

# On Key Assignment for Hierarchical Access Control

Jason Crampton

Keith Martin

Peter Wild

*Information Security Group*

*Royal Holloway, University of London*

*United Kingdom*

*{jason.crampton,keith.martin,p.wild}@rhul.ac.uk*

## Abstract

*A key assignment scheme is a cryptographic technique for implementing an information flow policy, sometimes known as hierarchical access control. All the research to date on key assignment schemes has focused on particular encryption techniques rather than an analysis of what features are required of such a scheme. To remedy this we propose a family of generic key assignment schemes and compare their respective advantages. We note that every scheme in the literature is simply an instance of one of our generic schemes. We then conduct an analysis of the Akl-Taylor scheme and propose a number of improvements. We also demonstrate that many of the criticisms that have been made of this scheme in respect of key updates are unfounded. Finally, exploiting the deeper understanding we have acquired of key assignment schemes, we introduce a technique for exploiting the respective advantages of different schemes.*

## 1 Introduction

**Background** The seminal work of Akl and Taylor [1] has inspired a considerable amount of research on implementing an *information flow policy* using cryptographic techniques. An information flow policy associates each user and each object in a computer system with a security label, where the set of security labels is partially ordered, and requires that a user must be at least as privileged as any object she reads. This is widely known as the *simple security property*, which forms part of the Bell-LaPadula security model [3]. Algebraically, the simple security property is expressed in the following way: for subject  $s$  to obtain read access to object  $o$  we require that  $\lambda(s) \geq \lambda(o)$ , where  $\lambda$  is a security function that associates an entity with a security label. The enforcement of the simple security property

is sometimes referred to as *hierarchical access control* because the partially ordered set of security labels can be represented as a hierarchy.

Although information flow policies were originally used in military systems, they have many other practical applications, particularly if we regard the set of users as being partitioned into a set of *security classes*, with each class being associated with a particular security label. Such applications exist in many different environments, including secure databases, broadcast services (such as Pay TV) and secure mail systems. The problem that needs to be addressed in all of these applications is how best to enforce the information flow policy.

**Key assignment schemes** The use of encryption to enforce information flow policies has been extensively studied [1, 2, 4–11, 18, 19, 21–26, 28–32, 35–38, 40, 41, 43–46, 48, 49, 51]. In solutions of this type, each security label is associated with a unique cryptographic key. The problem of enforcing the information flow policy thus becomes a problem of assigning and managing the keys in the system in a way that is compatible with the policy. For this reason they are typically referred to as *key assignment schemes*.

A key assignment scheme is administered by the policy owner, sometimes referred to as a *trusted centre*, who is responsible for generating and distributing keys, generating and managing any public data required by the scheme and updating the scheme in the event of changes to the policy. The existence of a trusted centre is assumed throughout this paper: statements of the form “choose a key  $k$ ”, for example, should be interpreted as “the trusted centre chooses a key  $k$ ”.

In addition to a key for each security label  $x$ , a key assignment scheme may also associate a secret value with  $x$  and publish some information that is accessible to all users of the scheme. This information can be used to derive the encryption key for any lower security label. In other words,

an object with security level  $y$  that has been encrypted with the associated key  $k(y)$  can be decrypted by any user with security level  $x \geq y$  because the user can derive key  $k(y)$  from the secret value associated with  $x$  and the public information.

**Motivation** There have been many papers in the last twenty years on key assignment schemes, the majority having appeared in the last ten. However, much of this research effort has been poorly motivated and narrowly focused. The results that have been produced are fragmented and are not presented within a common framework. Consequently, many proposals have been reinvented, while others have not been compared with state-of-the-art. Further, many proposals for key assignment schemes have made unsubstantiated claims and lack any satisfactory security analysis. It is thus not surprising that many have later been found to be flawed [7, 20, 22, 23, 25, 27, 39, 42, 43, 47, 50].

We believe that it is important to take stock of the body of research in this area in order to determine whether any substantial progress has been made since the early papers of Akl and Taylor [1] and Gudes [18]. A glaring omission from much of the existing research is any comparison either with fundamental key assignment schemes (as discussed in Section 2) or other proposals in the literature. We believe that the development of a generic model for key assignment schemes would be of considerable use in evaluating the wide variety of existing research.

**Contributions** A study of the literature has led us to the conclusion that there are actually a small number of generic key assignment schemes. Particular schemes only differ in the choice of cryptographic primitives that are chosen to implement one of the generic schemes. The first contribution of this paper is to define five generic key assignment schemes and to classify existing schemes as instances of these generic schemes. A comparison of the characteristics of these generic schemes allows us to infer the advantages of particular schemes in the literature.

It has long been thought that the Akl-Taylor scheme suffers from a number of disadvantages. This scheme is an instance of our generic node-based key assignment scheme. An analysis of the scheme allows us to propose an alternative version that does not suffer from many of the disadvantages identified by other researchers. Nevertheless, our analysis also reveals that the Akl-Taylor scheme does have significant drawbacks.

Different generic schemes offer different advantages. Our final contribution is to combine our deeper understanding of key assignment schemes with recent work in role-based administration to define a hybrid key assignment scheme that offers the advantages of its parent schemes without their respective disadvantages.

**Organization** In the next section we propose our framework, which includes a general definition of an information flow policy and a key assignment scheme. We define a number of generic schemes that can be used to create key assignment schemes and comment on the relative merits of these schemes. In Section 3 we consider the Akl-Taylor scheme in more detail. We modify the scheme, enabling us to improve its efficiency and compare the performance of this modified scheme with another scheme in the literature. In Section 4 we introduce our hybrid key assignment scheme. Finally in Section 5 we make some concluding remarks and suggest topics for future research.

**Notation** Henceforth we adopt the following conventions:  $x$ , when used as input to some (cryptographic) function, will denote a string identifying the security label  $x$ ;  $h$  denotes a hash function;  $E$  denotes a symmetric encryption algorithm,  $E_k(m)$  denotes the encryption of message  $m$  with key  $k$ ;  $\oplus$  denotes the bitwise XOR operation;  $s \parallel t$  denotes the concatenation of strings  $s$  and  $t$ ;  $m \mid n$  means that integer  $m$  divides integer  $n$  without remainder;  $(m, n)$  denotes the greatest common divisor of integers  $m$  and  $n$ ; in particular,  $(m, n) = 1$  means that  $m$  and  $n$  are co-prime.

## 2 A general model for key assignment schemes

A partially ordered set is a pair  $(L, \leq)$ , where  $\leq$  is a reflexive, anti-symmetric, transitive binary relation on  $L$ . We may write  $x \geq y$  whenever  $y \leq x$ . We say  $x$  covers  $y$ , denoted  $y < x$ , if  $y < x$  and there does not exist  $z \in L$  such that  $y < z < x$ . We say  $y$  is the *child* of  $x$  if  $y < x$  (and  $x$  is the *parent* of  $y$ ).  $X$  is a *tree* if no element of  $X$  has more than one parent.  $X$  is a total order if for  $x, y \in L$ , either  $x \leq y$  or  $y \leq x$ . The *Hasse diagram* of a poset is the directed graph  $(L, <)$  [14].

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

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

Such policies were of particular interest in the mid-1970s, and formed part of the famous Bell-LaPadula security model. The basic operation of the policy is that a user  $u$  can read an object  $o$  if  $\lambda(u) \geq \lambda(o)$ . Clearly, one way of

implementing such a policy is to encrypt an object with security label  $y \in L$  with a key  $k(y)$  and provide all users with security label  $x \geq y$  with the key  $k(y)$ . Henceforth, we will represent an information flow policy  $(L, \leq, U, O, \lambda)$  as a pair  $(L, \leq)$  with the tacit understanding that  $U, O$  and  $\lambda$  are given. We assume that the policy will be implemented by encrypting objects and distributing keys to users, enabling them to decrypt the objects to which they should have access.

## 2.1 Syntax of key assignment schemes

Most key assignment schemes seek to minimize the number of keys that need to be distributed to users. This entails either making certain additional information public or providing each user with additional secret information (or both). In general, a *key assignment scheme* (or *scheme*) for an information flow policy  $(L, \leq)$  defines four algorithms:

- `makeKeys` returns a labelled set of encryption keys  $(\kappa(x) : x \in L)$ , which we denote by  $\kappa(L)$ ;
- `makeSecrets` returns a labelled set of secret values  $(\sigma(x) : x \in L)$ , which we denote by  $\sigma(L)$ ;
- `makePublicData` returns some set of data  $Pub$  that is made public by the trusted centre;
- `getKey` takes  $x, y \in L, \sigma(x)$  and the public data, and returns  $\kappa(y)$  whenever  $y \leq x$ .

