Optimal Constructions for Chain-Based Cryptographic Enforcement of Information Flow Policies

Jason Crampton, Naomi Farley, Gregory Gutin, Mark Jones

Optimal Constructions for Chain-based Cryptographic Enforcement of Information Flow Policies

Jason Crampton, Naomi Farley, Gregory Gutin, and Mark Jones

Royal Holloway, University of London

Abstract. The simple security property in an information flow policy can be enforced by encrypting data objects and distributing an appropriate secret to each user. A user derives a suitable decryption key from the secret and publicly available information. A chain-based enforcement scheme provides an alternative method of cryptographic enforcement that does not require any public information, the trade-off being that a user may require more than one secret. For a given information flow policy, there will be many different possible chain-based enforcement schemes. In this paper, we provide a polynomial-time algorithm for selecting a chain-based scheme which uses the minimum possible number of secrets. We also compute the number of secrets that will be required and establish an upper bound on the number of secrets required by any user.

1 Introduction

Access control is a fundamental security service in modern computing systems and seeks to restrict the interactions between users of the system and the resources provided by the system. Generally speaking, access control is policy-based, in the sense that a policy is defined by the resource owner(s) specifying those interactions that are authorized. An attempt by a user to interact with a protected resource, typically called an *access request*, is evaluated by a trusted software component, the *policy decision point* (or *authorization decision function*), to determine whether the request should be permitted (if authorized) or denied (otherwise). The use of a policy decision point is entirely appropriate when we can assume the policy will be enforced by the same organization that defined it. However, use of third-party storage, privacy policies controlling access to personal data, and digital rights management all give rise to scenarios where this assumption does not hold.

An alternative approach to policy enforcement, and one that has attracted considerable interest in recent years, is to encrypt the protected object and enable authorized users to derive decryption keys. This approach is particularly suitable for data that changes infrequently, for read-only policies, and for policies that can be represented in terms of user attributes. Research into cryptographic access control began with the seminal work of Akl and Taylor [2] on the enforcement of information flow policies, and has seen a resurgence of interest in recent years.

Generally, it is undesirable to provide a user with all the keys she requires to decrypt protected objects. Instead, a user is given a small number of secrets from which she is able to derive all keys required. Thus a cryptographic enforcement scheme may be characterized by(i) the number of secrets each user has to store, (ii) the total number of secrets, (iii) the amount of auxiliary (public) information required for key derivation, and (iv) the amount of time required for key derivation.

Many schemes in the literature provide each user with a single secret [3, 11], the trade-off being that the amount of public information and derivation time may be substantial. In contrast, chain-based schemes require no public information but each user may require more than one secret [9, 14, 15]. In addition, chain-based schemes can achieve very strong security properties [15]. There are many different ways to instantiate a chain-based scheme for a given policy, each instantiation being defined by a chain partition of the partially ordered set that defines the policy.

However, existing work on chain-based CESs assumes the existence of a chain partition and simply generates the required secrets and keys for this partition [9, 14, 15]. This approach ignores the fact that there will be (exponentially) many choices of chain partition. Thus, it is important, if we are to make best use of chain-based CESs, that we know which chain partition to use for a given information flow policy. It is this issue that we address in this paper.

Contributions. Our first contribution (Theorem 2) is to show how $\widehat{K}(\Pi)$, the (total) number of secrets for a chain partition Π, is related to the set of edges in the representation of Π as an acyclic directed graph. We then prove that $\widehat{K}(\Pi)$ is determined by the end-points of the chains in Π (Lemma 2). This, in turn, allows us to prove there exists a chain partition that simultaneously minimizes the number of secrets required and the number of chains in the partition (Theorem 3). The last result is somewhat unexpected, as it is not usually possible to simultaneously minimize two different parameters. The result is also of practical importance, since the number of chains in Π provides a tight upper bound on the number of secrets required by any one user. Our main contribution (Theorem 1 and Section 4) is to develop a polynomial-time algorithm that enables us to find a chain partition Π such that $\widehat{K}(\Pi)$ and the number of chains is minimized (with respect to all chain partitions). Our algorithm is based on finding an optimal feasible flow in a network and makes use of the characterization of the number of secrets in terms of the set of edges (established in Theorem 2) to define the capacities of the edges in the network. We thereby provide rigorous foundations for the development of efficient chain-based enforcement schemes.

Paper structure. In the next section, we provide the relevant background on cryptographic enforcement schemes, formally define the problem, and state Theorem 1. In Sec. 3, we state and prove Theorems 2 and 3 and Lemma 2. In Sec. 4, we develop an efficient algorithm to derive the best chain partition and prove Theorem 1. We conclude the paper with a summary of our contributions and some ideas for future work.

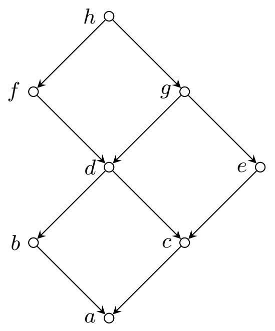


Fig. 1: The Hasse diagram of a simple poset

2 Background and Problem Statement

A *partially ordered set* (or *poset*) is a pair $(X, \leqslant)$, where $\leqslant$ is a reflexive, antisymmetric, transitive binary relation on X. We may write $x \geqslant y$ whenever $y \leqslant x$, and $y < x$ whenever $y \leqslant x$ and $y \neq x$. Given a poset $(X, \leqslant)$, it is convenient to introduce the following notation.

$$\downarrow x \overset{\text{def}}{=} \{y \in X : y \leqslant x\} \qquad \text{and} \qquad \uparrow x \overset{\text{def}}{=} \{y \in X : y \geqslant x\}$$

We will also make use of the following terminology and notation.

