Cryptographic Enforcement of Information Flow Policies without Public Information via Tree Partitions [1]

Jason Crampton [a,*], Naomi Farley [a], Gregory Gutin [a], Mark Jones [a], and Bertram Poettering [b]

[a] *Royal Holloway, University of London*
[b] *Ruhr University Bochum*

Abstract. We may enforce an information flow policy by encrypting a protected resource and ensuring that only users authorized by the policy are able to decrypt the resource. In most schemes in the literature that use symmetric cryptographic primitives, each user is assigned a single secret and derives decryption keys using this secret and publicly available information. Recent work has challenged this approach by developing schemes, based on a chain partition of the information flow policy, that do not require public information for key derivation, the trade-off being that a user may need to be assigned more than one secret. In general, many different chain partitions exist for the same policy and, until now, it was not known how to compute an appropriate one.

In this paper, we introduce the notion of a tree partition, of which chain partitions are a special case. We show how a tree partition may be used to define a cryptographic enforcement scheme and prove that such schemes can be instantiated in such a way as to preserve the strongest security properties known for cryptographic enforcement schemes. We establish a number of results linking the amount of secret material that needs to be distributed to users with a weighted acyclic graph derived from the tree partition. These results enable us to develop efficient algorithms for deriving tree and chain partitions that minimize the amount of secret material that needs to be distributed.

Keywords: access control, information flow policies, cryptographic enforcement, chains, forests, trees

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. Traditionally, 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, typi-

[1] This paper generalizes and extends our earlier results [16,15]. In particular, we define a new form of enforcement scheme that subsumes chain-based [15] and tree-based enforcement schemes [16]. We generalize results specific to these earlier schemes in order to support our more general framework.

[*] Corresponding author: Information Security Group, Royal Holloway, University of London, Egham, TW20 9QY, Egham; +44 1784 443117; jason.crampton@rhul.ac.uk

cally 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.

Cryptographic access control provides an alternative way of regulating access to data objects and has attracted considerable attention in recent years. In this setting, data objects are encrypted and appropriate decryption keys are issued to authorized users. Research into cryptographic access control began with the seminal work of Akl and Taylor [1], and has seen a resurgence of interest in recent years. For instance, there has been a considerable amount of research into attribute-based encryption [8,24], which is regularly used to support access control (see [28], for example). Attribute-based encryption is based on asymmetric cryptographic primitives, which means that any user is able to control read access to data (by encrypting), while only authorized users may decrypt. However, access control policies can also be enforced using symmetric cryptographic primitives (often a cheaper alternative to their asymmetric counterparts). Typically, in this scenario, a specific user – the data owner – encrypts all data objects before transmitting them to a storage provider that is only trusted to store data correctly. Users are able to retrieve data objects from the storage provider (in encrypted form) and only authorized users should be able to decrypt them.

In the symmetric setting, the focus of research has been on enforcing information flow policies [7], not least because many access control requirements may be articulated as information flow policies. An information flow policy is defined by a partially ordered set of security labels and a function mapping each user and data object to a security label. A user is authorized to read any data object associated with a security label that is less than or equal to that of the user.

Generally, it is undesirable to explicitly 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.[1] Hence, a common feature of cryptographic enforcement schemes for information flow policies is the derivation of decryption keys (since possession of the decryption key for label ℓ implies authorization for the decryption key for any label ℓ' less than ℓ). Informally, each security label is associated with a secret (which is issued to every user assigned to that security label) from which decryption keys for all subordinate security labels may be derived. The scheme may also publish additional information in order to support key derivation.

Therefore, the challenge is to compute efficiently the secrets and decryption keys associated with each security label, subject to constraints on the size of relevant parameters. Thus a cryptographic enforcement scheme may be characterized by(i) the number of secrets each user is given, (ii) the total number of secrets issued to users, (iii) the amount of auxiliary (public) information required for key derivation, and (iv) the computational effort required for key derivation.

Many schemes in the literature are space-efficient (on the user side) by providing each user with a single secret (see, for example, [2]), the trade-off being that the amount of public information and derivation time may be substantial. Moreover, the public information must either be transmitted to each

[1] We could, of course, simply view a set of secrets as a single secret and consider the amount of storage required by that secret. However, it is more convenient for the analysis later in the paper to consider a set of secrets and the number of elements in that set.

user or made available on some publicly accessible server, both possibilities giving rise to concerns either about costs of transmission and local storage, or availability and authenticity of the information.

Crampton, Daud and Martin [14] introduced the concept of a *chain-based* cryptographic enforcement scheme, which requires no public information but may require users to store more than one secret. Subsequent work has established that secure instantiations of chain-based schemes exist [20,21]. Chain-based schemes are based on a decomposition of the poset of security labels into disjoint chains (that are, in some appropriate sense, compatible with the poset). Informally, the secrets associated with the labels in each chain may be derived in a top-down manner and each user is issued with a number of secrets, at most one from each chain. Thus the number of secrets required by a user is no greater than the number of chains in the decomposition, which is significantly better, generally, than the naive solution of supplying each user with every secret for which she is authorized.

The motivation for the work in this paper can be summarized in two observations. First, there are, in general, many different ways to instantiate a chain-based scheme for a given information flow policy, each instantiation being defined by a particular chain partition of the partially ordered set used to specify the policy. The number of secrets and the amount of computation required to derive decryption keys in a given instantiation crucially depends on the chain partition chosen. However, existing work in the literature assumes that the chain partition is given as part of the input to the algorithm that outputs the secrets and decryption keys. One of the questions we address (in Section 5), therefore, is how to compute the "best" chain partition (with respect to some suitable metric) with which to instantiate a chain-based scheme. Our second observation is that each security label has at most one parent in the chain decomposition. The question we address (in Section 3) is whether it is possible to generalize chain-based schemes to tree-based schemes, given that each element in a tree also has at most one parent.

Our first set of contributions is associated with the novel concept of a *tree partition* of an information flow policy, from which we define the notion of a *forest-based* cryptographic enforcement scheme for information flow policies. We prove results establishing how the total number of secrets to be issued to users varies with the structure of the forest and demonstrate that an instantiation of our scheme retains the security property of strong key indistinguishability introduced by Freire, Paterson and Poettering [21]. We design and analyze an efficient algorithm for computing a forest that minimizes the total number of issued secrets. This work generalizes our previous work on tree-based enforcement schemes [16]. In addition, the more general framework enables us to simplify the techniques and formal exposition.

Our second set of contributions is based on specializing our generic scheme to chain-based schemes.[2] We prove that the total number of secrets issued is determined by the number of bottom elements of the chains in the chain partition (Lemma 3). This, in turn, allows us to prove (Theorem 3) there exists a chain partition that simultaneously minimizes the number of secrets that need to be issued and the number of chains in the partition (and thus the number of keys each user is required to store). The last result is of practical importance, since the number of chains provides a tight upper bound on the number of secrets required by any user. Moreover, the result is somewhat unexpected, as it is not usually possible to simultaneously minimize two different parameters. Our main contribution (Theorem 4 and Section 5.1) is to develop an efficient algorithm that enables us to find a chain partition such that the total number of distributed secrets and the number of chains are minimized (with respect to all chain partitions). Our

[2] One disadvantage with forest-based schemes is that one cannot, in general, simultaneously minimize the number of secrets issued on a per-user basis and the total number of secrets issued to users. Thus, chain-based schemes are still relevant, even though, in general, a forest-based scheme for the same policy will require fewer secrets in total to be issued.

algorithm is based on finding a minimum cost flow in a network whose construction is based on the technical results in Sections 3–5.

Overall, then, the contributions of this paper generalize and unify existing work on tree- and chain-based schemes using the novel concept of a tree partition and a forest-based enforcement scheme. Central to our work are the results in Section 4, which enable us to link two different characterizations of the additional secrets required, thereby allowing us to describe existing schemes using trees and chains within a single framework and to generalize tree-based enforcement schemes to forest-based schemes. An important consequence of our results is that there now exist efficient methods for instantiating cryptographic enforcement schemes that require no public information. We thereby provide rigorous foundations for the development of efficient chain-based enforcement schemes.

The remainder of the paper is organized as follows. In Section 2, we provide the relevant background on cryptographic enforcement schemes, and formally identify the problem. We also discuss related work, including preliminary versions of the ideas presented in this paper [15,16]. Then, in Section 3, we formally define a tree partition and a forest-based cryptographic enforcement scheme for an information flow policy. We establish some important results connecting the structure of a given forest and the total number of secrets required by the associated cryptographic enforcement scheme. We also establish that there exist secure instantiations of our scheme and briefly discuss cryptographic primitives that would be suitable for such an instantiation. In Section 4, we use the theoretical results of Section 3 to develop an efficient algorithm for computing the best tree partition, in terms of the total amount of secret material required. In Section 5, we prove that there exists a chain-based enforcement scheme in which no user requires more than w keys, where w is the width of the information flow policy; and that the total number of issued secrets in a chain-based enforcement scheme is determined entirely by the number of bottom elements of the chain partition. These results, however, are not constructive *per se*. Accordingly, we also develop an efficient algorithm to derive the best chain partition. We conclude the paper in Section 6 with a summary of our contributions and some ideas for future work.

2. Information Flow Policies

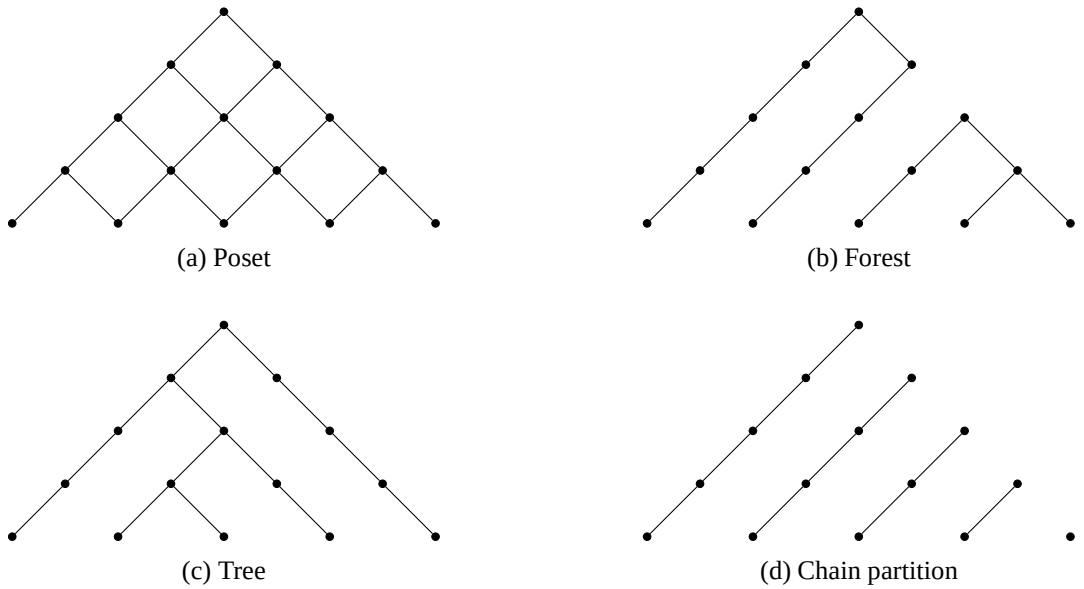
We first recall some basic definitions from discrete mathematics and establish some notation. We then define what is meant by an information flow policy [7] and discuss how such policies may be enforced using cryptographic mechanisms.

