_v \mid v > u\} \ .$$

Notice that, given $Corrupt_{G,S,u}$, the adversary can compute for himself all keys k_v for $v \in Corrupt_{G,S,u}$. As a challenge, the adversary additionally gets either key k_u or a random string of the same length, and it has to distinguish these two cases. We refer to Definition 2 below for the formal specification of this experiment. Observe that, from the obtained information, the adversary can gain access to k_v for any $v \in V \setminus \{u\}$.

In contrast to static adversaries, dynamic (also called adaptive) adversaries $\mathcal{A}_{dyn}$ may request keys k_v and secret information S_v in an adaptive manner before eventually committing to a security class $u \in V$ they want to attack. After receiving a challenge based on key k_u, they continue to request keys and secret information until terminating and outputting a bit. The adversary wins in the experiment if it successfully distinguishes

the key k_u from random, under the restriction that $u \not\leq v$ for all classes v in the corrupted set and that key k_u has not been requested.

It is not difficult to see that the static and dynamic models are actually polynomially equivalent. Indeed, in the corresponding reduction, the static adversary simply guesses which class will be the subject of the dynamic adversary's query, and aborts if the guess turns out to be incorrect; this reduction succeeds with probability $1/|V|$. A similar proof was used in [9] (and implicitly in [8]). So schemes proven secure against static adversaries are automatically also secure against dynamic adversaries (albeit with a less tight overall security reduction). In the remainder of the paper, we focus on the static case.

We next give our definition for security in the sense of *strong key indistinguishability with respect to static adversaries (S-KI-ST-security)*, formalising the above discussion.

Definition 2 (S-KI-ST). *Let Γ be a set of access graphs and let* $(\texttt{Gen}, \texttt{Derive})$ *be a hierarchical key assignment scheme for Γ. Consider the following experiment (where we assume that adversary $\mathcal{A}$ keeps state between invocations):*

$$
\begin{array}{l}
\text{Experiment } \mathrm{Exp}^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(1^\rho): \\
\quad u \leftarrow \mathcal{A}(1^\rho, G) \\
\quad (S, k, pub) \leftarrow \texttt{Gen}(1^\rho, G) \\
\quad \beta \xleftarrow{r} \{0,1\} \\
\quad \textit{If } \beta = 1 \textit{ then } T \leftarrow k_u \textit{ else } T \xleftarrow{r} \{0,1\}^{p(\rho)} \\
\quad d \leftarrow \mathcal{A}(pub, \mathit{Corrupt}_{G,S,u}, \mathit{Keys}_{G,u}, T) \\
\quad \textit{return } d
\end{array}
$$

For any $G \in \Gamma$, the advantage of $\mathcal{A}$ in the above experiment is defined as

$$Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho) = 2\left|\Pr\left[\mathrm{Exp}^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(1^\rho) = \beta\right] - 1/2\right|.$$

Note that if we write $\mathrm{Exp}^{\mathrm{S-KI-ST},\gamma}_{\mathcal{A},G}(1^\rho)$, $\gamma \in \{0,1\}$, *for the modification of* $\mathrm{Exp}^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(1^\rho)$ *where bit β is fixed to $\beta = \gamma$, we have that*

$$Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho) = \left|\Pr[\mathrm{Exp}^{\mathrm{S-KI-ST},1}_{\mathcal{A},G}(1^\rho) = 1] - \Pr[\mathrm{Exp}^{\mathrm{S-KI-ST},0}_{\mathcal{A},G}(1^\rho) = 1]\right|.$$

The key assignment scheme is said to be secure in the sense of strong key indistinguishability with respect to static adversaries (S-KI-ST-secure) *if $Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho)$ is negligible for every efficient adversary $\mathcal{A}$ and any graph $G \in \Gamma$.*

It will be evident that one can also define an S-KR-ST-security notion, in which the adversary is required to recover the key k_u rather than distinguish it from a random key. Clearly S-KI-ST-security implies S-KR-ST security.

We now explain why our model is stronger than the one introduced by Atallah *et al.* [8] that it is based on. While our S-KI-ST adversary receives both the set $\mathit{Corrupt}_{G,S,u} \subseteq S$ of secret information and the set $\mathit{Keys}_{G,u} \subseteq \{0,1\}^{p(\rho)}$ of computed (symmetric) keys, in the model from [8] the adversary receives only the former set when performing its attack. In the dynamic setting, our strong adversary has access to keys k_v for which $v > u$, where u is the challenge security class, whereas in the dynamic model of [8], the adversary has no access to such keys. Now in a real deployment of a scheme, some of the cryptographic keys k_v used in the scheme may leak, perhaps through cryptanalysis or misuse. In this case, we would like our selected security model to provide the strongest possible guarantees about the security of other keys that have not been leaked. But

note that the previous security model from [8] provides no such guarantees, whereas our model provides the strongest possible guarantee, in that *all* keys k_v with $v > u$ are given to the adversary. Indeed, as the next example makes clear, it is quite feasible that leakage of a key k_v for which $v > u$ can damage the security of the key k_u.

A separating example: Consider a graph (V, E) having linear structure, i.e. $V = \{v_0, \ldots, v_{n-1}\}$ with $v_{i+1} \lessdot v_i$ for all i. Let H be a one-way function, which we model as a random oracle. We select S_{v_0} at random from the domain of H and set $k_{v_i} = S_{v_i}$ and $S_{v_{i+1}} = H(S_{v_i})$ for all i. It is clear how the $\mathtt{Gen}$ and $\mathtt{Derive}$ algorithms should be defined, and that the resulting scheme is correct. It is also easy to see that the scheme is KR-ST-secure in the random oracle model, in the sense of [8]. However, it is also clear that with knowledge of key $k_{v_0} = S_{v_0}$, all keys k_v in the hierarchy can be efficiently determined (including challenge key k_u) and hence the scheme is insecure in the S-KR-ST model. We note that this separation is for key recovery (KR) security notions.

3 Security Analysis of the Chain Partition Construction

We begin by reviewing the Chain Partition Construction for key assignment schemes from [15]. Given a partially ordered set $(V, \leq)$, represented by the directed acyclic graph $P = (V, E)$, Dilworth's Theorem [23] asserts that every partially ordered set $(V, \leq)$ can be partitioned into w chains, where w is the *width* of V, that is, the cardinality of the largest antichain in V. The partition need not be unique. We select a particular partition of V into chains $\{C_0, \ldots, C_{w-1}\}$. The length of C_i is denoted by l_i, for $0 \leq i \leq w-1$. We let $l_{\max}$ denote $\max_i\{l_i\}$. The maximum class of C_i is regarded as the first class in C_i and the minimum class as the last class. Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V, each $u \in V$ belongs to precisely one chain.

Let $C = u_0 \gtrdot \ldots \gtrdot u_m$ be any chain in V. Then any chain of the form $u_j \gtrdot \ldots \gtrdot u_m$, $0 < j \leq m$ is said to be a *suffix* of C. Now, for any $u \in V$, the set $\downarrow u := \{v \in V : v \leq u\}$ has non-empty intersection with one or more chains $C_0, \ldots, C_{w-1}$. It is proved in [15] that the intersection of $\downarrow u$ and the chain C_i is a suffix of C_i or the empty set. Following, [15], this will enable us to define the private information that should be given to a user with label u.