- We say x *covers* y, denoted $y \lessdot x$, if $y < x$ and there does not exist $z \in X$ such that $y < z < x$. We say y is a *child* of x if $y \lessdot x$ (and x is a *parent* of y).
- The *Hasse diagram* of a poset is the directed acyclic graph $H = (X, E_0)$, where $xy \in E_0$ if and only if $y \lessdot x$.
- X is a *tree* if no element of X has more than one parent and X has a unique maximum element.
- $Y \subseteq X$ is a *chain* (or *total order*) if for $x, y \in Y$, $x < y$ or $x = y$ or $y < x$. $\{C_1, \dots, C_\ell\}$ is a *chain partition* (of $(X, \leqslant)$) if $C_i \subseteq X$ is a chain, $C_i \cap C_j = \emptyset$ if $i \neq j$, and $C_1 \cup \cdots \cup C_\ell = X$.
- $Y \subseteq X$ is an *antichain* if for $x, y \in Y$, $x \leqslant y$ if and only if $x = y$. (In other words, for $x \neq y$ in an antichain, $x \not\leqslant y$ and $y \not\leqslant x$.) The *width* of a poset is the cardinality of an antichain of maximum size.

An illustrative Hasse diagram is shown in Fig. 1. In the poset depicted, $\{a, d, f\}$ is a chain, for example, and $\{d, e\}$ is an antichain of maximum size. Thus the width of this poset is 2 and one chain partition of cardinality 2 is $\{\{a, c, e, g, h\}, \{b, d, f\}\}$.

Definition 1. *An* information flow policy *is a tuple* $(X, \leqslant, U, O, \lambda)$*, where:*

- $(X, \leqslant)$ *is a (finite) partially ordered set of* security labels*;*
- U *is a set of* users *and* O *is a set of* objects*;*
- $\lambda : U \cup O \to X$ *is a* security function *that associates users and objects with security labels.*

The simple security property *requires that user $u \in U$ can read an object $o \in O$ if and only if* $\lambda(u) \geqslant \lambda(o)$.

We may define an equivalence relation $\sim$ on U, where $u \sim v$ if and only if $\lambda(u) = \lambda(v)$. We write U_x to denote $\{u \in U : \lambda(u) = x\}$; U is partitioned into the set of equivalence classes $\{U_x : x \in X\}$. Similarly, $O_x \subseteq O$ is the set of objects having security label $x \in X$. Thus, the simple security property guarantees that any $o \in O_x$ can be read by a user $u \in U_y$ for any $y \geqslant x$. Conversely, $u \in U_y$ can read $o \in O_x$ for any $x \leqslant y$. Henceforth, we will represent an information flow policy $(X, \leqslant, U, O, \lambda)$ as a pair $(X, \leqslant)$ with the tacit understanding that U, O and λ are given.

2.1 Cryptographic Enforcement of Information Flow Policies

One way of enforcing the simple security property (for policy $(X, \leqslant)$) is to encrypt $o \in O_y$ with a (symmetric) key $k(y)$ and provide all users in U_x, where $x \geqslant y$ with the key $k(y)$. An alternative is to provide a user u in U_x with a smaller number of keys (typically a single key for label x) and enable u to derive keys for all y such that $y < x$. However, this introduces the possibility that users may be able to collude and use their keys to derive a key that no single user could derive.

More formally, there exists the notion of a *cryptographic enforcement scheme* (CES), defined by the SetUp and Derive algorithms, SetUp being used to generate secrets and keys and the data used to derive secrets and keys, and Derive being used to compute secrets and keys. Let $\mathcal{K}$ denote an arbitrary key space (typically $\mathcal{K} = \{0,1\}^l$ for some $l \in \mathbb{N}$). Then SetUp and Derive have the following characteristics.

– SetUp takes as input a security parameter ρ and information flow policy $(X, \leqslant)$.
  It outputs, for each element $x \in X$, a pair $(\sigma(x), \kappa(x))$: the *secret* $\sigma(x)$ is given to all users in U_x; $\sigma(x)$ is used to derive secrets and/or keys for labels $y \leqslant x$; and the *key* $\kappa(x) \in \mathcal{K}$ is used to encrypt data objects in O_x.
  The SetUp algorithm also outputs a set of public information Pub, which is used for the derivation of secrets and keys.
– Derive takes as input $(X, \leqslant)$, Pub, start and end points $x, y \in X$ and $\sigma(x)$.
  It outputs $\kappa(y) \in \mathcal{K}$ if and only if $y \leqslant x$. (In particular, $\kappa(x)$ can be derived from $\sigma(x)$.)

The requirement that Derive outputs $\kappa(y)$ (given $\sigma(x)$) if $y \leqslant x$ is a correctness criterion, which ensures an authorized user can derive the keys required to decrypt objects. We also require a security criterion. Informally, the *strong key-indistinguishability* criterion requires the following.

> There is no polynomial time algorithm, given $z \in X$, a set of secrets $\sigma(Y) = \{\sigma(y) : y \in Y\}$ such that $z \not\leqslant y$ for any $y \in Y$, and $\kappa(x)$ for all $x \neq z$ (and the public information *Pub*), that can distinguish between $\kappa(z)$ and a random key in $\mathcal{K}$.

That is, an adversary cannot distinguish a key from random unless it may be computed from one of the secrets or keys known to the adversary (which implies, in particular, that the adversary can only compute such a key if it can be computed from one of those secrets); see Freire *et al.* [15] for further details.

2.2 Chain-based Enforcement

For certain classes of cryptographic enforcement schemes, public information is not required. In particular, if X is a chain, then (by definition) there is a unique directed path from x to y (in the Hasse diagram of X) whenever $y < x$. Then for $y \lessdot x$, we may define the *secret* $\sigma(y)$ to be $F(\sigma(x))$, and $\kappa(y) = H(\sigma(y))$, where F and H are suitable one-way functions. Thus, if $y < x$, there exist $z_1, \dots, z_\ell \in X$ with $y = z_1 \lessdot z_2 \lessdot \cdots \lessdot z_\ell = x$; $\kappa(y)$ may be derived from $\sigma(x)$ by iteratively deriving $\sigma(z_i) = F(\sigma(z_{i+1}))$, $i = \ell - 1, \dots, 1$, and then deriving $\kappa(y) = H(\sigma(y)) = H(\sigma(z_1))$.

This observation has led to the development of chain-based CESs [9, 14, 15] for arbitrary information flow policies. The basic idea is to partition the information flow policy $(X, \leqslant)$ into chains and then construct multiple CESs, one for each chain.

More formally, let $(X, \leqslant)$ be a poset and $C = x_1 > x_2 > \cdots > x_m$ be a chain in X. Then we say any chain of the form $x_j > x_{j+1} > \cdots > x_m$, $1 \leqslant j \leqslant m$, is a *suffix* of C; the empty chain is (vacuously) also a suffix of C.