A *partially ordered set* (or *poset*) is a pair $\mathcal{P} = (X, \leqslant)$, where $\leqslant$ is a reflexive, anti-symmetric, transitive binary relation on a finite set X.

- We write $x < y$ to indicate $x \leqslant y$ and $x \neq y$, and we may write $x \geqslant y$ whenever $y \leqslant x$.
- We say x *covers* y, or x is a *parent* of y, denoted $y \lessdot x$, if $y < x$ and there does not exist $z \in X$ such that $y < z < x$. An element $x \in X$ is *maximal* if it has no parents.
- The *Hasse diagram* of $\mathcal{P}$ is the directed acyclic graph $H(\mathcal{P}) = (X, E_{\min})$, where the (directed) *edge* $xy \in E_{\min}$ if and only if $y \lessdot x$. We will also make use of the directed acyclic graph $H^*(\mathcal{P}) = (X, E_{\max})$, where $xy \in E_{\max}$ if and only if $y < x$. Representing the covering relation as an acyclic digraph the Hasse diagram provides a minimal amount of information required to reconstruct the full order relation.[3]

[3]The Hasse diagram $H(\mathcal{P}) = (X, \lessdot) = (X, E_{\min})$ is a unique representation of the poset $\mathcal{P} = (X, \leqslant)$. Conversely, as the Hasse diagram $H(\mathcal{P})$ of a poset $\mathcal{P}$ uniquely represents $\mathcal{P}$, we may consider $H(\mathcal{P})$ as a "shorthand" for $\mathcal{P}$ and even loosely say that $H(\mathcal{P})$ is a poset.

– The *in-degree* (*out-degree*, respectively) of node u of a directed graph $D = (V, E)$ is the number of nodes v such that $vu \in E$ ($uv \in E$, respectively). A directed graph D is an *out-forest* if every node of D has in-degree less than or equal to 1.
  We say $\mathcal{P}$ is a *forest* if $H(\mathcal{P})$ is an out-forest. We say $\mathcal{P}$ is a *tree* if it is a forest and has a unique maximal element. That is, there is a single node in its Hasse diagram of in-degree 0.
  Note that every forest is a disjoint union of trees. Hasse diagrams of a poset, forest and tree are shown in Figure 1. The edges in these Hasse diagrams (and all others in the paper) are assumed to be directed from top to bottom.
– A set $Y \subseteq X$ is a *chain* if for all distinct pairs of elements $x, y \in Y$, $x < y$ or $y < x$. A chain corresponds to a directed path in $H^*(\mathcal{P})$.
– A *chain partition* of poset $\mathcal{P}$ is a disjoint union of chains such that every element of $\mathcal{P}$ belongs to one of the chains. Figure 1d depicts a chain partition of the poset in Figure 1a.
– Let $x, y \in X$ with $y < x$. Then $\{z_0, \ldots, z_l\} \subseteq X$, where $x = z_0 \gtrdot z_1 \gtrdot \cdots \gtrdot z_l = y$ is a *derivation chain* (from x to y) in $\mathcal{P}$ of length l. A derivation chain from x to y corresponds to a directed path from x to y in $H(\mathcal{P})$.
– We write $x \parallel y$ to indicate that x, y are incomparable, i.e. $x \not\leqslant y$ and $x \not\geqslant y$. A set $Y \subseteq X$ is an *antichain* if for all distinct $x, y \in Y$, $x \parallel y$. The *width* of a poset is the cardinality of an antichain of maximum size.
– We write $\downarrow(x)$ to denote $\{y \in X : y \leqslant x\}$ and $\uparrow(x)$ to denote $\{y \in X : y \geqslant x\}$. Note that $\downarrow(x) \subseteq \downarrow(y)$ if and only if $x \leqslant y$.
– A *linear extension* of $\mathcal{P}$ is a chain $(X, \preccurlyeq)$ such that if $x \leqslant y$ then $x \preccurlyeq y$. Every (finite) partial order has at least one linear extension, which may be computed, in linear time, by representing the partial order as a directed acyclic graph and using a topological sort [12, §22.3].



(a) Poset (b) Forest

(c) Tree (d) Chain partition

Fig. 1. Hasse diagrams of a poset, a forest, a tree, and a chain partition

In many cases we will use subscripts to denote a function or relation relative to a poset $\mathcal{T}$. Thus, for example, we write $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$, we write $x \gtrdot_{\mathcal{T}} y$ if $x >_{\mathcal{T}} y$ and there is no $z \in X$ such that $x >_{\mathcal{T}} z >_{\mathcal{T}} y$, and we write $\downarrow_{\mathcal{T}}(x)$ to denote the set $\{y \in X : y \leqslant_{\mathcal{T}} x\}$.

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.*

A user $u \in U$ *is* authorized *to read an object* $o \in O$ *if and only if* $\lambda(u) \geqslant \lambda(o)$.

Given an information flow policy $(X, \leqslant, U, O, \lambda)$, we may define an equivalence relation $\sim$ on U, where, for any $u, v \in U$, $u \sim v$ if and only if $\lambda(u) = \lambda(v)$. We write U_x to denote $\{u \in U : \lambda(u) = x\}$. Similarly, $O_x \subseteq O$ denotes the set of objects having security label $x \in X$. In other words, user $u \in U_y$ is authorized to read $o \in O_x$ whenever $y \geqslant x$. Henceforth, we will represent an information flow policy $(X, \leqslant, U, O, \lambda)$ as a poset $\mathcal{P} = (X, \leqslant)$ with the tacit understanding that U, O and λ are given.

2.1. Cryptographic enforcement

The intuition behind the cryptographic enforcement of information flow policies is to encrypt data objects (using a symmetric encryption algorithm) and distribute appropriate secrets to authorized users (from which encryption keys are derived). Hence, there are two high-level algorithms that every cryptographic enforcement scheme (CES) provides: the first, SetUp, is run by the data owner and generates secrets, keys and any public information that is required for deriving decryption keys; the second, Derive, is used to derive decryption keys from secrets and public information. That is, in principle, SetUp and Derive have the following functionality.

- SetUp takes as input an information flow policy $(X, \leqslant)$.
  SetUp outputs $\{(x, \sigma(x), \kappa(x)) : x \in X\}$ and Pub, where $\sigma(x)$ and $\kappa(x)$ respectively determine the secret and encryption key associated with x, and the public information Pub is used as part of the input to the Derive algorithm.
- Derive takes as input the information flow policy, Pub, $x, y \in X$ and $\sigma(x)$.
  Derive outputs $\kappa(y)$ if $y \leqslant x$ (and some distinguished failure symbol $\perp$ otherwise); in particular, $\kappa(x)$ can be derived from $\sigma(x)$.

Prior CES schemes follow the above syntactical framework more or less closely. In particular, different representations of the information flow policy have been used as input to the SetUp and Derive algorithms, and some preprocessing may be required in order to produce those representations. Some schemes, for example, simply use the Hasse diagram of the poset [2] as the input to SetUp and (part of) the input to Derive, while others use a directed, acyclic graph whose edge set is a superset of $E_{\min}$ and a subset of $E_{\max}$ (and thus contains the same paths as the Hasse diagram) [4,13]. In this work, we transform the information flow policy into a partition of trees.

Part of the specification of Derive ensures the *correctness* of a scheme. That is, an authorized user belonging to U_x must be able to derive $\kappa(y)$ if $x \geqslant y$. In contrast, the *security* of a CES requires that users cannot derive keys for which they are not authorized, even if they collude by pooling secret information. In particular, a user in U_z where $z \not\geqslant y$ cannot derive $\kappa(y)$. Research in the last 10 years, pioneered by Atallah, Blanton, Frikken and Fazio [2] and Ateniese, de Santis, Ferrara and Masucci [5], has formalized security notions for CESs. Informally, the adversary learns the secrets and keys associated with some set of elements $A \subseteq X$ (modeling a group of colluding users) and selects a "target" x in X such that $x \not\leqslant a$ for any $a \in A$ (to avoid trivial cases). The adversary may be asked to determine $\kappa(x)$ or to determine, given a candidate key r, whether r is $\kappa(x)$ or a random element of the key space. These informal

scenarios lead to formal concepts of and definitions for *key recovery* and *key indistinguishability* [2].[4] We consider the security properties of CESs in more detail in Section 3.2.

2.2. Related Work

Essentially, designing a cryptographic enforcement scheme comes down to defining(i) what *secrets* each user will receive, (ii) how users will generate any *keys* they require to decrypt data objects, and (iii) how secrets and keys are related. Broadly speaking, there are two standard ways of designing a cryptographic enforcement scheme for information flow policies. These methods assume each user is given a single key from which all other relevant secrets and key may be derived, and are distinguished by the information used to derive secrets and keys. The first method, which we will call "node-based", relies only on secret information known to the user, while the second, which we will call "edge-based", assumes that some additional information must be made known to all users.[5]

Informally, a node-based scheme uses one-way functions: for $y < x$ the secret associated with y is some (one-way) function of $\sigma(x)$, the secret $\sigma(x)$ associated with x, and $\kappa(y)$, the key associated with y, is some (one-way) function of $\sigma(y)$. Some of the earliest work on cryptographic enforcement of information flow policies used these kinds of techniques [27]. However, in this setting, it is unclear how to distribute secrets such that $\sigma(y)$ can be derived from $\sigma(x_i)$ for each of the parents $x_1, \ldots, x_n$ that node y might have, without simultaneously exposing the scheme to collusion attacks.

In an edge-based scheme, public information is associated with each pair (x, y) where $x > y$ from which $\sigma(y)$ can be extracted with knowledge of $\sigma(x)$. Thus, informally, we might define $\mathsf{Pub}(x, y)$ to be $\mathrm{enc}_k(\sigma(y))$, where enc_k is some symmetric encryption algorithm with key k contained in $\sigma(x)$. An edge-based scheme can be used for arbitrary posets but requires public information [2].

Research into schemes that allocate a single secret to each user investigated what trade-offs were possible between the number of items of public data and the number of key derivation operations (in the worst case) [3,13]. Some of this work focused on posets with a particular structure (such as chains [3]). Such research was able to define specific data structures and algorithms, and perform exact complexity analyses [3,4,13]. Other work considered arbitrary posets and used results from graph and poset theory to develop analyses that were generic but arguably less useful in specific cases [5]. In all this work, the amount of public information required for key derivation necessarily increases.