Since $\{C_0, \ldots, C_{w-1}\}$ is a partition of V into chains, $\{\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}\}$ is a disjoint collection of chain suffixes. Additionally, the private information for each class in V should be chosen so that the private information for the j-th class of a chain can be used to compute keys for all lower classes in that chain. Hence, we can see that a user with label u should be given the private information for the maximal classes in the non-empty suffixes $\downarrow u \cap C_0, \ldots, \downarrow u \cap C_{w-1}$. Given $u \in V$, let $\hat{u}_0, \ldots, \hat{u}_{w-1}$ denote these maximal classes, with the convention that $\hat{u}_i = \perp$ if $\downarrow u \cap C_i = \emptyset$. Let u_j^i denote the j-th class in the chain C_i, where $0 \leq j \leq l_i - 1$.

The Chain Partition Construction: Let $(V, \leq)$ be a poset, $P = (V, E)$ the corresponding directed acyclic graph, and ρ a security parameter. Select a chain partition of V into w chains $C_0, \ldots, C_{w-1}$, so that C_i contains classes $u_0^i, u_1^i, \ldots, u_{l_i-1}^i$, with $u_{j+1}^i \lessdot u_j^i$, $0 \leq j < l_i - 1$. Let $l_{\max}$ denote $\max_i\{l_i\}$. Additionally, let $\mathtt{X} = (\mathtt{Gen}_{\mathtt{X}}, \mathtt{Derive}_{\mathtt{X}})$ be a KAS scheme for the set consisting of a single chain of length exactly $l_{\max}$. Then the chain partition scheme $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P) = (\mathtt{Gen}_{\mathtt{CP}}, \mathtt{Derive}_{\mathtt{CP}})$ (relative to the particular partition selected) is defined as follows.

Algorithm $\texttt{Gen}_{\texttt{CP}}(1^\rho, P)$:

1. For $0 \leq i \leq w-1$, run $\texttt{Gen}_{\texttt{X}}$ on inputs 1^ρ and a chain of length $l_{\max}$ to obtain (T^i, k^i, pub_i). Discard the last $l_{\max} - l_i$ elements of T^i and k^i to obtain the secret information and keys for a chain of length l_i. Note that this chain has the same $\texttt{Derive}$ algorithm as the starting chain. For ease of notation, we continue to denote the reduced sets by T^i and k^i, and we write $T^i = \{T_{u_0^i}, \ldots, T_{u_{l_i-1}^i}\}$ and $k^i = \{k_{u_0^i}, \ldots, k_{u_{l_i-1}^i}\}$. We stress here that we could run different algorithms $\texttt{Gen}_{\texttt{X}}$ to produce the different chains of lengths l_i, but for ease of notation we will assume they are all the same.
2. For each $u \in V$, define the private information S_u to be $\{T_{\hat{u}_i} : \hat{u}_i \neq \perp, 0 \leq i \leq w-1\}$ and the encryption key k_u to be $k_u = k_{u_j^i}$, where $u = u_j^i$.
3. Let S and k be the sets of private information and keys, respectively, in the above construction, and let $pub_{\texttt{CP}} = (pub_0, \ldots, pub_{w-1})$.
4. Output $(S, k, pub_{\texttt{CP}})$.

Algorithm $\texttt{Derive}_{\texttt{CP}}(P, u_j^i, u_h^g, S_{u_j^i}, pub_{\texttt{CP}})$:

1. For $u_j^i \geq u_h^g$, find $\hat{u}_g$, the maximal class in $\downarrow u_j^i \cap C_g$. This class is in chain C_g. We denote it by u_r^g, where $0 \leq r < l_g$. Note that, by construction, $u_r^g \leq u_j^i$ and $T_{u_r^g} \in S_{u_j^i}$.
2. Set $k_{u_h^g} \leftarrow \texttt{Derive}_{\texttt{X}}(C_g, u_r^g, u_h^g, T_{u_r^g}, pub_g)$.
3. Output $k_{u_h^g}$.

Theorem 1 (S-KI-ST Security of the Chain Partition Construction). *Let $P = (V, E)$ be a directed acyclic graph and let l_{max} be the maximum length of the chains in a chain partition of V. Let $\texttt{X}$ be an S-KI-ST-secure scheme for the set of graphs consisting of the single chain of length l_{max}. Then the scheme $\texttt{KAS}_{\texttt{CP}}(\texttt{X}, P) = (\texttt{Gen}_{\texttt{CP}}, \texttt{Derive}_{\texttt{CP}})$ for graph P obtained from the above chain partition construction is also S-KI-ST-secure. More precisely, for every S-KI-ST adversary $\mathcal{A}_{\texttt{CP}}$ that breaks the scheme $\texttt{KAS}_{\texttt{CP}}(\texttt{X}, P)$ with advantage $Adv^{\text{S-KI-ST}}_{\mathcal{A}_{\texttt{CP}}, P}(\rho)$, there exists an S-KI-ST adversary $\mathcal{A}_{\texttt{X}}$ that breaks $\texttt{X}$ with the same advantage. Moreover, the two adversaries run in roughly the same time.*

Proof. Assume $\mathcal{A}_{\texttt{CP}}$ attacks a class u_j^i of graph P. If $\mathcal{A}_{\texttt{CP}}$ is able to distinguish between the real key $k_{u_j^i}$ associated with class u_j^i, and a random string having the same length, we show that we can construct an S-KI-ST adversary $\mathcal{A}_{\texttt{X}}$ against the scheme $\texttt{X}$ that, using $\mathcal{A}_{\texttt{CP}}$ as a black box, is able to distinguish between real or random keys. Algorithm $\mathcal{A}_{\texttt{X}}$ plays the S-KI-ST security game described in Definition 2, receiving as initial input a security parameter 1^ρ and a chain on $l_{\max}$ classes. Adversary $\mathcal{A}_{\texttt{X}}$ simulates the environment of $\mathcal{A}_{\texttt{CP}}$ in such a way that $\mathcal{A}_{\texttt{CP}}$'s view is indistinguishable from its view when playing the S-KI-ST security game.

Algorithm $\mathcal{A}_{\texttt{X}}$:

1. Receive from the S-KI-ST experiment a chain C on $l_{\max}$ classes $v_0, \ldots, v_{l_{\max}-1}$.
2. Let $P = (V, E) \in \Gamma$ and run $\mathcal{A}_{\texttt{CP}}$ with input $(1^\rho, P)$ to get $\mathcal{A}_{\texttt{CP}}$'s choice of target class u.
3. Generate a chain partition of P containing chains $C_0, \ldots, C_{w-1}$. In this partition, class u is identified as some class u_j^i in some chain C_i of length $l_i \leq l_{\max}$. For $0 \leq t \leq w-1, t \neq i$, run $\texttt{Gen}$ on inputs 1^ρ and a chain of length $l_{\max}$ to obtain (S^t, k^t, pub_t), the set of secret information, the set of keys and the public information for that chain. Note that, as in the construction, these sets can be truncated to obtain

the set of secret information, the set of keys and the public information for a chain of length exactly l_t. By abuse of notation, we continue to use (S^t, k^t, pub_t) to denote this data.