Proposition 1. *For all $x \in X$ and any chain $C \subseteq X$, ${\downarrow}x \cap C$ is a suffix of C.*

The above result (due to Crampton *et al.* [9, Proposition 4]) enables us to define, for a given chain partition Π, the secrets that should be given to a user $u \in U_x$, since ${\downarrow}x$ defines the labels for which u is authorized. Given a chain partition $\Pi = \{C_1, \dots, C_\ell\}$, $\{{\downarrow}x \cap C_1, \dots, {\downarrow}x \cap C_\ell\}$ is a disjoint collection of chain suffixes. Hence, a user in U_x must be given the secrets for the maximal elements in the non-empty suffixes ${\downarrow}x \cap C_1, \dots, {\downarrow}x \cap C_\ell$. Thus, any user requires at most ℓ secrets. Let $\phi(x, \Pi) \subseteq X$ denote this set of maximal elements. (Clearly, $x \in \phi(x, \Pi)$ for all chain partitions Π and all $x \in X$.)

Remark 1. Let w be the width of a poset $(X, \leqslant)$. Clearly, $(X, \leqslant)$ cannot have a chain partition with less than w chains. Dilworth's theorem asserts that there exists a chain partition of $(X, \leqslant)$ into w chains [13]. Thus, if we can find a chain partition of X into w chains, no user will require more than w secrets. (If u were to have more secrets than there are chains in the partition, then there must exist a chain containing y and z for which u has secrets and one of the secrets may be derived from the other.)

Freire *et al.* [15] provide a formal description of the SetUp and Derive algorithms. Informally, the SetUp algorithm performs the following steps:

1. for each chain C_i in Π, select a secret for the top element in C_i and generate a secret for each element in the chain by applying the one-way function F to the secret of its parent in C_i;

2. for each element $x \in X$, generate $\kappa(x)$ by applying the one-way function H to $\sigma(x)$;
3. assign the secrets $\sigma(\phi(x, \Pi)) \overset{\text{def}}{=} \{\sigma(z) : z \in \phi(x, \Pi)\}$ to each user in U_x.

The Derive algorithm performs the following steps, given $x, y \in X$ and $\sigma(\phi(x, \Pi))$:

1. if $x = y$, then output $H(\sigma(x))$;
2. if $y < x$, then find $z \in \phi(x, \Pi)$ such that $z \geqslant y$, so there exist $z = z_0 \gtrdot_\Pi \cdots \gtrdot_\Pi z_t = y$, and compute $F(\sigma(z_0)) = \sigma(z_1), \dots, F(\sigma(z_{t-1})) = \sigma(y)$; output $H(\sigma(y))$.

This scheme has the strong key-indistinguishability property; see Freire *et al.* [15] for further details.

A user in U_x will need to be given $|\phi(x, \Pi)|$ secrets, in contrast to most CESs in the literature in which each user receives a single secret [3, 11]. However, chain-based CESs have substantial benefits:(i) they require no public information [9]; (ii) they can use cryptographic primitives that are very easy to compute; and (iii) it is easy to construct schemes with the strong key-indistinguishability property [15].

2.3 Problem Statement

Certain aspects of chain-based CESs are not well understood. As we have already noted, some users will require multiple secrets, each of which corresponds to a unique label in X. In particular, a user u in U_x will require a secret for each chain that contains an element y such that $y < x$. Three chain partitions of the poset in Fig. 1 are shown in Fig. 2. We have, for example, $\phi(g, \Pi_1) = \{b, e, g\}$, $\phi(g, \Pi_2) = \{b, d, g\}$, and $\phi(g, \Pi_3) = \{d, g\}$. Hence, the number of secrets required, on a per-user basis and in total, will vary, depending on the chain partition chosen. Thus, considering various chain partitions of X, we may ask:

– How do we minimize $k_{\max}$, the maximum number of secrets a user may require?
– How do we minimize K, the total number of secrets required?
– How do we minimize $\widehat{K}$, the total number of secrets that need to be issued to users?

More formally, given a chain partition Π of $(X, \leqslant)$, we may regard ϕ as a function from X to 2^X that is completely determined by Π. Thus, given a chain partition Π, we can define the following values.

$$k_{\max}(\Pi) \overset{\text{def}}{=} \max\{|\phi(x, \Pi)| : x \in X\}$$
$$K(\Pi) \overset{\text{def}}{=} \sum_{x \in X} |\phi(x, \Pi)|$$
$$\widehat{K}(\Pi) \overset{\text{def}}{=} \sum_{x \in X} |U_x| \cdot |\phi(x, \Pi)|$$

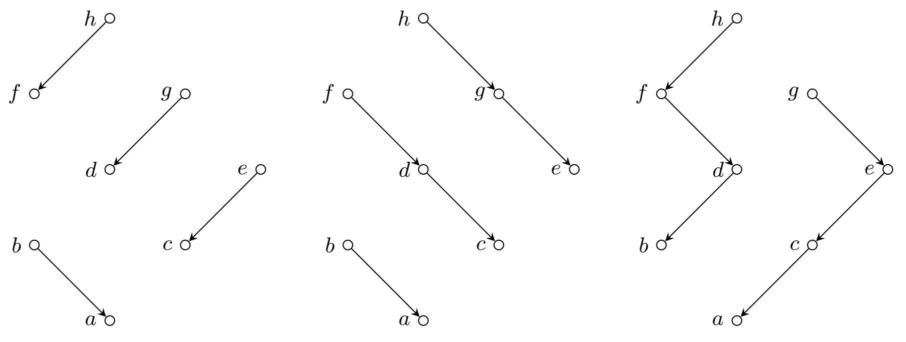


Fig. 2: Three chain partitions of the poset in Fig. 1

Values of $k_{\max}$ and K for the chain partitions in Fig. 2 are shown in Table 1; node h is used for illustrative purposes.[1]

Partition	$\phi(h)$	$k_{\max}$	K
$\mathit{\Pi}_1$	$\{b, e, g, h\}$	4	20
$\mathit{\Pi}_2$	$\{b, f, h\}$	3	17
$\mathit{\Pi}_3$	$\{g, h\}$	2	13

Table 1: $\phi(h)$, $k_{\max}$ and K for the chain partitions in Fig. 2

The important question is: Can we minimize these parameters (over all choices of chain partition $\mathit{\Pi}$ for X)? In short, given an information flow policy $(X, \leqslant)$, how do we determine $\mathit{\Pi}$ for use in a chain-based CES?[2] It is this question we address in the remainder of the paper. In particular, at the end of Section 4, we prove the following result.