A representation of the policy is required as input to the Derive algorithm. Hence, the data owner must publish the policy (or distribute it with the appropriate secrets to every user). The size of the policy is proportional to the number of edges (each representing a piece of public information) used for secret derivation; that is $O(n^2)$, where n is the cardinality of X (the set of security labels). However, compact representations, using an $n \times n$ binary matrix, exist. In the case of edge-based schemes, the data owner must also publish (or otherwise distribute) Pub, which is also proportional in size to the number of edges. However, the size of Pub will be several orders of magnitude bigger than the policy representation (due to the relative sizes of each datum of information). An alternative is to store Pub on a public server. In

[4]Note that a scheme in which Derive may be used to compute $\kappa(y)$ from $\kappa(x)$ whenever $y < x$ (rather than from $\sigma(x)$) does not possess the key indistinguishability property: the adversary may select x and A such that $x \geqslant a$ for some $a \in A$, use x, a, Pub and r (that is, assume $r = \kappa(x)$) as inputs to the Derive algorithm, and test the output for equality with $\kappa(a)$. Concerns about key indistinguishability in CESs led to the separation between secrets and keys [2].

[5]There are some other types of schemes but each of them suffer from a number of disadvantages (see [17], for example) so research has tended to focus on node- and edge-based schemes.

this case, the server must be on-line and accessible to any user that wishes to run the Derive algorithm. Thus, it may be advantageous to devise schemes that require no public information.

Crampton *et al.* [14] introduced the idea of cryptographic enforcement schemes, based on chain partitions of the information flow policy, that require no public information. The trade-off with such schemes is that some users may require more than one secret in order to be able to derive all the required encryption keys. Subsequent work established that secure instantiations of such schemes are possible [20,21].

To summarize, informally, the core trade-off made when designing a CES is the amount of public information that is required to assist in the derivation of secrets against the number of additional secrets that are associated with nodes. Broadly speaking, on the one hand one assumes each node is associated with a single secret and defines a "secret-derivation digraph" $G = (X, E)$, where $E_{\min} \subseteq E \subseteq E_{\max}$. (In other words, if $x > z$ in $(X, \leqslant)$ there is a derivation path in G, since $E \supseteq E_{\min}$; and if $x \not> z$ there is no derivation path in G, since $E \subseteq E_{\max}$.) On the other hand, one selects a secret-derivation digraph $G = (X, E)$ such that $E \subset E_{\min}$, G is an out-forest, and each node is associated with at least one secret. Then, if $x > z$, there is some node y such that every user in U_x is given the secret associated with node y and there is a directed path from y to z in G. Figure 2 provides a crude comparison of the generic schemes in the literature: E is the set of edges used to derive secrets; d is the length of the longest directed path in $G = (X, E)$; w is the width of X; n is the cardinality of X.

Generic scheme	**Edge set**	**Public information**	**Derivation time**	**Secrets per node**
Single-step secret derivation	$E = E_{\max}$	$O(\|E\|)$	$O(1)$	$k = 1$
Multi-step secret derivation	$E_{\min} \subseteq E \subset E_{\max}$	$O(\|E\|)$	$O(d)$	$k = 1$
Chain-based secret derivation	$E \subset E_{\min}$	None	$O(d)$	$k \in [1, w]$
All secrets distributed	$E = \emptyset$	None	0	$k \in [1, n]$

Fig. 2. A high-level comparison of generic cryptographic enforcement schemes

The significant open problem with prior work on chain-based schemes is the assumption that the chain partition is part of the input to the SetUp algorithm: there may be many such partitions and it is not immediately obvious how one should select a specific partition in order to optimize characteristics of the corresponding enforcement scheme (an example being to minimize the number of secrets issued). Hence, it seems very natural to ask how difficult it is to compute a "good" chain partition, given that (i) schemes based on chain partitions do not require public information, and (ii) the number of secrets that need to be distributed to users is determined by the choice of chain partition. Our recent work [15] shows that it is possible to compute a minimal chain partition in polynomial time using a minimum cost network flow algorithm.

Crampton *et al.* [16] made use of the fact that derivation paths are uniquely defined in trees (as well as in chains) to develop the idea of a tree-based cryptographic enforcement scheme. Their work established that it was possible to compute (in polynomial time) an optimal tree for the information flow policy.

2.3. Problem overview

While chain-based enforcement schemes require no public information, some users may be required to store more than one secret, unlike the majority of schemes in the literature. The number of secrets required by an instantiation of such a scheme depends on the chain partition chosen. Moreover, a natural

extension of the chain-based approach, explored in the current work, is to use a forest related to the poset defining the information flow policy. In this paper, therefore, we explore three questions:

- What is the optimal choice of chain partition and can we compute such a partition efficiently?
- How do we implement a cryptographic enforcement scheme based on a partition of the information flow policy into trees rather than chains?
- What is the optimal choice of tree partition and can we compute such a partition efficiently?

In the next section, we consider the second of these questions, the results of which enable us to answer the other two questions.

3. Enforcement Schemes from Tree Partitions

In this section, we generalize the approach taken by Crampton *et al.* [14] for chain-based enforcement schemes, and Crampton *et al.* [16] for tree-based enforcement schemes. In particular, we introduce the concept of a tree partition of a poset $(X, \leqslant)$ and show how such a partition may be used to construct a cryptographic enforcement scheme for an information flow policy defined by $(X, \leqslant)$.

Definition 2. *Let $\mathcal{P} = (X, \leqslant)$ be a poset, with Hasse diagram $H(\mathcal{P}) = (X, E)$. A* tree partition *of $\mathcal{P}$ is a poset $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ such that $H(\mathcal{T}) = (X, E_{\mathcal{T}})$ is an out-forest and $E_{\mathcal{T}} \subseteq E$.*

If $\mathcal{P} = (X, \leqslant)$ is a poset, $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ is a tree partition of $\mathcal{P}$ and $y \not< x$, then $y \not<_{\mathcal{T}} x$. However, we may have $y < x$ but $y \not<_{\mathcal{T}} x$. Thus, the problem with a tree partition, in the context of cryptographic enforcement schemes (CESs), is that some authorized labels that were "reachable" by a derivation chain in $\mathcal{P}$ will no longer be reachable in $\mathcal{T}$. Accordingly, we define the notion of forest-based enforcement scheme for a tree partition of $\mathcal{P} = (X, \leqslant)$.

Definition 3. *Given an information flow policy $\mathcal{P} = (X, \leqslant)$ and a tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$, a* forest-based enforcement scheme *is a pair $(\mathcal{T}, \psi)$, where $\psi : X \to 2^X$ and:*

1. *if $u \leqslant x$ then there exists $z \in \psi(x)$ such that $u \leqslant_{\mathcal{T}} z$;*
2. *if $u \not\leqslant x$ then for all $z \in \psi(x)$, $u \not\leqslant_{\mathcal{T}} z$.*

Informally, conditions 1 and 2 correspond to the correctness and security requirements of CESs, respectively. Note that $x \in \psi(x)$. To see this, suppose, in order to obtain a contradiction, that $x \notin \psi(x)$. Then, by the first property, there exists $z \in \psi(x)$ such that $x <_{\mathcal{T}} z$. This implies $x < z$ and thus $z \not\leqslant x$. By the second property for all $z^* \in \psi(x)$ we then have $z \not\leqslant_{\mathcal{T}} z^*$. This holds in particular for $z^* = z$ and we obtain $z \not\leqslant_{\mathcal{T}} z$, a contradiction.

Definition 4. *Let $\mathcal{P}$ be a poset and $\mathcal{T}$ a tree partition of $\mathcal{P}$. Then, given $x, z \in X$, the maximum element (if it exists) in $\downarrow_{\mathcal{P}}(x) \cap \uparrow_{\mathcal{T}}(z)$, is the* anchor *between x and z and denoted by $\alpha(xz)$.*

We note the following facts, which we state without proof:

- $\alpha(xz)$ exists iff $x \geqslant z$;
- $\alpha(xz)$ is a unique maximal element (that is, a maximum element) since $\uparrow_{\mathcal{T}}(z)$ is a chain;
- if $x \geqslant z$ and $x >_{\mathcal{T}} z$ then there exists a derivation chain in $\mathcal{T}$ from x to z and $\alpha(xz) = x$ (since x is the maximum element in $\downarrow_{\mathcal{P}}(x)$); and
- if $x \geqslant z$ and $x \not>_{\mathcal{T}} z$ then there exists a derivation chain in $\mathcal{T}$ from $\alpha(xz)$ to z and $x > \alpha(xz)$.

Given $\mathcal{P}$ and a tree partition $\mathcal{T}$, define $\phi_{\mathcal{T}} : X \to 2^X$ as follows:

$$\phi_{\mathcal{T}}(x) = \{\alpha(xz) : x \geqslant z\}$$

Proposition 1. *For any poset $\mathcal{P}$ and any tree partition $\mathcal{T}$ of $\mathcal{P}$, $(\mathcal{T}, \phi_{\mathcal{T}})$ is a forest-based enforcement scheme.*

Proof. If $u \leqslant x$, then $z = \alpha(xu)$ belongs to $\phi_{\mathcal{T}}(x)$ and $u \leqslant_{\mathcal{T}} z$. And if $u \not\leqslant x$ then for every $z \in \phi_{\mathcal{T}}(x)$ we have $z \not\geqslant_{\mathcal{T}} u$. □

In other words, given the secrets corresponding to the elements in $\phi_{\mathcal{T}}(x)$, a user in U_x can derive the secret for all elements $z \leqslant x$ using a derivation chain starting at $\alpha(xz)$.

Lemma 1. *Let $\mathcal{P} = (X, \leqslant)$ be a poset, $\mathcal{T}$ be a tree partition of $\mathcal{P}$, and $(\mathcal{T}, \psi)$ be a forest-based enforcement scheme. Then $\phi_{\mathcal{T}}(x) \subseteq \psi(x)$ for all $x \in X$.*

Proof. Suppose, in order to obtain a contradiction, that $y \in \phi_{\mathcal{T}}(x)$ and $y \notin \psi(x)$. By definition, $y \leqslant x$; therefore, there must exist $y' \in \psi(x)$ such that $y' >_{\mathcal{T}} y$, and thus $x \geqslant y'$. Moreover, $y \geqslant_{\mathcal{T}} z$ so we have $x \geqslant y' >_{\mathcal{T}} y >_{\mathcal{T}} z$; that is, $y' \in {\downarrow_{\mathcal{P}}}(x) \cap {\uparrow_{\mathcal{T}}}(z)$. Thus y is not the maximal element in ${\downarrow_{\mathcal{P}}}(x) \cap {\uparrow_{\mathcal{T}}}(z)$, the desired contradiction. □

The following simple lemma characterizes the elements of $\phi_{\mathcal{T}}$ and will be used to prove Proposition 2 and Theorem 2.

Lemma 2. *Let $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ be a tree partition of poset $\mathcal{P} = (X, \leqslant)$. Then for every x in X and every z in X, $z \in \phi_{\mathcal{T}}(x)$ if and only if exactly one of the following conditions holds:(i) $z = x$; (ii) $z < x$, z has a parent in $\mathcal{T}$ and $x \not\geqslant \mathrm{par}_{\mathcal{T}}(z)$; (iii) $z < x$ and z has no parent in $\mathcal{T}$.*

Proof. Suppose $x \geqslant z$ and $x \not\geqslant \mathrm{par}_{\mathcal{T}}(z)$. Since $x \not\geqslant \mathrm{par}_{\mathcal{T}}(z) \succcurlyeq_{\mathcal{T}} z$, z is the maximal element in ${\downarrow_{\mathcal{P}}}(x) \cap {\uparrow_{\mathcal{T}}}(z)$. Similarly, if z has no parent or $z = x$, then z is the maximal element in ${\downarrow_{\mathcal{P}}}(x) \cap {\uparrow_{\mathcal{T}}}(z)$. In either case, $z = \alpha(xz)$ and $z \in \phi_{\mathcal{T}}(x)$.

