A Fast Attribute Based Encryption

Yacov Yacobi
5050 West Mercer Way, Mercer Island, WA, USA
yacov@live.com

March 16, 2016

Abstract

Our new *Access Control Encryption* is an implementation of CP-ABE, when used as part of a key delivery mechanism for an encrypted Data Base. It focuses on improving performance. In ACE the access policies are any predicates over the set of object attributes. Efficiency gains are most pronounced when the DNF representations of policies are compact. In ACE, within the life span of the keys, each user has to perform very few ABE decryptions, regardless of the number of policies accessible to her. Keys to most objects are then computed using only symmetric key decryptions.

ACE is not the first to utilize symmetric key cryptography to reduce the number of CP-ABE operations, when access policies form a multi-level partially ordered set. However, in addition to this significant saving, ACE also takes advantage of overlaps among policies on clauses of the policies, thus further reducing computational complexity.

Let R denote the number of user roles, N be the number of object access policies, k the ratio between the cost of CP-ABE encryption and symmetric key encryption complexities (for 10 attributes $k \approx 10^6$), and $N = cR$. The gain factor of ACE vs. a competing hybrid system is $\eta = kc/(k+c)$. Usually $c >> 1$, but in some systems it may happen that $c < 1$.

ACE is composed of two sub systems encrypting the same messages: A CP-ABE and a symmetric key encryption system. We prove that ACE is secure under a new *Uniform Security Game* that we propose and justify, assuming that its building blocks, namely CP-ABE and block ciphers are secure. We require that CP-ABE be secure under the *Selective Set Model,* and that the block cipher be secure under *Multi-User CPA*, which we define.

We present *Policy Encryption* (PE) that can replace CP-ABE as a component of ACE. In many cases, PE is more efficient than CP-ABE. However PE does not prevent collusions. Instead it limits collusions. PE is useful in those cases where owners can compartmentalize objects and subjects, so that within each compartment the owners can tolerate collusions. PE prevents inter compartmental collusions. PE has also the following appealing properties: It relies on older hence more reliable intractability assumption, the Computational Diffie-Hellman assumption, whereas CP-ABE relies on the newer Bilinear Diffie-Hellman assumption. PE uses off-the shelf standard crypto building blocks with one small modification, with proven security. For a small number of compartments PE is much faster than CP-ABE. PE and CP-ABE can coexist in the same system, where ABE is used in high security compartments.

We apply ACE to a practical financial example, the Consolidate Audit Trail (CAT), which is expected to become the largest repository of financial data in the world.

Keywords: Access Control, Attribute Based Encryption, Hierarchical Key Derivation, Monotone Access Structures, Semantic Security, Hybrid Argument proof technique.

1 Introduction

Efficient cryptographically enforced high granular access control and secure communication are among the main security challenges facing the Cloud Security industry [11]. Our new system, Access Control Encryption (ACE) aims at improving the computational efficiency of such systems. It uses a combination of CP-ABE sub-systems and symmetric key cryptography to accelerate computations.

ATTRIBUTES are binary variables. Their real world meaning can be "`works in the payroll department`", "`is a Smith family member`", etc.

THE PLAYERS in ACE are *writers, readers*, and *owners*. The owners set up the system, decide *read* and *write* access policies (aka access structures) for their objects, and create the Key Delivery layers accordingly. The owners are fully trusted, and have full access to their protected objects. During ordinary user's transaction owners may be off-line.

In ACE, access may mean *read* or *write*, a user is granted access to an object by getting a symmetric payload cryptographic key. The *read* and the *write* access policies may be distinct. In order to prevent a legitimate reader from impersonating a legitimate writer the writers sign their encrypted objects and the readers verify the signatures. We assume that writers are always legitimate readers of the objects that they wrote.

The other off-line Trusted Third Parties (TTP) are Certification & Revocation Authorities (CRA, aka *Accreditation Authority*) that issue certificates and secret CP-ABE keys, and revoke them when necessary. The various CRAs may be autonomous and uncoordinated.

Attribute Based Encryption (ABE [6, 9, 8, 13, 15, 17, 19]) is a way to encrypt data in multi-tenant storage, such as byte-oriented file system, record-oriented file system, and database system. It can enforce compliance with diverse highly granular, possibly distinct, ***`read`*** and ***`write`*** access polices. Compared with classic access control mechanisms it reduces the amount of trust required in the central facility and its personnel. `CP-`ABE is a type of public-key encryption in which the secret key of a user and the ciphertext are dependent upon attributes. In such a system, the decryption of a ciphertext is possible only if the set of attributes of the user key matches the attributes of the ciphertext.

In this paper we use *Ciphertext Policy ABE*. In CP-ABE objects are associated with access polices, which are subsets of subsets of attributes, and users with subsets of attributes. Suppose that some user is associated with a subset of attributes ω and that some object is associated with access policy $\mathbb{A}$. The user has access to that object iff $\omega \in \mathbb{A}$. Sometimes it is convenient to write it in terms of predicates. Viewing $\mathbb{A}()$ as a predicate over the attributes, the condition for access is $\mathbb{A}(\omega) = True$. For more details about access policies see the appendix.

CP-ABE is best used as part of an Authenticated Key Exchange (AKE), so that users get cryptographic keys to perform read and write operations on data iff they are authorized to do so. Our goal is to achieve this functionality more efficiently than by straightforward encrypting those keys using CP-ABE. A few versions of CP-ABE were proven secure in [6] and in the other references already mentioned above. We do not describe any particular CP-ABE in this paper.

ACE can be viewed also in the context of ABAC (NIST Special Publication 800-162, January 2014 [1]). ABAC is worth mentioning, since it includes a detailed explanation of a rich environment with many players that are needed in practice. In the ACE model given here we abstracted many of them into fewer entities.

[1] http://dx.doi.org/10.6028/NIST.SP.800-162

PARTIALLY ORDERED SETS: We assume that protected objects and their access policies are organized as a partially ordered set (posets [2]), where each element corresponds to a subset of attributes, each object's access policy corresponds to some subset of its elements, and a user's role is associated with an element of the set. This is a common view in applications of CP-ABE.