Theorem 1. *Let $(X, \leqslant)$ be an information flow policy of width w and let $\widehat{K}$ denote the minimum number of secrets required by a chain-based enforcement scheme for X. Then in $O(|X|^4 w)$ time, we can find a chain partition $\mathit{\Pi}$ for which the corresponding chain-based enforcement scheme only requires $\widehat{K}$ secrets and $k_{\max} \leqslant w$.*

[1] Note that we can deduce K from $\widehat{K}$ by letting $|U_x| = 1$ for all $x \in X$.

[2] Crampton *et al.* [9] observed that further research was needed to identify the best choice of chain partition for a given information flow policy. While subsequent research has formalized [14] and strengthened the security properties of chain-based CESs [15], we are not aware of any research that specifies how to select a chain partition.

Remark 2. We assume throughout that our information flow policy has a maximum element. We may assume this without loss of generality: given an information flow policy $(X, \leqslant)$ without a maximum element, we simply add a maximum element r and define $r \gtrdot m$ for all maximal elements m in X; no users are assigned to r. Observe that such a transformation does not affect the values of $k_{\max}$ and $\widehat{K}$.

3 Computing $k_{\max}$ and $\widehat{K}$

Informally, we take a poset $(X, \leqslant)$ and construct a second poset $(X, \leqslant')$, where $x <' y$ implies $x < y$ (but $x < y$ does not necessarily imply $x <' y$). We will say $\leqslant'$ is *contained* in $\leqslant$. In particular, any chain partition Π of $(X, \leqslant)$ defines a second poset $(X, \leqslant_\Pi)$, where $x <_\Pi y$ if and only if x and y belong to the same chain and $x < y$; thus $\leqslant_\Pi$ is contained in $\leqslant$ for any Π. Note, however, that $x \lessdot_\Pi y$ does not necessarily imply $x \lessdot y$.[3]

Given a poset $(X, \leqslant)$ and $z < y$, we define

$$\gamma(yz) = \{x \in X : x \geqslant z, x \not\geqslant y\} .$$

Thus $z \in \gamma(yz)$ and $y \notin \gamma(yz)$. For the maximum element $r \in X$ and any $y, z \in X$ such that $z < y$, $r \notin \gamma(yz)$. Informally, the intuition behind γ is that its cardinality measures the "damage" that would be done by creating a chain partition Π such that $z \lessdot_\Pi y$, because having $z \lessdot_\Pi y$ means that $z \not\leqslant_\Pi x$ for any $x \in \gamma(yz)$. Thus, every user in U_x will require an extra secret in order to derive $\kappa(z)$. We will capture this intuition more precisely in Lemma 1.

Remark 3. For maximum element r and any chain partition $\Pi = \{C_1, \dots, C_\ell\}$, $\phi(r, \Pi) = \{t_1, \dots, t_\ell\}$, where t_i is the maximum element in chain C_i. Moreover, $r = t_i$ for some i. Hence, we can construct a tree $\widetilde{\Pi} = (X, \leqslant_{\widetilde{\Pi}})$, where $y \lessdot_{\widetilde{\Pi}} x$ if and only if one of the following conditions holds:(i) $y = t_j$, $j \neq i$, and $x = r$; (ii) $y \lessdot_\Pi x$.

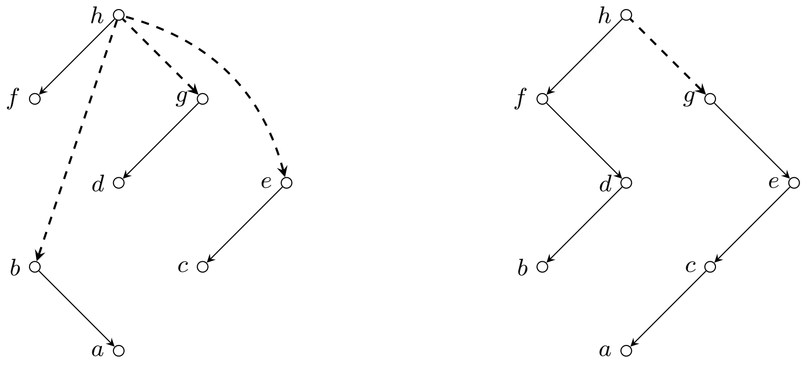
Figure 3 illustrates the construction of two such trees, using chain partitions from Fig. 2; the arcs used to create the trees are shown as dashed lines.

Lemma 1. *Let $(X, \leqslant)$ be a poset and let Π be a chain partition of X. Then, for all $x, y, z \in X$ such that $x \neq r$ and $z \lessdot_{\widetilde{\Pi}} y$,*

$$z \in \phi(x, \Pi) \text{ if and only if } x \in \gamma(yz).$$

Proof. Given $z \in \phi(x, \Pi)$ and chain partition $\Pi = \{C_1, \dots, C_\ell\}$, $y_i \in \phi(x, \Pi) \cap C_i$ if and only if $C_i \cap {\downarrow}x$ is non-empty and y_i is the maximum element in $C_i \cap {\downarrow}x$ (Sec. 2.3). Thus, $z \leqslant x$. Moreover, $x \not\geqslant y$ (otherwise there would exist $t \in \phi(x)$ such that $y \leqslant_{\widetilde{\Pi}} t$ and hence $z \lessdot_{\widetilde{\Pi}} y \leqslant_{\widetilde{\Pi}} t$, violating the condition that z is the maximum element in the suffix $C_i \cap {\downarrow}x$). That is, $x \in \gamma(yz)$.

[3] To see this, consider the poset of four elements, in which $a \lessdot b \lessdot d$ and $a \lessdot c \lessdot d$ with $b \not\leqslant c, c \not\leqslant b$. Then $\{\{b\}, \{c\}, \{a, d\}\}$ is a chain partition and $a \lessdot_\Pi d$, but $a \not\lessdot d$.