Conversely, if $z \in \phi_{\mathcal{T}}(x)$, then $x \geqslant z$, by definition, and $\alpha(xz) = z$. Thus, $x \not\geqslant \mathrm{par}_{\mathcal{T}}(z)$ if z has a parent (otherwise, $\mathrm{par}_{\mathcal{T}}(z) \in {\downarrow_{\mathcal{P}}}(x) \cap {\uparrow_{\mathcal{T}}}(z)$ and $z \neq \alpha(xz)$). □

Proposition 2. *Let $\mathcal{P} = (X, \leqslant)$ be an information flow policy and let $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ be a tree partition. Then $\phi_{\mathcal{T}}$ can be computed in time $O(n^2)$, where $n = |X|$.*

Proof. By Lemma 2, for all $x \in X$, besides x itself, we add all those elements $z \in X$, $z < x$, to $\phi_{\mathcal{T}}(x)$ that are either maximal in $\mathcal{T}$ or, if not, satisfy $x \not\geqslant \mathrm{par}_{\mathcal{T}}(z)$. In both cases, we must determine whether $x > z$ for some $z \in X$.

After $O(n^2)$ time preprocessing, we may assume that we have data structures allowing us to check whether $x > z$ in $O(1)$ time, and test whether z is a maximal element in T (and compute $\mathrm{par}_{\mathcal{T}}(z)$ otherwise) in $O(1)$ time. Hence, we can compute $\phi_{\mathcal{T}}$ in $O(n^2)$ time. □

3.1. Generic instantiation

The above results enable us to specify the algorithms of a cryptographic enforcement scheme. The construction can be considered a generalization of the one using chains (rather than trees) defined by Freire *et al.* [21]. When defining SetUp and Derive we assume that the information flow policy $\mathcal{P} = (X, \leqslant)$ is presented in the form of a tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$, and that for the latter a specific forest-based enforcement scheme $(\mathcal{T}, \psi)$ has been selected (such as $(\mathcal{T}, \phi_{\mathcal{T}})$). Further, for the $n = |X|$ labels of X we assume a numbering convention that follows a (reverse) linear extension $\prec$ of $\leqslant$; more precisely, we assume that $X = \{x_1, \ldots, x_n\}$ where $x_n \prec x_{n-1} \prec \cdots \prec x_2 \prec x_1$ (in particular, x_n is a minimal element in X and x_1 is a maximal element). The cryptographic building block of our construction is a pseudorandom function (PRF) where the key space and the output space are the same set $\mathcal{K}$. Given such a function $\mathcal{F}\colon \mathcal{K} \times \{0,1\}^* \to \mathcal{K}$ and an (injective) label naming function $\ell\colon X \to \{0,1\}^*$ we define:

Algorithm SetUp, on input an information flow policy in the format described above:

1. For $i = 1$ to n do (i.e., count from a maximal down to a minimal label):
   - if x_i is maximal in $(X, \leqslant_{\mathcal{T}})$ pick fresh random key $s(x_i) \leftarrow_{\$} \mathcal{K}$;
   - otherwise, identify the (unique) parent y of x_i in $\mathcal{T}$ and assign $s(x_i) \leftarrow \mathcal{F}(s(y), \ell(x_i))$ (where $s(y)$ is the PRF key and $\ell(x_i)$ is the PRF input);
2. For each $x \in X$ output $\sigma(x) = \{(v, s(v)) : v \in \psi(x)\}$ and $\kappa(x) = \mathcal{F}(s(x), \ell(x))$; no public information is needed, i.e., $\mathsf{Pub} = \emptyset$.

The general principle of this CES is to derive secrets in a top-down fashion: top nodes (according to $\leqslant_{\mathcal{T}}$) are assigned random keys, and the keys of all other nodes are deterministically derived from their parent using the PRF. Observe that, as we arranged $\prec$ to be a linear extension of $\leqslant$ (and thus $\leqslant_{\mathcal{T}}$), step (1) of SetUp is actually well-defined. We next define the corresponding Derive algorithm:

Algorithm Derive, on input the information flow policy, labels $x, y \in X$, and secret $\sigma(x)$:

1. Return $\perp$ if $x \not\geqslant y$;
2. Identify the (unique) $z \in \psi(x)$ such that $y \leqslant_{\mathcal{T}} z$ and recover $s(z)$ from $\sigma(x)$;
3. Let $z = z_0 \succcurlyeq z_1 \succcurlyeq \cdots \succcurlyeq z_m = y$ be the complete derivation chain in $\mathcal{T}$ between z and y;
4. For $i = 1$ to m do: $s(z_i) \leftarrow \mathcal{F}(s(z_{i-1}), \ell(z_i))$;
5. Output $\kappa(y) = \mathcal{F}(s(y), \ell(y))$.

In this instantiation, the same pseudorandom function $\mathcal{F}$ is used as a secret- and key-generation function; secret values, and values derived from secret values, serve as PRF keys, and fixed strings that uniquely identify the corresponding node are its inputs.

3.2. Security analysis

We assess the security of our enforcement scheme using the principles of provable security. We start by formalizing the properties of the cryptographic building block, the pseudorandom function $\mathcal{F}$. Our definition is not the most general possible: rather, it is tailored to the requirements of our construction; specifically, we require that the keyspace and the range of the PRF are the same set.

Definition 5. *A* pseudorandom function *(PRF) with keyspace and range $\mathcal{K}$ is any efficient function $\mathcal{F}\colon \mathcal{K} \times \{0,1\}^* \to \mathcal{K}$. We also write $\mathcal{F}_K(x)$ to denote $\mathcal{F}(K,x)$. We define the* advantage *of an adversary $\mathcal{D}$ in distinguishing $\mathcal{F}$ from a random function as*

$$\mathrm{Adv}^{\mathcal{F}}(\mathcal{D}) = \left|\Pr[K \leftarrow_{\$} \mathcal{K}; \mathcal{D}^{\mathcal{F}_K} \Rightarrow 1] - \Pr[\varphi \leftarrow_{\$} \langle\{0,1\}^* \to \mathcal{K}\rangle; \mathcal{D}^{\varphi} \Rightarrow 1]\right| .$$

We say that PRF $\mathcal{F}$ is (ϵ,τ)-indistinguishable *from a random function if ϵ upper-bounds the advantage of all distinguishers $\mathcal{D}$ that run in time at most τ.*

In the definition above, $\langle\{0,1\}^* \to \mathcal{K}\rangle$ denotes the universe of all functions mapping $\{0,1\}^*$ to $\mathcal{K}$, and writing "$\mathcal{D}^F \Rightarrow 1$" for a function F means that algorithm $\mathcal{D}$ has oracle access to F and terminates outputting value 1. In Definition 5, F either implements access to a keyed PRF instance $\mathcal{F}_K$, or it implements a completely random function. That is, the smaller we can choose ϵ, the closer a particular PRF $\mathcal{F}$ is to a random function. We discuss some practical candidate functions in Section 3.3.

We next make precise the level of security that we target for our enforcement scheme. Many different cryptographic models for CES with security guarantees of various strengths have been proposed (see [11] for a comparative overview). The notion we target and reproduce below, strong key indistinguishability [21], was not only proven to imply all other notions (i.e., to define the highest level of security),[6] but is also, we believe, the most natural and versatile one. It is based on the security experiment $\mathrm{Expt}^{\mathsf{kist},b}_{X,x}$ defined in Fig. 3, where we use the following notation:

$$\begin{aligned}
\bar{\sigma} &= \{(v,\sigma(v)) : v \in X\} ,\\
\bar{\kappa} &= \{(v,\kappa(v)) : v \in X\} ,\\
\mathit{Corrupt}_{X,x} &= \{(v,\sigma(v)) : v \in X, x \not\leqslant v\} ,\\
\mathit{Keys}_{X,x} &= \{(v,\kappa(v)) : v \in X \setminus \{x\}\} .
\end{aligned}$$

In the experiment we assume that the adversary receives the information flow policy $(X,\leqslant)$ in the same format as the SetUp algorithm does.

Definition 6. *Let $(X,\leqslant)$ be an arbitrary poset. A CES for $(X,\leqslant)$ is* (ϵ,τ)-strongly key indistinguishable with respect to static adversaries *[21] if, for all $x \in X$, the advantage of all adversaries $\mathcal{A}$ that interact in experiment $\mathrm{Expt}^{\mathsf{kist},b}_{X,x}(\mathcal{A})$ and run in time at most τ is bounded by ϵ, where we define*

$$\mathrm{Adv}^{\mathsf{kist}}_{X,x}(\mathcal{A}) = \left|\Pr\left[\mathrm{Expt}^{\mathsf{kist},1}_{X,x}(\mathcal{A}) \Rightarrow 1\right] - \Pr\left[\mathrm{Expt}^{\mathsf{kist},0}_{X,x}(\mathcal{A}) \Rightarrow 1\right]\right| .$$

Observe that in this definition the adversary obtains, in principle, all secrets embedded in the system (that is, all $\sigma(x)$ and $\kappa(x)$ values), excluding only those that would allow distinguishing the challenge key by trivial means (e.g., by invoking the Derive algorithm).

[6] [11] show that not all of these implications are strict; in particular strong key indistinguishability is polynomially equivalent to the notion of (plain) key indistinguishability of [2], with tightness loss $n = |X|$. Note also our model considers a static setup where the challenge label is fixed a priori. A variant of Definition 6 would consider dynamic adversaries: such an adversary is able to choose the challenge label x *during* the experiment, rather than having it fixed as one of the experiment's parameters. However, it has been shown that static and dynamic definitions of strong key indistinguishability are polynomially equivalent [21]; corresponding results for (plain) key indistinguishability have also been obtained [5]. To simplify the exposition, therefore, we restrict our attention to the static case.

$\mathrm{Expt}^{\mathsf{kist},b}_{X,x}(\mathcal{A})$:

1. $(\mathsf{Pub}, \bar{\sigma}, \bar{\kappa}) \leftarrow_{\$} \mathsf{SetUp}(X)$
2. Derive $Corrupt_{X,x}$ and $Keys_{X,x}$ from $\bar{\sigma}$ and $\bar{\kappa}$
3. $K_0 \leftarrow_{\$} \mathcal{K}$
4. $K_1 \leftarrow \kappa(x)$, extracted from $\bar{\kappa}$
5. $b' \leftarrow_{\$} \mathcal{A}(X, x, \mathsf{Pub}, Corrupt_{X,x}, Keys_{X,x}, K_b)$
6. Return b'

Fig. 3. Security experiment for strong key indistinguishability

The final step of our analysis is to prove that our forest-based enforcement scheme from the preceding section is strongly key indistinguishable in the sense of Definition 6. More precisely, we have the following result.