ACE is applicable to any poset, however, for the sake of concreteness we consider two such posets. Let $\mathcal{A}$ denote the set of attributes, and let $X = 2^{\mathcal{A}}$ be its powerset. The poset (X,≤) orders the elements of X by their subsets (of attributes) inclusion relations. Throughout most of this paper we discuss (X,≤), and in section 5 we discuss another poset, denoted (L,≤). In (X,≤) if node x corresponds to a subset of attributes that include the subset of attributes of node y then $x \geq y$. In that case it is convenient to describe posets using Hasse Diagrams, where in the above example, the direction of the edge is from x to y. In (L,≤), which we use in the financial example in section 5, there is in addition a hierarchy among basic attributes, which influences the poset hierarchy.

ACE has the following properties over any poset: Let $x \geq y$ be elements of the poset. Each node of the poset is associated with a secret symmetric key. *(i)* If a user can access the symmetric key of node x then she can access the symmetric key of node y and thereby gain access to any object whose access policy includes node y. *(ii)* We use symmetric key cryptosystems to enforce the one-wayness of the symmetric key derivation implied in *(i)* . *(iii)* Each user gets a maximal symmetric key using a single CP-ABE decryption. *(iv)* Another mechanism that we use to improve efficiency, and which we view as our main contribution, is utilizing overlaps among access policies. The symmetric key of each poset element is encrypted only once using ABE, regardless of the number of object access policies in which it is involved. The saving is proportional to the accumulated size of the overlaps as shown later.

In [1, 2], the symmetric key systems that we use to accelerate key delivery computations are called "Key Management for Access Hierarchies". In [7] they are called "Cryptographic Enforcement of Information Flow Control." Elsewhere it is called "Hierarchical Key Derivation" (HKD) and we adopt here this shortened name. For $m = 10$ attributes, for reasonable security parameters, HKD is about a million times faster than ABE encryption (the factor grows linearly in m). For detailed descriptions of the operations carried out by each player in the system see section 2.3.2.

Let R denote the number of user roles, N be the number of object access policies, k the ratio between the cost of CP-ABE encryption and symmetric key encryption complexities (for 10 attributes $k \approx 10^6$), and $N = cR$. The gain factor of ACE vs. a competing hybrid system is $\eta = kc/(k + c)$. Usually $c >> 1$, but in some systems it may happen that $c < 1$.

In this paper we use the following TERMINOLOGY: (a) ABE always means CP-ABE. (b) AKE stands for *Authenticated Key Exchange*. We can use any secure Encryption Based 1-pass AKE protocol such as [16], PP. 510. (c) *Modified Crampton system* [7] (or in short *Modified* [7]) is the system in [7] (also surveyed briefly in Appendix E) modified to be used as the encryption component of AKE, and such that each user gets her most privileged key using ABE.

ACE is an AKE whose encryption component is composed of two sub systems encrypting the same messages: a CP-ABE and a symmetric key encryption system, denoted (HKD || ABE). In section 3 we prove that ACE is secure in the following sense: We prove that (HKD || ABE) is secure under a new *Uniform Security Game* that we propose and justify, assuming that its building

[2] A partial order is a binary relation "≤" over a set S which is reflexive, antisymmetric, and transitive, i.e., which satisfies for all a, b, and c in S: $a \leq a$; if $a \leq b$ and $b \leq a$ then $a = b$; if $a \leq b$ and $b \leq c$ then $a \leq c$.

blocks, namely CP-ABE and block ciphers are secure. We require that CP-ABE be secure under the *Selective Set Model*, and that the block cipher be secure under *Multi-User CPA*, which we define. Therefore, if the AKE protocol is secure assuming its encryption component is secure then ACE is secure.

We present *Policy Encryption* (PE) that can replace ABE as a component of ACE. In many cases, PE is more efficient than ABE. However PE does not prevent collusions. Instead it limits collusions. PE is useful in those cases where owners can compartmentalize objects and subjects (namely, these compartments apply to access policies and to user's roles), so that within each compartment the owners can tolerate collusions by users. PE prevents inter-compartmental collusions. PE comes at the cost of adding new dummy attributes. For m PE compartments we need $2^m - 1$ additional dummy attributes. Computational complexity, while linear in the number of attributes, has very small constants. PE has also the following appealing properties: PE relies on older hence more reliable intractability assumption, the Computational Diffie-Hellman assumption, whereas CP-ABE relies on the newer Bilinear Diffie-Hellman assumption. PE uses off-the shelf standard crypto building blocks with one small modifications, with proven security. For a small number of compartments PE is much faster than CP-ABE. For example, for 10 real attributes and 5 compartments, for reasonable security parameters, it takes about 4 millisecond (worst case on high end PC) to encrypt a poset node using PE, Vs. 300 ms when using CP-ABE on comparable machines with comparable security. Above 20 compartments there is no advantage to PE. PE and CP-ABE can coexist in the same system, where CP-ABE is used in high security compartments.

We apply ACE to a practical financial example, the Consolidate Audit Trail (CAT[3] section 5). We create a special partially ordered set (poset) (L,$\leq$) over which each user's subset of attributes is a node, each object's access policy is a subset of incomparable nodes, each query is a node, and each object is a leaf.

RELATED WORK: ABE was first published by Goyal, Pandey, Sahai, and Waters in [9]. It is a KP-ABE. In [6] Bethencourt, Sahai, and Waters presented the first CP-ABE system. In [19] Waters presented a new methodology for realizing Ciphertext-Policy Attribute Encryption (CP-ABE) under concrete and noninteractive cryptographic assumptions in the standard model. In [8] Goyal, Jain, Pandey and Sahai presented the first construction of a ciphertext-policy attribute based encryption scheme having a security proof based on a number theoretic assumption and supporting advanced access structures. Previous CP-ABE systems could either support only very limited access structures or had a proof of security only in the generic group model. In [17] Ostrovsky, Sahai and Waters constructed an Attribute-Based Encryption scheme that allows a user's private key to be expressed in terms of any access formula over attributes. Previous ABE schemes were limited to expressing only monotonic access structures. In [13] Lewko, Okamoto, Sahai, Takashima and Waters presented two fully secure functional encryption schemes: A fully secure ABE scheme and a fully secure attribute hiding predicate encryption scheme for inner-product predicates. For an exposition of many other types of ABE see [11]. Identity Based Encryption (IBE) is related to ABE, and preceded it. GIBE, which generalizes IBE was published by Boneh and Hamburg in [5]. It gives the proper intuition of viewing IBE over poset $(X, \leq)$, where X is the set user roles, where each role is associated with a unique subset of attributes, and $\leq$ is the natural subset (of attributes) inclusion relation. HIBE [14] by Lewko and Waters is a way to generate an IBE secret key for a lower ID given the keys of a dominating ID. It is not ABE, but the hierarchy