(a) $\widetilde{\Pi}_1$ (b) $\widetilde{\Pi}_3$

Fig. 3: Creating trees from partitions Π_1 and Π_3 in Fig. 2

Now suppose $x \in \gamma(yz)$. Then $x \not\geqslant y$, by definition, and hence y does not belong to $\downarrow x \cap C_i$ for any i. However, $x \geqslant z$; hence, there exists $t \in \phi(x)$ such that $z \leqslant_{\widetilde{\Pi}} t$. Since Π is a chain partition, the only parent of z in $\widetilde{\Pi}$ is y. Hence it must be the case that $z = t$ (and thus $z \in \phi(x)$). □

Let $(X, \leqslant)$ be an information flow policy and let $y, z \in X$ with $z < y$. Then, following Crampton *et al.* [10], we define

$$\omega(yz) \overset{\text{def}}{=} \sum_{x \in \gamma(yz)} |U_x| .$$

We will be interested in minimizing the $\sum \omega(yz)$, where the sum is taken over all pairs (y, z) such that $z \lessdot_{\widetilde{\Pi}} y$. The intuition behind this definition is that it captures, in some appropriate sense, the connectivity that is lost from $(X, \leqslant)$ by using $(X, \leqslant_\Pi)$. Since every element in $(X, \leqslant_\Pi)$ has at most one parent, $\gamma(yz)$ represents those elements in X that become "disconnected" from z by defining $z \lessdot_\Pi y$. The next result establishes an exact correspondence between $\phi(x, \Pi)$ and $\gamma(yz)$, and enables us to use network flow techniques to compute a chain partition that minimizes $\widehat{K}$ (as we explain in Sec. 4).

Theorem 2. *Let $(X, \leqslant_\Pi)$ be a chain partition of $(X, \leqslant)$ with maximum element r. Then*

$$\widehat{K}(\Pi) = \ell\, |U_r| + \sum_{z \lessdot_{\widetilde{\Pi}} y} \omega(yz)$$

where ℓ is the number of chains in Π.

Proof. By definition,

$$\widehat{K}(\Pi) = \sum_{x \in X} |U_x|\, |\phi(x, \Pi)| = |U_r|\, |\phi(r, \Pi)| + \sum_{x \in X \setminus r} \sum_{z \in X} |U_x|\, \delta(x, z),$$

where $\delta(x,z)$ equals 1 if $z \in \phi(x,\Pi)$ and 0 otherwise. By Lemma 1, we have $\delta(x,z) = 1$ if and only if $x \in \gamma(yz)$ for $z \lessdot_{\widetilde{\Pi}} y$. Moreover, y is unique, since $\widetilde{\Pi}$ is a tree. Therefore,

$$\sum_{x \in X \setminus r} \sum_{z \in X} |U_x| \, \delta(x,z) = \sum_{z \lessdot_{\widetilde{\Pi}} y} \sum_{x \in \gamma(yz)} |U_x| = \sum_{z \lessdot_{\widetilde{\Pi}} y} \omega(yz)$$

As $r \geqslant x$ for all $x \in X$, $\phi(r,\Pi)$ must contain exactly one element from each chain in Π. Therefore $|U_r| \, |\phi(r,\Pi)| = \ell \, |U_r|$, as required. □

The following result shows that the number of secrets required by a chain partition can be computed by considering only the minimum elements in the chain partition.

Lemma 2. *Let $\Pi = \{C_1, \dots, C_\ell\}$ be a chain partition of $(X, \leqslant)$ and let chain C_i have bottom element b_i, $1 \leqslant i \leqslant \ell$. Then*

$$K(\Pi) = \sum_{i=1}^{\ell} |{\uparrow} b_i| \qquad \textit{and} \qquad \widehat{K}(\Pi) = \sum_{i=1}^{\ell} \sum_{x \in {\uparrow} b_i} |U_x| \, .$$

Proof. We have, by definition,

$$\begin{aligned}
\widehat{K}(\Pi) &= \sum_{x \in X} |U_x| \, |\phi(x,\Pi)| = \sum_{x \in X} |U_x| \, |\{C_i : C_i \cap {\downarrow} x \neq \emptyset, 1 \leqslant i \leqslant \ell\}| \\
&= \sum_{x \in X} |U_x| \, |\{b_i : x \geqslant b_i, 1 \leqslant i \leqslant \ell\}| \\
&= \sum_{x \in X} \sum_{i=1}^{\ell} |U_x| \, \delta(x,b_i) \qquad \text{where } \delta(x,b_i) = 1 \text{ if } x \geqslant b_i \text{ and 0 otherwise} \\
&= \sum_{i=1}^{\ell} \sum_{x \in X} |U_x| \, \delta(x,b_i) = \sum_{i=1}^{\ell} \sum_{x \in {\uparrow} b_i} |U_x|
\end{aligned}$$

Clearly, we may prove the result for K in an analogous fashion. □

In Fig. 2a, for example, the bottom elements are a, c, d and f and $|{\uparrow} a| = 8$, $|{\uparrow} c| = 6$, $|{\uparrow} d| = 4$ and $|{\uparrow} f| = 2$. Thus, the number of secrets required in total is 20.