Many schemes do not require secret values. In other words, the `makeSecrets` algorithm is not required. We call these schemes *simple*. In such schemes,  $\kappa(x)$  is used to derive  $\kappa(y)$ . Hence we assume in such cases that  $\sigma(x)$  is equal to  $\kappa(x)$ . If `makeSecrets` is defined, it takes the information flow policy  $(L, \leq)$  and  $\kappa(L)$  as inputs.

We say a scheme has *independent keys* if the keys can be chosen independently (and *dependent keys* otherwise). In some schemes, `makePublicData` is run before `makeKeys`, with `makeKeys` taking  $Pub$  as input, as well as  $(L, \leq)$ . Such schemes usually have dependent keys, because the secret key  $\kappa(x)$  may depend on a public value associated with  $x$ . The practical effects of a scheme having independent keys are that

- the `makeKeys` algorithm can effectively choose the keys  $\kappa(L)$  at random from the key space (subject to any requirements on key selection of the underlying encryption system), and
- the redistribution of keys or secrets to other users in the event of key compromise is simpler.

Obviously, `getKey` can only be run if the scheme has been initialized by running the other three algorithms. In

the remainder of this paper, we will define a key assignment scheme by stating  $\kappa(L), \sigma(L), Pub$  and the method by which keys are derived. When describing the key derivation method, we always assume that a user with security label  $x$  wishes to obtain  $\kappa(y)$ , where  $y \leq x$ .

## 2.2 Special cases

If  $L$  is a tree, then (by definition) there is a unique directed path from  $y$  to  $x$  whenever  $y \leq x$ . This makes the construction of a key assignment scheme particularly simple [1, 23, 28, 36, 46, 48, 49], the basic approach being to define  $\kappa(y) = E_{\kappa(x)}(y)$  whenever  $y < x$ . (A typical implementation would use a hash function [36].) In order to recover  $\kappa(y)$ , where  $y \leq x$ , a user with security label  $x$  traverses the path between  $y$  and  $x$  in reverse – first recovering the key of the first node in the path between  $x$  and  $y$ , and then repeating the process – until  $\kappa(y)$  is obtained. However, it is generally difficult to build a key assignment scheme for arbitrary poset-based policies from such schemes, although there have been several attempts to do so [46, 48, 49]. Given the limited utility of key assignment schemes designed specifically for total orders and trees, such schemes will not be considered further in this paper.

Traditionally, an information flow policy  $(L, \leq)$  is a *lattice* [3, 15]. Since every lattice is a poset, all key assignment schemes for posets can be applied to lattices (as well as total orders and trees). There are no key assignment schemes in the literature that can be applied to lattices but not arbitrary posets.

## 2.3 Some benchmark schemes

In this section we define several generic schemes and categorize the schemes in the literature using these “benchmarks”. We also compare the advantages and disadvantages of the generic schemes. We discuss a number of schemes in the literature to illustrate the fact that many researchers have ignored the underlying issues that need to be considered when developing a key assignment; in doing so, they have often produced schemes that are rather poor implementations of the benchmark schemes.

**Scheme 1** A trivial key assignment scheme (*TKAS*) has the following properties:

- *Independent keys*;
- $\sigma(x) = (\kappa(y) : y \leq x)$ ;
- $Pub = \emptyset$ ;
- $\kappa(y) \in \sigma(x)$ , so key derivation is trivial.

Clearly, a key assignment scheme requires the secure distribution of keys to users and users will be required to store a certain amount of information securely. Hence, two factors that will affect the suitability of a key assignment scheme will be the amount of data that must be distributed to and stored by end users.

Note also that it will be necessary at times to change  $\kappa(x)$  for some  $x \in L$ . This may be because the key has been compromised or a user with security label  $x$  has been removed from the scheme. In both instances all objects encrypted with  $\kappa(x)$  have to be re-encrypted with a new key. In fact, we must assume that  $\kappa(y)$ ,  $y \leq x$ , is also compromised and must re-encrypt all objects with security label  $y$  and distribute the new  $\kappa(y)$  to all users with security label  $y$ .

Hence, a third desirable feature of key assignment schemes is that updating a key  $\kappa(x)$  should not require changes to  $\sigma(x)$ ,  $\kappa(y)$  or  $\sigma(y)$  for any  $y \not\leq x$ . In this respect, TKAS is a poor scheme, because a change to  $\kappa(x)$ , will require changes to  $\sigma(z)$  for all  $z$  such that  $z \geq y$  for some  $y \leq x$ .

Of course, both key generation and derivation are very straightforward in TKAS, so it is important to remember that this scheme exists. All other key assignment schemes are, to varying extents, attempts to improve on TKAS.

**Scheme 2** A trivial key encrypting key assignment scheme (TKEKAS) has the following properties:

- Independent keys;
- $\sigma(x) = (K(y) : y \leq x)$ , where  $K(x)$  denotes a key encrypting key for  $x \in L$ ;
- $Pub = (E_{K(x)}(\kappa(x)) : x \in L)$
- $\kappa(y)$  is obtained by decrypting  $E_{K(y)}(\kappa(y)) \in Pub$  using  $K(y) \in \sigma(x)$ .

Variants of the trivial key encrypting scheme have appeared in the literature [26, 38, 40], accompanied by the claim that such schemes make key updates easier. It is certainly true that it is possible to change  $\kappa(y)$  without having to change  $\sigma(x)$  or  $\kappa(x)$  for any  $x \neq y$ , as we simply choose a new value for  $\kappa(y)$  and encrypt the new key with  $K(y)$ .

However, in the event of a user (with security label  $y$ ) leaving, the key that needs to be changed is  $K(y)$  not  $\kappa(y)$ . In other words, TKEKAS is only useful in the event that a key  $\kappa(y)$  is compromised. Moreover, this slight advantage has been gained at the expense of the introduction of public information. While the storage requirements for public information are likely to be less of a problem than those for private information, it is still desirable that a key assignment scheme should minimize the amount of public data that is

required. Finally, notice that a user is required to store the same number of (key encrypting) keys as he was in TKAS.

We now introduce a scheme in which each user (with security label  $x$ ) is only required to store a single secret value ( $\kappa(x)$ ). In other words, this is the first example of a simple key assignment scheme (that is,  $\sigma(x) = \kappa(x)$ ). Essentially the scheme encodes the information flow policy in the public information rather than the private information. The amount of public information becomes proportional to the cardinality of the  $\leq$  relation rather than the cardinality of  $L$ .

**Scheme 3** A direct key encrypting key assignment scheme (DKEKAS) has the following properties:

- Independent keys;
- $Pub = (E_{\kappa(x)}(\kappa(y)) : y < x, x \in L)$ ;
- $\kappa(y)$  is obtained by decrypting  $E_{\kappa(x)}(\kappa(y)) \in Pub$  using  $\kappa(x)$ .

DKEKAS has two desirable properties: each user requires a single key, thereby minimizing private storage requirements, and key updates are easier to perform. If a key  $\kappa(x)$  needs to be changed, we only need to change  $\kappa(y)$ ,  $y \leq x$ , and  $E_{\kappa(x)}(\kappa(y))$ ,  $y \leq x$ . The main disadvantage of DKEKAS is that a considerable amount of public data is required.

Notice that in each of the above schemes a user can derive a key in a single step, so-called *direct key derivation*. This is possible because each scheme encodes information about the full partial order relation, either in the private or in the public data. In contrast, it is possible to use the transitivity of the partial order relation to reduce the amount of information that needs to be stored. As we would expect, this reduction in storage requirements comes at a price: the derivation of keys is now an iterative process.

**Scheme 4** An iterative key encrypting key assignment scheme (IKEKAS) has the following properties:

- Independent keys;
- $Pub = (E_{\kappa(x)}(\kappa(y)) : y < x, x \in L)$ ;
- There exists a path (in the Hasse diagram of  $L$ )  $(z_0, z_1), \dots, (z_{m-1}, z_m)$ , where  $y = z_0$  and  $x = z_m$ ,  $m \geq 0$ . We decrypt  $E_{\kappa(x)}(\kappa(z_{m-1})) \in Pub$  using  $\kappa(x)$  to obtain  $\kappa(z_{m-1})$  and iterate to obtain  $\kappa(z_0) = \kappa(y)$ .

All the schemes that we have considered thus far use the public information to store encrypted keys. These schemes rely on the strength of the encryption schemes used to encrypt the keys and make use of the ordering relationship

between elements of  $L$ . Clearly, each of these schemes could use a symmetric encryption algorithm  $E$ , although many key encrypting schemes in the literature choose to use asymmetric techniques [26, 38, 40].