[3] http://catnmsplan.com/

is similar. It is based on expensive bilinear operations, whereas we accelerate by utilizing cheap symmetric key operations, and by taking advantage of overlaps among access policies. Hierarchical Key Derivation (HKD;[1, 2, 3], Atallah et. al., Akl et. al.)) is a symmetric key method for key distribution over any poset. We use a version that Crampton uses in [7], where he showed that most ABE operations over poset $(X, \leq)$ can be replaced by symmetric key operations. However, we use it with much smaller posets. Crampton did not suggest how to deliver maximally privileged keys to users. Also, formally [7] is wrong; ABE is an asymmetric system while HKD is symmetric. All our comparisons to [7] are made with respect to modified [7] (see TERMINOLOGY above).

THE STRUCTURE OF THE REST OF THE PAPER: In section 2 we explain the meaning of basic terms like *attributes*, and *access policies*, and define the *players* and their roles. Section 3 includes formal definitions and system details. In section 3 we prove that ACE is secure. In section 4 we present Policy Encryption and in section 5 we apply ACE to a practical example: The Consolidated Audit Trail (CAT).

2 System

2.1 Background

The CRA publishes certificates linking attributes with unique public keys, and gives eligible users the corresponding secret keys. The policy setter (aka owner) decides access policies and provides keying material to eligible users based on their credentials, with which they can access objects. The users communicate with the protected data base without the need for real time involvement of either CRA or the owner. See Figure 1. Recall that we use ACE as the encryption component of an AKE protocol (see TERMINOLOGY in 1).

Our system has two layers of encryption. At the lower layer, the payload (PL) layer, eligible writers encrypt data, and eligible readers decrypt data. A higher layer, the key delivery (KD) layer, delivers PL crypto keys to eligible users. The KD layer has its *KD algorithms* and *KD data*, to be detailed in subsection 2.3. We proceed to concentrate on the KD layer. Our KD layer can support any PL layer.

OWNERS decide the access policies of each object. For a given object, the *read* and *write* access policies may be distinct. The owners create the Key Delivery layers for each object accordingly. When an owner initiates an object, she picks a payload key and (in a first simple naive incarnation), CP ABE encrypts them according to her chosen object read/write access polices, so that exactly users possessing matching secret CP-ABE keys can decrypt the payload keys. The encrypted payload keys are part of the Key Delivery data. Each object is associated with a *capsule* containing a description of its access polices, the corresponding key delivery data, and additional AKE data, such as a signature by the owner, and her certificate[4]. Eligible users can decrypt the PL keys. Capsules are defined precisely below.

[4]The owner signs every cryptogram that she creates on the nodes and on the edges of the poset, and links the signature to the corresponding node, edge, resp. It is sufficient however to link the owner's certificate to the poset just once.

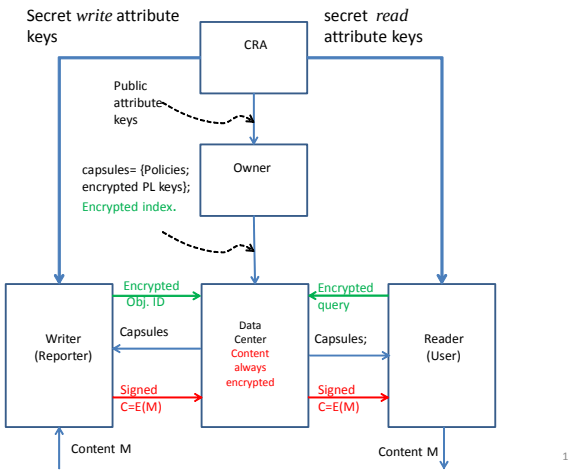

Figure 1: A general block diagram of the ACE system.

2.2 Formal definitions of HKD, CP-ABE, and Access

As we already mentioned, ACE can run over any poset, however, for the sake of concreteness we describe it over the poset $(X, \leq)$. As before, let $\mathcal{A}$ denote the set of attributes. Let $X = 2^{\mathcal{A}}$. As before, we use $(X, \leq)$ to denote a poset were partial orders are dictated by the natural subset of attributes inclusion relations. When running CP-ABE over $(X, \leq)$ objects are associated with subsets of nodes of X, and users with single nodes of X. In the body of the paper we focus on $(X, \leq)$, while in the CAT example (Section 5) we use another poset. In the appendix we formally define w.l.g. access policies as Monotone Access polices (MAS).

The sets U and O represent users and object, resp. For CP-ABE, let $\lambda : U \rightarrow X$, and $\lambda' : O \rightarrow 2^X$. The former function assigns a subset of attributes to each user, and the latter function assigns a policy to each object.

Let $\mathcal{E} = (F, D)$ be a secure symmetric-key encryption function defined over spaces $(\mathcal{K}, \mathcal{M}, \mathcal{C})$; the key, message and cryptogram spaces, resp. Let $F(\kappa, M)$ denote the symmetric encryption of message M with key κ. We next define Hierarchical Key Derivation (HKD) for any poset represented by a directed loopless graph $G = (V, E)$, where V is the set of nodes and E is the set of edges. In particular it applies to the posets $(X, \leq)$ and to $(L, \leq)$ which is defined in section 5. The HKD defined below was used for example in [7].

Definition 1 *(`Hierarchical Key Derivation (HKD)`): Let $G = (V, E)$, be any directed loopless graph, where V is the set of nodes and E is the set of edges. Assign a secret random symmetric key $h(x)$ to each $x \in V$, and for each directed edge $(x, y) \in E$ include $F(h(x), h(y))$ in the public information.*

As we show in subsection 2.3, in ACE a user u can derive the key for any $y \in V$ if there exists a directed path from some $x \in \lambda(u)$ to y in V. If node y is included in the access policy of object o then the ability to derive the key for y implies access to o. We accelerate users' computations

using symmetric key HKD. We proceed to define CP-ABE. The following generic description is taken from [6].

