Constructing Key Assignment Schemes from Chain Partitions

Jason Crampton, Rosli Daud, Keith M. Martin

Constructing Key Assignment Schemes from Chain Partitions

Jason Crampton, Rosli Daud, and Keith M. Martin

Information Security Group, Royal Holloway, University of London

Abstract. In considering a problem in access control for scalable multimedia formats, we have developed new methods for constructing key assignment schemes. Our first contribution is to improve an existing cryptographic access control mechanism for scalable multimedia formats. We then show how our methods can be applied to a chain partition to develop alternative mechanisms for scalable multimedia formats and how these methods can themselves be extended to create a new type of key assignment scheme.

1 Introduction

Scalable multimedia formats, such as MPEG-4 [1] and JPEG2000 [2], consist of two components: a non-scalable base component and a scalable enhancement component. Decoding the base component will yield low quality results. The quality of the decoded data can be improved by decoding the enhancement component as well as the base component. The enhancement component may comprise multiple "orthogonal" layers, orthogonal in the sense that each layer controls a distinct aspect of the quality of the encoded content. The MPEG-4 FGS (fine granularity scalability) format [1], for example, has a bit-rate layer and a peak signal-to-noise ratio (PSNR) layer.

Zhu *et al* proposed a layered access control scheme for MPEG-4 FGS called SMLFE (scalable multi-layer FGS encryption) [3]. The purpose of SMLFE is to provide different end-users with access to the same content at different levels of quality (by controlling access to the enhancement component).

SMLFE assumes that each enhancement frame is decomposed into different *segments*, each of which is associated with some bit-rate level and some PSNR level. In other words, the *enhancement component stream* (a sequence of enhancement frames) is split into a number of distinct segment streams. Each of these segment streams is encrypted with a different key, and the ability of an end-user (or, more accurately, the decoder available to the end-user) to reconstruct the enhancement component is determined by the keys that are accessible to the user.

However, SMLFE had a number of inadequacies and subsequent research sought to address these deficiencies [4–6]. This later research uses a labeling technique, which associates each segment with a k-tuple and then uses iterative hashing to derive key components for each segment. Most of these labeling

schemes suffer from the distinct disadvantage that different users can combine their respective key components to derive keys for which no single user is authorized. The one exception [5] uses a very complicated labeling process that makes it very difficult to reason about the properties of the scheme (including whether it is secure against colluding users or not). Our first contribution is to construct a labeling scheme that can be proved to be secure against colluding users and has other significant advantages over existing schemes. We discuss labeling schemes in Sec. 3.

We then consider alternative approaches to the problem of layered access control for scalable multimedia formats. Our second contribution is to define several schemes in Sec. 4 that make use of chain partitions. One of our constructions makes use of the labeling scheme we introduce in Sec. 3. The constructions in Sec. 4 have demonstrable advantages, in the context of layered access control, over labeling schemes and existing approaches to cryptographic access control.

It can be shown that the enforcement of layered access control for scalable multimedia formats can be regarded as a instance of a *key assignment scheme*. Such schemes are used to enforce a no-read-up information flow policy using cryptographic techniques. A recent survey of such schemes proposed a classification into four generic types of scheme [8]. These schemes offer different trade-offs in terms of the amount of storage required and the complexity of key derivation. Our final contribution, presented in Sec. 5, is to show that the schemes in Sec. 4 can be generalized to create new types of generic key assignment schemes. These generic schemes offer different trade-offs from existing schemes, which may prove useful for certain applications.

We conclude the paper with some suggestions for future work. Before proceeding further, we introduce some relevant background material.

2 Background

In this section, we first recall some relevant concepts from mathematics and cryptography. The section concludes with a more formal statement of the problem of layered access control and a discussion of its relationship to work on key assignment schemes.

2.1 Definitions and Notation

A *partially ordered set* (or *poset*) is a pair $(X, \leqslant)$, where $\leqslant$ is a reflexive, antisymmetric, transitive binary relation on X. X is a *total order* (or *chain*) if for all $x, y \in X$, either $x \leqslant y$ or $y \leqslant x$. We say $A \subseteq X$ is an *antichain* if for all $x, y \in A$, $x \not\leqslant y$ and $x \not\geqslant y$. We may write $y < x$ if $y \leqslant x$ and $y \neq x$, and we may write $x \geqslant y$ if $y \leqslant x$.

The (directed, acyclic) graph $(X, \leqslant)$ would include all "reflexive edges" and all "transitive edges", so it is customary to represent a poset using a smaller set of edges. We say x *covers* y, denoted $y \lessdot x$, if $y < x$ and there does not exist $z \in X$ such that $y < z < x$. Then the *Hasse diagram* of a poset $(X, \leqslant)$ is

defined to be the (directed, acyclic) graph $(X, \prec)$ [9]. A simple Hasse diagram is shown in Fig. 1(a). Note that all edges in the diagram are assumed to be directed upwards.

A *partition* of a set X is a collection of sets $\{Y_1, \ldots, Y_k\}$ such that (i) $Y_i \subseteq X$ (ii) $Y_1 \cup \cdots \cup Y_k = X$, and (iii) $Y_i \cap Y_j \neq \emptyset$ if and only if $i = j$. The *greatest common divisor* of x and y is written $\gcd(x, y)$; we say x and y are *co-prime* if $\gcd(x, y) = 1$.

We assume the existence of an *RSA key generator* [10], a randomized algorithm that takes a security parameter k as input and outputs a triple (N, e, d) such that:

- $N = pq$, where p and q are distinct odd primes;
- $e \in \mathbb{Z}^*_{\phi(N)}$, where $\phi(N) = (p-1)(q-1)$, $e > 1$, and $\gcd(e, \phi(N)) = 1$;
- $d \in \mathbb{Z}^*_{\phi(N)}$, where $ed \equiv 1 \mod \phi(N)$.