4. Output v_j in chain C as $\mathcal{A}_{\mathtt{X}}$'s choice of target class. $\mathcal{A}_{\mathtt{X}}$ now receives as input the public information, pub, output by $\mathtt{Gen}_{\mathtt{X}}$, along with secret information S_{v_t} for all classes $v_t < v_j$ in C, and all secret keys k_{v_t} in C such that $v_t > v_j$. $\mathcal{A}_{\mathtt{X}}$ also receives as input a value T which is either the real key k_{v_j} or a random key of the same length. In what follows, $\mathcal{A}_{\mathtt{X}}$ will identify the first l_i classes in C with the chain C_i in the chain partition construction.
5. Set $pub_{\mathtt{CP}} = (pub_0, \ldots, pub_{i-1}, pub, pub_{i+1}, \ldots, pub_{w-1})$. Use the secret information S_{v_t} for classes $v_t < v_j$ in C together with the secret information in the sets S^t for $0 \leq t \leq w-1, t \neq i$ to build the set $Corrupt_{P,S,u}$. Use keys k_{v_t} in C such that $v_t > v_j$ and the keys from the sets k^t to build the set $Keys_{P,u}$.
6. Run $\mathcal{A}_{\mathtt{CP}}$ with inputs $(pub_{\mathtt{CP}}, Corrupt_{P,S,u}, Keys_{P,u}, T)$. It is easy to see that $\mathcal{A}_{\mathtt{X}}$ has the information required to properly construct the sets $Corrupt_{P,S,u}, Keys_{P,u}$ in such a way that $\mathcal{A}_{\mathtt{CP}}$'s input here is valid in $\mathcal{A}_{\mathtt{CP}}$'s experiment against the scheme $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P)$, and such that T is the real key (resp. the random key) in $\mathcal{A}_{\mathtt{CP}}$'s experiment if and only if T is the real key (resp. the random key) in $\mathcal{A}_{\mathtt{X}}$'s experiment.
7. When $\mathcal{A}_{\mathtt{CP}}$ outputs a bit, output the same bit.

Now as $\mathcal{A}_{\mathtt{X}}$'s simulation is perfect, we see that the advantage of $\mathcal{A}_{\mathtt{X}}$ in winning its S-KI-ST indistinguishability game for the chain C of length $l_{\max}$ is the same as the advantage of $\mathcal{A}_{\mathtt{CP}}$ in playing the S-KI-ST indistinguishability game against $\mathtt{KAS}_{\mathtt{CP}}(\mathtt{X}, P)$. The theorem now follows. □

Note that, in the above theorem, $\mathtt{X}$ need only be an S-KI-ST-secure scheme for chains of length exactly $l_{\max}$. Because of the truncation trick, this is equivalent to $\mathtt{X}$ being an S-KI-ST-secure scheme for the set of graphs consisting of chains of lengths up to $l_{\max}$.

4 A Scheme based on PRFs

In this section we construct an S-KI-ST-secure key assignment scheme for totally-ordered hierachical access structures of arbitrary depth, based on pseudorandom functions. By combining our construction with the result from Section 3, a general key assignment scheme for arbitrary posets is obtained.

We admit that also Atallah *et al.* [8] give an efficient PRF-based construction for arbitrary posets. However, their construction achieves only a security notion called 'key recovery' (where an adversary attacking a class u has to compute the challenge key k_u, instead of distinguishing it from a random key), which is weaker than our S-KI-ST security notion. Moreover, our scheme is much simpler, and requires no public information to be stored.

We start by recalling the definition of a PRF, the central building block of our construction:

Definition 3 (Pseudorandom Function, PRF). *Let $\mathcal{K}, \mathcal{D}, \mathcal{R}$ be finite sets*[1] *and $F : \mathcal{K} \times \mathcal{D} \to \mathcal{R}$ be an efficient function. For all $\kappa \in \mathcal{K}$ and $x \in \mathcal{D}$ we also write $F_\kappa(x) = F(\kappa, x)$ and call $F_\kappa : \mathcal{D} \to \mathcal{R}$ an* instance *of F. We call $\mathcal{K}$ the set of* keys*, while $\mathcal{D}$*

[1] More precisely, we assume that $\mathcal{K}, \mathcal{D}, \mathcal{R}$ are families of finite sets, indexed by a security parameter ρ. That is, we require $\mathcal{K} = (\mathcal{K}_\rho)_{\rho \in \mathbb{N}}$, and similarly for $\mathcal{D}$ and $\mathcal{R}$. For the sake of a cleaner exposition, however, we do not write down the security parameter explicitly.

and $\mathcal{R}$ *are the* domain *and* range *of* F*, respectively. Function* F *is* pseudorandom *if the input/output behaviour of a random instance* F_κ *is computationally indistinguishable from that of a random function* $\mathcal{D} \to \mathcal{R}$*.*

More formally, let $\texttt{Rand} = \mathcal{R}^{\mathcal{D}} = \{g \,|\, g : \mathcal{D} \to \mathcal{R}\}$ *denote the set of all functions* $\mathcal{D} \to \mathcal{R}$*. Let* $\mathcal{A}_F$ *be an algorithm that has oracle access to a function* $\mathcal{D} \to \mathcal{R}$*, and returns a bit. In our security model, this function is either drawn at random from* $\texttt{Rand}$*, or is* F_κ *for a random* $\kappa \xleftarrow{r} \mathcal{K}$*, and* $\mathcal{A}_F$ *has to distinguish these cases. Consider the following two experiments:*

Experiment $\mathrm{Exp}^{\mathrm{PRF}-1}_{\mathcal{A}_F,F}(1^\rho)$:	Experiment $\mathrm{Exp}^{\mathrm{PRF}-0}_{\mathcal{A}_F,F}(1^\rho)$:
$\kappa \xleftarrow{r} \mathcal{K}$	$g \xleftarrow{r} \texttt{Rand}$
$d \leftarrow \mathcal{A}_F^{F_\kappa}(1^\rho)$	$d \leftarrow \mathcal{A}_F^{g}(1^\rho)$
return d	*return* d

The advantage of $\mathcal{A}_F$ *is defined as*

$$Adv^{\mathrm{PRF}}_{\mathcal{A}_F,F}(\rho) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{PRF}-1}_{\mathcal{A}_F,F}(1^\rho) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{PRF}-0}_{\mathcal{A}_F,F}(1^\rho) = 1\right]\right|.$$

We say that F *is* pseudorandom *(or: is a PRF) if* $Adv^{\mathrm{PRF}}_{\mathcal{A}_F,F}(\rho)$ *is negligible for every efficient adversary* $\mathcal{A}_F$*.*

In our following construction, we will use special PRFs where $\mathcal{K} = \mathcal{R} = \{0,1\}^\rho$ for security parameter ρ, and $\mathcal{D}$ is any set. We remark that some constructions in [8] also require PRFs with similar restrictions on keyspace and range. For concreteness, we propose to deploy the (hash-based) HMAC primitive [19] as a PRF (see also analysis in [24]). In addition, it might be possible to find suitable constructions based on number-theoretic assumptions, e.g. derived from the PRF obtained by converting the BBS [20] PRG into a PRF via the Goldreich-Goldwasser-Micali (GGM) construction [25].