Note that  $Pub$  in IKEKAS is defined in terms of the set of edges in the Hasse diagram of  $L$ . Indeed, all the schemes that we have introduced in some way make direct use – either in  $Pub$  or in  $\sigma(L)$  – of either the order or covering relation defined by the information flow policy. For this reason, we collectively refer to these schemes as *edge-based* key assignment schemes. However, it is possible to define schemes in which the public information encodes the structure of the poset by assigning a value  $e(x)$  to each node  $x \in L$ . The public information associated with  $y$  is used to compute key  $\kappa(y)$  given another key  $\kappa(x)$ .

**Scheme 5** A node-based key assignment scheme (NBKAS) has the following properties:

- $Pub \supseteq (e(x) : x \in L)$ ;
- $\kappa(x) = f(e(x))$ , where  $f$  is a secret function chosen in such a way that there exists a public algorithm  $g$  such that  $g(f(e(x)), e(x), e(y)) = \kappa(y)$  for all  $y \leq x$ ;
- By construction,  $\kappa(y)$  can be derived by any user with knowledge of  $\kappa(x)$ .

It is rather more difficult to visualize a practical instance of a NBKAS, so we briefly describe the best known example in the literature. The Akl-Taylor scheme is a NBKAS whose security relies on the assumption that it is difficult to compute integral roots modulo  $n$  (where  $n$  is composite). (It is known that computing square roots modulo  $n$  is as hard as factorizing  $n$  [34].)

The public information is constructed in such a way that  $e(x) \mid e(y)$  if and only if  $y \leq x$ . Large primes  $p$  and  $q$  are chosen and  $n = pq$  forms part of the public information. Finally, a system secret  $s \in \mathbb{Z}_n^*$  is chosen and  $f(e(x))$  is defined to be  $s^{e(x)} \pmod n$ . Then  $g$  is the function defined by

$$g(\alpha, a, b) = \alpha^{\frac{b}{a}} \pmod n.$$

Hence,  $\kappa(y) = s^{e(y)}$  can be computed from  $\kappa(x)$  since

$$g(\kappa(x), e(x), e(y)) = (s^{e(x)})^{\frac{e(y)}{e(x)}} = s^{e(y)} \pmod n = \kappa(y).$$

Note that if  $y \leq x$ ,  $e(y)$  is divisible by  $e(x)$ , and key derivation is an exponentiation operation. However, if  $e(y)$  is not divisible by  $e(x)$ ,  $\kappa(y)$  can only be obtained by taking integral roots of  $\kappa(x)$ , so it is only feasible to compute  $\kappa(y)$  from  $\kappa(x)$  if  $y \leq x$ .

Note that each user is only required to store a single secret value, so NBKAS has a distinct advantage over TKEKAS. Note also that the number of items of public

information is determined by the cardinality of  $L$  rather than the cardinality of the covering relation, so it has also has some advantages over DKEKAS. Perhaps the most important practical difference between NBKAS and IKEKAS though is that key derivation is direct. Hence we would expect a NBKAS to have quicker key derivation and lower public storage than an IKEKAS. However, it has been claimed that the complexity of key derivation and the size of the public values that are stored in existing NBKASs means that they are worse than iterative schemes [2]. In Section 3, we examine these claims in more detail.

Note that any simple scheme with dependent keys (such as a NBKAS) can be converted into a non-simple scheme with independent keys in the following way:

- For each  $x \in L$ , choose some key  $\kappa'(x)$ ;
- Generate  $\kappa(L)$  using the simple key assignment scheme;
- Set  $\sigma(x) = \kappa(x)$ ;
- Define  $Pub' = Pub \cup (E_{\kappa(x)}(\kappa'(x)) : x \in L)$ , where  $E$  is a suitable symmetric key encryption algorithm.

Then if  $y \leq x$ ,  $\sigma(x)$  can be used to derive  $\kappa(y)$ , using the original key assignment scheme, and  $\kappa(y)$  can be used to decrypt  $\kappa'(y)$ . The resulting scheme is related to the trivial and direct key encrypting key schemes (Schemes 2 and 3): a change to  $\kappa'(x)$  only requires a change to  $E_{\kappa(x)}(\kappa'(x))$ , as in the trivial key encrypting scheme; and the encrypted keys are held in the publicly available data, as in the direct key encrypting scheme.

## 2.4 Security considerations

Very few key assignment schemes are unconditionally secure, although TKAS (Scheme 1) is an exception. The unconditionally secure scheme investigated by Ferrara and Masucci is a variant of this scheme [17]. In fact, their scheme is identical to TKAS except that it is possible to compress the representation of certain information flow policies and thereby achieve a scheme that requires slightly less secret information.

Most key assignment schemes, however, are not unconditionally secure. The first consideration is the assumption underlying the key assignment scheme that determines how difficult it would be for an adversary to derive a key to which they were not entitled. The security of most key assignment schemes depends on the difficulty of solving a computational problem that is assumed to be hard. The security of each of Schemes 2, 3 and 4, for example, depends on the security of the encryption algorithm  $E$ . The security of a NBKAS depends on the difficulty of determining  $\kappa(y)$  from  $\kappa(x)$ ,  $e(x)$  and  $e(y)$  when  $y \not\leq x$ . The Akl-Taylor

scheme, for example, relies on the difficulty of finding integer roots modulo  $n$ , where  $n$  is product of two primes  $p$  and  $q$ .

Given this assumption, the most important practical security issue concerns the ability of a user or sets of users to derive keys. In particular, no key assignment scheme should allow a user with security label  $x$  to derive the key of security label  $y$  if  $y \not\leq x$ . It should be noted here that what we actually mean by “should allow” is that it should be no easier for the user to derive the key than it is to solve the computational problem that underpins the key assignment scheme. We now define what we mean by collusion security.

**Definition 2** Let  $M \subseteq L$  and let  $\sigma(M)$  denote  $(\sigma(x) : x \in M)$ . We say that a key assignment scheme  $\mathcal{S}$  is collusion secure with respect to  $M$  (or  $M$ -secure) if for all  $y \in L$ , it is feasible to derive  $\kappa(y)$  knowing  $\sigma(M)$  and  $Pub$  only if  $y \leq x$  for some  $x \in M$ . We say  $\mathcal{S}$  is node secure if it is  $\{x\}$ -secure for all  $x \in L$ , and collusion secure if it is  $M$ -secure for all  $M \subseteq L$ .

A scheme that is collusion secure means that no group of users can collude to derive keys to which they are not entitled. (This is equivalent to the definition given by Atallah, Frikken and Blanton [2, Definition 4].) Schemes 2–4 described in Section 2.3 are collusion secure. Clearly we would require that a key assignment scheme is at least node secure, otherwise the key assignment scheme does not implement the information flow policy. The Akl-Taylor scheme is node secure, but it is necessary to impose some extra conditions on  $e(L)$  to produce a collusion secure scheme. We discuss this further in Section 3.

## 2.5 Summary

We conclude this section with a summary of the desirable characteristics of key assignment schemes and the degree to which our benchmark schemes possess these features. Ideally, then, a key assignment scheme should:

- require a small amount of private storage for  $\kappa(x)$  and  $\sigma(x)$ ;
- require a small amount of public storage for  $Pub$ ;
- provide a computationally efficient method for deriving keys;
- provide a computationally efficient method for updating keys;
- be collusion secure.

Each of our benchmark schemes possess at least two of these features. Table 1 summarizes the important features of each of the schemes:  $l$  is the cardinality of  $L$ ,  $e$  (“edges”) is the cardinality of the partial order relation  $\leq$ , and  $c$  is the cardinality of the covering relation  $\triangleleft$ ; typically  $l < c < e < \frac{1}{2}l^2$ . The private storage column states the number of elements that need to be stored by a user with security label  $x$ ; the number of bits of storage is proportional to this value. DKEKAS, for example, requires each user to store one key (like all simple schemes). Similarly, DKEKAS stores  $e$  public values, the constant of proportionality being the length of the output of the encryption algorithm  $E$  (for some fixed key length). The update column is to be read in an analogous way. The table does not include the storage required for the information flow policy in the public information; this is required by all schemes and can be disregarded.

We write  $\Delta x$  to denote the set  $\{y \in L : y \triangleleft x\}$  and  $\nabla x$  to denote the set  $\{y \in L : x \triangleleft y\}$ . We write  $\downarrow x$  to denote the set  $\{y \in L : y \leq x\}$  and  $\uparrow x$  to denote the set  $\{y \in L : y \geq x\}$ . The update column indicates the number of items that would need to be changed if  $\kappa(x)$  is changed (and hence  $\kappa(y)$ ,  $y \leq x$ , needs to be changed). Two sets of figures are given for TKEKAS in this column: the first set indicates the number of items required following a change to  $\kappa(x)$  and the set in brackets indicates the number following a change to  $K(x)$ . It is not possible to give figures for the cost of updates in node-based schemes because this varies from scheme to scheme.