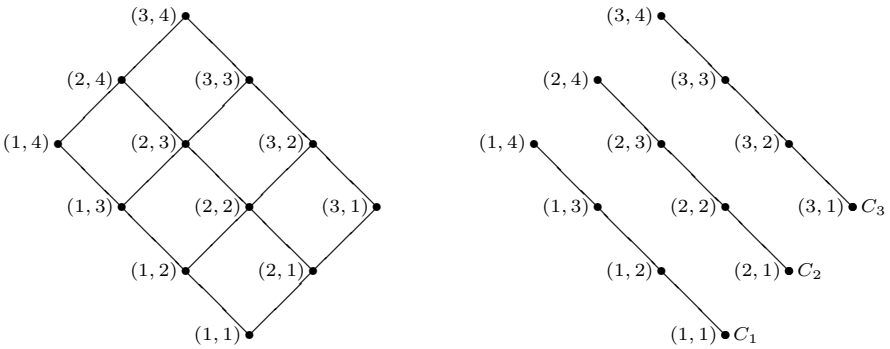
Finally, let $h : \{0,1\}^* \to \{0,1\}^\ell$ be a hash function and let $k \geqslant 0$ be an integer. Then we define the *iterative hash function* $h^k : \{0,1\}^\ell \to \{0,1\}^\ell$ in the following way: $h^0(x) = x$ and $h^k(x) = h(h^{k-1}(x))$.

2.2 Key Assignment Schemes

We now rephrase the problem at hand in more formal terms and illustrate how this problem is related to existing work on *key assignment schemes*. Let us assume that we are concerned with a scalable multimedia format with two distinct layers (such as bit-rate and PSNR), containing m and n levels respectively.

Define $R_{m,n} = \{(x, y) : 1 \leqslant x \leqslant m,\ 1 \leqslant y \leqslant n\}$ and define $(x_1, y_1) \leqslant (x_2, y_2)$ if and only if $x_1 \leqslant x_2$ and $y_1 \leqslant y_2$. Then $(R_{m,n}, \leqslant)$ is a partially ordered set. Each segment (and segment stream) represents a distinct protected object and is labeled with a pair (i, j) indicating the corresponding levels in the bit-rate and PSNR layers, respectively. Each pair $(i, j) \in R_{m,n}$ is associated with an encryption key $\kappa_{i,j}$. Segment streams are encrypted with the corresponding key. Each user is authorized to access layered multimedia of some quality $q_{i,j}$, which implies that such a user must be able to compute $\kappa_{x,y}$ for all $x \leqslant i$ and all $y \leqslant j$ in order to decode the relevant segment streams. Figure 1(a) illustrates the poset $R_{3,4}$.

Clearly the access control requirements described above closely resemble the "no-read-up" component of an *information flow policy* [11, 12]. There are many schemes in the literature for enforcing an information flow policy using cryptographic techniques (see the survey paper of Crampton *et al* [8], for example). Given a security lattice $(L, \leqslant)$, a set of subjects U, a set of protected objects O, and a security function $\lambda : U \cup O \to L$, we define a set of cryptographic keys $\{\kappa(x) : x \in L\}$. Then, adopting a cryptographic approach to policy enforcement, we encrypt object o with (symmetric) key $\kappa(\lambda(o))$. In order to correctly implement the information flow policy, a user u with security label should be given, or be able to derive, $\kappa(y)$ for all $y \leqslant x$. There are several generic solutions, the most obvious of which is to give u the set of keys $\{\kappa(y) : y \leqslant \lambda(u)\}$.



(a) $R_{3,4}$ (b) Canonical chain decomposition

Fig. 1. A typical poset used in layered access control for scalable multimedia formats

More commonly, we give u a single key $\kappa(\lambda(u))$ and publish additional information that enables the user to derive $\kappa(y)$ whenever $y < \lambda(u)$. The additional information "encrypts edges" in the graphical representation of L: that is, if $G = (L, E)$, then we publish $\mathsf{Enc}_{\kappa(x)}(\kappa(y))$ for all $(x, y) \in E$. There are two obvious choices for E: the edge set corresponding to the full order relation $\leqslant$, in which case, key derivation can be performed in a single step; or the edge set corresponding to the cover relation $\lessdot$, in which case key derivation may take several steps. We call the former a *direct key encrypting* (DKE) key assignment scheme and the latter an *iterative key encrypting* (IKE) scheme [8]. Clearly, there is a trade-off between the amount of storage required and the number of steps required for key derivation.

Any key assignment scheme for enforcing an information flow policy should satisfy two criteria.

- The scheme is *correct* if for all y, $\kappa(x)$ can be derived from $\sigma(y)$ and the public information if $y \geqslant x$.[1]
- The scheme is *collusion secure* if, for all $x \in X$ and all $Y \subseteq X$ such that for all $y \in Y$, $y \not\geqslant x$, it is not possible to derive $\kappa(x)$ from $\{\sigma(y) : y \in Y\}$ and the public information. Note that this definition includes the case of a singleton subset Y, which corresponds to a single user "colluding" to recover a key for which she is not authorized.[2]

[1] It should be emphasized here that "derived from" means "derived from in a feasible amount of time". Very few cryptographic schemes provide unconditional security in an information-theoretic sense; rather, they guarantee with a high probability that a scheme is secure against an adversary with reasonable resources. The interested reader is referred to the literature for a more detailed discussion of these issues [10].

[2] Recent work has introduced the notions of *key recovery* and *key indistinguishability* [13]. A proof that a scheme is secure against key recovery is analogous to proving that a scheme is collusion secure. The main difference is that collusion security assumes that colluding users will try to compute a key using the particular methods of

Clearly, the problem of enforcing layered accessed control for scalable multimedia formats can be addressed by defining an appropriate key assignment scheme for the partially ordered set $(R_{m,n}, \leqslant)$. However, because of the particularly simple structure of $R_{m,n}$, in the next two sections we consider some key assignment schemes that are tailored to the problem of layered access control for scalable multimedia formats. In Sec. 3, we consider *labeling schemes*, in which each key is defined by a set of *key components*, each of which is obtained by iteratively hashing some secret value. In Sec. 4, we consider some alternative approaches using chain partitions of $R_{m,n}$.