A generic ABE system

Definition 2 *An (Ciphertext-Policy)* `Attribute Based Encryption` *scheme consists of four algorithms.*
Setup. *This is a randomized algorithm that takes no input other than the implicit security parameter. It outputs the public parameters* PK *and a master secret key* MK.
Encryption. *This is a randomized algorithm that takes as input a message M, an access policy $\mathbb{A}$, and the public parameters* PK. *It outputs the ciphertext E. Notation: $E \leftarrow Encrypt(\mathrm{PK}, \mathbb{A}, M)$,*
KeyGen *. This is a randomized algorithm that takes as input a set of attributes γ, the secret master key* MK *and the public parameters* PK. *It outputs a decryption key D. $D \leftarrow KeyGen(\mathrm{PK}, \mathrm{MK}, \gamma)$*
Decryption. *This algorithm takes as input the ciphertext E that was encrypted according to access policy $\mathbb{A}$, the decryption key D for the set γ of attributes and the public parameters* PK. *It outputs the message M with overwhelming probability iff $\gamma \in \mathbb{A}$. Notation: $M \leftarrow Decrypt(\mathrm{PK}, D, E)$.*

We now define the meaning of a user's permission to access an object. Access to *read* and to *write* may be distinct, but are handled the same way, so here we do not specify the type of access.

Definition 3 *(Access in CP-ABE) Let $\omega \in 2^{Att}$ and $\mathbb{A} \subseteq 2^{Att}$ be a set of attributes associated with user u, and object o's access policy, respectively.* `User` *u* `can access object` *o iff $\omega \in \mathbb{A}$.*[5]

We can use any ABE as a component in ACE, and therefore we do not describe any particular ABE in this paper. ABE was proven secure under SSM (see definition in sec. 3.1) in [9] for one version, and elsewhere for other versions (e.g. `[6, 15]`).

2.2.1 The Total Number of ABE Operations

Let Att denote the set of attributes, and let $X = 2^{Att}$. A Monotone Access Structure (MAS) is a collection $S \subseteq 2^{Att}$, s.t. for $A, B \subseteq 2^{Att}$, if $A \in S$, and $A \subseteq B$ then $B \in S$. This implies that for MAS S, if a chain $C \in S$ then it is fully represented by its minimal element. Therefore every MAS S is fully represented by an antichain over Att. Let $\mathcal{A}^{Att}$ denote the set of antichains over Att. Let the cardinality of Att be $|Att| = m$. Clearly the number of possible MAS over Att is upper bounded by $2^{(2^m)}$, since $|X| = 2^m$, and the number of binary vectors of length 2^m is $2^{(2^m)}$. This upper bound is not tight, since we have to count only antichains over Att (see definition 26). Let $M(m) = |\mathcal{A}^{Att}|$. $M(m)$ is the Dedekind number of m. It is well known that $\log_2 M(m) \approx \binom{m}{\lfloor m/2 \rfloor}$. From Stirling's approximation $\binom{m}{\lfloor m/2 \rfloor} \approx \sqrt{\frac{2}{\pi m}} \cdot 2^m$. Therefore $M(m) \approx 2^{(\sqrt{\frac{2}{\pi m}} \cdot 2^m)}$. Assuming that system [7] uses ABE to deliver top keys to users, the upper bound on the number of such operations is $M(m)$, while for ACE it is 2^m.

[5] This means that in CP-ABE, ω must dominate in $(X, \leq)$ at least one element of $\mathbb{A}$ in order to allow user u access to object o.

2.2.2 The meaning of running ABE together with HKD

Description Each node of $(X, \leq)$ may correspond to the most privileged node for some user, therefore we run ABE and HKD together. Each node γ of $(X, \leq)$ is associated with a secret HKD key $h(\gamma)$, which is encrypted once using ABE and once using HKD, to be defined precisely in section 3.3.2. We denote this system as (ABE || HKD). Our system, ACE, is an Authenticated Key Exchange so that writers and readers get assurance that the owner, who sets access policies, is who they expect her to be. In a store-and-forward system it is natural to use a 1-pass AKE (such as [16], pp. 510). (ABE || HKD) is the encryption component of this AKE. In section 3 we prove that (ABE || HKD) is semantically secure; a necessary condition for AKE security. Below we give a description of ACE. This version is suitable for small objects of high value. For a large system we recommend replacing public key encryption at the payload level with symmetric key encryption, and adding writer's signature on the payload.

1. An object class[6] may have distinct read and write access policies. Therefore we use asymmetric key cryptosystems to encrypt them. Payload encryptions and decryptions are done by writers and readers resp. The system must have the additional property that given any of the keys it must be infeasible to find its counterpart matching key[7].

2. The owner associates each node in $(X, \leq)$ with a unique HKD symmetric-key drawn independently from a uniform distribution and a unique ABE secret-key.

3. In each node the HKD key of the node is accessible via both HKD and ABE. Namely, the owner encrypts it using both HKD (with the HKD key of a parent node, using symmetric-key encryption[8]), and using ABE with the ABE key of that node (see section 3.3.2 for details). In addition, the owner signs these cryptograms as required in a 1-pass AKE ([16], pp. 510).

4. Let $\psi_R(o)$ and $\psi_W(o)$ denote the read and write access structures of object class o resp. The owner encrypts the read (write) payload key of object-class o using a symmetric key cryptosystem in each of the nodes corresponding to $\psi_R(o)$ $(\psi_W(o))$ using the HKD key of that node.

5. A plurality of access structures may intersect on a node, each having its own payload key encrypted under the same HKD key of that node.

2.3 System Details

2.3.1 The two savings

For the sake of concreteness we use here the poset (X,≤). The ACE system works for any poset. Each node $\gamma \in (X, \leq)$, which is associated with some user or belongs in an object's access policy is associated with a symmetric key, denoted $h(\gamma)$. We encrypt $h(\gamma)$ a few times. Once using CP-ABE and then using the HKD key of each father of γ, and we keep these cryptograms in the

[6] Recall that all the objects that have the same read and the same write access policies comprise an object class.

[7] A simple modification of RSA meets this additional condition; have both exponents long and secret, and do not divulge the factorization of the modulus to the users. The writer gets one of the exponents and a reader gets its matching counterpart.