If we assume that updates to private data are undesirable (because of the need to redistribute keys to users) and updates to public data acceptable, it is evident from the table that DKEKAS and IKEKAS both have some attractive features. In particular, key derivation and key updates are relatively easy to accomplish. DKEKAS has an advantage over IKEKAS because key derivation is not iterative. Conversely, IKEKAS requires less public storage than DKEKAS. The utility of node-based schemes is harder to assess without considering a particular example, because the storage requirements and difficulty of key updates are largely determined by the cryptographic implementation chosen. We will develop an efficient NBKAS in Section 3 and compare it to a representative IKEKAS scheme in Section 3.2.

It is worth noting that many schemes in the literature prefer to consider the ease with which changes to the information flow policy, such as the addition of new security labels, can be incorporated into a key assignment scheme. We believe that the most important consideration is the ease with which keys can be changed, since this is likely to be the most common update required in a key assignment scheme. Moreover, the ease with which keys can be updated has a direct impact on the ease with which other changes can be

**Table 1. Summary characteristics of generic key assignment schemes**

Scheme	Storage		Key derivation	Independent keys	Update $\kappa(x)$		Examples in the literature
	Private	Public			Private	Public	
TKAS	$ \downarrow x $	0	Direct	Yes	$ \uparrow(\downarrow x) $	0	[5, 46]
TKEKAS	$ \downarrow x $	$l$	Direct	Yes	$0( \uparrow(\downarrow x) )$	$1( \uparrow(\downarrow x) )$	[26, 38, 40]
DKEKAS	1	$e$	Direct	Yes	$ \downarrow x $	$ \uparrow x  +  \downarrow x $	[18, 51]
IKEKAS	1	$c$	Iterative	Yes	$ \downarrow x $	$ \downarrow x $	[2, 6, 11, 29, 30, 48, 49]
NBKAS	1	$l$	Direct	No	?	?	[1, 4, 7–9, 19, 22–25, 28, 32, 43]

accommodated.

The final column of Table 1 contains many of the schemes that have appeared in the literature. The motivation for many of these schemes has been the perceived drawbacks of another scheme. However, the resulting “improved” schemes simply substitute one set of disadvantages (corresponding to those of the generic scheme underlying the original scheme) for another set (those of the generic scheme underlying the new scheme).

Many of the schemes also suffer from inappropriate use of cryptographic primitives. By way of illustration, we mention two schemes [26, 38] that use the Rabin cryptosystem [34] to encrypt keys.

The basic idea is to associate each  $x \in L$  with a public-private key pair: the private key is a pair of large primes  $(p(x), q(x))$  and the public key is  $(d(x), n(x))$ , where  $n(x) = p(x)q(x)$ . Then the scheme is set up in the following way:

- Independent keys;
- $\sigma(x) = (y, p(y), q(y) : y \leq x)$ ;
- $Pub = (d(x) : x \in L) \cup (n(x) : x \in L) \cup ((\kappa(x) \parallel x)((\kappa(x) \parallel x) + d(x)) \bmod n(x) : x \in L)$ .

It should be clear that this is no more than a TKEKAS that uses the Rabin cryptosystem to encrypt the keys  $\kappa(L)$ . Given that a symmetric encryption algorithm would be suitable and that symmetric keys are far smaller than asymmetric keys of equivalent strength, it is unclear what advantages such a scheme offers. The schemes in the literature try to reduce the amount of storage required by using the Chinese Remainder theorem, but we believe that this really misses the point: TKEKAS schemes do not need to employ complicated cryptographic techniques. (The Chinese Remainder theorem has also been used in instances of TKAS [5] and DKEKAS [51].)

There are many other schemes that use complicated methods to implement what are rather simple schemes. There have been several papers on node-based schemes, for example, that use interpolating polynomials, none of which appear to be collusion secure [4, 7, 22, 23, 25, 28, 43].

We suggest that the most effective schemes in the literature have been those that implement an IKEKAS. The first of these schemes was proposed by Lin [29]: the trusted centre publishes a prime  $p$  and a primitive element  $a$  modulo  $p$  and defines

$$Pub = ((a^{\kappa(x) \oplus y} \bmod p) \oplus \kappa(y) : y \leq x, x \in L).$$

A number of schemes have used hash functions as the key encrypting mechanism [2, 6, 11, 49]. Atallah, Frikken and Blanton [2], for example, define

$$Pub = (\kappa(y) - h(\kappa(x), y) : y \leq x, x \in L).$$

### 3 An analysis of the Akl-Taylor schema

Akl and Taylor described a method for defining NBKASs [1]. Recall that

- the public data comprises  $\{n\} \cup \{e(x) : x \in L\}$ , where  $n = pq$  for large primes  $p$  and  $q$  and  $e(x) \mid e(y)$  if and only if  $y \leq x$ ;
- $\kappa(x) = s^{e(x)}$ , where  $s \in \mathbb{Z}_n^*$  is a system secret.

We will call this general approach the *Akl-Taylor schema*: an *Akl-Taylor scheme* is an instance of the schema distinguished by the choice of  $e$ . Note that if  $y \not\leq x$ , it is computationally hard to compute  $\kappa(y)$  using  $\kappa(x)$ . In other words, any Akl-Taylor scheme is node secure. Moreover, the following result was proved by Akl and Taylor [1].

**Proposition 3** *Given an Akl-Taylor scheme with labelling  $e$ ,  $\kappa(y)$  can be feasibly computed from a set of keys  $\kappa(M)$ ,  $M \subseteq L$ , if and only if*

$$\gcd\{e(x) : x \in M\} \mid e(y). \quad (1)$$

By Proposition 3, the problem of finding collusion secure Akl-Taylor schemes is reduced to finding a labelling with the following property:

$$\text{for all } y \in L, \gcd\{e(x) : x \not\leq y\} \nmid e(y). \quad (2)$$

Akl and Taylor proposed the following labelling: for each  $x \in L$  choose a distinct prime  $p(x)$  and define

$$e(x) = \prod_{z \not\leq x} p(z).$$

This has the property that  $y \leq x$  if and only if  $e(x) \mid e(y)$  and satisfies condition (2), and hence the scheme is collusion secure. The Akl-Taylor scheme corresponding to this choice of  $e(L)$  is illustrated in Figures 1(a)–1(c).

It has been argued that this Akl-Taylor scheme is unlikely to be useful in practice because of its computational and storage overheads (see [1, 2], for example). In a large poset, products of primes that make up the public information are likely to be large and  $e(x)$  will require  $\mathcal{O}(l \log l)$  bits of storage, where  $l$  is the cardinality of  $L$ . Hence, the total public storage required will be  $\mathcal{O}(l^2 \log l)$ . Moreover, the division of two such products in order to derive a key requires  $\mathcal{O}(l^2 \log^2 l)$  operations. In the next section, we demonstrate that the storage requirements and key derivation time of the Akl-Taylor scheme can be considerably reduced. In Section 3.2, we compare the Akl-Taylor scheme to an IKEKAS and examine the claims that IKEKASs are better than the Akl-Taylor scheme.

### 3.1 Simplifying the Akl-Taylor scheme

We make the following observations: the security of an Akl-Taylor scheme relies on the fact that it is difficult to compute integral roots modulo  $n$ ; for any scheme,  $y \leq x$  implies  $e(x) \mid e(y)$ . In other words, in order to define an Akl-Taylor scheme we simply have to find a function  $e : L \hookrightarrow D(m)$ , where  $D(m)$  is the set of integer divisors of  $m$ , for some integer  $m$ , and  $d \leq d'$  if  $d' \mid d$ . In other words,  $e$  is an *order embedding*<sup>1</sup> of  $L$  into  $D(m)$ .

Further consideration of the scheme reveals that the embedding  $e$  chosen by Akl and Taylor simply represents an order embedding of  $L$  into  $D(m)$ , where

$$m = \prod_{x \in L} p(x) \quad \text{and} \quad e(x) = \frac{m}{\prod_{y \leq x} p(y)}.$$

Trivially, we can treat the association of  $x \in L$  with a prime  $p(x)$  as an isomorphism; we can extend this to an order isomorphism<sup>2</sup>  $\phi$  from  $(2^L, \subseteq)$  to  $(D(m), \mid)$ , where

$$\phi(M) = \prod_{x \in M} p(x).$$

Moreover, we can define the order embedding  $e' : L \hookrightarrow 2^L$ , where  $e'(x) = L \setminus \downarrow x$ .

<sup>1</sup>Informally, an order embedding  $e : X \hookrightarrow Y$  is a function that preserves the ordering on elements in  $X$  [14]. In other words, the Hasse diagram of  $Y$  contains the Hasse diagram of  $X$  as a subgraph.

<sup>2</sup>Informally, two posets are order isomorphic if their Hasse diagrams are identical.