4.1 A PRF-based Key Assignment Scheme for Totally Ordered Hierarchies

We briefly recall the setting of key assignment for chains. Let Γ be the family of graphs corresponding to totally ordered hierarchies, and let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \lessdot u_i$ for all i. To each security class $u_i \in V$, private information S_i and key k_i are assigned, where S_i can be used to compute subordinated keys. Note that here we abuse notation, for better exposition (we can do this because we are in the linear setting), writing S_i for S_{u_i} and k_i for k_{u_i}.

Let ρ be a security parameter and let $F : \{0,1\}^\rho \times \mathcal{D} \to \{0,1\}^\rho$ be a PRF. Let c_0 and c_1 be two different elements in $\mathcal{D}$. The `Gen` and `Derive` algorithms work as follows.

Algorithm $\texttt{Gen}(1^\rho, G)$:

1. Pick random $S_0 \xleftarrow{r} \{0,1\}^\rho$ and set $k_0 \leftarrow F_{S_0}(c_1)$.
2. For each class $u_i \in V, i > 0$, set $S_i \leftarrow F_{S_{i-1}}(c_0)$ and $k_i \leftarrow F_{S_i}(c_1)$.
3. Set $S \leftarrow (S_0, \ldots, S_{n-1})$, $k \leftarrow (k_0, \ldots, k_{n-1})$, and $pub \leftarrow \emptyset$.
4. Output (S, k, pub).

Algorithm $\texttt{Derive}(G, u_i, u_j, S_i, pub)$: (note that we may assume $j \geq i$)

1. If $i = j$ then return $k_j = F_{S_i}(c_1)$.
2. For $h = i + 1$ to j:

$S_h \leftarrow F_{S_{h-1}}(c_0)$.

3. Return $k_j = F_{S_j}(c_1)$.

Observe that computing key k_j from secret information S_i requires exactly $j - i + 1$ evaluations of the underlying PRF.

Theorem 2 (S-KI-ST Security of the PRF-based Scheme for Totally Ordered Hierarchies). *The above PRF-based scheme is key indistinguishable, in the sense of Definition 2, for any totally ordered graph G, assuming security of pseudorandom function F.*

Proof. Fix any totally ordered graph $G = (V, E) \in \Gamma$. Proving the theorem amounts to showing that the only way to break the key indistinguishability of our PRF-based scheme, in the sense of Definition 2, is by breaking the pseudorandom function F. To this aim, we need to show how to turn an S-KI-ST adversary $\mathcal{A}$ attacking our KAS scheme into an adversary $\mathcal{A}_F$ attacking F. Assume $\mathcal{A}$ attacks a class $u_i \in V$. We define a sequence of computationally indistinguishable games Game 0, Game 1, ..., Game $i+1$. Game 0 is the actual adversarial game (as defined in Definition 2), and in the last game, Game $i + 1$, the probability that $\mathcal{A}$ is successful in outputting the correct bit β is only $1/2$. Let Succ_ι be the event that $\mathcal{A}$ is successful in Game ι, i.e. that $d = \beta$ in the experiment.

Game 0. Let Game 0 be the original attack game as described in Definition 2.

Game ι ($1 \leq \iota \leq i + 1$). This game is identical to Game $\iota - 1$, except that here the assignment of private information and keys is modified in such a way that key $k_{\iota-1}$ and private information S_ι are substituted with values randomly selected from $\{0,1\}^\rho$. This modification amounts to substituting the occurrences of $F_{S_{\iota-1}}$ in Game $\iota - 1$ with a truly random function, which is warranted by the security of the PRF, as we will see in Lemma 1. Hence,

$$|\Pr[\text{Succ}_\iota] - \Pr[\text{Succ}_{\iota-1}]| \leq \epsilon_{\text{PRF}}(\rho) \ ,$$

where ϵ_{PRF} is a bound on the advantage $Adv^{\text{PRF}}_{\mathcal{A}_F,F}(\rho)$ for any polynomial-time adversary $\mathcal{A}_F$.

Now, we see that in Game $i+1$ the adversary's view is independent of bit β: in both cases it gets as challenge a random value in $\{0,1\}^\rho$. Thus,

$$\Pr[\text{Succ}_{i+1}] = 1/2 \ .$$

Therefore, we have

$$Adv^{\text{S-KI-ST}}_{\mathcal{A},G}(\rho) = 2\,|\Pr[\text{Succ}_0] - 1/2| \leq 2(i+1)\epsilon_{\text{PRF}}(\rho) \ .$$

As, by assumption, $\epsilon_{\text{PRF}}(\rho)$ can be chosen to be negligible, and this completes the proof. □

Lemma 1. $|\Pr[Succ_\iota] - \Pr[Succ_{\iota-1}]| \leq \epsilon_{\text{PRF}}(\rho)$.

Proof. Assume we have an S-KI-ST adversary $\mathcal{A}$ against our PRF-based scheme for totally ordered sets that attacks a class u_i and is able to distinguish between Game $\iota - 1$

and Game ι. We describe below how to construct an algorithm $\mathcal{A}_F$ that, using $\mathcal{A}$ as a black-box, is able to distinguish between pseudorandom and truly random functions.

Algorithm $\mathcal{A}_F$ plays the PRF game described in Definition 3 and is thus given access to a function $g(\cdot)$ that is either an instance of a pseudorandom function, keyed with a key κ, or a truly random function. In order to use algorithm $\mathcal{A}$, $\mathcal{A}_F$ simulates the environment of $\mathcal{A}$ in such a way that interpolates between Game $\iota - 1$ and Game ι. This means that if $\mathcal{A}_F$ is interacting with a pseudorandom function, then the simulation proceeds as in Game $\iota - 1$. Otherwise, if $\mathcal{A}_F$ is interacting with a random function, then the simulation proceeds as in Game ι. More formally, algorithm $\mathcal{A}_F$ works as follows.