[8] We associate that cryptogram to the proper edge.

public domain (to be defined precisely below). The ABE encryption consists of one cryptogram per node, while the number of HKD encryptions is equal to the number of ancestor nodes of node γ. Therefore, in each node other than the root the HKD key of the node is accessible via HKD and via ABE. The HKD key of the root is accessible only via ABE. We denote this system as (ABE || HKD). Our system, ACE, is part of an Authenticated Key Exchange so that writers and readers get assurance that the owner, who sets access policies, is who they expect her to be. In a store-and-forward system it is natural to use a 1-pass AKE (such as [16], pp. 510). (ABE || HKD) is the encryption component of this AKE. ACE reduces computational complexity in two ways:

1st saving: In ACE, using CP-ABE, each user does a single CP-ABE decryption (in a key update cycle). After accessing her maximally privileged HKD key using CP-ABE decryption, the user proceeds to find less privileged HKD keys using the HKD key derivation algorithm (Def. 30).

2nd saving: *We take advantage of overlaps on nodes among access polices. The overlaps mean that access polices can share the same ABE and HKD encryptions of the shared node.*[9]

Each overlap replaces a CP-ABE encryption (i.e. pairing operation) with a few symmetric key encryptions which are about a million times faster for realistic parameter sizes. Below we give a detailed description of ACE.

2.3.2 Detailed player's routines

In poset (X,$\leq$) we assign each node of the poset a unique integer, $i = 1, 2, ...$ Since each such node is also associated with a unique subset of attributes $\omega(i)$ we treat i and $\omega(i)$ synonymously. Let $\mathbb{A}(o)$ denote the access policy of object o, corresponding to nodes $\lambda'(o) = \{j_1, j_2, ...j_k\}$ of the poset (X,$\leq$), namely $\mathbb{A}(o) = \{\omega(j_1), ...\omega(j_k)\}$. The owner encrypts the payload key of object o k times, once for each of the nodes in $\lambda'(o)$ using the HKD key of each node. Below we describe the process in detail for a single node j.

Let f_2, f_3, f_4 be three symmetric encryption functions, where f_2 is the basic HKD encryption function, f_3 is used to encrypt a payload key p, and f_4 is used to encrypt the payload using key p. Their corresponding decryption functions are f_i^*, $i = 2, ..4$. We use the syntax $cryptogram \leftarrow f_i(key, message)$; $message \leftarrow f_i^*(key, cryptogram)$. As before, let $h(j)$ denote the HKD key of node j. We may use the same symmetric key encryption to implement all three functions f_i . However we find that these distinct notations help in understanding the system.

Definition 4 *Suppose that object o is associated with access policy* $\mathbb{A}(o) = \{\omega(1), \omega(2), ...\omega(k)\}$. *The* capsule *for object o, is a collection of k individual capsules, one for each subset of attributes* $\omega(i)$, $i = 1, 2, ...k$. *For each* $i = 1, 2, ...k$ *the individual capsule of* $\omega(i)$ *is capsule*$(\omega(i)) = \{c_1, \{c_2\}, c_3, c_4\}$, *where* c_1 *is the ABE encryption of the HKD key* $h(\omega(i))$ *associated with node* $\omega(i)$ *using the attributes of node* $\omega(i)$*:* $c_1 \leftarrow Encrypt(\mathrm{PK}, \omega(i), h(\omega(i)))$; $\{c_2\}$ *is a subset of HKD cryptograms: for each node x such that x is a direct ancestor of* $\omega(i)$ *in poset* $(X, \leq)$, $c_2(x, \omega(i)) \leftarrow f_2(h(x), h(\omega(i)))$. *Cryptogram* c_3 *is the payload key p of object o encrypted under key* $h(\omega(i))$, *namely,* $c_3 \leftarrow f_3(h(\omega(i)), p)$. *The same payload key is encrypted in each of the k individual capsules of nodes* $\omega(1), \omega(2), ...\omega(k)$. *A node may be associated with more than one*

[9] Usually, unlike in ACE, CP-ABE systems do an ABE encryption operation for each policy, without utilizing the overlap among policies.

object. Finally, c_4 *is the object* o *itself encrypted under* p : $c_4 \leftarrow f_4(p, o)$. *The* Key Delivery algorithms *are* f_1 *=ABE,* f_2 *=HKD, and* f_3*, while the* Key Delivery data *is a triple of cryptograms* (c_1, c_2, c_3) *that these encryption functions output.*

Signatures

To the simplified diagram of Figure 1 we have to add the following signatures:

1. CRA certifies the linkages between attributes and their associated public attribute keys, as usual. Owners and users verify the certificates.
2. Owners sign (certify) capsules, and users verify them[10].
3. Writers sign encrypted payloads, and readers verify these signatures. Within capsules, owners deliver to writers the signature key and to readers the corresponding verification keys. Users trust owners. All the legitimate writers of an object use the same signature key.

In the rest of this paper we ignore the above signatures, and focus on the encryptions.
On each key update cycle the players do:

Certification & Revocation Authority (CRA):

1. Publish fresh public attribute keys;
2. Give each user the secret attribute keys that she deserves.
3. Publishes certificates linking attributes and unique public keys and gives eligible users the corresponding secret keys.

Each Owner:

1. Chooses the set of attributes, *Att*. Let $X = 2^{Att}$. Constructs the poset $(X, \leq)$. Prune unused nodes. Let $\omega(j)$ denote the subset of attributes associated with node $j \in X$.
2. Run the CP-ABE Setup procedure to produce the public parameters PK and a master secret key MK.
3. To each node $j \in X$ *(a)* compute its ABE key $D(j) \leftarrow KeyGen(\mathrm{PK}, \mathrm{MK}, \omega(j))$, *(b)* Associate a random HKD key $h(j)$, *(c)* Compute $c_1(j) \leftarrow Encrypt(\mathrm{PK}, \omega(j), h(j))$; *(d)* For every i a direct ancestor of j in poset $(X, \leq)$. Compute $c_2(i, j) \leftarrow f_2(h(i), h(j))$.
4. For each object o, pick a payload key $p(o)$.
5. In general, each object is associated with two access policies, one corresponds to its *write* access, and the other to its *read* access. An access policy may include a plurality of nodes. The owner does:

[10] This is the AKE signature. In [16], pp. 510 the signed message looks simpler. It is just a single encrypted key. Here it is a set of encrypted keying material (the KD data).