**Proposition 4**  $e = \phi \circ e'$ .

**Proof** For all  $x \in L$ ,

$$\begin{aligned} e(x) &= \prod_{z \not\leq x} p(z) \\ &= \phi(\{z \in L : z \not\leq x\}) \\ &= \phi(\{z \in L : z \notin \downarrow x\}) \\ &= \phi(L \setminus \downarrow x) \\ &= \phi(e'(x)). \end{aligned}$$

■

In other words, the Akl-Taylor scheme is simply an encoding of the sets  $L \setminus \downarrow x$ ,  $x \in L$ , using a set of primes. This means that it is sufficient to use a representation of  $L \setminus \downarrow x$  as the public information. Computing the quotient of  $e(x)$  in  $e(y)$  corresponds to computing the set of elements in  $e'(y)$  that are not in  $e'(x)$ , which we denote  $e'(y) \setminus e'(x)$ . Hence we have

$$\kappa(y) = (\kappa(x))^{\phi(e'(y) \setminus e'(x))}.$$

Figure 1(d) illustrates  $e'$  for our running example. ( $e'$  is actually an order embedding of  $L$  into  $2^L$ , but for simplicity we have only shown the relevant subsets of  $L$ .)

In practice,  $e'(x)$  can be represented as an  $l$ -bit string in the natural way. That is, enumerate the elements of  $L$  and define the  $i$ th bit of  $e'(x)$  to be 1 if  $x_i \in e'(x)$  and 0 otherwise. We call  $e'(x)$  the *characteristic value* of  $x$ . Hence  $Pub = \{e'(x) : x \in L\}$  requires  $l^2$  bits of storage. Since  $e'(x)$  agrees with  $e'(y)$  on every bit position that contains a 1 (when  $y \leq x$ ),  $e'(y) \setminus e'(x)$  is obtained simply by computing  $e'(x) \oplus e'(y)$ , which can obviously be computed very quickly. Consider elements  $x_2$  and  $x_5$ , for example:  $e'(x_2) = 101001$  and  $e'(x_5) = 111101$ ;  $e'(x_2) \oplus e'(x_5) = 010100$ , which corresponds to the quotient 3.7 of 2.5.13 and 2.3.5.7.13.

In summary, the implementation of our modified Akl-Taylor scheme requires  $l^2$  bits of storage for the public information and  $l \log l$  bits for a list of distinct primes each of which is associated (via  $\phi$ ) with a unique element of  $L$ . Derivation of  $\kappa(y)$  requires a single XOR operation, the multiplication of at most  $l$  primes, and one exponentiation; it is achieved by computing

$$(\kappa(x))^{\phi(e'(y) \oplus e'(x))}.$$

Hence, under the usual assumption that that computing integral roots modulo  $n$  is hard, we have the following result.

**Proposition 5** *The modified Akl-Taylor scheme is collusion secure.*

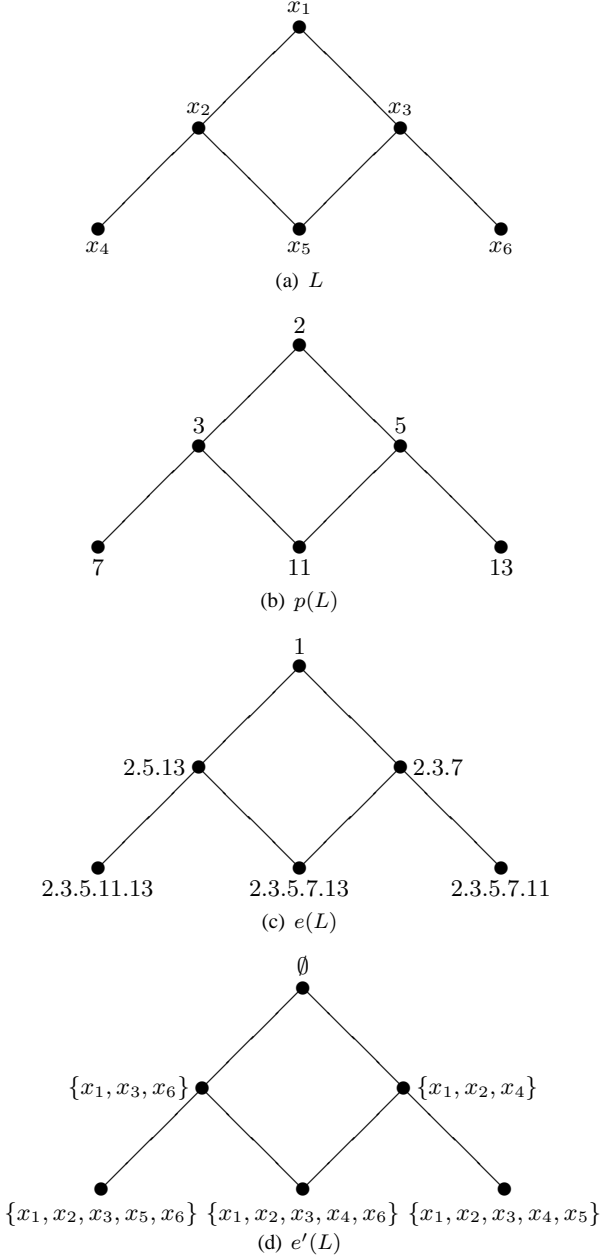


Figure 1. Akl-Taylor embeddings of  $L$

**Proof** Condition (1) in Proposition 3 is equivalent to saying that for any set of keys  $\kappa(M)$ ,  $M \subseteq L$ , we can derive  $\kappa(y)$  if and only if

$$\bigcap_{x \in M} e'(x) \subseteq e'(y).$$

Now, by definition,  $y \in e'(x)$  for all  $x \not\leq y$ , which implies that

$$y \in \bigcap_{x \not\leq y} e'(x).$$

Moreover  $y \notin e'(y)$  and hence

$$\bigcap_{x \not\leq y} e'(x) \not\subseteq e'(y).$$

Hence  $\{\kappa(x) : x \not\leq y\}$  cannot be used to derive  $\kappa(y)$ .<sup>3</sup> ■

In summary, we have shown that the Akl-Taylor scheme can be implemented more efficiently without compromising its security.

### 3.2 A comparison with an iterative key encrypting scheme

We now compare our modified Akl-Taylor scheme with perhaps the simplest and most efficient IKEKAS in the literature. Atallah, Frikken and Bykova recently proposed a scheme (henceforth referred to as the AFB scheme) in which  $Pub = \{\kappa(y) - h(\kappa(x), y) : y \leq x\}$ , where  $h : \{0, 1\}^* \rightarrow \{0, 1\}^\rho$  is a hash function [2]. All keys are assumed to be  $\rho$  bits in length. Given  $\kappa(x)$ , it is possible to compute  $\kappa(y)$ , for any  $y \leq x$ , by computing  $h(\kappa(x), y)$  and adding this to the relevant item of public data.

A claim that remains unchallenged in the literature, and which is repeated by Atallah *et al*, is that a change to  $\kappa(x)$  in an IKEKAS only requires changes to a small amount of public data: namely,  $(E_{\kappa(x)}(\kappa(y)) : y \leq x)$  and  $(E_{\kappa(z)}(\kappa(x)) : z \geq x)$ . However, if  $\kappa(x)$  has been compromised or a user with security label  $x$  is to be removed from the scheme, then  $\kappa(y)$  may have been derived from  $\kappa(x)$  and should be considered to be compromised. In other words, it is actually necessary to change  $\kappa(y)$  for all  $y \leq x$ , and hence to change the public information  $(E_{\kappa(z)}(\kappa(y)) : y \leq x, y \leq z)$ .

A further assumption that has not been challenged is that it is far more difficult to update keys in the Akl-Taylor scheme than in an IKEKAS. In fact, such a change requires very little change to the public information and may not require many more changes to private keys than an IKEKAS.

**Proposition 6** *In order to change the set of keys  $\{\kappa(y) : y \leq x\}$ , in the Akl-Taylor scheme, it is sufficient to change one prime  $p(z)$ , where  $z \in L \setminus \downarrow x$ .*

**Proof** Note that for all  $y \leq x$ ,  $\kappa(y) = s^{e(y)}$  and

$$e(x) = \prod_{z \not\leq x} p(z).$$

Hence a change to  $p(z)$  for any  $z \in L \setminus \downarrow x$  causes a change to  $e(x)$ , and hence to  $\kappa(x)$ . Moreover, for any  $y \leq x$ ,  $z \notin \downarrow y$  (otherwise we have  $z \leq y$  and  $z \leq x$  by transitivity), so  $e(y)$  is also changed. ■

<sup>3</sup>A more direct proof simply observes that the labelling  $e$  satisfies condition 2 and that  $e = \phi \circ e'$ , where  $\phi$  is the order isomorphism between  $2^L$  and  $D(m)$ .