Algorithm $\mathcal{A}_F$:

1. Run $\mathcal{A}$ with input $(1^\rho, G)$ to get $\mathcal{A}$'s choice of target class u_i.
2. Set up the access hierarchy for graph G by running $\texttt{Gen}(1^\rho, G)$ with the following modifications:
   (a) Private information S_ι and key $k_{\iota-1}$ are computed via oracle g as follows:
   $$S_\iota \leftarrow g(c_0),\ k_{\iota-1} \leftarrow g(c_1).$$
   (b) For all $\iota' < \iota$, set $S_{\iota'} \xleftarrow{r} \{0,1\}^\rho$.
   (c) For all $\iota' < \iota - 1$, set $k_{\iota'} \xleftarrow{r} \{0,1\}^\rho$.
   So far, this is almost equivalent to the actions of Game $\iota - 1$ when $\mathcal{A}_F$'s oracle computes g according to a pseudorandom function, and equivalent to Game ι when it computes g completely at random. The only difference is that, in case g is a PRF, the private information $S_{\iota-1}$ is not the same as g's internal key (unknown to $\mathcal{A}_F$). This is completely indistinguishable from $\mathcal{A}$'s view since $\mathcal{A}$ is not allowed to ask for private information $S_{i'}$, $i' < i+1$, and thus as $\iota \leq i+1$, $\mathcal{A}$ is not allowed to ask for $S_{\iota'}$, $\iota' < \iota$.
3. Pick a random bit $\beta \in \{0,1\}$. If $\beta = 1$, $\mathcal{A}_F$ sets $\mathcal{A}$'s challenge, denoted here by T, to be the real key k_i. If $\beta = 0$, $\mathcal{A}_F$ sets T to be a random key of the same length of k_i.
4. Run $\mathcal{A}$ with inputs $(pub, Corrupt_{G,S,u_i}, Keys_{G,u_i}, T)$, where $Corrupt_{G,S,u_i} = \{S_{i+1}, \ldots, S_{n-1}\}$ and $Keys_{G,u_i} = \{k_0, \ldots, k_{i-1}\}$, to obtain a bit d. (Note that according to $\mathcal{A}_F$'s setup of the access hierarchy, it can compute all this information.) Here d is $\mathcal{A}$'s guess as to whether it was given the real key associated with class u_i or a random string having the same length.
5. If $d = \beta$, output 1, guessing for a pseudorandom function; otherwise, output 0, guessing for a truly random function.

Now we have

$$\begin{aligned}
\epsilon_{\mathrm{PRF}}(\rho) &\geq Adv^{\mathrm{PRF}}_{\mathcal{A}_F,F}(\rho) \\
&= \left|\Pr[\mathrm{Exp}^{\mathrm{PRF}-1}_{\mathcal{A}_F,F}(1^\rho) = 1] - \Pr[\mathrm{Exp}^{\mathrm{PRF}-0}_{\mathcal{A}_F,F}(1^\rho) = 1]\right| \\
&= |\Pr[\mathcal{A}_F \text{ outputs } 1 \mid g \text{ is a PRF}] - \Pr[\mathcal{A}_F \text{ outputs } 1 \mid g \text{ is random}]| \\
&= |\Pr[\mathcal{A} \text{ guesses } \beta \text{ correctly} \mid g \text{ is a PRF}] - \Pr[\mathcal{A} \text{ guesses } \beta \text{ correctly} \mid g \text{ is random}]| \\
&= |\Pr[\mathrm{Succ}_{\iota-1}] - \Pr[\mathrm{Succ}_\iota]|\,.
\end{aligned}$$

□

5 A Scheme based on Forward-Secure PRGs

FS-PRGs, introduced by Bellare and Yee in [22], are stateful/iterated pseudorandom generators (PRGs) that deterministically derive sequences of fixed-length bit strings from an initial (random) seed. More precisely, in each iteration they output a string of bits, update their internal state, and securely erase the old state. Like in regular PRGs, the output sequences are required to be indistinguishable from sequences of random strings. The pivotal property of FS-PRGs is *forward security*, i.e. the adversary has the ability to eventually corrupt generator's internal state, but indistinguishability of output strings is guaranteed to still hold up to that point. Different constructions of FS-PRGs are proposed in [22], including highly efficient ones based on symmetric building blocks like blockciphers or HMAC [19], and also a construction based on a number-theoretic assumption (specifically, on the Blum-Blum-Shub PRG [20]).

In this section, building on generic FS-PRGs, we construct a key assignment scheme which achieves S-KI-ST security for totally-ordered access graphs and, in combination with the results from Section 3, for arbitrary posets. It is worth pointing out that we actually widely generalize the construction from [16], which implicitly exploits the property of forward security of the BBS pseudorandom generator. As our construction generically builds on FS-PRGs, it is amenable to the efficiency gain obtained by replacing the BBS-based FS-PRG by, for instance, an HMAC-based one.

Before describing our scheme, let us first recall the definition and security notion of forward-secure pseudorandom number generators (FS-PRGs). Observe that we slightly weaken the model from [22] (considering static adversaries instead of adaptive ones), what renders our construction of a key assignment scheme more general. Clearly the FS-PRG constructions proposed and proved secure in [22] naturally remain secure in our adapted model.

Definition 4 (Forward-Secure PRG). *Let* $G_{\mathtt{FS}} = (G_{\mathtt{FS}}.\mathtt{setup}, G_{\mathtt{FS}}.\mathtt{key}, G_{\mathtt{FS}}.\mathtt{next})$ *be a set of efficient algorithms such that* $G_{\mathtt{FS}}.\mathtt{setup}$ *is a probabilistic algorithm that, on input a security parameter* 1^ρ*, outputs a set of system parameters 'params';* $G_{\mathtt{FS}}.\mathtt{key}$ *is a probabilistic key generation algorithm that takes 'params' as input and outputs an initial state* $St_0 \in \{0,1\}^\rho$ *(the initial seed);* $G_{\mathtt{FS}}.\mathtt{next} : \{0,1\}^\rho \to \{0,1\}^\rho \times \{0,1\}^{p(\rho)}$ *is a deterministic algorithm that turns state* $St_{i-1} \in \{0,1\}^\rho$ *(the 'seed' at iteration* i*) into a pair* (St_i, Out_i)*, where* $St_i \in \{0,1\}^\rho$ *is the updated state, and* Out_i *is a* $p(\rho)$*-bit string.*

Let $\mathcal{D}$ *be an adversary against* $G_{\mathtt{FS}}$*.* $\mathcal{D}$ *is fed with a number of output blocks,* Out_1*,* $Out_2, \ldots, Out_i$*, each of length* $p(\rho)$*, and is given the then current state of the generator,* St_i*.* $\mathcal{D}$*'s job is to decide whether these blocks are the real output of the generator, or just a sequence of random bits. To formalize this, we consider the following experiments.*

The advantage of $\mathcal{D}$ *is defined as*

$$Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\mathtt{FS}}}(\rho) = \left|\Pr\left[\mathrm{Exp}^{\mathrm{FS-PRG-1}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{FS-PRG-0}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho) = 1\right]\right| \ .$$

We say that $G_{\mathtt{FS}}$ *is a forward-secure pseudorandom number generator (FS-PRG) if* $Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\mathtt{FS}}}(\rho)$ *is negligible for every efficient adversary* $\mathcal{D}$*.*

Experiment $\mathrm{Exp}^{\mathrm{FS-PRG-1}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho)$:
  $i \leftarrow \mathcal{D}$
  $params \xleftarrow{r} G_{\mathtt{FS}}.\mathtt{setup}(1^\rho)$
  $St_0 \xleftarrow{r} G_{\mathtt{FS}}.\mathtt{key}(params)$
  $i' \leftarrow 0$
  Repeat
    $i' \leftarrow i' + 1$
    $(St_{i'}, Out_{i'}) \leftarrow G_{\mathtt{FS}}.\mathtt{next}(St_{i'-1})$
  Until $i' = i$
  $Out \leftarrow Out_1, Out_2, \ldots, Out_i$
  $d \leftarrow \mathcal{D}(St_i, Out)$
  return d