(a) For each node j_1 in the *write* access policy compute:

$$c_{3,w}(j_1) \leftarrow f_3(h(j_1), p); \tag{1}$$

(b) For each node j_2 in the *read* access policy compute:

$$c_{3,r}(j_2) \leftarrow f_3(h(j_2), p); \tag{2}$$

Each eligible writer for node j :

Below we describe the process for eligible writers. A writer is *eligible for node* $j \in X$ if she is associate with a set of attributes corresponding to a node equal or higher in the hierarchy in $(X, \leq)$ than node j. When a writer wants to write into an object, she sends the data center an object-ID, and gets in return the object capsule.

1. Acquire HKD key $h(j)$:

   If the writer already has $h(j)$ then goto 2.

   Else, if the writer has $h(i)$ for some direct ancestor i of node j, then compute $h(j) \leftarrow f_2^*(h(i), c_2(i,j))$; goto 2. If node i is an indirect ancestor of node j then repeat the process iteratively along the path $(i \rightarrow j)$;goto 2.

   Else, compute ; $h(j) \leftarrow Decrypt(\mathrm{PK}, D(j), c_1(j))$; this succeeds when $j \leq i$ in the poset, where i is the writer's node.

2. Decrypt payload key $p \leftarrow f_3^*(h(j), c_{3,w}(j))$;

3. Encrypt object o using $c_4 \leftarrow f_4(p, o)$; (c_4 replaces object o in the DB and is linked to node j).

Each eligible reader for node j :

To simplify we exclude here the issues of queries. In the example in section 5 we include queries. When a reader wants to read an object, he sends the data center an object-ID, and gets in return the object capsule. A reader (like a writer) is *eligible for node* $j \in X$ if she is associate with a set of attributes corresponding to a node equal or higher in the hierarchy in $(X, \leq)$ than node j.

1. Acquire key $h(j)$ (same as for writer):

2. Decrypt payload key $p \leftarrow f_3^*(h(j), c_{3,r}(j))$;

3. Decrypt object o using $o \leftarrow f_4^*(p, c_4)$;

2.3.3 Computational Complexity

Recall that a MAS (sec. 24) is represented over $(X, \leq)$ as a subset of nodes $\psi(o) \in 2^X$, comprising an antichain, and that from Lemma 31 user u can access object o iff $\psi(u)$ dominates at least one element of $\psi(o)$ in poset $(X, \leq)$. Recall also that in ACE we run ABE in parallel to HKD over the same poset (X,≤). Saving for owners comes from taking advantage of overlaps on $(X, \leq)$ nodes among access structures. The overlaps mean that access structures can share the same ABE and HKD encryptions of that node. In ACE each user does just one ABE decryption (in a key update cycle). Then after accessing her top HKD key using ABE decryption, the user proceeds to find lower HKD keys using the HKD key derivation algorithm (see section 30).

We compare ACE to C' (Crampton's system to which we add ABE and they run together over the same poset [7]). For $m = |Att| = 10$ a single ABE encryption costs about $k \approx 10^6$ times as much as symmetric key encryption of comparable strength. For users, both ACE and C' save a factor $\tilde{}k$ over pure CP-ABE.

For owners, C' is about the same as pure ABE (since any poset node can be the top key of some user, any node needs ABE encryption). ACE is more efficient than C' since we work with the smaller poset $(X, \leq)$ while C' works with the much larger poset $(Auth, \sqsubseteq)$. Let R denote the number of roles and N the number of MAS. The tight upper bounds are $R \leq 2^m$, and $N \leq 2^{\sqrt{\frac{2}{\pi m}} \cdot 2^m}$ (see section 2.2.1). In C' the total number of ABE operations and symmetric key operations (for owners during setup) is upper bounded by N, whereas in ACE owners do up to R ABE operations and up to N symmetric key operations. We now look at the gain factor for particular cases rather than on asymptotics. For a particular case, let $N = cR$. Then the saving factor, η, for owners of ACE over C' is:

$$\eta \approx \frac{kN}{kR + N} = \frac{kc}{(k + c)} \tag{3}$$

Note that cases where $c < 1$ are possible, however, the upper bounds suggest that usually $c > 1$, in which case $\eta > 1$. In fact we believe that usually $c >> 1$, and $\eta >> 1$.

The upper bounds on R and on N are specific to poset $(X, \leq)$, whereas the non asymptotic expression for η is true for any poset, including $(L, \leq)$ used for CAT (see 5.3).

3 Security of ACE

We define and prove the semantic security of running HKD together with ABE. This is a necessary condition for the security of ACE. The system should be used as the encryption component of an Encryption-Based 1-pass Authenticated Key Exchange (AKE) protocol (e.g. [16], PP. 510).

3.1 Definition of Security for ABE

We can use any ABE as a component in ACE, and therefore we do not describe any particular ABE in this paper. For a definition of generic ABE see section 2.2. ABE was proven secure under the security game Selective Set Model (SSM) defined below in [9] for one version, and elsewhere for other ABE versions (e.g. `[6, 15]`).

A crucial observation, made first in [5] and later in [7] is that it is very natural to describe ABE over a poset of roles, where each role is associated with a subset of attributes. All of ABE, HKD,

or ACE, are considered over the same poset $(X, \leq)$. The following definition is equivalent to the Selective Set Model of [6], although the poset $(X, \leq)$ is not mentioned there (the posets reflect the natural partial order of roles; A user having the secret keys for role ρ associated with a subset of attributes $s \subseteq Att$ has the keys for role $\rho' \leq \rho$, since ρ' is associated with attributes $s' \subseteq s$.

Attack Game (Selective Set Model (SSM)). For a given ABE system, and for a given adversary $\mathcal{A}$, we define:

Init. The adversary declares the node,$\gamma \in X$, that he wishes to be challenged upon.

Setup. The challenger runs the Setup algorithm of ABE and gives the public parameters to the adversary. The challenger does not expose secret keys.

Phase 1. The adversary is allowed to issue queries for secret keys for polynomially many nodes $v_j \in (X, \leq)$, where for all j, v_j does not dominate γ in $(X, \leq)$ (this is equivalent to saying that role v_j is not included in monotone access structure γ, in the standard ABE terminology).

Challenge. The adversary submits two equal length messages p_0 and p_1. The challenger flips a random coin b, and encrypts p_b using the encryption of node γ. The ciphertext is passed to the adversary.