Hence we can choose to change the prime that causes the least disruption to elements that do not belong to  $\downarrow x$ . Specifically, we change the prime associated with a minimal element in  $L \setminus \downarrow x$  because this will not change  $e(w)$  for any other element  $w \geq z$ . In particular, if there is a unique minimal element in  $L \setminus \downarrow x$ , then only those keys for elements in  $\downarrow x$  are changed.

To clarify key updates in the Akl-Taylor scheme, consider a change to  $\kappa(x_2)$  in our example. In order to change  $\kappa(x_2)$  we must change  $e(x_2)$ , which can be effected by altering  $p(x_1)$ ,  $p(x_3)$  or  $p(x_6)$ . It is natural to choose a new prime for  $x_6$ , the minimal element in  $L \setminus \downarrow x_2$ . If we replace 13 with 17 say, we have  $e(x_2) = 2.5.17$ ,  $e(x_4) = 2.3.5.11.17$  and  $e(x_5) = 2.3.5.7.17$ . In other words, all we need to change is the prime associated with  $x_6$ . Note that no other public values are affected, since we do not need to compute products of primes in our modified representation of the Akl-Taylor scheme.

In this particular example, no more keys need to be changed in the Akl-Taylor scheme than would have been necessary in the AFB scheme (or in any other IKEKAS). In other words, IKEKASs do not necessarily enjoy any advantage over the Akl-Taylor scheme with respect to key updates. Indeed, more public information will need to be changed in an IKEKAS than in the Akl-Taylor scheme.

It is also worth noting that global key updates can be performed in the Akl-Taylor scheme without any changes to the public information. The trusted centre simply chooses a new secret system value  $s'$  and computes  $\kappa(x) = (s')^{e(x)} \bmod n$ ,  $x \in L$ .

Nevertheless, the fact remains that key generation and key derivation in the Akl-Taylor scheme require exponentiation modulo  $n$ , and the same operations in the AFB scheme only require the evaluation of a hash function. (The fact that key derivation is iterative in the AFB scheme does not mitigate this fact, as the number of hash function computations is bounded by the maximal length of a path in the Hasse diagram of  $L$ .) This means that these fundamental operations in a key assignment scheme will be orders of magnitude slower in the Akl-Taylor scheme. In practical applications, this consideration is likely to outweigh all others. This leads us to the conclusion that IKEKASs are preferable to existing NBKASs.

## 4 Domain-based key assignment schemes

Ignoring issues of key derivation and generation, the disadvantage of node-based schemes such as Akl-Taylor is that a change to a single key may require several other keys to be updated (and redistributed to the appropriate users). Specifically, if we need to change  $\kappa(x)$ , then we need to change  $e(x)$ . We have seen in Section 3.2 that it is sufficient to change  $p(z)$  for some  $z \notin \downarrow x$ . Nevertheless, this has the

effect of changing  $e(y)$  for all  $y \notin \uparrow z$ .<sup>4</sup> In other words, a number of keys may need to be changed, in addition to those for elements in  $\downarrow x$ . It is clear that node-based schemes are particularly unsuitable for posets of small height and large width.<sup>5</sup> Conversely, the disadvantages of IKEKASs such as AFB are that key derivation is not direct and a considerable amount of public information may need to be changed following a key update. Such schemes are unsuitable for posets of large height. In short, it might be useful to be able to combine the advantages of different key assignment schemes.

In this section we describe how to combine several key assignment schemes to produce a domain-based key assignment scheme. Each of the domains in such a scheme is a partially ordered set and has its own key assignment scheme. Informally, each domain is then treated as a node in an IKEKAS to “stitch” the key assignment schemes together.

### 4.1 Overview

**Definition 7** Let  $(L, \leq)$  be a poset. A family of subsets of  $L$  is a domain quasi-partition (DQP) of  $L$ , denoted  $\mathcal{D}(L)$ , if

- for all  $D \in \mathcal{D}(L)$ ,  $D$  has a maximal element;
- for all distinct  $D, D' \in \mathcal{D}(L)$ ,  $D \cap D' = \emptyset$ ;
- for all  $x \in D$  and for all  $y \in \uparrow x \setminus D$ ,  $y \geq z$ , where  $z$  is the maximal element in  $D$ .

Each  $D \in \mathcal{D}(L)$  is a domain.

Note that we do not require the union of elements in  $\mathcal{D}(L)$  to equal  $L$ , so  $\mathcal{D}(L)$  is not a partition in the true mathematical sense. In the next section we show that it is possible to construct a DQP for any  $L$ . Henceforth we write  $\mathcal{D}$  rather than  $\mathcal{D}(L)$  when  $L$  is obvious from context.

By a slight abuse of notation, we write  $L \setminus \mathcal{D}$  to denote the set of elements in  $L$  that do not belong to any domain in the DQP. We extend a DQP to a true partition of  $L$  by including the set  $L \setminus \mathcal{D}$ . We say  $(y, x)$  is a *domain edge* if  $y \leq x$  and either  $x$  and  $y$  belong to the same domain or  $x$  and  $y$  belong to  $L \setminus \mathcal{D}$ ; we say  $(y, x)$  is a *non-domain edge* if  $y \leq x$  and  $(y, x)$  is not a domain edge. We will write  $y \triangleleft x$  if  $(y, x)$  is a non-domain edge.

**Scheme 6** A domain-based key assignment scheme for an information flow policy  $(L, \leq)$  comprises:

- a DQP  $\mathcal{D}$ ;

<sup>4</sup>In Figure 1,  $L \setminus \downarrow x_4$ , for example, has two minimal elements ( $x_5$  and  $x_6$ ). Hence if we change  $\kappa(x_4)$ , by changing  $p(x_5)$ , say, we will change  $\kappa(x_6)$  in addition to  $\kappa(x_4)$ .

<sup>5</sup>The *height* of a poset  $L$  is the maximum cardinality of a chain in  $L$ ; the *width* of  $L$  is the maximum cardinality of an antichain in  $L$ .

- for each  $D \in \mathcal{D}$ , a key assignment scheme for  $(D, \leq)$ ;
- a key assignment scheme for  $(L \setminus \mathcal{D}, \leq)$ ;
- a set of public information associated with the non-domain edges  $Pub_{\triangleleft} = \{E_{\kappa(x)}(\kappa(y)) : y \triangleleft x\}$ .

In practice, at least one domain will employ a key assignment scheme that has direct key derivation (otherwise, we may as well use a single IKEKAS). For the purposes of clear exposition, we henceforth assume that each domain uses a NBKAS.

Then derivation of  $\kappa(y)$  from  $\kappa(x)$  ( $y \leq x$ ) can be performed in a single step if  $y$  and  $x$  belong to the same domain or if  $y$  and  $x$  belong to  $L \setminus \mathcal{D}$ , since the appropriate NBKAS can be used. If  $x \in L \setminus \mathcal{D}$  (and  $y \in D$  for some  $D \in \mathcal{D}$ ), then, by definition of a DQP,  $D$  has a maximal element  $z$  and  $x \geq z$ .  $Pub_{\triangleleft}$  is used to compute  $\kappa(z)$ , which can then be used to derive  $\kappa(y)$ . If  $x \in D$  for some  $D \in \mathcal{D}$  (and  $y \in L \setminus \mathcal{D}$ ), there exists an element  $z \in D$  and an element  $w \in L \setminus \mathcal{D}$ , such that  $w \triangleleft z$ . Hence the NBKAS for the domain containing  $x$  can be used to compute  $\kappa(z)$ , and  $Pub_{\triangleleft}$  can be used to compute  $\kappa(w)$ , which can be used to derive  $\kappa(y)$ .

In practice, the trusted centre selects a DQP, and creates a NBKAS for each domain. The trusted centre then creates an NBKAS for the set of elements that do not belong to any domain. The trusted centre uses these keys, the non-domain edge set and the set of keys defined by the domain NBKASs to construct  $Pub_{\triangleleft}$ .

In our running example, we might define  $\mathcal{D}$  to be  $\{D_2, D_3\}$ , where  $D_2 = \{x_2, x_4\}$  and  $D_3 = \{x_3, x_6\}$ . Then  $L \setminus \mathcal{D} = \{x_1, x_5\}$  and the set of non-domain edges is

$$\{(x_5, x_2), (x_5, x_3), (x_2, x_1), (x_3, x_1)\}.$$

For simplicity, we write  $\kappa_i$  to denote  $\kappa(x_i)$ . The trusted centre creates a NBKAS for each of  $D_2$ ,  $D_3$  and  $\{x_1, x_5\}$ , and publishes

$$Pub_{\triangleleft} = \{E_{\kappa_1}(\kappa_2), E_{\kappa_1}(\kappa_3), E_{\kappa_2}(\kappa_5), E_{\kappa_3}(\kappa_5)\}.$$

Then a user assigned to security label  $x_1$  can derive  $\kappa_4$ , for example, by first computing  $\kappa_2$  from  $\kappa_1$  and  $Pub_{\triangleleft}$ , and then computing  $\kappa_4$  directly from  $\kappa_2$  using the NBKAS for  $D_2$ .