Theorem 3. *Let $(X, \leqslant)$ be an information flow policy of width w and let $\widehat{K}$ denote the minimum number of secrets required by a chain-based enforcement scheme for X. Then there exists a chain partition containing w chains such that $\widehat{K}(\Pi) = \widehat{K}$.*

Proof. Let Π be a chain partition of X into $t \geqslant w$ chains such that $\widehat{K}(\Pi) = \widehat{K}$ and let B be the set of bottom vertices in the chains of Π. A result of Gallai and Milgram asserts that if a chain partition Π of a poset $(X, \leqslant)$ contains t chains, where $t > w$, then there exists a chain partition Π' into $t-1$ chains such that the set of bottom vertices in Π' is a subset of B [16].[4] Hence, by

[4] The result is phrased in the language of digraphs, but every poset may be represented by an equivalent transitive acyclic digraph.

iterated applications of the Gallai-Milgram result, there exists a chain partition Π^* of width w such that the set of bottom vertices B^* in Π^* is a subset of B. Moreover, by Lemma 2,

$$\widehat{K}(\Pi^*) = \sum_{b \in B^*} \sum_{x \in \uparrow b} |U_x| \leqslant \sum_{b \in B} \sum_{x \in \uparrow b} |U_x|$$

By the minimality of $\widehat{K}$, we deduce that $\widehat{K}(\Pi^*) = \widehat{K}$. □

Corollary 1. *Let $(X, \leqslant)$ be an information flow policy. There exists a chain partition such that the total number of secrets $\widehat{K}$ is minimized and $k_{\max} \leqslant w$.*

Proof. The result follows immediately from Theorem 3, the definition of $k_{\max} = \max\{|\phi(x, \Pi)| : x \in X\}$, and the fact that $|\phi(x, \Pi)|$ is bounded above by the number of chains in Π for all $x \in X$. □

4 Finding a Chain Partition Requiring $\widehat{K}$ Keys

Suppose $(X, \leqslant)$ is a poset of width w. In general, a chain partition of X has $\ell \geqslant w$ chains. Theorem 3 asserts that there exists a partition of X into w chains such that the corresponding enforcement scheme requires the minimum number of secrets. We now show how such a chain partition may be constructed. In particular, we show how to transform the problem of finding a chain partition Π such that $\widehat{K}(\Pi)$ attains the minimum value into a problem of finding a minimum cost flow in a network.

Informally, a *network* is a directed graph in which each edge is associated with a *capacity*. A *network flow* associates each edge in a given network with a flow, which must not exceed the capacity of the edge. Networks are widely used to model systems in which some quantity passes through channels (edges in the network) that meet at junctions (vertices); examples include traffic in a road system, fluids in pipes, or electrical current in circuits. In our setting, we model an information flow policy as a network in which the capacities are determined by the weights ω. Our definitions for networks and network flows follow the presentation of Bang-Jensen and Gutin [5].

Definition 2. *A* network *is a tuple $\mathcal{N} = (D, l, u, c, b)$, where:*

- *$D = (V, A)$ is a directed graph with vertex set V and arc set A;*
- *$l : V \times V \to \mathbb{N}$ such that $l(vv') = 0$ if $vv' \notin A$ and $l(vv') \geqslant 0$ otherwise;*
- *$u : V \times V \to \mathbb{N}$ such that $u(vv') = 0$ if $vv' \notin A$ and $u(vv') \geqslant l(vv') \geqslant 0$ otherwise;*
- *$c : V \times V \to \mathbb{R}$;*
- *$b : V \to \mathbb{R}$ such that $\sum_{v \in V} b(v) = 0$.*

Intuitively, l and u represent lower and upper bounds, respectively, on how much flow can pass through each arc, and c represents the cost associated with each unit of flow in each arc. The function b represents how much flow should enter or leave the network at a given vertex. If $b(x) = 0$, then the flow going into x should be equal to the flow going out of x. If $b(x) > 0$, then there should be $b(x)$ more flow coming out of x than going into x. If $b(x) < 0$, there should be $|b(x)|$ more flow going into x than coming out of x.

Definition 3. *Given a network $\mathcal{N} = (D, l, u, c, b)$, a function $f : V \to \mathbb{N}$ is a* feasible flow *for $\mathcal{N}$ if the following conditions are satisfied:*

- *$u(vv') \geqslant f(vv') \geqslant l(vv')$ for every $vv' \in V \times V$;*
- *$\sum_{v' \in V}(f(vv') - f(v'v)) = b(v)$ for every $v \in V$.*

The cost *of f is defined to be*

$$\sum_{vv' \in A} c(vv')f(vv').$$

Our aim is to find a tree $\widetilde{\Pi}$ such that Π is a chain partition of X with w chains that minimizes $\widehat{K}$. To do this, we will construct a network $\mathcal{N}$ such that the minimum cost flow of $\mathcal{N}$ corresponds to the desired tree. We can then find the minimum cost flow of $\mathcal{N}$ in polynomial time.

In $\widetilde{\Pi}$, we want every vertex except r to have at most one parent and at most one child. We cannot represent this requirement directly in a network. However, we can use the *vertex splitting procedure* [5] to simulate it. Specifically, given poset $(X, \leqslant)$, define $X_{\text{in}} = \{x_{\text{in}} : x \in X \setminus \{r\}\}$ and $X_{\text{out}} = \{x_{\text{out}} : x \in X\}$; and define $v' \prec v$ if and only if either $v = x_{\text{in}}$ and $v' = x_{\text{out}}$ for some $x \in X \setminus r$, or $v = x_{\text{out}}$ and $v' = y_{\text{in}}$ for some $x, y \in X$ such that $y < x$. We now add a minimum element $\perp$, where $\perp \prec x_{\text{out}}$ for all $x \in X$.

Then define $D = (X_{\text{in}} \cup X_{\text{out}} \cup \{\perp\}, A)$, where $xy \in A$ if and only if $y \prec x$, and the network (D, l, u, c, b), where

$$l(vv') = \begin{cases} 1 & \text{if } v = x_{\text{in}}, v' = x_{\text{out}}, x \in X \setminus r \\ 0 & \text{otherwise;} \end{cases}$$

$$u(vv') = \begin{cases} 1 & \text{if } v' \prec v \\ 0 & \text{otherwise;} \end{cases}$$

$$c(vv') = \begin{cases} \omega(xy) & \text{if } v = x_{\text{out}}, v' = y_{\text{in}}, y \leqslant x \\ 0 & \text{otherwise;} \end{cases}$$

$$b(v) = \begin{cases} w & \text{if } v = r_{\text{out}} \\ -w & \text{if } v = \perp \\ 0 & \text{otherwise.} \end{cases}$$

We call this network the *network chain-representation of* $(X, \leqslant)$. Note that any feasible flow f for this network must have $0 \leqslant f(xy) \leqslant 1$ for all $xy \in A$.