3 A New Labeling Scheme for Layered Access Control

Apart from SMLFE [3], all existing schemes for layered access control (to our knowledge) associate a distinct k-tuple with each element of $R_{m,n}$ [4–6]. This k-tuple is used to construct k key components using iterative hashing. We write $\phi(x, y) \in \mathbb{Z}^k$ to denote the label assigned to $(x, y) \in R_{m,n}$ and we write $\phi_i(x, y)$ to denote the ith co-ordinate of $\phi(x, y)$. In this section, we first summarize the basic technique and then describe our new labeling scheme and compare it to existing work.

First we introduce some additional definitions. Let $a = (a_1, \ldots, a_k)$ and $b = (b_1, \ldots, b_k)$ be elements of $\mathbb{Z}^k$. Then we define $(a_1, \ldots, a_k) \leqslant (b_1, \ldots, b_k)$ (in $\mathbb{Z}^k$) if and only if $a_i \leqslant b_i$ for all i, and we define $a - b = (a_1 - b_1, \ldots, a_k - b_k)$. We say a is *positive* if $a_i \geqslant 0$ for all i.

Labeling schemes have the property that $(x, y) \geqslant (x', y')$ in $R_{m,n}$ if and only if $\phi(x', y') - \phi(x, y)$ is positive. It is this property that ensures the correctness of each scheme, since $\phi(x', y') - \phi(x, y)$ is used to construct k secrets per node using iterative hashing.

The content provider, hereafter called the *scheme administrator*, chooses a hash function $h : \{0,1\}^* \to \{0,1\}^\ell$ and k secrets $\sigma_1, \ldots, \sigma_k \in \mathbb{Z}^\ell$. Then the secret $\sigma_{x,y}$ assigned to $(x, y) \in R_{m,n}$ comprises k key components:

$$\sigma_{x,y} \stackrel{\text{def}}{=} (h^{\phi_1(x,y)}(\sigma_1), \ldots, h^{\phi_k(x,y)}(\sigma_k)).$$

For brevity, we may abuse notation and write $h^{\phi(x,y)}(\sigma)$ to denote $\sigma_{x,y}$. We define the key assigned to (x, y) to be

$$\kappa_{x,y} \stackrel{\text{def}}{=} h(h^{\phi_1(x,y)}(\sigma_1) \parallel \cdots \parallel h^{\phi_k(x,y)}(\sigma_k)),$$

where $s_1 \parallel s_2$ denotes the concatenation of s_1 and s_2.[3] Again, we may abuse notation and write $h(\sigma_{x,y})$ to denote $\kappa_{x,y}$.

key derivation associated with the scheme, whereas a proof of security against key recovery establishes that the recovery of a key is as difficult as solving some known hard problem. While formal security proofs of this nature are certainly important in modern cryptographic research, space constraints mean they are out of scope for this paper.

[3] The schemes in the literature simply define the "key" associated with (x, y) to be the concatenation of the key components. We take the hash of the concatenation of

Correctness. By construction $(x, y) \geqslant (x', y')$ if and only if $\phi_i(x', y') - \phi_i(x, y)$ is positive. Now the ith key component of $\kappa_{x,y}$ is $h^{\phi_i(x,y)}(\sigma_i)$ and the ith key component of $\kappa_{x',y'}$ is $h^{\phi_i(x',y')}(\sigma_i)$. Hence, if $(x, y) \geqslant (x', y')$, then we simply hash the ith component of $\kappa_{x,y}$ a total of $\phi_i(x', y') - \phi_i(x, y)$ times to obtain the ith key component of $\sigma_{x',y'}$. Conversely, if $(x, y) \not\geqslant (x', y')$, then for some i, $\phi_i(x', y') - \phi_i(x, y) < 0$, which implies that we can only obtain the ith component of $\kappa_{x,y}$ by inverting h, which is computationally infeasible provided h is chosen appropriately.

The IWFK-1 scheme [6, §3.1.1], for example, simply defines $\phi(x, y)$ for $(x, y) \in R_{m,n}$ to be $(m - x, n - y)$. So, for example, $\phi(2, 4) = (1, 0)$ and $\phi(1, 1) = (2, 3)$ in $R_{3,4}$. Then $\sigma_{2,4} = (h(\sigma_1), \sigma_2)$ and $\sigma_{1,1} = (h^2(\sigma_1), h^3(\sigma_2))$. Hence, $\sigma_{2,4}$ can be used to derive $\sigma_{1,1}$ by hashing the first component of $\sigma_{2,4}$ once and hashing the second component twice.

Collusion Security. It is known that all but one of the schemes in the literature are not collusion secure. Indeed, it is trivial to find examples that break each of the schemes: in the IWFK-1 scheme for $R_{3,4}$, for example, $\sigma_{2,4} = (h(\sigma_1), \sigma_2)$ and $\sigma_{3,3} = (\sigma_1, h(\sigma_2))$; clearly these keys can be combined to recover $(\sigma_1, \sigma_2) = \sigma_{3,4}$. The IFAK scheme is claimed to be collusion secure [5], although no proof of this claim is given.

3.1 The CDM Scheme

We now explain how our scheme works, which we call the CDM scheme for ease of reference.

Definition 1. *Let $(x, y) \in R_{m,n}$. Then we define the* CDM label *of (x, y) to be*

$$\phi_{\text{CDM}}(x, y) \stackrel{\text{def}}{=} (\underbrace{n - y, \ldots, n - y}_{x}, \underbrace{n, \ldots, n}_{m-x}).$$

Henceforth, we will simply write $\phi(x, y)$ to denote the CDM labeling of $(x, y) \in R_{m,n}$. Note the CDM labeling has m components. We now state several elementary results concerning the properties of the CDM labeling.[4]

Proposition 1. *Let $(x, y), (x', y') \in R_{m,n}$. Then $\phi(x', y') - \phi(x, y)$ is positive if and only if $(x, y) \geqslant (x', y')$.*

Proposition 2. *Let $(x, y), (x', y') \in R_{m,n}$ such that $(x, y) \geqslant (x', y')$. Then $\sigma_{x',y'}$ can be derived from $\sigma_{x,y}$ using precisely $xy - x'y'$ hash computations.*

---

those components to make the distinction between key and key components clearer. It also means that we have fixed-length, short symmetric keys, determined by the size of h's output.