Experiment $\mathrm{Exp}^{\mathrm{FS-PRG-0}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho)$:
  $i \leftarrow \mathcal{D}$
  $params \xleftarrow{r} G_{\mathtt{FS}}.\mathtt{setup}(1^\rho)$
  $St_0 \xleftarrow{r} G_{\mathtt{FS}}.\mathtt{key}(params)$
  $i' \leftarrow 0$
  Repeat
    $i' \leftarrow i' + 1$
    $(St_{i'}, Out_{i'}) \leftarrow G_{\mathtt{FS}}.\mathtt{next}(St_{i'-1})$
    $Out_{i'} \xleftarrow{r} \{0,1\}^{p(\rho)}$
  Until $i' = i$
  $Out \leftarrow Out_1, Out_2, \ldots, Out_i$
  $d \leftarrow \mathcal{D}(St_i, Out)$
  return d

5.1 The FS-PRG-based Scheme for a Single Chain

Key assignment schemes for totally-ordered access graphs are readily constructed from FS-PRGs: In our construction, we identify the FS-PRG's state St_i with the private information S_i stored for class u_i, while key k_i is set to the FS-PRG's output Out_{i+1}.

More precisely, let Γ be the family of graphs corresponding to totally ordered hierarchies, let $G = (V, E) \in \Gamma$ be a graph, where $V = \{u_0, \ldots, u_{n-1}\}$ for some n, and $u_{i+1} \preccurlyeq u_i$ for all i. As in Section 4.1, we write S_i for private information S_{u_i}, and k_i for key k_{u_i}. Let ρ be a security parameter, and let $G_{\mathtt{FS}} = (G_{\mathtt{FS}}.\mathtt{setup}, G_{\mathtt{FS}}.\mathtt{key}, G_{\mathtt{FS}}.\mathtt{next})$ be an FS-PRG. Then `Gen` and `Derive` algorithms work as follows.

Algorithm $\mathtt{Gen}(1^\rho, G)$:

1. Run $params \leftarrow G_{\mathtt{FS}}.\mathtt{setup}(1^\rho)$ and $S_0 \leftarrow G_{\mathtt{FS}}.\mathtt{key}(params)$;
2. For all $0 \le i < n$:
   Compute $(S_{i+1}, k_i) \leftarrow G_{\mathtt{FS}}.\mathtt{next}(S_i)$;
3. Set $S \leftarrow (S_0, \ldots, S_{n-1})$, $k \leftarrow (k_0, \ldots, k_{n-1})$, and $pub \leftarrow \emptyset$;
4. Output (S, k, pub).

Algorithm $\mathtt{Derive}(G, u_i, u_j, S_i, pub)$: (note that we may assume $j \ge i$)

1. For $h = i$ to j:
   $(S_{h+1}, k_h) \leftarrow G_{\mathtt{FS}}.\mathtt{next}(S_h)$;
2. Return k_j.

Theorem 3 (S-KI-ST Security of the FS-PRG-based Scheme for Totally Ordered Hierarchies). *The above FS-PRG-based scheme is key indistinguishable, in the sense of Definition 2, for any totally ordered graph G, assuming security of the FS-PRG, $G_{\mathtt{FS}}$.*

Proof. Fix any totally ordered graph $G = (V, E) \in \Gamma$. This proof amounts to showing that for every S-KI-ST adversary $\mathcal{A}$ that breaks our FS-PRG-based scheme with advantage $Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho)$, there exists a PPT algorithm $\mathcal{D}$ that breaks the security of the FS-PRG, $G_{\mathtt{FS}}$, with advantage $Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\mathtt{FS}}}(\rho) \ge Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho)/2$. Assuming we have an S-KI-ST adversary $\mathcal{A}$ that attacks a class u_i and is able to distinguish between the

real key k_i and a random string of the same length, we can construct an algorithm $\mathcal{D}$ that, using $\mathcal{A}$ as a black box, is able to tell apart output blocks generated by $G_{\mathtt{FS}}$, up to some iteration t, from a sequence of purely random blocks.

Algorithm $\mathcal{D}$ has to distinguish the two experiments described in Definition 4. For this, it initially outputs an iteration number, say t, and then is given access to a sequence of t output blocks $Out_1, \ldots, Out_t$, that are either random or the first t output blocks of $G_{\mathtt{FS}}$. $\mathcal{D}$ is also given the current state, St_t, of the FS-PRG. Given all that information, $\mathcal{D}$ simulates the environment of $\mathcal{A}$ in a way that $\mathcal{A}$'s view is indistinguishable from its view when playing the S-KI-ST game described in Definition 2. We need to prove that

$$Adv^{\mathrm{S-KI-ST}}_{\mathcal{A},G}(\rho) = \left|\Pr[\mathrm{Exp}^{\mathrm{S-KI-ST},1}_{\mathcal{A},G}(1^\rho) = 1] - \Pr[\mathrm{Exp}^{\mathrm{S-KI-ST},0}_{\mathcal{A},G}(1^\rho) = 1]\right| \leq 2\epsilon_{\mathrm{FS-PRG}}(\rho) \ ,$$

where $\epsilon_{\mathrm{FS-PRG}}$ is an upper bound on the advantage $Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\mathtt{FS}}}(\rho)$ for any polynomial-time distinguisher $\mathcal{D}$. We define a sequence of games and then analyse it.

Game 0. Define Game 0 to be identical to $\mathrm{Exp}^{\mathrm{S-KI-ST},0}_{\mathcal{A},G}(1^\rho)$. In particular, challenge key T is chosen to be random in $\{0,1\}^{p(\rho)}$.

Game 1. This game is like Game 0, except that all elements in $Keys_{G,u}$ are replaced by random strings. Challenge key T remains as before, i.e. random.

Game 2. This game is identical to $\mathrm{Exp}^{\mathrm{S-KI-ST},1}_{\mathcal{A},G}(1^\rho)$. In particular, challenge key T is the real key, as computed via `Derive` algorithm.