Lemma 3. *Let $\mathcal{N}$ be the network chain-representation of poset $(X, \leqslant)$. Then the minimum number of secrets required by a chain-based enforcement scheme for $(X, \leqslant)$ with w chains is $w\,|U_r| + \widehat{f}$, where $\widehat{f}$ is the minimum cost of a feasible flow in $\mathcal{N}$.*

Proof. Suppose we are given a chain partition Π with w chains. Then we may construct the tree $\widetilde{\Pi}$. Consider the following flow:

$$\begin{aligned} f(x_{\text{in}}x_{\text{out}}) &= 1 && \text{for all } x \in X \setminus r; \\ f(x_{\text{out}}y_{\text{in}}) &= 1 && \text{if } y \lessdot_{\widetilde{\Pi}} x; \\ f(x_{\text{out}}\bot) &= 1 && \text{if } x \text{ is a bottom element in a chain in } \Pi; \\ f &= 0 && \text{otherwise.} \end{aligned}$$

Then we can show that f is a feasible flow. Indeed, by construction all arcs xy satisfy $u(xy) \geqslant f(xy) \geqslant l(xy)$. In the graph formed by arcs xy with $f(xy) = 1$, it is clear that every vertex x has in-degree and out-degree 1, except for r_{out} and $\bot$. As there is one element y such that $y \lessdot_{\widetilde{\Pi}} r$ for each chain in Π, r_{out} has in-degree 0 and out-degree w in this graph, and similarly $\bot$ has in-degree w and out-degree 0. As all arcs xy have $f(xy) = 1$ or $f(xy) = 0$, we have that

$$\sum_{v \in V(D)} (f(xv) - f(vx)) = b(x)$$

for all x, as required. Moreover, the cost of f equals $\sum_{x \lessdot_{\widetilde{\Pi}} y} \omega(yx)$.

Conversely, suppose f is a feasible flow for $\mathcal{N}$. Then we define $y \lessdot_f x$ if and only if $f(x_{\text{out}}, y_{\text{in}}) = 1$. For each $x \in X \setminus r$, the arc $x_{\text{in}}x_{\text{out}}$ is the only in-coming arc for x_{out} and the only out-going arc for x_{in} in D, and by definition of $\mathcal{N}$, $f(x_{\text{in}}x_{\text{out}}) = 1$. As $b(x_{\text{in}}) = b(x_{\text{out}}) = 0$ and all in-coming arcs for x_{in} are of the form $y_{\text{out}}x_{\text{in}}$, it follows that there is exactly one element $y \in X$ such that $x \lessdot_f y$, and at most one element $z \in X$ such that $z \lessdot_f x$. As $b(r_{\text{out}}) = w$ and r_{out} has no in-coming arcs in D, and all its out-going arcs are of the form $r_{\text{out}}x_{\text{in}}$, there are exactly w elements y such that $y \lessdot_f r$. Let these elements be labelled $t_1, \dots, t_w$.

Now choose an arbitrary i, $1 \leqslant i \leqslant w$, and define $y \lessdot_\Pi x$ if and only if $x = r$ and $y = t_i$, or $x \neq r$ and $y \lessdot_f x$. Then for every element $x \in X$, there is at most one element $y \in X$ such that $x \lessdot_\Pi y$, and at most one element $z \in X$ such that $z \lessdot_\Pi x$.

It is easy to see that $\leqslant_\Pi$, the reflexive, transitive closure of $\lessdot_\Pi$, defines a chain partition of X. (Observe that as D is an acyclic digraph, the transitive reflexive closure of $\lessdot_\Pi$ is antisymmetric, and therefore a partial order. The fact that $(X, \leqslant_\Pi)$ is a chain partition can be shown by induction on $|X|$, considering X with a minimal element removed for the induction step.) By construction, the only maximal elements for $\leqslant_\Pi$ are r and the elements t_j for $j \neq i$. Thus, $(X, \leqslant_\Pi)$ has w chains.

Recall the definition of $\lessdot_{\widetilde{\Pi}}$, that $y \lessdot_{\widetilde{\Pi}} x$ if and only if either $y \lessdot_\Pi x$, or $y = t_j$, $j \neq i$, and $x = r$. Note that $\lessdot_{\widetilde{\Pi}}$ is exactly the relation $\lessdot_f$. By Theorem 2, the number of secrets required by Π is

$$w\,|U_r| + \sum_{z \lessdot_{\widetilde{\Pi}} y} \omega(yz).$$

As $z \lessdot_{\widetilde{\Pi}} y$ if and only if $f(y_{\text{out}}z_{\text{in}}) = 1$, $c(y_{\text{out}}z_{\text{in}}) = \omega(yz)$, and $c(uv) = 0$ for all other arcs with $f(uv) = 1$, we have that $\sum_{z \lessdot_{\widetilde{\Pi}} y} \omega(yz)$ is exactly the cost of f, as required. $\square$

Lemma 4. *We can find a minimum cost flow for $\mathcal{N}$ in $O(|X|^4 w)$ time.*

Proof. The Negative Cycle algorithm (see [1, §5.3], for example) finds a minimum cost flow for a network with n vertices and m arcs in time $O(nm^2CU)$, where C denotes the maximum cost on an arc, and U denotes the maximum of all upper bounds on arcs and the absolute values of all balance demands on vertices. By construction of $\mathcal{N}$, we have that $n = 2|X| = O(|X|)$, $m = O(n^2) = O(|X|^2)$, $C = \max\{\omega(xy) : xy \in E_0^*\} = O(|X|)$, $U = 1$ and $C = w$. Thus we get the desired running time. □

Remark 4. Strictly speaking, the Negative Cycle algorithm assumes that all lower bounds on arcs are 0. However, we can satisfy this assumption, given $\mathcal{N} = (D, l, u, c, b)$, by defining the network $\mathcal{N}' = (D, l', u', c, b')$, where

$$\begin{aligned} l'(xy) &= 0 & b'(x) &= b(x) - l(xy) \\ u'(xy) &= u(xy) - l(xy) & b'(y) &= b(y) + l(xy) \end{aligned}$$

Then the minimum cost flow f' for $\mathcal{N}'$ will have cost exactly $\sum_{xy} l(xy)c(xy)$ less than the minimum cost flow for $\mathcal{N}$, and f' can be transformed into a minimum cost feasible flow f for $\mathcal{N}$ by setting $f(xy) = f'(xy) + l(xy)$.