[4] Lack of space precludes the inclusion of proofs in this version of the paper: the interested reader is referred to the extended version of the paper [7] for the relevant details.

Corollary 1 *The number of hash computations required is bounded by* $mn - 1$.

We now give some intuition behind the labeling and an example. The element $(x, y) \in R_{m,n}$ defines a sub-rectangle $R_{x,y}$. Removing $R_{x,y}$ truncates the first i chains and leaves the remaining chains intact. Our labeling simply records the lengths of the chains that are left following the removal of $R_{x,y}$. Hence, for example, $\phi(2, 4) = (0, 0, 4)$ and $\phi(1, 1) = (3, 4, 4)$. Note that $3+4 = 7$ operations are required to derive $\kappa_{1,1}$ from $\kappa_{2,4}$ (as we would expect from Proposition 2).

The geometric intuition behind the scheme also provides some understanding of why our scheme is collusion secure.

Proposition 3. *Let* $(x_1, y_1), \ldots, (x_j, y_j) \in R_{m,n}$ *such that* $(x_i, y_i) \not\geqslant (x, y)$ *for all* i. *Then there exists* t, $1 \leqslant t \leqslant m$, *such that* $\phi_t(x, y) < \phi_t(x_i, y_i)$ *for all* i.

Hence, no set of m colluding users can recover the tth component of $\sigma_{x,y}$. In other words, we have the following corollary.

Corollary 2 *The CDM scheme is collusion secure.*

3.2 Related Work

Table 1 provides a summary of the four schemes in the literature for layered access control (IWFK-1 [6, §3.1.1], IWFK-2 [6, §3.2.3], IFAK [5], and HIFK [4]), presented in chronological order and identified by the initial letters of the authors' surnames. Each component of $\sigma_{x,y} = h^{\phi(x,y)}(\sigma)$ is a distinct secret key component, as each component has to be hashed independently of the others. Hence, we believe it is appropriate to minimize the number of key components and the number of derivation steps that are required. The table reports precise storage requirements (given by the number of key components k) and worst case derivation (in terms of the number of hash computations required).

All of these schemes are correct, but only the IFAK scheme is claimed to be collusion secure, in the sense that a set of collaborating users cannot combine the secret components of their respective keys (and possibly use iterative hashing) to derive a key for which no one of them was authorized.

Table 1. A summary of labeling schemes for layered access control

Scheme	k	**Key derivation**	**Collusion secure**
IWFK-1	2	$m + n - 2$	N
IWFK-2	3	$m + 2n - 3$	N
IFAK	$m + n - 1$	$\frac{1}{2}(m + n - 2)(m + n - 1)$	Y
HIFK	3	$2m + 2n - 4$	N
CDM	m	$mn - 1$	Y

The characteristics of our scheme are shown in the last row of the table. Our scheme is collusion secure under the same assumptions that the IFAK scheme is (claimed to be) secure. However, we use a smaller value of k and we require fewer derivation steps. Moreover, we have a systematic and easily implementable way of generating our labels (unlike the IFAK scheme); because of this we can also compute the number of derivation steps required for any $(x, y), (x', y') \in R_{m,n}$ and prove that our scheme is collusion secure. IFAK, in contrast, has an extremely complicated labeling scheme, which makes it difficult to reason about (i) the number of derivation steps required in the general case (ii) the collusion security of the scheme.

4 New Schemes for Layered Access Control

In this section, we propose a number of key assignment schemes for implementing layered access control for scalable multimedia formats. These schemes assume that the poset $(R_{m,n}, \leqslant)$ has been partitioned into chains. Dilworth's Theorem [14] asserts that every partially ordered set $(X, \leqslant)$ can be partitioned into w chains, where w is the *width* of X.[5]

Evidently, there are many different ways to partition the poset $R_{m,n}$ into chains, but we choose a particular partition that enables us to define two very simple schemes. We assume without loss of generality that $m \leqslant n$, and we define the *canonical* partition of $R_{m,n}$ into chains to be $\{C_1, \ldots, C_m\}$, where $C_i = \{(i, j) : 1 \leqslant j \leqslant n\}$. Figure 1(b) illustrates the canonical partition of $R_{3,4}$ into chains.

4.1 Schemes with No Public Information

Generally, key assignment schemes rely on public information for key derivation [8]. An interesting feature of the schemes in the previous section is that no public information is used. In this section we consider two different schemes that require no public information – one based on hash functions and one based on RSA – and have lower storage requirements than the CDM scheme.

A Scheme Based on Hash Functions. The scheme administrator first selects a family of m hash functions $h_i : \{0,1\}^* \to \{0,1\}^\ell$, $1 \leqslant i \leqslant m$. The scheme administrator also selects m secret values, $\sigma_1, \ldots, \sigma_m \in \{0,1\}^\ell$, where σ_i is associated with chain C_i. The scheme administrator then computes a secret key for each element in $R_{m,n}$, where $\kappa_{i,j}$ is defined to be $h_i^{n-j}(\sigma_i)$. Then a user authorized for content quality $q_{i,j}$ is given the keys $\{\kappa_{x,j} : 1 \leqslant x \leqslant i\}$. For reasons that will be apparent from the above description, we call this a *multiple-key iterated hash scheme*.

[5] The width of X is the cardinality of the largest antichain in X. Clearly, any partition into chains must contain at least w chains. It is harder to prove that no more than w are required.

Correctness. We first show that a user can derive all keys for which she is authorized. Suppose that a user is authorized for quality $q_{i,j}$. (Equivalently, the user is associated with label $(i, j) \in R_{m,n}$.) Henceforth, we will simply write $u_{i,j}$ for such a user. Then $u_{i,j}$ must be able to derive all keys in the rectangle $R_{i,j}$. Now, by construction, $u_{i,j}$ has $\kappa_{x,j}$ for all $x \leqslant i$. Moreover, $\kappa_{x,y} = h_x^{j-y}(\kappa_{x,j})$, $1 \leqslant y < j$. Hence, a user $u_{i,j}$ can derive any key in $R_{i,j}$ in no more than $j - 1$ steps.