In the analysis, let Succ_ι be the event that $\mathcal{A}$ outputs 1 in Game ι. First we show that $|\Pr[\mathrm{Succ}_0] - \Pr[\mathrm{Succ}_1]| \leq \epsilon_{\mathrm{FS-PRG}}(\rho)$. Consider the following distinguisher $\mathcal{D}$ against the FS-PRG $G_{\mathtt{FS}}$: It runs $\mathcal{A}$ with input $(1^\rho, G)$ to get u_i, the class that $\mathcal{A}$ is aiming to attack. $\mathcal{D}$ chooses an integer $t = i$ and sends it to its own challenger, receiving St_i and $Out = Out_1, \ldots, Out_i$, where the latter is either the honest output generated by $G_{\mathtt{FS}}.\mathtt{next}$, or a collection of random strings in $\{0,1\}^{p(\rho)}$. $\mathcal{D}$ sets $Keys_{G,u_i} = Out_1, \ldots, Out_i$ and uses St_i to compute the private information collected in $Corrupt_{G,S,u_i}$. $\mathcal{D}$ sets $Corrupt_{G,S,u_i} = \{St_{i+1}, \ldots, St_{n-1}\}$ and $pub \leftarrow \emptyset$. Finally, $\mathcal{D}$ executes $\mathcal{A}$ on input $(pub, Keys_{G,u_i}, Corrupt_{G,S,u_i}, T)$, where like in Games 1 and 2, T is chosen to be random in $\{0,1\}^{p(\rho)}$. Whenever $\mathcal{D}$ receives a bit d from $\mathcal{A}$, $\mathcal{D}$ forwards the same bit to its own challenger. We see that if Out is the sequence of outputs generated by $G_{\mathtt{FS}}.\mathtt{next}$, then $\mathcal{A}$'s view is exactly as in Game 0. On the other hand, if the values are random strings, then $\mathcal{A}$'s view is the one from Game 1. Now we have

$$\begin{aligned}
\epsilon_{\mathrm{FS-PRG}}(\rho) &\geq Adv^{\mathrm{FS-PRG}}_{\mathcal{D},G_{\mathtt{FS}}}(\rho) \\
&= \left|\Pr\left[\mathrm{Exp}^{\mathrm{FS-PRG-1}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho) = 1\right] - \Pr\left[\mathrm{Exp}^{\mathrm{FS-PRG-0}}_{\mathcal{D},G_{\mathtt{FS}}}(1^\rho) = 1\right]\right| \\
&= |\Pr[\mathrm{Succ}_0] - \Pr[\mathrm{Succ}_1]| \, .
\end{aligned}$$

Next we bound $|\Pr[\mathrm{Succ}_1] - \Pr[\mathrm{Succ}_2]|$. The reduction is similar to the one above, the difference is that this time the FS-PRG distinguisher $\mathcal{D}$ specifies $t = i + 1$ instead of $t = i$, and assigns $k_j \leftarrow Out_{j+1}$ for all $0 \leq j \leq i$. This means that not only the keys in $Keys_{G,u_i}$ are taken from Out, but also the 'challenge key' k_i. Notice that here, $\mathcal{D}$ has access to the value St_{i+1} and thus is able to compute the set of private information

$Corrupt_{G,S,u_i} = \{St_{i+1}, \ldots, St_{n-1}\}$. Similarly to above, this establishes $|\Pr[\text{Succ}_1] - \Pr[\text{Succ}_2]| \leq \epsilon_{\text{FS-PRG}}(\rho)$.

All in all, this proves that

$$\begin{aligned} Adv^{\text{S-KI-ST}}_{\mathcal{A},G}(\rho) &= \left|\Pr[\text{Exp}^{\text{S-KI-ST},1}_{\mathcal{A},G}(1^\rho) = 1] - \Pr[\text{Exp}^{\text{S-KI-ST},0}_{\mathcal{A},G}(1^\rho) = 1]\right| \\ &= |\Pr[\text{Succ}_2] - \Pr[\text{Succ}_0]| \\ &\leq 2\epsilon_{\text{FS-PRG}}(\rho), \end{aligned}$$

as required. □

6 Comparison with Previous Schemes

In this section, we compare instantiations of our constructions with other provably secure schemes from the literature. Note that all these previous schemes have proofs only in weaker models than our strong models. (However, in some cases, these schemes can also be proven secure in our strong models.)

In [8, 13], Atallah *et al.* proposed a first construction based on pseudorandom functions, which they prove to be KR-secure; and a second construction which they prove to be KI-secure, but which requires the additional use of a symmetric encryption scheme secure under chosen plaintext attack (SE-CPA). In both constructions the private information of a class consists of a single symmetric key associated with that class. In the first construction, the amount of public information grows with the number of edges in the directed acyclic graph (each edge has an associated PRF output). In the second construction, the amount of public information grows with the number of classes (each class has an associated ciphertext). In both constructions, key derivation uses only symmetric operations and its cost grows linearly with the number of levels between the classes.

De Santis *et al.* [12] proposed a construction which is based on symmetric encryption schemes only, achieves KI security and is simpler than the KI-secure scheme proposed in [8]. The construction uses only a chosen plaintext secure symmetric encryption scheme and the required private key storage is small at one key per class. The amount of public storage needed grows linearly with the number of classes and the number of edges in the graph. Key derivation requires roughly h symmetric decryptions, where h is the number of levels between the classes.

Ateniese *et al.* [9,17] proposed two different constructions for time-bound key assignment schemes, which achieve KI security. Their first construction is based on symmetric encryption schemes and the second makes use of bilinear maps. The security of the latter construction is based on the Bilinear Decisional Diffie-Hellman (BDDH) assumption. The advantage of these constructions is that they provide very efficient procedures for key derivation, requiring only one decryption or one pairing evaluation, no matter the number of levels between the classes. However, the public information for a scheme obtained from the first construction can be very large since it depends not only on the number of classes, but also on the number of time periods. The downside of the second construction is that the private information could be as large as the number of time periods (and the public information is already very large). These constructions are not directly comparable to normal (i.e. non-time-bound) schemes, but we include them in the table to get an idea of their efficiency compared to normal schemes.

D'Arco *et al.* [14] proposed a generic construction yielding a key assignment scheme offering KI security, using as components a KR-secure scheme and the Goldreich-Levin

Scheme	Private storage	Public storage	Key derivation	Type of security	Security based on
Atallah *et al.* [8, 13]	ρ	$\rho\|E\|$	$(c_H + c_{XOR})h$	KR	PRFs
	ρ	$\rho_1\|E\|$	$(c_H + c_D)h$	KI	PRF+SE-CPA
De Santis *et al.* [12]	ρ	$\rho_1(\|E\| + 2\|V\|)$	$(c_D + 1)h$	KI	SE-CPA
Freire and Paterson [16]	$w.\rho_2$	ρ_2	$c_S(h+1)\ell$	KI	PRG:BBS
Ateniese *et al.* [9, 17]	ρ	$\rho_1\mathcal{O}(\|V\|^2.\|T\|^3)$	c_D	KI	SE-CPA
	$\rho_4\mathcal{O}(\|T\|)$	$\rho_3\mathcal{O}(\|V\|^2)$	c_P	KI	BDDH
D'Arco *et al.* [14] (+ [1])	ρ_2	$\rho_2(\|V\|(1+\ell)+2)$	$c_E\ell$	KI	Random exp. RSA
Ours (PRF-based)	$w.\rho$	0	$c_{\mathrm{PRF}}(h+1)$	S-KI	PRF
Ours (FS-PRG-based)	$w.\rho_2$	0	$c_{\mathrm{PRG}}(h+1)$	S-KI	FS-PRG

Table 1. Comparison with Previous Schemes

hard-core bit (GL bit) [26]. They instantiated their construction with an Akl-Taylor scheme, which they proved to be KR-secure under the RSA assumption, therefore achieving KI-security under the same assumption. Their construction can also be instantiated, for example, with the KR-secure scheme from Atallah *et al.* [8], yielding a KI-secure scheme based on PRFs. However, their construction involves a significant "blow up" when going from KR-security to KI-security, since it requires the use of a KR-secure scheme for a poset having ℓ classes for each class in the poset for the final KI-secure scheme, where ℓ is the length of the keys k_u. Typically, we would desire $\ell = 128$ bits in applications. Their construction also consumes a large amount of public storage, requiring roughly ℓ times as many public values as in the starting KR-secure scheme. Key derivation also involves an increase by a factor of ℓ relative to the starting scheme.