Phase 2. Same as Phase 1.

Guess. The adversary outputs a guess b' of b.

The *advantage* of adversary $\mathcal{A}$ in attacking ABE system $\mathcal{E}$ in this game is
$Adv_{SSM}[\mathcal{A}, \mathcal{E}] = |Pr[b' = b] - \frac{1}{2}|$.

Definition 5 *ABE system $\mathcal{E}$ is* ***secure*** *under the Selective Set Model if for all polynomial time adversaries $\mathcal{A}$ $Adv_{SSM}[\mathcal{A}, \mathcal{E}]$ is negligible.*

3.2 HKD

3.2.1 A Topological Definition of Security for HKD

We motivate a new definition of semantic security of HKD and prove that the HKD system of [7] (section 30) is secure under this definition. The current definitions of security of HKD (Key recovery and Key Indistinguishability [1, 2]) are not close enough to the SSM definition to allow the combination of HKD and ABE and its security analysis . Therefore we propose a new definition of HKD security which is similar enough to SSM to allow the definition of semantic security game for (HKD || ABE), and prove its security under this game. The most notable new element of this game is that in the challenge phase the challenger does not return a single cryptogram as usual. Rather she returns a neighborhood of distance 1 (notation 6 below) around the attacked node of poset $(X, \leq)$, complete with its topology and the cryptograms linked to its edges.

Notation 6 *Let $N(\gamma, x)$ denote a neighborhood of distance 1 around node γ of $(X, \leq)$. It consists of the edges connected to node γ (ingoing and outgoing) and their labels when the secret HKD key of node γ has value x.*

Let $S(\gamma)$ denote the set of directed edges in the neighborhood of distance 1 from node γ, and (i, j) denote a directed edge going from node i to node j. Then $S(\gamma) = \{(i, j)\}_{\gamma \in \{i,j\}}$. $S(\gamma)$ is just the topology of the neighborhood. The neighborhood includes the topology and the cryptograms associated with the edges of this neighborhood (we use "edge-labels" and "edge-cryptograms" synonymously), namely, let the HKD key of node i be $h(i)$. Then $N(\gamma, h(\gamma)) =$

$\{(i,j), E(h(i),h(j))\}_{(i,j)\in S(\gamma)}$. There is a somewhat unorthodox flavor to the new definition below. When the attacker issues a challenge (m_0, m_1) against node γ the challenger's response is $N(\gamma, m_b)$ for a random bit b, rather than a single cryptogram as usual.

Attack Game Topological Selective Set Model (TSSM) for HKD

Given poset $(X, \leq)$:

Init. The adversary declares the node,$\gamma \in X$, that he wishes to be challenged upon,

Setup. The challenger runs the Setup algorithm of system HKD *except for* $N(\gamma, .)$, and gives the public parameters (edge labels) to the adversary. The challenger does not expose secret keys.

Phase 1. The adversary is allowed to issue queries for secret keys for polynomially many nodes $v_j \in (X, \leq)$, where for all j, v_j does not dominate γ in $(X, \leq)$ (this is equivalent to saying that role v_j is not included in monotone access structure γ, in the standard ABE terminology).

Challenge: Attacker selects equal length messages (m_0, m_1) as candidates for $h(\gamma)$ and passes (m_0, m_1) to the challenger. The challenger flips a coin b, assigns $h(\gamma) \leftarrow m_b$, computes $N(\gamma, m_b)$ and passes it to the attacker.

Phase 2. Repeat Phase-1.

Guess. The adversary outputs a guess b' of b.

The *advantage* of adversary $\mathcal{A}$ against HKD system S in this game is
$Adv_{TSSM}(\mathcal{A}, S) = |Pr[b' = b] - \frac{1}{2}|$.

Definition 7 *HKD system $\mathcal{S}$ is **secure** under TSSM if for all polynomial time adversaries $\mathcal{A}$ $Adv_{TSSM}(\mathcal{A}, S)$ is negligible.*

3.2.2 Extended Multi-key CPA security

We are not aware of publicly available definition of *Multi-User CPA security* for symmetric key cryptosystems, so we adapt and modify definition 3 of [4], which is written for asymmetric cryptosystems. We use the name *Multi-key CPA security (MCPA).* The differences are that, of course, instead of generating matching asymmetric key pairs, symmetric keys are chosen independently from a uniform distribution on the key space $\mathcal{K}$. Each such key has a public *selector*, s, where the key itself is denoted $h(s)$. We denote the process: $h(s) \in_R \mathcal{K}$. We extend the MCPA game to allow additional type of queries, and use EMCPA to denote *Extended MCPA security.* In MCPA the attacker uses queries of the form $(s_i, m_{i,o}, m_{i,1})$ $i = 1, 2, ...$, where for each i the messages have equal length, to which the challenger ("oracle") responds with $E(h(s_i), m_{i,b})$, where the left input is the key and the right input is the message and the same bit $b \in \{0, 1\}$ is used for all the responses. We call these queries *type (i) queries.* In EMCPA attackers can issue type (i) and *type (ii) queries*, where a type (ii) query is of the form (s_i, s_j), to which the challenger's response is $E(h(s_i), h(s_j))$. Type (ii) queries represent the additional data that attackers of HKD have, namely they can read the edge cryptograms on poset $(X, \leq)$ (see [7], and the end of section 30). Clearly for HKD to be secure it is necessary that the underlying cryptosystem upon which HKD is built, $\mathcal{E} = (E, D)$, be EMCPA secure. We prove the other way; that if $\mathcal{E}$ is EMCPA secure then HKD is TSSM secure. Formally:

Attack Game (Extended Multi-key CPA security, EMCPA). For a given symmetric key cipher $\mathcal{E} = (E, D)$, defined over spaces $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ (key, message, cryptogram, resp.), and for a

given adversary $\mathcal{A}$, we define two experiments, Experiment 0 and Experiment 1. For $b = 0, 1$, we define

Experiment b:

• Initially, the challenger sets $r \leftarrow 0$.

• The adversary submits a sequence of queries to the challenger. For $i = 1, 2, ...,$the i$_{th}$ query is either (i) a pair of messages, $(m_{i,0}, m_{i,1}) \in \mathcal{M}^2$, of the same length, along with a key selector s_i, where $1 \leq s_i \leq r + 1$, or (ii) a pair of selectors $1 \leq s_i, s_j \leq r + 1$. W.l.g we assume that $s_i \geq s_j$.