Collusion Security. Any "good" hash function will have the property that it is computationally hard to compute x given $y = h(x)$ (that is, *pre-image resistance*). Since keys are obtained by successively hashing elements in a chain, it is computationally hard to recover $\kappa_{i,j+1}$ from $\kappa_{i,j}$, as this would require the computation of the pre-image of $\kappa_{i,j}$. Hence, a user certainly cannot use a key from one key chain to derive a key higher up the same key chain (and hence for which she is not authorized), providing the scheme administrator chooses a suitable hash function. However, a user may have several keys: assuming that the key chains are independent – in the sense that knowledge of an element in C_i provides no information about any element in C_j, for all $j \neq i$ – then it is not possible for the user to derive any keys for which she is not authorized. We have chosen a different hash function for each chain in order to provide this key chain independence.

If two or more users collude – equivalently, if an adversary is able to obtain the keys of several users – then the set of keys available do not correspond to the nodes of a sub-rectangle (as they do for a single user). Suppose that an adversary (whether it is a group of colluding users or a single malicious entity) collectively has the keys $\kappa_{1,j_1}, \ldots, \kappa_{m,j_m}$. Then κ_{i,j_i} cannot be used to recover κ_{i,j_i+k} for any $k > 0$ if h_i has pre-image resistance. Hence, assuming the independence of key chains, as before, we see that such an adversary has no additional advantage over a single user.

A Scheme Based on RSA. In this scheme, we make use of a special case of the Akl-Taylor scheme [15], which can be applied to any poset. Specifically, we apply the scheme to each of the chains in the partition.

The scheme administrator first obtains m large compound integers $N_1, \ldots, N_m$ using an RSA key generator and makes these values public. For each chain C_i, the scheme administrator:

- chooses a secret $\sigma_i \in \mathbb{Z}_{N_i}^*$, such that for all σ_i and σ_j are co-prime if $i \neq j$;
- defines $\kappa_{i,j} = (\sigma_i)^{2^{n-j}} \bmod N_i$.

We call the sequence of keys

$$\kappa_{i,n} = (\sigma_i)^1,\ \kappa_{i,n-1} = (\sigma_i)^2 \bmod N_i, \ldots,\ \kappa_{i,1} = (\sigma_i)^{2^{n-1}} \bmod N_i$$

an *RSA key chain*. As before, user $u_{i,j}$ is given the keys $\{\kappa_{x,j} : 1 \leqslant x \leqslant i\}$. Henceforth, for reasons of clarity and brevity, we will write x rather than $x \bmod N_i$, when N_i is clear from context.

Correctness and Collusion Security. Key derivation is quite different using RSA key chains. To obtain $\kappa_{x,y}$, where $x < i$ and $y < j$, the user selects $\kappa_{x,j}$ and then computes

$$(\kappa_{x,j})^{2^{j-y}} = (\kappa_{x,j})^{\frac{2^{n-y}}{2^{n-j}}} = \left((\sigma_x)^{2^{n-j}}\right)^{\frac{2^{n-y}}{2^{n-j}}} = (\sigma_x)^{2^{n-y}} = \kappa_{x,y}$$

To illustrate, consider Fig. 1(b) and suppose that the keys for C_2 are

$$\kappa_{2,4} = \sigma_2,\ \kappa_{2,3} = \sigma_2^2,\ \kappa_{2,2} = \sigma_2^4,\ \kappa_{2,1} = \sigma_2^8.$$

Suppose we wish to derive $\kappa_{2,1}$ and we have $\kappa_{2,3}$. Then we compute

$$(\kappa_{2,3})^{2^{3-1}} = \kappa_{2,3}^4 = (\sigma_2^2)^4 = \sigma_2^8 = \kappa_{2,1}.$$

However, user with key $\kappa_{i,j}$ cannot derive $\kappa_{i,y}$ if $y > j$, since this would require the user to solve the *RSA problem*.[6] Similarly, no collection of keys that includes $\kappa_{i,j}$ (but no key higher up the ith chain) can be used to derive $\kappa_{i,y}$.

4.2 Schemes with Single Keys

Most key assignment schemes in the literature require the end-user to store a single key. The multiple-key schemes described above clearly do not satisfy this criterion.

In this section, we describe schemes that only require the end user to store a single key. The trade-off is that such schemes require a certain amount of public information.

A Scheme Based on Hash Functions. The scheme we now describe could be considered to be a hybrid of an iterative key encrypting (IKE) scheme [8] and a hash chain. Atallah *et al*, for example, define a concrete construction of an IKE scheme [13].

In our scheme, the content provider selects m hash functions $h_1, \ldots, h_m$ and m secrets $\sigma_1, \ldots, \sigma_m$, and defines key $\kappa_{i,j} = h_i^{n-j}(\sigma_i)$, as before. Now, however, the content provider publishes enough information to enable the computation of $\kappa_{x,j}$ from $\kappa_{i,j}$ for all $x < i$, by publishing $\left\{\mathsf{Enc}_{\kappa_{i,j}}(\kappa_{i-1,j}) : 1 < i \leqslant m, 1 \leqslant j \leqslant n\right\}$. Hence, we require $(m-1)n$ items of public information.

Correctness and Collusion Security. Again, it is very easy to demonstrate that a user $u_{i,j}$ can derive the key for any node in $R_{i,j}$. First, $u_{i,j}$ is given $\kappa_{i,j}$ and this key, in conjunction with the public information, can be used to derive $\kappa_{x,j}$ for all $x < i$. Moreover, $\kappa_{x,y}$ can be obtained from $\kappa_{x,j}$ by $j - y$ applications of h. Hence, $u_{i,j}$ can obtain $\kappa_{x,y}$ in no more than $i - 1 + j - 1 = i + j - 2$ steps.