In this simple example, little is gained in terms of key derivation. Note, however, that changes can be made to  $\kappa_4$  and  $\kappa_6$  without affecting the keys associated with nodes in other domains. Moreover, new leaf nodes can be added to  $D_2$  or  $D_3$  without affecting keys outside the respective domains. Recall from Section 3.2 that  $e(y)$  is only changed if  $p(z)$ ,  $z \not\leq y$ , is changed. In particular, if  $x$  is a maximal element in an Akl-Taylor scheme,  $\kappa(x)$  is unaffected by any key changes or the addition of new nodes within this domain. In other words, the effect of changes to non-maximal elements in a domain remain local to a domain.

## 4.2 Choosing domains

In this section, we describe how recent work on role-based administration [12, 13] can be used to define  $\mathcal{D}(L)$  for any  $L$ . Given a poset  $L$ , define  $\alpha : L \rightarrow 2^L$ , where

$$\alpha(x) = \{y \in \downarrow x : \uparrow y \subseteq \downarrow x \cup \uparrow x\}.$$

We say  $\alpha(x)$  is the *administrative scope* of  $x$  [13]. We say that  $D \subseteq L$  is an *administrative domain* with *administrator*  $x$  if  $D = \alpha(x)$  for some  $x \in L$ . It can be shown that an administrative domain  $D = \alpha(x)$  has a unique maximal element  $x$  [13]. It can also be shown that any two administrative domains are either nested or disjoint [12]. Moreover, it follows from the definition of administrative scope that given  $x \in \alpha(z)$ , for all  $y \in \uparrow x \setminus \alpha(z)$ ,  $y \geq z$ . Therefore, any set of disjoint administrative domains can be used to define a DQP. In our running example, the administrative domains are

$$\begin{aligned} \alpha(x_1) &= \{x_1, x_2, x_3, x_4, x_5, x_6\}, \\ \alpha(x_2) &= \{x_2, x_4\}, \alpha(x_3) = \{x_3, x_6\}, \\ \alpha(x_4) &= \{x_4\}, \alpha(x_5) = \{x_5\}, \alpha(x_6) = \{x_6\}. \end{aligned}$$

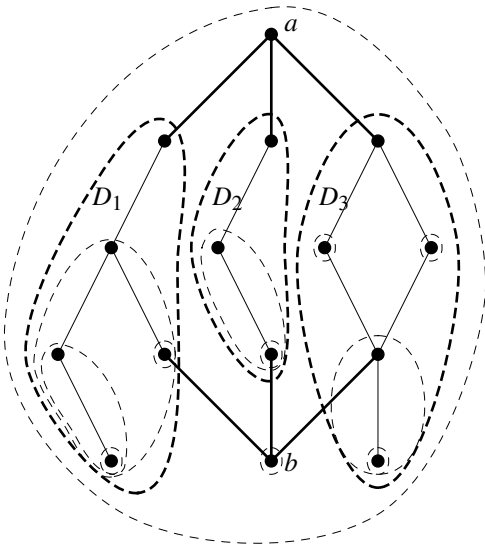
Note that  $x_5 \notin \alpha(x_2)$  since  $x_3 \in \uparrow x_5$  and  $x_3 \notin \uparrow x_2$ . (By symmetry,  $x_5 \notin \alpha(x_3)$ .)

In practice, not all administrative domains would be used in constructing a domain-based key assignment scheme. In fact, since the set of administrative domains ordered by subset inclusion is a forest, it will usually be the case that the best choice of domains is the set of administrative domains that are immediate children of the maximal elements in the administrative domain forest.

Figure 2 illustrates a more complex poset and its administrative domains. In practice, we might use the domains  $D_1$ ,  $D_2$  and  $D_3$  as the building blocks for a domain-based scheme. The non-domain edges are marked in bold.

The advantages of domain-based key assignment schemes can be seen more clearly in this example. It would require four steps to derive the key of the minimal element in domain  $D_3$  from  $\kappa(a)$ , for example. Using a domain-based scheme, it would only require two steps: one to calculate the key for the maximal element in  $D_3$  (using the IKEKAS public information and  $\kappa(a)$ ) and one to compute the required key using the NBKAS for  $D_3$ . Moreover,  $\kappa(b)$  can be derived from  $\kappa(a)$  in a single step.

Naturally, we have chosen an example of a poset that illustrates the utility of our hybrid approach. Clearly, this approach will not be appropriate for posets in which each administrative domain is very small. In such cases, we may as well use an IKEKAS. Conversely, a domain-based scheme will not be particularly useful for a poset that is a chain, because it is natural to use a NBKAS for such a poset. In short, domain-based key assignment schemes are likely to



**Figure 2. Choosing domains for a domain-based key assignment scheme**

be useful if it is possible to find a DQP of cardinality  $k$  such that  $k \lll l$  and the cardinality of the non-domain edge set is much smaller than that of the covering relation on  $L$ .

## 5 Concluding remarks

Key assignment schemes provide a cryptographic way of controlling access to resources, given that users and resources can be associated with some security label. While there has been considerable interest in key assignment schemes, there has been little attempt to understand the basic features of such schemes and the relationships that exist between these features. We have provided the first general framework for key assignment schemes and identified the relative merits of five types of scheme. This has enabled us to classify all the schemes in the literature of which we are aware. Many of these schemes are unnecessarily elaborate.

We can also make some general recommendations about the use of key assignment schemes. Specifically, an IKEKAS will generally be the best type of scheme to employ in practice. Perhaps the only exception to this is when the user population is assumed to very stable and key updates are likely to be rare. In this case, a DKEKAS provides direct key derivation at the cost of an increase in the amount of public information. We note that the AFB scheme can be used to implement a DKEKAS that is far simpler than any of the DKEKASs described in the literature.

We have also shown that the Akl-Taylor scheme, the

archetypal node-based scheme, can be implemented more efficiently in terms of storage and key derivation time than had previously been known. There are a number of interesting research questions regarding node-based schemes: Do there exist embeddings of  $L$  that reduce the storage requirements in the Akl-Taylor schema still further or is the embedding into the set of order filters that we introduce in Section 3.1 optimal?<sup>6</sup> More importantly, are there other, more efficient, secret functions that can be used to underpin a node-based key assignment scheme?

Finally, we have introduced the idea of a domain-based key assignment scheme. This is a hybrid scheme, that uses features of both iterative key encrypting schemes and node-based schemes. Our approach is particularly useful for posets that contain large administrative domains. An efficient NBKAS would significantly increase the advantages of a hybrid approach.

A number of extensions have been proposed for key assignment schemes. Perhaps the most common in the literature is the use of temporal constraints on the use of keys [10, 21, 41, 47]. In such schemes, keys are updated periodically and users are able to derive the keys for certain pre-determined time intervals. Modelling the set of contiguous time intervals as a partially ordered set (under subset inclusion) and taking the direct product of this set with the information flow policy  $(L, \leq)$  yields a generic approach to temporally constrained key assignment schemes that we have not seen used in the literature. We intend to explore this approach in more detail.

Finally, we note that the classic information flow policy is based on a security lattice  $C \times 2^K$ , where  $C$  is a totally ordered set of security classifications and  $K$  is a set of need-to-know categories [3]. This lattice is order isomorphic to the chain product  $S(c, 1, \dots, 1)$  [16], where  $c = |C|$ . We conjecture that there may be efficient key assignment schemes for such lattices, since they can be generated (in some technical sense) by  $|K| + 1$  elements.

On a more speculative note, we observe that key assignment schemes have some similarity to broadcast encryption schemes. We believe it may be instructive to investigate whether the techniques used in one context might be usefully applied in the other. We hope to report some interesting results in the near future.

**Acknowledgements** We would like to thank the anonymous referees for their helpful comments, which have led to significant improvements in the final paper.

<sup>6</sup>We have already proved that the most economical embedding for the Akl-Taylor schema, defined by MacKinnon *et al* [32], can be treated as an embedding into  $D(p_1^{\lambda_1} \dots p_w^{\lambda_w})$ , where  $w$  is the width of the poset,  $p_1, \dots, p_w$  are distinct primes, and  $\lambda_1, \dots, \lambda_w$  are positive, non-zero integers. Unfortunately, space constraints preclude a detailed description of this work.