Theorem 1. *For any poset $(X, \leqslant)$, $x \in X$, and adversary $\mathcal{A}$ that runs in time at most τ, there exists a constant $0 \leqslant c \leqslant |X|$ and distinguishers $\mathcal{D}^0_1, \ldots, \mathcal{D}^0_c, \mathcal{D}^1_1, \ldots, \mathcal{D}^1_c$ against the underlying PRF such that*

$$\mathrm{Adv}^{\mathsf{kist}}_{X,x}(\mathcal{A}) \quad \leqslant \quad \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}^0_1) + \cdots + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}^0_c) + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}^1_1) + \cdots + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}^1_c)$$

and the respective running times are at most $\tau^b_i = \tau + O(|X|)$. That is, if the PRF is $(\epsilon', \tau + O(|X|))$-indistinguishable then our CES construction is (ϵ, τ)-strongly key indistinguishable with $\epsilon = 2|X|\epsilon'$.

Proof. The argument proceeds using a sequence of $|X| = n$ hybrid games that interpolate between experiments $\mathrm{Expt}^{\mathsf{kist},0}_{X,x}$ and $\mathrm{Expt}^{\mathsf{kist},1}_{X,x}$. In each hybrid step, if specific conditions are met, we replace one PRF instance by a random function; from the point of view of the adversary, the distance between each two consecutive hybrids is not greater than $\mathrm{Adv}^{\mathcal{F}}(\mathcal{D})$ for a specific PRF distinguisher $\mathcal{D}$.

Fix a poset $(X, \leqslant)$ together with a (reverse) linear extension $x_n \prec x_{n-1} \prec \cdots \prec x_2 \prec x_1$ of X, a label $x \in X$, and a CES adversary $\mathcal{A}$ that runs in time at most τ. We use sequence $x_n \prec \cdots \prec x_1$ to define our hybrid experiments: For $b \in \{0, 1\}$, we set $G^b_0 = \mathrm{Expt}^{\mathsf{kist},b}_{X,x}$ and define games $G^b_1, \ldots, G^b_n$ (in that order) such that if $1 \leq k \leq n$ and $x_k \geqslant x$ then the difference between games G^b_k and G^b_{k-1} is precisely that all PRF invocations with key $s(x_k)$ are replaced by assignments with values drawn uniformly at random from $\mathcal{K}$ (correspondingly, also the keys considered in lines (2) and (4) are changed). For the remaining indices k, i.e., in case $x_k \not\geqslant x$, games G^b_k and G^b_{k-1} are identical. Let S^b_k denote $\Pr[G^b_k(\mathcal{A}) \Rightarrow 1]$ for all b, k.

Observe that we replace PRF invocations by random assignments for precisely those labels $x \in X$ that do not have a corresponding entry in $Corrupt_{X,x}$. Observe also that, as we consider the labels in a suitable order, for all switchings from a PRF to a random function we have that the corresponding PRF key $s(x)$ was replaced with a uniform random value before. Thus, the difference between any two consecutive games is bounded by a PRF advantage: by a standard reductionist argument, in the cases $x \leqslant x_k$, we have

$$|S^b_k - S^b_{k-1}| = |\Pr[G^b_k(\mathcal{A}) \Rightarrow 1] - \Pr[G^b_{k-1}(\mathcal{A}) \Rightarrow 1]| = \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}) \ , \tag{1}$$

for a specific distinguisher $\mathcal{D}$ with running time approximately $\tau + |X| \cdot T_{\mathrm{prf}} \in \tau + O(|X|)$, where T_{prf} is the time required for one PRF evaluation; in addition, whenever $x \not\leqslant x_k$ we have $G^b_k = G^b_{k-1}$ and

hence $|S_k^b - S_{k-1}^b| = 0$. Now, by repeated application of the triangle inequality and (1), we have

$$\left|S_0^b - S_n^b\right| \leqslant \sum_{k=1}^{n} \left|S_{k-1}^b - S_k^b\right| \leqslant \sum_{k=1}^{c} \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}_k^b) \ ,$$

where $c = |\{x' \in X : x \leqslant x'\}|$ and distinguishers $\mathcal{D}_k^b$ are constructed as specified. We now consider games G_n^0 and G_n^1. In both cases $\kappa(x)$ is picked uniformly at random, thus lines (3) and (4) in the experiment implement the same operation. Hence G_n^0 is identical to G_n^1 and $\left|S_n^0 - S_n^1\right| = 0$. Thus, we obtain

$$\begin{aligned}\mathrm{Adv}^{\mathsf{kist}}_{X,x}(\mathcal{A}) = |S_0^1 - S_0^0| &\leqslant |S_0^1 - S_n^1| + |S_n^1 - S_n^0| + |S_n^0 - S_0^0| \\ &\leqslant \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}_1^1) + \ldots + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}_c^1) + 0 + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}_1^0) + \ldots + \mathrm{Adv}^{\mathcal{F}}(\mathcal{D}_c^0)\end{aligned}$$

as required. □

Note that by results of [11] it would have sufficed to prove (plain) key indistinguishability of our scheme, as the latter would imply the notion of strong key indistinguishability that we target. Observe however that going this way introduces a tightness loss of $n = |X|$. Besides saving this factor, we believe our direct approach is also more intuitive.

3.3. On practical instantiations of the PRF component

We now briefly consider how one might instantiate our CES in practice. Although pseudorandom functions are a standard building block in the domain of provable security, corresponding constructions do not explicitly appear in most international cryptographic standards documents (e.g., by ANSI, IEEE, NIST, IETF, etc.). However, certain standardized MACs and block ciphers can be used as a PRF replacement, as we discuss next.

The primary aim of message authentication codes (MACs) is integrity protection and data authentication. A standard result says that any PRF may also be used as a MAC. The converse is in general not true: a good MAC is not automatically a good PRF. Fortunately, however, essentially all standardized MAC constructions are in fact good PRFs, including the popular HMAC [26], CMAC [18], GMAC [19], and PMAC [10] schemes.

In our application, the data input of the PRF and hence of the MAC is the name $\ell(x)$ of a node $x \in X$. For the sake of generality we did not impose any constraints on the format of these names (in particular, strings of arbitrary length are allowed). We note that all of the MAC schemes mentioned above are designed to process arbitrary-length strings, of any format. By consequence, all of them are suitable to securely instantiate our enforcement scheme. However, we point out that if we imposed a constant-length restriction on $\ell(x)$, then a much simpler PRF than the MACs mentioned above can be used: by the PRF/PRP switching lemma [9], any block cipher (a.k.a. pseudorandom permutation, PRP) also constitutes a PRF, where the input length is equal to the output length and coincides with the cipher's block size. In particular, if one is satisfied with using 128 bit keys and may require 128-bit labels for elements in X then the AES block cipher can be used without modification as the pseudorandom function of our CES construction. Further, if the target is a security level of 256 bit and one uses 127-bit labels, then the following function would be a suitable PRF:

$$\mathcal{F}\colon \{0,1\}^{256} \times \{0,1\}^{127} \to \{0,1\}^{256}, \text{ where } (K, s) \mapsto \mathrm{AES}_K(0 \parallel s) \parallel \mathrm{AES}_K(1 \parallel s).$$

4. Selecting a Good Tree Partition

Each poset admits many possible tree partitions and each tree partition gives rise to many possible enforcement schemes. In this section, we investigate which enforcement scheme to select for a given tree partition and which tree partition to select for a given poset. Our analysis is based on the assumption that we wish to minimize the total number of secrets that need to be distributed to users. Thus, given a tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ and a forest-based enforcement scheme $(\mathcal{T}, \psi)$, we define

$$\mathcal{S}(\mathcal{T}, \psi) = \sum_{x \in X} |\psi(x)| \cdot |U_x| \,.$$

Note that $|\psi(x)|$ denotes the number of secrets issued to each $u \in U_x$ for the enforcement scheme $(\mathcal{T}, \psi)$. Thus, $\mathcal{S}(\mathcal{T}, \psi)$ is the total number of secrets that need to be distributed to users when we apply scheme $(\mathcal{T}, \psi)$. By Lemma 1, for a given tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$, any forest-based enforcement scheme $(\mathcal{T}, \psi)$ and any $x \in X$, we have $\phi_{\mathcal{T}}(x) \subseteq \psi(x)$; thus $|\phi_{\mathcal{T}}(x)| \leqslant |\psi(x)|$ and $\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}}) \leqslant \mathcal{S}(\mathcal{T}, \psi)$. Hence, for a given tree partition $\mathcal{T}$, we will assume the use of the forest-based enforcement scheme $(\mathcal{T}, \phi_{\mathcal{T}})$.

Let $\mathcal{P} = (X, \leqslant)$ be an information flow policy and let $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ be a tree partition of $\mathcal{P}$. Then we say that $\mathcal{T}$ is a *minimal tree partition* of $\mathcal{P}$ if, for any tree partition $\mathcal{T}'$ of $\mathcal{P}$, we have $\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}}) \leqslant \mathcal{S}(\mathcal{T}', \phi_{\mathcal{T}'})$. (In other words, $\mathcal{T}$ is a tree partition that minimizes the total number of distributed secrets.)

For any tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ and for all $x \in X$, x must have at most one parent in $(X, \leqslant_{\mathcal{T}})$. Informally, then, to construct a tree partition $\mathcal{T}$ from $\mathcal{P} = (X, \leqslant)$, for all $x \in X$ we must discard all but (at most) one parent of x in $\mathcal{P}$. Hence, if we can associate the choice of parent y for z with an appropriate cost of the edge yz in $H^* = (X, E_{\max})$, then computing a minimal tree partition can be translated into a problem of selecting a suitable weighted forest.

We now describe how to compute such a cost function. Given an information flow policy $\mathcal{P} = (X, \leqslant)$, for each pair yz such that $y > z$, we define $\gamma_{\mathcal{P}}(yz) = \{x \in X : x \geqslant z, x \not\geqslant y\}$.

Proposition 3. *For all $x > y > z$, $\gamma_{\mathcal{P}}(xz) \supset \gamma_{\mathcal{P}}(yz)$.*

Proof. Let $t \in \gamma_{\mathcal{P}}(yz)$. Then $t \geqslant z$ and $t \not\geqslant y$. Now if $t \geqslant x$, we would have $t \geqslant y$, by transitivity. Thus $t \not\geqslant x$ and hence $t \in \gamma_{\mathcal{P}}(xz)$. Moreover, $y \in \gamma_{\mathcal{P}}(xz)$, since $y > z$ and $y \not\geqslant x$, and $y \notin \gamma_{\mathcal{P}}(yz)$, so the inclusion is strict. □

Define a weight function $\omega_{\mathcal{P}} : X \times X \to \mathbb{N}$, where

$$\omega_{\mathcal{P}}(yz) = \begin{cases} \sum_{x \in \gamma_{\mathcal{P}}(yz)} |U_x| & \text{if } y > z, \\ 0 & \text{otherwise.} \end{cases}$$

Note that for any tree partition $\mathcal{T}$, z has at most one parent in $\mathcal{T}$, so we may write $\gamma_{\mathcal{T}}(z)$ for $\gamma_{\mathcal{P}}(\mathrm{par}_{\mathcal{T}}(z)z)$ without ambiguity. Given a tree partition $\mathcal{T}$ of X, we define the weight function $\Omega_{\mathcal{T}} : X \to \mathbb{N}$, where

$$\Omega_{\mathcal{T}}(z) = \begin{cases} \displaystyle\sum_{x \geqslant z} |U_x| & \text{if } z \text{ is maximal in } \mathcal{T}, \\ \displaystyle\sum_{x \in \gamma_{\mathcal{T}}(z)} |U_x| & \text{otherwise.} \end{cases}$$

Informally, $\Omega_{\mathcal{T}}(z)$ represents the number of users that will require the secret associated with z, on the one hand if z is maximal in $\mathcal{T}$ and on the other if edge $\mathrm{par}_{\mathcal{T}}(z)z$ is used in $\mathcal{T}$. We can now prove the main result of this section, which establishes a relationship between $\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}})$ and $\Omega_{\mathcal{T}}$, and thus enables us to define an (efficient) algorithm for computing a minimal tree partition.