Collusion security follows from the fact that pre-image resistance of the hash function prevents the computation of $\kappa_{i,j+k}$ from $\kappa_{i,j}$ for any $k > 0$. The assumption that it is computationally hard to decrypt without knowledge of the secret key ensures that $\kappa_{i+k,j}$ cannot be derived from $\kappa_{i,j}$.

[6] That is, given N, $y \in \mathbb{Z}_N^*$ and an integer $e > 0$ that is co-prime to $\phi(N)$, compute $y^{1/e} \bmod N$.

A Scheme Based on RSA. Finally, we note that we can use the CDM labeling (Definition 1), in which modular exponentiation is used to recover keys. It is important to note that this scheme does not rely on the idea of encrypting edges, and is therefore quite different from the schemes described above. (It is, however, closely related to the Akl-Taylor scheme [15].)

Recall that we associate each $(x, y) \in R_{m,n}$ with a CDM label $\phi(x, y) \in \mathbb{Z}^m$. Moreover, $\phi(x', y') - \phi(x, y)$ is positive if and only if $(x, y) \geqslant (x', y')$. In this new scheme the scheme administrator

- obtains a large compound integer N using the RSA key generator;
- chooses small, distinct primes $p_1 = 2, p_2 = 3, \ldots, p_m \in \mathbb{Z}_N^*$ and makes them public;
- chooses a master secret $\sigma \in \mathbb{Z}_N^*$;
- defines
$$\pi(x, y) = \prod_{i=1}^{m} p_i^{\phi_i(x,y)};$$
- defines $\kappa_{x,y} = \sigma^{\pi(x,y)} \bmod N$.

Correctness and Collusion Security. Consider (x, y) and (x', y'), where $(x, y) \geqslant (x', y')$. Then $\phi(x', y') - \phi(x, y)$ is positive and

$$\frac{\pi(x', y')}{\pi(x, y)} = \prod_{i=1}^{m} p_i^{\phi_i(x',y') - \phi_i(x,y)} = \prod_{i=1}^{x'} p_i^{y-y'} \prod_{i=x'+1}^{x} p_i^{y}$$

Hence,

$$(\kappa_{x,y})^{\frac{\pi(x',y')}{\pi(x,y)}} = (\sigma^{\pi(x,y)})^{\frac{\pi(x',y')}{\pi(x,y)}} = \sigma^{\pi(x',y')} = \kappa_{x',y'}$$

In other words, if $\phi(x', y') - \phi(x, y)$ is positive, we can compute $\kappa_{x',y'}$ from $\kappa_{x,y}$ since we can compute $\frac{\pi(x',y')}{\pi(x,y)}$. Specifically, given $\kappa_{x,y}$:

1. compute $\phi(x, y)$ and $\phi(x', y')$, which is trivial if m and n are known;
2. compute $\phi(x', y') - \phi(x, y)$ and hence $\pi(x', y')/\pi(x, y)$;
3. finally, compute $\kappa_{x',y'}$.

We cannot compute $\kappa_{x'',y''}$ from $\kappa_{x,y}$ if $(x, y) \not\geqslant (x'', y'')$ since this would imply that $\phi(x'', y'') - \phi(x, y)$ is not positive and we would have to compute integral roots modulo N to compute $\kappa_{x'',y''}$. (In other words, solve the RSA problem.) Moreover, Proposition 3 implies that any adversary with keys $\kappa_{x_1,y_1}, \ldots, \kappa_{x_j,y_j}$, such that $(x_i, y_i) \not\geqslant (x, y)$, would have to solve the RSA problem to compute $\kappa_{x,y}$.

4.3 Related Work

In Table 2, we summarize the properties of several schemes in the literature and compare them to the schemes we have introduced in this section. The table includes, for ease of reference, the best labeling scheme from Sec. 3. We also

include IKE and DKE schemes for $R_{m,n}$. Atallah *et al* have demonstrated how an IKE scheme (and hence a DKE scheme) can be implemented using hash functions [13]; we will assume that this implementation is used for the purposes of our comparison. We write MKIH to denote the multiple-key iterative hash scheme and MKRSA to denote the multiple-key RSA scheme and replace 'M' with 'S' for the equivalent single-key schemes.

We write T_{Hsh} to denote the time taken to compute a hash function and T_{Mul} to denote the time taken to perform a modular multiplication. (Recall that the modular exponentiation a^n can be performed by a square-and-multiply algorithm using no more than $2\log_2 n$ modular multiplications.) We assume that our unit of storage is 128 bits. That is, all storage costs in Table 2 are expressed as multiples of 128 bits. We assume that the output of each hash function is 128 bits, and the RSA modulus is 1024 bits (that is, 8 units of storage). The table reports the worst case for storage costs and the number of key derivation steps.

Table 2. A summary of related work and a comparison with our schemes

Scheme	**Private storage**	**Public storage**	**Key derivation**
CDM	m	0	$(mn-1)T_{\text{Hsh}}$
IKE	1	$(m-1)n+m(n-1)$	$(m+n-2)T_{\text{Hsh}}$
DKE	1	$\frac{1}{4}mn((m+1)(n+1)-4)$	T_{Hsh}
MKIH	m	0	$(m-1)T_{\text{Hsh}}$
MKRSA	$8m$	0	$(m-1)T_{\text{Mul}}$
SKIH	1	$(m-1)n$	$(m+n-2)T_{\text{Hsh}}$
SKRSA	8	$8m$	$(2n\sum_{i=1}^{m}\log_2 p_i)T_{\text{Mul}}$

It is clear that even the best labeling scheme (CDM) does not compare well with either the generic schemes in the literature or the schemes we have introduced in this section. The main reason for this is that the key components in the labeling schemes do not provide as much information about keys as the other schemes do. In MKIH, for example, a single key is required to derive all the keys on any particular chain in the canonical decomposition, in contrast to the labeling schemes. We can see from the table that the RSA-based schemes, although attractive in principle, are unlikely to be as attractive in practice: the storage required per key is an order of magnitude greater than hash-based schemes and the key derivation method requires the comparatively expensive modular multiplication operation.