Freire and Paterson [16] proposed a construction which exploits the chain partition approach proposed in [15] and which achieves KI security. In their scheme, the maximum amount of private information associated with a class is related to the width w of the poset representing the hierarchy. The public storage is much smaller than in previous schemes: when instantiated using the BBS generator, the public storage is just one RSA modulus. Key derivation requires multiple squaring operations modulo an RSA modulus, the number of such operations growing as ℓh where ℓ is the bit-size of the scheme's keys and h is the number of levels between the classes.

Our constructions provide stronger security guarantees (*i.e.*, key indistinguishability in our *strengthened* model) than all the above schemes and indeed all other hierarchical key assignment schemes in the literature. As with [16], our constructions provide schemes having a trade-off between storage of private information and efficiency of key derivation, depending on how the poset is partitioned into chains. The overall efficiency of key derivation is bounded by the length of the longest chain in the partition, and the amount of private information depends on w, the poset width (which is equal to the number of chains in the partition). Moreover, due to the cryptographic components used in our constructions (PRFs and FS-PRGs), key derivation is relatively efficient, growing linearly in h, the number of levels between classes. In addition, our constructions require *no* public storage.

Table 1 gives us a comparison of our schemes and other provably secure schemes in the literature. Private storage is measured per class in the access graph, public storage is measured for the entire hierarchy, and key derivation shows the maximum amount of

computation that is needed to traverse h levels down in the hierarchy. We also include the type of security that the scheme achieves, key recovery (KR), key indistinguishability (KI) or our strengthened version, S-KI, and the basic components of the scheme. In the table, ρ is a security parameter that corresponds to the size of the keys for a PRF or for a symmetric encryption scheme $\mathcal{E}$; ρ_1 represents the size of ciphertexts for a semantically secure symmetric encryption scheme; ρ_2 is a security parameter for a pseudorandom number generator; ρ_3 represents the size of pairing-friendly group elements (which is typically a little larger than twice the security parameter, e.g. 171 bits at the 80-bit security level); ρ_4 is the size of the order q of a pairing-friendly group (which can usually be taken as twice the security parameter); c_H denotes the computation required to compute the output of a hash function; c_D is the computation needed for a symmetric key decryption; c_P is the computation needed for one pairing evaluation; c_S is the cost of squaring modulo an integer N of size ρ_2; c_E is the cost of exponentiation modulo an integer N of size ρ_2; c_{PRF} is the computation needed to compute the output of a PRF; c_{PRG} is the computation needed to compute the output of an FS-PRG; ℓ is the bit-size of the scheme's keys; and w is the width of a poset. Finally, $|T|$ represents the number of distinct time periods in the time-bound schemes of [9, 17].

References

1. Akl, S.G., Taylor, P.D.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems **1**(3) (1983) 239–248
2. MacKinnon, S.J., Taylor, P.D., Meijer, H., Akl, S.G.: An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. IEEE Transactions on Computers **34**(9) (1985) 797–802
3. Harn, L., Lin, H.Y.: A cryptographic key generation scheme for multilevel data security. Computers & Security **9**(6) (1990) 539–546
4. Chen, T.S., Chung, Y.F.: Hierarchical access control based on Chinese Remainder Theorem and symmetric algorithm. Computers & Security **21**(6) (2002) 565–570
5. Shen, V., Chen, T.S.: A novel key management scheme based on discrete logarithms and polynomial interpolations. Computers & Security **21**(2) (2002) 164–171
6. Wu, T.C., Chang, C.C.: Cryptographic key assignment scheme for hierarchical access control. Int. Journal of Computer Systems Science and Engineering **16**(1) (2001) 25–28
7. Yeh, J.H., Chow, R., Newman, R.: A key assignment for enforcing access control policy exceptions. In: Int. Symposium on Internet Technology. (1998) 54–59
8. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. In: ACM Conference on Computer and Communications Security. (2006) 190–202
9. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. In: ACM Conference on Computer and Communications Security. (2006) 288–297
10. Tzeng, W.G.: A secure system for data access based on anonymous authentication and time-dependent hierarchical keys. In: ACM Symposium on Information, Computer and Communications Security. (2006) 223–230
11. Wang, S.Y., Laih, C.S.: An efficient solution for a time-bound hierarchical key assignment scheme. IEEE Transactions on Dependable and Secure Computing **3**(1) (2006) 91–100
12. De Santis, A., Ferrara, A.L., Masucci, B.: Efficient provably-secure hierarchical key assignment schemes. In: MFCS. (2007) 371–382
13. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. **12**(3) (2009)
14. D'Arco, P., De Santis, A., Ferrara, A.L., Masucci, B.: Variations on a theme by Akl and Taylor: Security and tradeoffs. Theoretical Computer Science **411**(1) (2010) 213–227
15. Crampton, J., Daud, R., Martin, K.M.: Constructing key assignment schemes from chain partitions. In: Working Conference on Data and Applications Security (DBSec 2010). (2010) 130–145
16. Freire, E.S.V., Paterson, K.G.: Provably secure key assignment schemes from factoring. In: ACISP. (2011) 292–309

17. Ateniese, G., De Santis, A., Ferrara, A.L., Masucci, B.: Provably-secure time-bound hierarchical key assignment schemes. J. Cryptology **25**(2) (2012) 243–270
18. Crampton, J., Martin, K.M., Wild, P.R.: On key assignment for hierarchical access control. In: Computer Security Foundations Workshop. (2006) 98–111
19. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: CRYPTO. (1996) 1–15
20. Blum, L., Blum, M., Shub, M.: A simple unpredictable pseudo-random number generator. SIAM Journal on Computing **15**(2) (1986) 364–383
21. Blum, M., Goldwasser, S.: An efficient probabilistic public-key encryption scheme which hides all partial information. In: CRYPTO. (1984) 289–302
22. Bellare, M., Yee, B.S.: Forward-security in private-key cryptography. In: CT-RSA. (2003) 1–18
23. Dilworth, R.P.: A decomposition theorem for partially ordered sets. Annals of Mathematics **51**(1) (1950) 161–166
24. Dodis, Y., Gennaro, R., Hastad, J., Krawczyk, H., Rabin, T.: Randomness extraction and key derivation using the CBC, Cascade and HMAC modes. In: CRYPTO. Volume 3152. (2004) 115–133
25. Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. J. ACM **33**(4) (1986) 792–807
26. Goldreich, O., Levin, L.A.: A hard-core predicate for all one-way functions. In: ACM STOC. (1989) 25–32