Theorem 2. *Let $\mathcal{P} = (X, \leqslant)$ be a poset with Hasse diagram $H(\mathcal{P}) = (X, E_{\min})$ and let $\mathcal{T}$ be a tree partition $\mathcal{T}$ of $\mathcal{P}$. Then*

$$\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}}) = \sum_{z \in X} \Omega_{\mathcal{T}}(z). \tag{2}$$

Moreover, we can compute a minimal tree partition $\widehat{\mathcal{T}}$ of $\mathcal{P}$ in time $O(|E_{\min}| + |X|^2)$.

Proof. We first prove (2). Let X'' denote the set of maximal elements in $\mathcal{T}$ and X' denote the set of non-maximal elements. By definition,

$$\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}}) = \sum_{x \in X} |\phi_{\mathcal{T}}(x)||U_x|$$

and, by Lemma 2, we have

$$|\phi_{\mathcal{T}}(x)| = \left|\left\{z \in X' \setminus \{x\} : \ x \in \gamma_{\mathcal{T}}(z)\right\}\right| + \left|\left\{z \in X'' : \ x > z\right\}\right| + 1.$$

Hence

$$\begin{aligned}
\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}}) &= \sum_{x \in X} \left(\left|\left\{z \in X' : x \in \gamma_{\mathcal{T}}(z)\right\}\right| + \left|\left\{z \in X'' : x > z\right\}\right| + 1\right)|U_x| \\
&= \sum_{x \in X} \left|\left\{z \in X' : x \in \gamma_{\mathcal{T}}(z)\right\}\right| |U_x| - \sum_{x \in X'} |U_x| + \sum_{x \in X} \left|\left\{z \in X'' : x > z\right\}\right| |U_x| + \sum_{x \in X} |U_x| \\
&= \sum_{x \in X} \left|\left\{z \in X' : x \in \gamma_{\mathcal{T}}(z)\right\}\right| |U_x| + \sum_{x \in X} \left|\left\{z \in X'' : x > z\right\}\right| |U_x| + \sum_{x \in X''} |U_x| \\
&= \sum_{z \in X'} \sum_{x \in \gamma_{\mathcal{T}}(z)} |U_x| + \sum_{z \in X''} \sum_{x \geqslant z} |U_x| \\
&= \sum_{z \in X} \Omega_{\mathcal{T}}(z)
\end{aligned}$$

We next establish the choice of $\mathcal{T}$ that minimizes $\mathcal{S}(\mathcal{T}, \phi_{\mathcal{T}})$. Observe that if z is not a maximal element of X, a minimal tree partition $\widehat{\mathcal{T}}$ will not have z as a maximal element either. Indeed, suppose z is a maximal element in a tree partition $\mathcal{T}$ and let y be a parent of z in X. Then $\Omega_{\mathcal{T}}(z) > \Omega_{\mathcal{T}'}(z)$, where T' is obtained from $\mathcal{T}$ by adding edge yz to the Hasse diagram of $\mathcal{T}$, since $\{x \in X : \ x \in \gamma_{\mathcal{T}'}(z)\} \subset \{x \in X : \ x \geqslant z\}$; the inclusion is strict since y is in the first set but not the second. Thus, z is a maximal in $\widehat{\mathcal{T}}$ if and only if z is maximal in X. It remains to decide on parents in $\widehat{\mathcal{T}}$ for non-maximal elements in X.

Let $\mathcal{T}$ be a tree partition and z is not maximal in $\mathcal{T}$. Note that $\Omega_{\mathcal{T}}(z) = \omega_{\mathcal{P}}(\mathrm{par}_{\mathcal{T}}(z)z)$. By Proposition 3, we have $\gamma_{\mathcal{P}}(yz) \subset \gamma_{\mathcal{P}}(xz)$ for $x > y > z$. It follows that $\omega_{\mathcal{P}}(yz) \leqslant \omega_{\mathcal{P}}(xz)$, the inequality

being strict if we assume that at least one user is assigned to each node in X. Thus it suffices to consider only parents of z in X when constructing a minimum tree partition. Moreover, to build $\widehat{\mathcal{T}}$, for each non-maximal $z \in X$, we select a parent y of z in X such that $\omega_{\mathcal{P}}(yz) \leqslant \omega_{\mathcal{P}}(y'z)$ for all other parents y' of z.

Finally, we analyze the running time to compute a minimum tree partition. We can compute $\omega_{\mathcal{P}}(yz)$ for each non-maximal z and each parent y of z in $\mathcal{P}$ in time $O(|X|^2)$ using an algorithm similar to that used for computing $\phi_{\mathcal{T}}$. Now a minimal tree partition $\mathcal{T}$ of $\mathcal{P}$ can be obtained by setting $y = \mathrm{par}_{\mathcal{T}}(z)$, where $\omega_{\mathcal{P}}(yz) \leqslant \omega_{\mathcal{P}}(xz)$ for all $x \in X$ such that $x > z$. This will require time $O(|E_{\min}|)$. Thus, the total time required is $O(|E_{\min}| + |X|^2)$.[7] □

We have shown that we can compute a minimal tree partition efficiently. Recall that $|\phi_{\mathcal{T}}(x)|$ measures the number of secrets a user in U_x will require to derive all authorized secrets (and keys). We now consider whether it is possible to compute a minimal tree partition that simultaneously bounds $\max_{x \in X} \{|\phi_{\mathcal{T}}(x)|\}$. Let $\mathcal{T}$ be a minimal tree partition of $\mathcal{P} = (X, \leqslant)$. We will say that $\mathcal{T}$ is an *optimal tree partition* of $\mathcal{P}$ if $\mathcal{T}$ has the minimum number of minimal elements among all minimal tree partitions. An optimal tree partition with ℓ leaves has the property that no user will require more than ℓ secrets.

For each non-maximal $z \in \mathcal{P} = (X, \leqslant)$, let $Y(z)$ be the set of $y \in X$ such that $y > z$ and $\omega_{\mathcal{P}}(yz)$ is minimum. Construct a directed acyclic graph H with vertex set X; for every non-maximal $y \in X$, the in-neighborhood of y is $Y(y)$, and each maximal $y \in X$ has no in-neighbors. Add to H a new vertex r which is an in-neighbor of every $x \in X$. Now apply the polynomial-time algorithm MINLEAF [25], that allows us to find an out-tree rooted at r with minimum number of leaves, i.e., vertices with no out-neighbors. As a result, we obtain, among all tree partitions with minimum number of secrets, one with minimum number of minimal elements. Let X' denote the set of non-maximal elements in $\mathcal{P}$. Then MINLEAF's runtime is $O(s + |X|^{3/2} s^{1/2})$, where $s = \sum_{z \in X'} |Y(z)|$. Observe that $s \leqslant |E_{\max}|$ and $|E_{\max}| \leqslant |X|^2$. This implies that $O(s + |X|^{3/2} s^{1/2}) = O(|X|^{3/2} |E_{\max}|^{1/2})$. Thus, we have the following result.

Corollary 1. *Given an information flow policy $\mathcal{P} = (X, \leqslant)$, we can find an optimal tree partition $\mathcal{T} = (X, \leqslant_{\mathcal{T}})$ of $\mathcal{P}$ in time $O(|X|^{3/2} |E_{\max}|^{1/2})$.*

We conclude this section with an example illustrating our results. Let $[n] = \{1, 2, \ldots, n\}$ and let $[i, j] = \{i, i+1, \ldots, j-1, j\}$ for $i \leq j$. Then define the poset

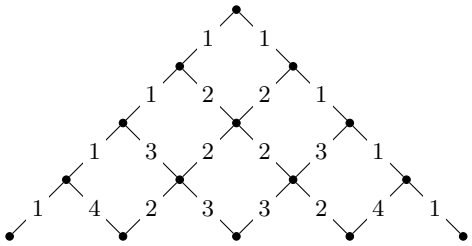
$$\mathcal{I}(n) = \{[i, j] : 1 \leqslant i \leqslant j \leqslant n\},$$

where $[i, j] \leqslant [i', j']$ if and only if $i' \leqslant i$ and $j' \geqslant j$. The Hasse diagram for $\mathcal{I}(5)$ is illustrated in Figure 1a. The poset $\mathcal{I}(n)$ has attracted considerable interest because of its application to "time-bound" access control (see [4,13], for example). In particular, the numbers $1, \ldots, n$ represent time points or time intervals, and elements in $\mathcal{I}(n)$ represent contiguous intervals of time (either consecutive points or a sequence of consecutive intervals). A user u assigned the interval $[i, j]$ is authorized to access any object assigned an interval $[i', j'] \subseteq [i, j]$.

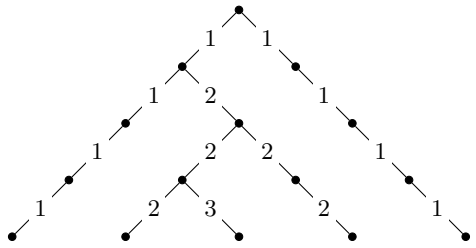
The cardinality of $\gamma_{\mathcal{P}}(yz)$, $y, z \in \mathcal{I}(5)$, $y \succ z$, is shown in Figure 4a. A tree of minimum weight is shown in Figure 4b and the corresponding values of $\Omega_{\mathcal{T}}(z)$ are shown in Figure 4c. It is possible to

[7] Since $|E_{\min}| \leqslant |X|^2$ we can simplify the total time to $O(|X|^2)$. However, we decided to keep $|E_{\min}|$ to stress that only parents of elements need to be considered to compute a minimum tree partition.

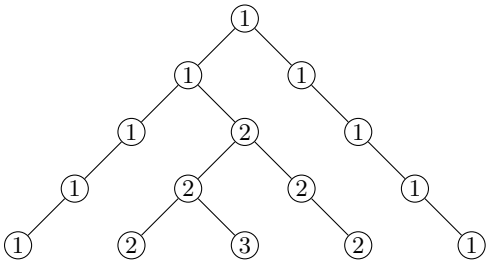
show that the minimum number of secrets required in total, assuming $|U_x| = 1$ for each $x \in \mathcal{I}(n)$, is $\frac{1}{6}m(m+1)(4m-1)$ if $n = 2m-1$, and $\frac{1}{6}m(m+1)(4m+5)$ if $n = 2m$.