5 New Key Assignment Schemes

Our original motivation was to construct better schemes for layered access control. However, it became apparent that the schemes described in the preceding

section could be generalized to create key assignment schemes that could be applied to any poset. Moreover, the resulting schemes do not fit into the taxonomy of generic key assignment schemes proposed by Crampton *et al* [8]. In this section, we describe briefly how two of our schemes for layered access control can be extended to create generic key assignment schemes.

Given a poset X, we first select a partition of X into chains $\{C_1, \ldots, C_w\}$, where w is the width of X.[7] We denote the length of C_i by ℓ_i, $1 \leqslant i \leqslant w$. We regard the maximum element of C_i as the first element in C_i and the minimum element as the last (or ℓ_ith) element.

Let $C = x_1 \geqslant x_2 \geqslant \cdots \geqslant x_m$ be any chain in X. Then we say any chain of the form $x_j \geqslant \cdots \geqslant x_m$, $1 < j \leqslant m$, is a *suffix* of C. Now, for any $x \in X$, the set $\downarrow x \stackrel{\text{def}}{=} \{y \in X : y \leqslant x\}$ has non-empty intersection with one or more chains $C_1, \ldots, C_w$. We now prove that the intersection of $\downarrow x$ and a chain C_i is a suffix of C_i. This result enables us to define the keys that should be given to a user with label x.

Proposition 4. *For all $x \in X$ and any chain $C \subseteq X$, either $\downarrow x \cap C$ is a suffix of C or $\downarrow x \cap C = \emptyset$.*

The above proposition indicates how we should allocate keys to users. Since $\{C_1, \ldots, C_w\}$ is a partition of X into chains, $\{\downarrow x \cap C_1, \ldots, \downarrow x \cap C_w\}$ is a disjoint collection of chain suffixes. Moreover, the keys for each element in X have been chosen so that the key for the jth element of a chain can be used to compute all lower elements in that chain. Hence, we can see that a user with label x must be given the keys for the maximal elements in the non-empty suffixes $\downarrow x \cap C_1, \ldots, \downarrow x \cap C_w$. Given $x \in X$, let $\hat{x}_1, \ldots, \hat{x}_w$ denote these maximal elements, with the convention that $\hat{x}_i = \perp$ if $\downarrow x \cap C_i = \emptyset$. Clearly the number of $\hat{x}_i$ such that $\hat{x}_i \neq \perp$ is no greater than w. The above result and observations provide the foundations of both the schemes that follow.

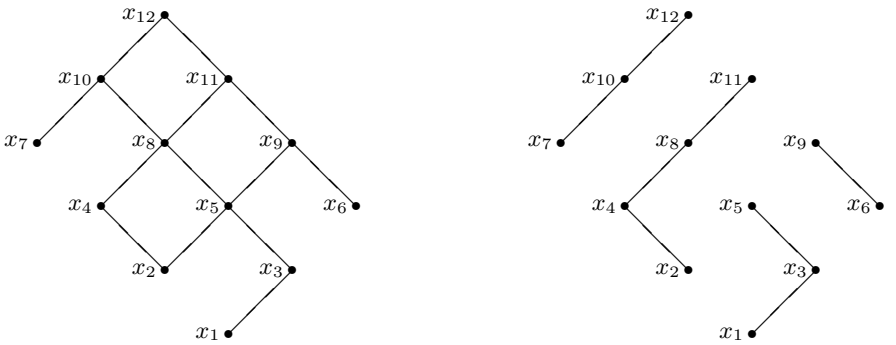
5.1 Multiple-Key Iterated Hash Scheme

We first consider the use of iterated hashing. The scheme administrator

- selects a chain partition of X into w chains $C_1, \ldots, C_w$;
- selects w secret values $\sigma_1, \ldots, \sigma_w$ and w hash functions $h_1, \ldots, h_w$;
- defines the key for the maximum element of chain C_i to be σ_i;
- for each pair $x, y \in C_i$ such that $x \lessdot y$, defines $\kappa(x) = h_i(\kappa(y))$;
- for each $x \in X$, defines the private information for x to be $\{\kappa(\hat{x}_i) : \hat{x}_i \neq \perp\}$.

We denote the key for the jth element of C_i by $\kappa_{i,j}$. Clearly (as in Sec. 4), a user in possession of $\kappa_{i,j}$ can compute $\kappa_{i,y}$, for any $y > j$, by $y - j$ iterative hash computations. Figure 2 illustrates a poset X of width 4 and one possible partition of X into 4 chains. If the chain $x_{11} > x_8 > x_4 > x_1$ is associated with the secret value σ, for example, then $\kappa(x_{11}) = \sigma$ and $\kappa(x_8) = h(\sigma)$, etc.

[7] Unlike $R_{m,n}$, there is no canonical partition for an arbitrary poset X. At this stage, we do not consider what features a "good" partition might have. We return to this question towards the end of the section.



(a) A poset X (b) A partition of X into chains

Fig. 2. Partitioning an arbitrary poset into chains

Clearly, the number of steps required for key derivation is bounded by the length of the longest chain in the partition. With this is mind, it might be sensible to choose a chain partition in which the chains are as similar in length as possible. In terms of correctness and collusion security, the MKIH scheme for arbitrary posets is no different from the corresponding schemes for $R_{m,n}$.

5.2 Multi-Key RSA Scheme

In the second scheme, we use RSA key chains. The scheme administrator generates and publishes $N_1, \ldots, N_w$ (as in Sec. 4.1). As in Sec. 5.1, the scheme administrator selects a chain partition of X and defines the key for the maximum element of the ith chain to be σ_i. Now, for each pair $x, y \in C_i$ such that $x \lessdot y$, the scheme administrator defines $\kappa(x) = (\kappa(y))^2 \bmod N_i$. Finally, the private information associated with $x \in X$ is defined to be $\{\kappa(\hat{x}_i) : \hat{x}_i \neq \perp\}$ (as in the preceding scheme).