We are now able to prove our main result, which is, essentially, a corollary of Theorem 3 and Lemmas 3 and 4.

Proof (of Theorem 1). By Theorem 3, there exists a chain partition that has exactly w chains, for which the corresponding chain-based enforcement scheme only requires $\widehat{K}$ secrets. Then by Lemma 3, $\widehat{K}$ is equal to the minimum cost of a feasible flow in $\mathcal{N}$, the network chain-representation of $(X, \leqslant)$. By Lemma 4, such a flow can be found in $O(|X|^4 w)$ time, and this flow can be easily transformed into the corresponding chain partition Π. Finally, by definition of $\phi(x, \Pi)$, $|\phi(x, \Pi)| \leq w$ for each $x \in X$ and therefore $k_{\max} \leqslant w$. □

5 Concluding Remarks

Cryptographic enforcement schemes (CESs) fall into two broad categories: those that use symmetric cryptographic primitives and those that use asymmetric ones (notably attribute-based encryption [6, 17]). The focus of this paper is on symmetric schemes, which may be characterized by(i) the total number of secrets required, (ii) the number of secrets required per user, (iii) the total amount of public information required for the derivation of secrets, and (iv) the number of derivation steps required.

Until recently, symmetric CESs for information flow policies have assumed each user would be given a single secret, from which other secrets and decryption keys would be derived using public information generated by the scheme administrator (see, for example, [3, 11]). In this setting, there is a considerable literature on the trade-offs that are possible by reducing the number of steps required for the derivation of secrets, at the cost of increasing the amount of public information (see, for example, [4, 8, 12]).

One drawback of these types of CESs is that the administrator must generate and publish information to facilitate the derivation of secrets (and decryption keys). Moreover, the amount of public information required may be substantial, particularly when security labels are defined in terms of (subsets of) attributes. Chain-based CESs obviate the requirement for public information, the trade-off being that each user may require several secrets. The chain-based approach may well be much more practical, particularly if the poset is large and its Hasse diagram contains many edges (as in a powerset, for example). Moreover, chain-based CESs may be implemented using one-way functions, typically the fastest of cryptographic primitives in practice.

However, it was not known which choice of chain partition was most appropriate for a given information flow policy. Our work provides formal and practical methods for constructing a chain partition with the smallest number of keys in total, with the additional property that no user is required to have more than w keys, where w is the width of the information flow policy.

One question remains: If there exist multiple chain partitions that minimize the number of keys in total and per-user, which of these should we choose and can we compute it efficiently? The one parameter that our work does not address is the number of derivation steps d required by a user in the worst case. Our future work, then, will attempt to find a polynomial-time or fixed-parameter algorithm that takes a poset as input and outputs a chain partition into w chains that minimizes d. We also hope to investigate whether the insight provided by Lemma 2—that $\widehat{K}(\Pi)$ is completely determined by the bottom elements in Π—can be exploited to design an algorithm whose performance improves on that of the algorithm described in Section 4.

Acknowledgements. The authors would like to thank Betram Poettering for his valuable feedback and the reviewers for their comments.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows: Theory, Algorithms, and Applications. Prentice Hall, Englewood Cliffs, NJ (1993)
2. Akl, S., Taylor, P.: Cryptographic solution to a problem of access control in a hierarchy. ACM Transactions on Computer Systems 1(3), 239–248 (1983)
3. 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)
4. Atallah, M.J., Blanton, M., Frikken, K.B.: Incorporating temporal capabilities in existing key management schemes. In: Biskup, J., Lopez, J. (eds.) ESORICS. Lecture Notes in Computer Science, vol. 4734, pp. 515–530. Springer (2007)
5. Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications. Springer, 2nd edn. (2009)
6. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA. pp. 321–334. IEEE Computer Society (2007)

7. Ciriani, V., De Capitani di Vimercati, S., Foresti, S., Jajodia, S., Paraboschi, S., Samarati, P.: Combining fragmentation and encryption to protect privacy in data storage. ACM Trans. Inf. Syst. Secur. 13(3) (2010)
8. Crampton, J.: Practical and efficient cryptographic enforcement of interval-based access control policies. ACM Trans. Inf. Syst. Secur. 14(1), 14 (2011)
9. Crampton, J., Daud, R., Martin, K.M.: Constructing key assignment schemes from chain partitions. In: Foresti, S., Jajodia, S. (eds.) DBSec. Lecture Notes in Computer Science, vol. 6166, pp. 130–145. Springer (2010)
10. Crampton, J., Farley, N., Gutin, G., Jones, M., Poettering, B.: Cryptographic enforcement of information flow policies without public information. CoRR abs/1410.5567 (2014), `http://arxiv.org/abs/1410.5567`, to appear in *Proceedings of ACNS 2015*
11. Crampton, J., Martin, K.M., Wild, P.R.: On key assignment for hierarchical access control. In: 19th IEEE Computer Security Foundations Workshop, (CSFW-19 2006), 5-7 July 2006, Venice, Italy. pp. 98–111. IEEE Computer Society (2006)
12. D'Arco, P., De Santis, A., Ferrara, A.L., Masucci, B.: Security and tradeoffs of the Akl-Taylor scheme and its variants. In: Královic, R., Niwinski, D. (eds.) MFCS. Lecture Notes in Computer Science, vol. 5734, pp. 247–257. Springer (2009)
13. Dilworth, R.: A decomposition theorem for partially ordered sets. Annals of Mathematics 51, 161–166 (1950)
14. Freire, E.S.V., Paterson, K.G.: Provably secure key assignment schemes from factoring. In: Parampalli, U., Hawkes, P. (eds.) Information Security and Privacy - 16th Australasian Conference, ACISP 2011, Melbourne, Australia, July 11-13, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6812, pp. 292–309. Springer (2011)
15. Freire, E.S.V., Paterson, K.G., Poettering, B.: Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In: Dawson, E. (ed.) Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7779, pp. 101–114. Springer (2013)
16. Gallai, T., Milgram, A.N.: Verallgemeinerung eines Graphentheoretischen Satzes von Rédei. Acta Sci. Math. 21, 181–186 (1960)
17. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Ning, P., di Vimercati, S.D.C., Syverson, P.F. (eds.) Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. pp. 195–203. ACM (2007)