(a) $|\gamma(yz)|$



(b) $\widehat{\mathcal{T}}$



(c) $\Omega_{\widehat{\mathcal{T}}}(z)$

Fig. 4. A minimal tree partition of $(\mathcal{I}(5), \subseteq)$

5. Selecting a Good Chain Partition

In this section, we consider chain-based schemes. Recall that a chain partition of a poset $\mathcal{P}$ is a disjoint union of chains such that every element of $\mathcal{P}$ belongs to one of the chains. An element z of a chain C is called *top* (*bottom*, respectively) if the in-degree (out-degree, respectively) of z in $H(C)$ is zero.

We first show that the number of secrets to be issued in a chain-based enforcement scheme is determined by the bottom elements of the chains in the corresponding chain partition. This in turn implies that there exists a chain partition with a minimum number of secrets issued for which the number of chains is exactly the width of the poset.

Lemma 3. *For any poset $\mathcal{P} = (X, \leqslant)$ and any chain partition $\mathcal{C} = (X, \leqslant_{\mathcal{C}})$ of $(X, \leqslant)$ with chains $\{C_1, \ldots, C_\ell\}$, let chain C_i have bottom element b_i, $1 \leqslant i \leqslant \ell$. Then*

$$\mathcal{S}(\mathcal{C}, \phi_{\mathcal{C}}) = \sum_{i=1}^{\ell} \sum_{x \in \uparrow_{\mathcal{P}}(b_i)} |U_x| \,. \tag{3}$$

Proof. Let C_i comprise elements $z_1, z_2, \ldots, z_c$ such that $z_1 > z_2 > \cdots > z_c$ (i.e., $b_i = z_c$) and observe that $\uparrow_{\mathcal{P}}(b_i)$ is the disjoint union of sets X_i, $1 \leqslant i \leqslant c$, where $X_1 = \{x : x \geqslant z_1\}$ and $X_j = \{x : x \not\geqslant z_{j-1}, x \geqslant z_j\}$, $2 \leqslant j \leqslant c$. Observe that $X_j = \{x : x \in \gamma_{\mathcal{C}}(z)\}$, $2 \leqslant j \leqslant c$. This decomposition of $\uparrow_{\mathcal{P}}(b_i)$ into sets X_i, $1 \leqslant i \leqslant c$, will be used in the following derivation.

By (2) and the definition of $\Omega_{\mathcal{C}}(z)$,

$$\begin{aligned}\mathcal{S}(\mathcal{C},\phi_{\mathcal{C}}) &= \sum_{z\in X}\Omega_{\mathcal{C}}(z)\\ &= \sum_{i=1}^{\ell}\sum_{x\in X_1}|U_x| + \sum_{i=1}^{\ell}\sum_{j=2}^{\ell}\sum_{x\in X_j}|U_x|\\ &= \sum_{i=1}^{\ell}\sum_{x\in\uparrow_{\mathcal{P}}(b_i)}|U_x|\end{aligned}$$

□

By Dilworth's Theorem, a poset $(X,\leqslant)$ of width w of has a chain partition with w chains. Such a chain partition can be obtained in time $O(|X|^{2.5})$ [23]. Thus, in particular, we can compute w in time $O(|X|^{2.5})$. The next theorem can be viewed as a strengthening of Dilworth's Theorem. In Subsection 5.1, we will show how to compute a minimal chain partition of width w in polynomial time.

Theorem 3. *Let $\mathcal{P}=(X,\leqslant)$ be an information flow policy of width w. Then there exists a minimal chain partition of width w.*

Proof. Let $\mathcal{C}=(X,\leqslant_{\mathcal{C}})$ be a minimal chain partition of X into $t\geqslant w$ chains and let B be the set of bottom elements in the chains of $\mathcal{C}$. A theorem of Gallai and Milgram asserts that if a chain partition $\mathcal{C}$ of a poset $\mathcal{P}$ contains t chains, where $t>w$, then there exists a chain partition $\mathcal{C}'=(X,\leqslant_{C'})$ into $t-1$ chains such that the set of bottom elements in $\mathcal{C}'$ is a subset of B [22].[8] Hence, by iterated applications of the Gallai-Milgram theorem, there exists a chain partition $\mathcal{C}^*=(X,\leqslant_{C^*})$ of width w such that the set of bottom elements B^* in $\mathcal{C}^*$ is a subset of B. Moreover, by Lemma 3,

$$\mathcal{S}(\mathcal{C}^*,\phi_{\mathcal{C}^*}) = \sum_{b\in B^*}\sum_{x\in\uparrow_{\mathcal{P}}(b)}|U_x| \leqslant \sum_{b\in B}\sum_{x\in\uparrow_{\mathcal{P}}(b)}|U_x| = \mathcal{S}(\mathcal{C},\phi_{\mathcal{C}})$$

As $\mathcal{C}$ is a minimal chain partition, we conclude that $\mathcal{C}^*$ is also a minimal chain partition. □

Corollary 2. *Let $\mathcal{P}=(X,\leqslant)$ be an information flow policy. There exists a chain partition $\mathcal{C}=(X,\leqslant_{\mathcal{C}})$ such that $\mathcal{S}(\mathcal{C},\phi_{\mathcal{C}})$ is minimized and $\max\{|\phi_{\mathcal{C}}(x)| : x\in X\}\leqslant w$.*

Proof. The result follows immediately from Theorem 3 and the fact that $|\phi_{\mathcal{C}}(x)|$ is bounded above by the number of chains in $\mathcal{C}$ for all $x\in X$. □

The above corollary shows that no user requires more than w secrets in a chain-based enforcement scheme.

Returning to our example of $\mathcal{I}(n)$, note that the width of $\mathcal{I}(n)$ is n as the minimal elements form the largest antichain. Thus, any chain partition with n chains requires the same number of secrets. It is not hard to show that this number is $\frac{1}{6}n(n+1)(n+2)$, which is minimum possible. Thus the minimal tree partition of $\mathcal{I}(n)$ (discussed in Section 4) requires approximately half the number of secrets required by the minimal chain partition.

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

5.1. Computing a minimal chain partition

A chain partition imposes stronger constraints than a tree partition. Specifically, each element in a chain partition has at most one parent and one child, whereas a tree partition only requires that each element has at most one parent. Thus, the straightforward algorithm for computing a minimal tree partition cannot be used to compute a minimal chain partition.

Suppose $\mathcal{P} = (X, \leqslant)$ is a poset of width w. In general, a chain partition of $\mathcal{P}$ has $\ell \geqslant w$ chains. Theorem 3 asserts that there exists a minimal chain partition comprising w chains. We now show how such a chain partition may be constructed. In particular, we show how to transform the problem of finding a minimal chain partition $\mathcal{C} = (X, \leqslant_{\mathcal{C}})$ 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. Our definitions for networks and network flows follow the presentation of Bang-Jensen and Gutin [6].

Definition 7. *A* network *is a tuple $\mathcal{N} = (D, l, u, c, \beta)$, where:*

- *$D = (V, A)$ is a directed graph with vertex set V and edge 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;*
- *$c : V \times V \to \mathbb{R}$;*
- *$\beta : V \to \mathbb{R}$ such that $\sum_{v \in V} \beta(v) = 0$.*

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

Definition 8. *Given a network $\mathcal{N} = (D, l, u, c, \beta)$, a function $f : V \times V \to \mathbb{N}$ is a* feasible flow *for $\mathcal{N}$ if the following conditions are satisfied:*

- *$u(vv') \geqslant f(vv') \geqslant l(vv')$ for every $vv' \in V \times V$;*
- *$\sum_{v' \in V} (f(vv') - f(v'v)) = \beta(v)$ 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 $\mathcal{C} = (X, \leqslant_{\mathcal{C}})$ such that $\mathcal{C}$ is a chain partition of X with precisely w chains that minimizes $\mathcal{S}(\mathcal{C}, \phi_{\mathcal{C}})$. To do this, we will construct a network $\mathcal{N}$ such that the minimum cost flow of $\mathcal{N}$ corresponds to the desired chain partition. We can then find the minimum cost flow of $\mathcal{N}$ in polynomial time.

Every top vertex in $\mathcal{C}$ must have one child and no parent in C, every bottom vertex in C must have one parent and no child in C, and every other vertex in C must have one parent and one child. We cannot

represent this requirement directly in a network. However, we can use the *vertex splitting procedure* [6] to simulate it. Specifically, given poset $\mathcal{P} = (X, \leqslant)$, define first a directed graph $D = (V, A)$. Let $X_{\text{in}} = \{x_{\text{in}} : x \in X\}$ and $X_{\text{out}} = \{x_{\text{out}} : x \in X\}$, and define the vertex set $V = X_{\text{in}} \cup X_{\text{out}} \cup \{s, t\}$, where $\{s, t\} \cap (X_{\text{in}} \cup X_{\text{out}}) = \emptyset$. Define the edge set A as follows: for $v, v' \in X_{\text{in}} \cup X_{\text{out}}$, $vv' \in A$ if and only if either $v = x_{\text{in}}$ and $v' = x_{\text{out}}$ for some $x \in X$, or $v = x_{\text{out}}$ and $v' = y_{\text{in}}$ for some $x, y \in X$ such that $y \leqslant x$; for every $v \in X_{\text{in}}$ we have $sv \in A$; and for every $v \in X_{\text{out}}$ we have $vt \in A$.

Then define a network $\mathcal{N} = (D, l, u, c, \beta)$, where

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

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

$$c(vv') = \begin{cases} \sum_{x \in \uparrow_{\mathcal{P}}(v)} |U_x| & \text{if } v' = t, v = x_{\text{out}}, \text{ where } x \in X \\ 0 & \text{otherwise;} \end{cases}$$

$$\beta(v) = \begin{cases} w & \text{if } v = s \\ -w & \text{if } v = t \\ 0 & \text{otherwise.} \end{cases}$$

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 4. *Let $\mathcal{N}$ be the network chain-representation of an information flow policy $\mathcal{P} = (X, \leqslant)$. Then the minimum number of secrets issued by a chain-based enforcement scheme for $(X, \leqslant)$ with w chains is $\widehat{f}$, where $\widehat{f}$ is the minimum cost of a feasible flow in $\mathcal{N}$.*

Proof. Suppose we are given a chain partition $\mathcal{C} = (X, \leqslant_{\mathcal{C}})$.Consider the following flow:

$$\begin{aligned} f(x_{\text{in}}x_{\text{out}}) &= 1 && \text{for all } x \in X; \\ f(x_{\text{out}}y_{\text{in}}) &= 1 && \text{if } x = \text{par}_{\mathcal{C}}(y); \\ f(sx_{\text{in}}) &= 1 && \text{if } x \text{ is the top element in a chain in } \mathcal{C}; \\ f(x_{\text{out}}t) &= 1 && \text{if } x \text{ is the bottom element in a chain in } \mathcal{C}; \\ f &= 0 && \text{otherwise.} \end{aligned}$$

Observe that f is a feasible flow. Indeed, by construction all edges xy satisfy $u(xy) \geqslant f(xy) \geqslant l(xy)$. In the graph formed by edges xy with $f(xy) = 1$, it is clear that every vertex x has in-degree and out-degree 1, except for s and t. Also, s has in-degree 0 and out-degree w in this graph, and t has in-degree w and out-degree 0. As all edges xy have $f(xy) = 1$ or $f(xy) = 0$, we have that

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

for all x, as required. Moreover, the cost of f equals $\sum_{b\in B}\sum_{x\in\uparrow_{\mathcal{P}}(b)}|U_x|$, where B is the set of bottom elements of chains in $\mathcal{C}$, which by (3) equals $\mathcal{S}(\mathcal{C},\phi_{\mathcal{C}})$.