5.3 Related Work

In Table 3, we summarize the differences between our schemes and existing generic key assignment schemes. We also compare the performance of these schemes for the poset and chain partition illustrated in Fig. 2. We write c for the cardinality of the cover relation $\lessdot$ and r for the cardinality of the order relation $\leqslant$. For example, DKE, in general, requires a single key, r items of public information, and one step to derive a key; the scheme requires 51 items of public information for the poset illustrated in Fig. 2. The table states the storage in terms of number of keys and number of operations. For simplicity we omit MKRSA from the comparison, enabling us to assume that all keys have the same length.

Table 3. A comparison of our schemes with existing generic key assignment schemes

Scheme	Private storage				Public storage		Key derivation			
	x	x_{12}	x_{10}	x_9			x	x_{12}	x_{10}	x_9
Trivial	$\lvert{\downarrow}x\rvert$	12	7	5	0		0			
DKE	1				r	51	1			
IKE	1				c	14	d	5	4	3
MKIH	$\leqslant w$	4	3	3	0		$\leqslant d$	3	2	2

The MKIH scheme provides a different trade-off from the three existing schemes: users may have multiple keys[8] but no public information is required and key derivation will generally be quicker than for an equivalent IKE scheme.[9]

6 Conclusion

We have shown how to construct a new type of generic key assignment scheme using chain partitions, the inspiration for the original constructions being provided by the problem of enforcing layered access control in scalable multimedia formats. Our schemes, both for layered access control and as generic key assignment schemes, compare favorably with those in the literature.

We have many ideas for future work. Of primary interest is whether we can prove that our schemes are secure against key recovery [13], a more exacting criterion than that of collusion security used in this paper. We also hope to gain some insight, either from a mathematical analysis or through experimental work, into what might be the best choice(s) of chain partition for an arbitrary poset. A third area for potential research is to generalize our constructions to more than two scalable components (most likely using a recursive construction with one of our schemes from Sections 3 and 4 as a base case). Finally, we would like to apply our schemes to access control for geo-spatial data [17], because the policies used are rather similar to those for scalable multimedia formats.

Acknowledgements. The authors would like to thank the anonymous referees for their valuable comments.

[8] Schemes with multiple keys were usually disregarded in the early literature [8], although several recent schemes have made use of multiple keys [13, 17].

[9] Note that key derivation in our multi-key iterative hash scheme cannot be worse than key derivation in IKE and, in many cases, will be considerably better. As we observed earlier, it would be sensible to choose a chain partition in which all chains have approximately the same length. If this is possible, key derivation is approximately $|X|/w$. The poset in Fig. 2, for example, can be partitioned into 4 chains of length 3. Then any key can be derived in no more than 2 hops, whereas an IKE scheme would require 5 hops to derive $\kappa(x_1)$ from $\kappa(x_{12})$.

References

1. Li, W.: Overview of fine granularity scalability in MPEG-4 video standard. IEEE Transactions on Circuits and Systems for Video Technology **11**(3) (2001) 301–317
2. Christopoulos, C., Skodras, A., Ebrahimi, T.: The JPEG2000 still image coding system: An overview. IEEE Transactions on Consumer Electronics **46**(4) (2000) 1103–1127
3. Zhu, B., Feng, S., Li, S.: An efficient key scheme for layered access control of MPEG-4 FGS video. In: Proceedings of the 2004 IEEE International Conference on Multimedia and Expo. Volume 1. (2004) 443–446
4. Hashimoto, N., Imaizumi, S., Fujiyoshi, M., Kiya, H.: Hierarchical encryption using short encryption keys for scalable access control of JPEG 2000 coded images. In: Proceedings of the 2008 IEEE International Conference on Image Processing. (2008) 3116–3119
5. Imaizumi, S., Fujiyoshi, M., Abe, Y., Kiya, H.: Collusion attack-resilient hierarchical encryption of JPEG 2000 codestreams with scalable access control. In: Proceedings of the 2007 IEEE International Conference on Image Processing. Volume 2. (2007) 137 –140
6. Imaizumi, S., Watanabe, O., Fujiyoshi, M., Kiya, H.: Generalized hierarchical encryption of JPEG 2000 codestreams for access control. In: Proceedings of the 2005 IEEE International Conference on Image Processing. Volume 2. (2005) 1094–1097
7. Crampton, J., Daud, R., Martin, K.: Constructing key assignment schemes from chain partitions. Technical Report RHUL-MA-2010-10, Royal Holloway, University of London (2010) Available at `http://www.ma.rhul.ac.uk/static/techrep/2010/RHUL-MA-2010-10.pdf`.
8. Crampton, J., Martin, K., Wild, P.: On key assignment for hierarchical access control. In: Proceedings of 19th Computer Security Foundations Workshop. (2006) 98–111
9. Davey, B., Priestley, H.: Introduction to Lattices and Order. Cambridge University Press, Cambridge, United Kingdom (1990)
10. Katz, J., Lindell, Y.: Introduction to Modern Cryptography. Chapman & Hall/CRC (2007)
11. Bell, D., LaPadula, L.: Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Mitre Corporation, Bedford, Massachusetts (1976)
12. Denning, D.: A lattice model of secure information flow. Communications of the ACM **19**(5) (1976) 236–243
13. Atallah, M., Blanton, M., Fazio, N., Frikken, K.: Dynamic and efficient key management for access hierarchies. ACM Transactions on Information and System Security **12**(3) (2009) 1–43
14. Dilworth, R.: A decomposition theorem for partially ordered sets. Annals of Mathematics **51** (1950) 161–166
15. Akl, S., Taylor, P.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems **1**(3) (1983) 239–248
16. Atallah, M., Blanton, M., Frikken, K.: Key management for non-tree access hierarchies. In: Proceedings of 11th ACM Symposium on Access Control Models and Technologies. (2006) 11–18
17. Atallah, M., Blanton, M., Frikken, K.: Efficient techniques for realizing geo-spatial access control. In: Proceedings of the 2007 ACM Symposium on Information, Computer and Communications Security. (2007) 82–92