## 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, K. B. Frikken, and M. Blanton. Dynamic and efficient key management for access hierarchies. In *Proceedings of 12th ACM Conference on Computer and Communications Security*, pages 190–202, 2005.
- [3] D. Bell and L. LaPadula. Secure computer systems: Unified exposition and Multics interpretation. Technical Report MTR-2997, Mitre Corporation, Bedford, Massachusetts, 1976.
- [4] C.-C. Chang, R.-J. Hwang, and T.-C. Wu. Cryptographic key assignment scheme for access control in a hierarchy. *Information Systems*, 17(3):243–247, 1992.
- [5] T.-S. Chen and Y.-F. Chung. Hierarchical access control based on Chinese Remainder Theorem and symmetric algorithm. *Computers & Security*, 21(6):565–570, 2002.
- [6] T.-S. Chen and J.-Y. Huang. A novel key management scheme for dynamic access control in a user hierarchy. *Applied Mathematics and Computation*, 162:339–351, 2005.
- [7] T.-S. Chen, K.-H. Huang, and Y.-F. Chung. Modified cryptographic key assignment scheme for overcoming the incorrectness of the CHW scheme. *Applied Mathematics and Computation*, 159:147–155, 2004.
- [8] W.-M. Chew, C.-H. Chi, and T.-Y. Li. Efficient key assignment scheme for mobile agent systems. In *Proceedings of the 15th International Conference on Tools with Artificial Intelligence (ICTAI'03)*, pages 338–345, 2003.
- [9] G. Chick and S. Tavares. Flexible access control with master keys. In *Advances in Cryptology - CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 316–322. Springer-Verlag, 1990.
- [10] H.-Y. Chien. Efficient time-bound hierarchical key assignment scheme. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1301–1304, 2004.
- [11] H.-Y. Chien and J.-K. Jan. New hierarchical assignment without public key cryptography. *Computers & Security*, 22(6):523–526, 2003.
- [12] J. Crampton. Understanding and developing role-based administrative models. In *Proceedings of the 12th ACM Conference on Computer and Communications Security*, pages 158–167, 2005.
- [13] J. Crampton and G. Loizou. Administrative scope: A foundation for role-based administrative models. *ACM Transactions on Information and System Security*, 6(2):201–231, 2003.
- [14] B. Davey and H. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, Cambridge, United Kingdom, 1990.
- [15] D. Denning. A lattice model of secure information flow. *Communications of the ACM*, 19(5):236–243, 1976.
- [16] K. Engel. *Sperner Theory*. Cambridge University Press, Cambridge, United Kingdom, 1997.
- [17] A. Ferrara and B. Masucci. An information-theoretic approach to the access control problem. In C. Blundo and C. Laneve, editors, *ICTCS 2003*, volume 2841 of *Lecture Notes in Computer Science*, pages 342–354. Springer-Verlag, 2003.
- [18] E. Gudes. The design of a cryptography based secure file system. *IEEE Transactions on Software Engineering*, SE-6(5):411–419, 1980.
- [19] L. Harn and H. Lin. A cryptographic key generation scheme for multilevel data security. *Computers and Security*, 9(6):539–546, 1990.
- [20] C.-L. Hsu and T.-S. Wu. Cryptanalyses and improvements of two cryptographic key assignment schemes for dynamic access control in a user hierarchy. *Computers & Security*, 22(5):453–456, 2003.
- [21] H.-F. Huang and C.-C. Chang. A new cryptographic key assignment scheme with time-constraint access control in a hierarchy. *Computer Standards & Interfaces*, 26:159–166, 2004.
- [22] M.-S. Hwang. Extension of CHW cryptographic key assignment scheme in a hierarchy. *IEE Proceedings – Computers & Digital Techniques*, 146(4):219, 1999.
- [23] M.-S. Hwang. An improvement of a dynamic cryptographic key assignment scheme in a tree hierarchy. *Computers and Mathematics with Applications*, 37:19–22, 1999.
- [24] M.-S. Hwang and W.-P. Yang. Controlling access in large partially ordered hierarchies using cryptographic keys. *Journal of Systems and Software*, 67:99–107, 2003.
- [25] M.-S. Hwang, W.-P. Yang, and C.-C. Chang. Modified Chang-Hwang-Wu access control scheme. *Electronics Letters*, 29(24):2095–2096, 1993.
- [26] F. Kuo, V. Shen, T. Chen, and F. Lai. Cryptographic key assignment scheme for dynamic access control in a user hierarchy. *IEE Proceedings – Computers & Digital Techniques*, 146(5):235–240, 1999.
- [27] N.-Y. Lee and T. Hwang. Comments on dynamic key management schemes for access control in a hierarchy. *Computer Communications*, 22:87–89, 1999.
- [28] H. Liaw, S. Wang, and C. Lei. A dynamic cryptographic key assignment scheme in a tree structure. *Computers and Mathematics with Applications*, 25(6):109–114, 1993.
- [29] C.-H. Lin. Dynamic key management schemes for access control in a hierarchy. *Computer Communications*, 20:1381–1385, 1997.
- [30] C.-H. Lin. Hierarchical key assignment without public key cryptography. *Computers & Security*, 20(7):612–619, 2001.
- [31] I.-C. Lin, M.-S. Hwang, and C.-C. Chang. A new key assignment scheme for enforcing complicated access control policies in a hierarchy. *Future Generation Computer Systems*, 19:457–462, 2003.
- [32] S. MacKinnon, P. Taylor, H. Meijer, and S. Akl. An optimal algorithm for assigning cryptographic keys to control access in a hierarchy. *IEEE Transactions on Computers*, C-34(9):797–802, 1985.
- [33] National Institute of Standards and Technology. *Recommendation for Key Management - Part 1: General*, 2005. NIST Special Publication 800-57.
- [34] M. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical Report TR-212, Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1979.
- [35] I. Ray, I. Ray, and N. Narasimhamurthi. A cryptographic solution to implement access control in a hierarchy and more. In *Proceedings of the Seventh ACM Symposium on Access Control Models and Technologies*, pages 65–73, 2002.

- [36] R. Sandhu. Cryptographic implementation of a tree hierarchy for access control. *Information Processing Letters*, 27(2):95–98, 1988.
- [37] A. D. Santis, A. Ferrara, and B. Masucci. Cryptographic key assignment schemes for any access control policy. *Information Processing Letters*, 92:199–2005, 2004.
- [38] V. Shen, T.-S. Chen, and F. Lai. A novel key management scheme based on discrete logarithms and polynomial interpolations. *Computers & Security*, 21(2):164–171, 2002.
- [39] Q. Tang and C. Mitchell. Comments on a cryptographic key assignment scheme. *Computer Standards & Interfaces*, 27:323–326, 2005.
- [40] H.-M. Tsai and C.-C. Chang. A cryptographic implementation for dynamic access control in a user hierarchy. *Computers & Security*, 14(2):159–166, 1995.
- [41] W.-G. Tzeng. A time-bound cryptographic key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 14(1):182–188, 2002.
- [42] S.-Y. Wang and C.-S. Lai. Cryptanalysis of Hwang-Yang scheme for controlling access in large partially ordered hierarchies. *The Journal of Systems and Software*, 75:189–192, 2005.
- [43] J. Wen, J. Sheu, and T. Chen. Cryptographic key assignment scheme for overcoming the incorrectness of the CHW scheme. *IEE Proceedings – Communications*, 148(4):260–264, 2001.
- [44] J. Wu. An access control scheme for partially ordered set hierarchy with provable security. Master’s thesis, Lakehead University, Ontario, Canada, 2004. Available from [www.eprint.iacr.org/2004/295](http://www.eprint.iacr.org/2004/295).
- [45] T.-C. Wu and C.-C. Chang. Cryptographic key assignment scheme for hierarchical access control. *Computer Systems Science and Engineering*, 1(1):25–28, 2001.
- [46] C. Yang and C. Li. Access control in a hierarchy using one-way functions. *Computers & Security*, 23:659–664, 2004.
- [47] X. Yi and Y. Ye. Security of Tzeng’s time-bound key assignment scheme for access control in a hierarchy. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):1054–1055, 2003.
- [48] Y. Zheng, T. Hardjono, and J. Seberry. New solutions to the problem of access control in a hierarchy. Technical Report 93-2, Department of Computer Science, University of Wollongong, 1993.
- [49] S. Zhong. A practical key management scheme for access control in a user hierarchy. *Computers & Security*, 21(8):750–759, 2002.
- [50] S. Zhong and T. Lin. A comment on the Chen-Chung scheme for hierarchical access control. *Computers & Security*, 22(5):450–452, 2003.
- [51] X. Zou, B. Ramamurthy, and S. Magliveras. Chinese Remainder Theorem based hierarchical access control for secure group communications. In *Proceedings of International Conference on Information and Communication Security, ICICS 2001*, volume 2229 of *Lecture Notes in Computer Science*, pages 381–385. Springer-Verlag, 2001.