Conversely, suppose f is a feasible flow for $\mathcal{N}$. Then we define $y \preccurlyeq_{\mathcal{C}} x$ if and only if $x, y \in X$ and $f(x_{\text{out}}y_{\text{in}}) = 1$. By the construction of $\mathcal{N}$ and definition of f, it is not hard to see that $\mathcal{C}$ is a chain partition of X with w chains. By construction of $\mathcal{N}$, the cost of f equals $\sum_{b\in B}\sum_{x\in\uparrow_{\mathcal{P}}(b)}|U_x|$, where B is the set of bottom elements of chains in $\mathcal{C}$, which by (3) equals $\mathcal{S}(\mathcal{C},\phi_{\mathcal{C}})$. □

Lemma 5. *We can find a minimum cost flow for $\mathcal{N}$ in $O(|X|^4 w)$ time.*

Proof. Recall that computing w can be done in time $O(|X|^{2.5})$. To compute $\sum_{x\in\uparrow_{\mathcal{P}}(y)}|U_x|$ for each $y \in X$ requires time $O(|E_{\max}| + |X|)$ using depth-first search from y in the digraph obtained from $H^*(X)$ by changing orientation of every edge. Thus, to compute $\sum_{x\in\uparrow_{\mathcal{P}}(y)}|U_x|$ for all $y \in X$ requires time $O(|X|(|E_{\max}| + |X|))$.

The well-known buildup algorithm (see [6, §4.10.5], for example) finds a minimum cost flow for a network with n vertices and m edges in time $O(n^2mM)$, where M denotes the maximum of all absolute values of balance demands on vertices. By construction of $\mathcal{N}$, we have that $n = 2|X| + 2 = O(|X|)$, $m = O(n^2) = O(|X|^2)$, and $M = w$. Thus we get the desired running time. □

Remark 1. *Strictly speaking, the buildup algorithm assumes that all lower bounds on edges are 0. In its current form, our network does not satisfy this condition. However, we can satisfy this condition, given $\mathcal{N} = (D, l, u, c, \beta)$, by defining the network $\mathcal{N}' = (D, l', u', c, \beta')$, where*

$$l'(xy) = 0 \qquad \beta'(x) = \beta(x) - l(xy)$$
$$u'(xy) = u(xy) - l(xy) \qquad \beta'(y) = \beta(y) + l(xy)$$

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, for this section which is, essentially, a corollary of Theorem 3 and Lemmas 4 and 5.

Theorem 4. *Let $\mathcal{P} = (X, \leqslant)$ be an information flow policy of width w. Then we can find a minimal chain partition comprising w chains in time $O(|X|^4 w)$. In such a chain partition no user requires more than w secrets.*

Proof. Let $\mathcal{S}$ denote the minimum number of secrets issued by a chain-based enforcement scheme for X. By Theorem 3, there exists a chain partition that has exactly w chains, for which the corresponding chain-based enforcement scheme only requires $\mathcal{S}$ secrets. Then by Lemma 4, $\mathcal{S}$ is equal to the minimum cost of a feasible flow in $\mathcal{N}$, the network chain-representation of $\mathcal{P}$. By Lemma 5, such a flow can be found in $O(|X|^4 w)$ time, and this flow can be easily transformed into the corresponding chain partition $\mathcal{C} = (X, \leqslant_{\mathcal{C}})$. Finally, by definition of $\phi_{\mathcal{C}}(x)$, $|\phi_{\mathcal{C}}(x)| \leqslant w$ for each $x \in X$ and therefore no user requires more than w secrets. □

6. Concluding Remarks

In this paper, we introduced the concept of a tree partition, generalizing prior work on chain partitions and tree-based enforcement schemes. We have proved that it is possible to compute optimal chain and tree partitions for an arbitrary information flow policy in polynomial time. And we have proved that there exist secure instantiations of enforcement schemes based on tree partitions. In short, we have shown that it is possible to construct forest-based cryptographic enforcement schemes for information flow policies efficiently.

Perhaps the most important contribution of our work on cryptographic enforcement schemes based on tree and chain partitions is to provide alternative trade-offs between the parameters of such enforcement schemes. These additional trade-offs provide data owners with a greater range of potential enforcement schemes, enabling them to select the most appropriate for their particular information flow policy and deployment constraints (such as storage and connectivity capabilities of end-user devices). We might, for example, wish to use an existing scheme that requires each device to store a single secret when storage is limited. Alternatively, we might wish to use a chain-based scheme when the distribution of public information is difficult and we wish to impose a small upper bound on the number of secrets that any device needs to store. We might use a tree-based scheme if distribution of public information is difficult and we wish to minimize the amount of data we wish to transmit to the user population.

Another difference between minimal tree-based and chain-based schemes is that computing the former is significantly faster than the latter as the former can essentially be computed by a simple greedy algorithm, while the latter requires a more sophisticated and much slower minimum cost flow algorithm. While still polynomial-time, minimum cost flow algorithms may be too slow when $|X|$ is large.

In future work, we hope to investigate the difficulty of finding a tree partition in which the worst-case derivation time is as similar as possible for all users (whilst still minimizing the number of secrets issued).

References

[1] S. Akl and P. Taylor. Cryptographic solution to a problem of access control in a hierarchy. *ACM Transactions on Computer Systems*, 1(3):239–248, 1983.

[2] M. J. Atallah, M. Blanton, N. Fazio, and K. B. Frikken. Dynamic and efficient key management for access hierarchies. *ACM Trans. Inf. Syst. Secur.*, 12(3), 2009.

[3] M. J. Atallah, M. Blanton, and K. B. Frikken. Key management for non-tree access hierarchies. In D. F. Ferraiolo and I. Ray, editors, *SACMAT 2006, 11th ACM Symposium on Access Control Models and Technologies, Lake Tahoe, California, USA, June 7-9, 2006, Proceedings*, pages 11–18. ACM, 2006.

[4] M. J. Atallah, M. Blanton, and K. B. Frikken. Incorporating temporal capabilities in existing key management schemes. In J. Biskup and J. Lopez, editors, *ESORICS*, volume 4734 of *Lecture Notes in Computer Science*, pages 515–530. Springer, 2007.

[5] G. Ateniese, A. D. Santis, A. L. Ferrara, and B. Masucci. Provably-secure time-bound hierarchical key assignment schemes. *J. Cryptology*, 25(2):243–270, 2012.

[6] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2nd edition, 2009.

[7] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Mitre Corporation, Bedford, Massachusetts, 1976.

[8] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[9] J. Black and P. Rogaway. CBC MACs for arbitrary-length messages: The three-key constructions. In M. Bellare, editor, *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, volume 1880 of *Lecture Notes in Computer Science*, pages 197–215. Springer, 2000.

[10] J. Black and P. Rogaway. A block-cipher mode of operation for parallelizable message authentication. In L. R. Knudsen, editor, *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*, volume 2332 of *Lecture Notes in Computer Science*, pages 384–397. Springer, 2002.

[11] A. Castiglione, A. D. Santis, and B. Masucci. Key indistinguishability vs. strong key indistinguishability for hierarchical key assignment schemes. *IACR Cryptology ePrint Archive*, 2014:752, 2014.

[12] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 3rd edition, 2009.

[13] J. Crampton. Practical and efficient cryptographic enforcement of interval-based access control policies. *ACM Trans. Inf. Syst. Secur.*, 14(1):14, 2011.

[14] J. Crampton, R. Daud, and K. M. Martin. Constructing key assignment schemes from chain partitions. In S. Foresti and S. Jajodia, editors, *Data and Applications Security and Privacy XXIV, 24th Annual IFIP WG 11.3 Working Conference, Rome, Italy, June 21-23, 2010, Proceedings*, volume 6166 of *Lecture Notes in Computer Science*, pages 130–145. Springer, 2010.

[15] J. Crampton, N. Farley, G. Gutin, and M. Jones. Optimal constructions for chain-based cryptographic enforcement of information flow policies. In P. Samarati, editor, *Data and Applications Security and Privacy XXIX - 29th Annual IFIP WG 11.3 Working Conference, DBSec 2015, Fairfax, VA, USA, July 13-15, 2015, Proceedings*, volume 9149 of *Lecture Notes in Computer Science*, pages 330–345. Springer, 2015.

[16] J. Crampton, N. Farley, G. Gutin, M. Jones, and B. Poettering. Cryptographic enforcement of information flow policies without public information. In T. Malkin, V. Kolesnikov, A. B. Lewko, and M. Polychronakis, editors, *Applied Cryptography and Network Security - 13th International Conference, ACNS 2015, New York, NY, USA, June 2-5, 2015, Revised Selected Papers*, volume 9092 of *Lecture Notes in Computer Science*, pages 389–408. Springer, 2015.

[17] J. Crampton, K. M. Martin, and P. R. Wild. On key assignment for hierarchical access control. In *CSFW*, pages 98–111. IEEE Computer Society, 2006.

[18] M. J. Dworkin. SP 800-38B: Recommendation for block cipher modes of operation: The CMAC mode for authentication. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2005. `http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf`.

[19] M. J. Dworkin. SP 800-38D: Recommendation for block cipher modes of operation: Galois/Counter Mode (GCM) and GMAC. Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007. `http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf`.

[20] E. S. V. Freire and K. G. Paterson. Provably secure key assignment schemes from factoring. In U. Parampalli and P. Hawkes, editors, *Information Security and Privacy - 16th Australasian Conference, ACISP 2011, Melbourne, Australia, July 11-13, 2011. Proceedings*, volume 6812 of *Lecture Notes in Computer Science*, pages 292–309. Springer, 2011.

[21] E. S. V. Freire, K. G. Paterson, and B. Poettering. Simple, efficient and strongly KI-secure hierarchical key assignment schemes. In E. Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 101–114. Springer, 2013.

[22] T. Gallai and A. N. Milgram. Verallgemeinerung eines Graphentheoretischen Satzes von Rédei. *Acta Sci. Math.*, 21:181–186, 1960.

[23] V. K. Garg. *Introduction to Lattice Theory with Computer Science Applications*. Wiley, 2015.

[24] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 89–98. ACM, 2006.

[25] G. Gutin, I. Razgon, and E. J. Kim. Minimum leaf out-branching and related problems. *Theor. Comput. Sci.*, 410(45):4571–4579, 2009.

[26] National Institute of Standards and Technology. FIPS 198-1, The Keyed-Hash Message Authentication Code, Federal Information Processing Standard (FIPS), Publication 198-1. Technical report, Department of Commerce, 2008.

[27] R. S. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Inf. Process. Lett.*, 27(2):95–98, 1988.

[28] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, pages 534–542. IEEE, 2010.