The challenger then does the following:

if $s_i = r + 1$ then

$r \leftarrow r + 1$

$k_r \in_R \mathcal{K}$

For type (i) query the challenger's response is $E(h(s_i), m_{ib})$. *The same value of bit b is used for all the type (i) queries.* for type (ii) query the response is $E(h(s_i), h(s_j))$. The challenger sends the response to the adversary.

• The adversary outputs a bit $\widetilde{b} \in \{0, 1\}$.

⊡

For $b = 0, 1$, let W_b be the event that $\mathcal{A}$ outputs $\widetilde{b} = 1$ in Experiment b. We define $\mathcal{A}$'s advantage with respect to $\mathcal{E}$ as $Adv_{EMCPA}[A, \mathcal{E}] := |Pr[W_0] - Pr[W_1]|$.

Definition 8 *(Extended Multi-key CPA security). A cipher $\mathcal{E}$ is called Extended Multi-key CPA secure (or EMCPA secure in short) if for all efficient adversaries $\mathcal{A}$, the value $Adv_{EMCPA}[A, \mathcal{E}]$ is negligible.*

Remark 9 *An instance of EMCPA where there is not even one type (i) query with $m_{i0} \neq m_{i1}$ is trivially easy to solve (any value of $\widetilde{b}$ is "correct").*

Consider HKD built using $\mathcal{E} = (E, D)$ in the sense of [7] (Definition 30).

Theorem 10 *If $\mathcal{E} = (E, D)$ is EMCPA secure then HKD built using $\mathcal{E}$ over any poset $(X, \leq)$ is TSSM secure.*

Proof. See Appendix B (7). ■

3.3 The security of parallel Connections

We start with a simple case of semantically secure block ciphers having the same message space and prove that the parallel connection of semantically secure block ciphers is semantically secure. Then we define precisely the combination of HKD and ABE as key allocation system and generalize the proof to cover it.

3.3.1 The security of parallel connection of simple Block Ciphers

Let $\mathcal{E}_1 = (E_1, D_1)$, $\mathcal{E}_2 = (E_2, D_2)$ be symmetric block ciphers over $\mathcal{K}, \mathcal{M}, \mathcal{C}$, the key, message and cryptogram spaces, resp. (Appendix A Sec. 6). The two ciphers have the same message space, but not necessarily the same key and the same cryptogram spaces. Let $\mathcal{K}_i, \mathcal{C}_i$ denote the corresponding key and cryptogram spaces of cipher $\mathcal{E}_i$, $i = 1, 2$.

Definition 11 *We say that $\mathcal{E}_1$ and $\mathcal{E}_2$ having the same message space are connected in parallel (denoted $(\mathcal{E}_1||\mathcal{E}_2)$) if they encrypt the same messages, namely, for all $m \in \mathcal{M}$, $k_i \in \mathcal{K}_i$, their output is a pair $(c_1, c_2) \in \mathcal{C}_1 \times \mathcal{C}_2$, where $c_i \leftarrow E_i(k_i, m)$.*

In this section, for the sake of simplicity, we use the most basic definition of semantic security of symmetric-key block ciphers (Appendix A, Sec. 6).

Lemma 12 *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be semantically secure block ciphers having the same message space. Then $(\mathcal{E}_1||\mathcal{E}_2)$ is semantically secure.*

Proof. See Appendix C (Sec. 8). ■

3.3.2 The security of combining HKD and ABE

In section 3.2.1 we proposed a new semantic security model for HKD (TSSM), which is similar to the standard SSM model of ABE. Then in Theorem 10 we showed that HKD of [7] is secure under TSSM if the underlying block cipher $\mathcal{E} = (E, D)$ is EMCPA secure. We now define another model, USM, which generalizes both SSM and TSSM, and show that our combination of ABE and HKD is secure under USM.

We first define Key Allocation system uniformly to capture HKD, ABE and then give a precise definition of the meaning of running HKD together with ABE over the same poset $(X, \leq)$. We take the liberty of using the notation $(ABE \ || \ HKD)$ for this combination of larger systems because of its similarity to parallel combination of basic cryptosystems. We then formally define and prove the security of $(ABE \ || \ HKD)$ under a game that generalizes the standard Selective Set Model (SSM) of ABE. In the descriptions below each step specializes to ABE, HKD, and $(ABE \ || \ HKD)$. We put them together to highlight their structural similarities.

A Uniform Key Allocation (UKA) Let the basic ABE algorithms be $Setup, Encrypt, Decrypt$. as defined in Sec. 2.2. HKD is built around encryption function $\mathcal{E} = (E, D)$ defined over spaces $(\mathcal{K}, \mathcal{M}, \mathcal{C})$ (definition 30). For a given poset of roles $(X, \leq)$ (Sec. 10.1) we describe the treatment of any node j with direct ancestor i. Let PP and MK denote the Public Parameters and secret Master Key of the ABE system. We equate a node name with the subsets of attributes corresponding to that node and with the access policy of that node. For $\rho, j \in X$ a user with role $\rho \geq j$ has secret ABE key $K_{ABE}(\rho)$ that enables the decryption of ABE cryptograms in node j. In (ABE || HKD) $h(j)$ of node j (the HKD key of node j) is encrypted independently of other nodes belonging to the same access policy. Therefore in the description below, we treat each node independently[11]. Recall that (i, j) denotes a directed edge going from node i to node j in $(X, \leq)$.

Set: In (i) below Certification & Revocation Authority (CRA) sets up the ABE system, while the owner sets up the HKD system.

(i) Setup the underlying cryptosystem for security parameter λ :

(a) ABE: Generate PP and MK.

(b) HKD: $\forall\, j \in X$, pick a random secret key $h(j)$,

(c) $(ABE \ || \ HKD)$: Do (a) and (b) as above.

[11] Recall that in all the nodes belonging to the same MAS (represented as an antichain) the same PL key is encrypted. In node j it is symmetrically encrypted under $h(j)$ (Section 2.2.2).

(ii) Owner does $\forall\ j \in X$, Encrypt $h(j)$:

(a) ABE: $c_1(j) \leftarrow Encrypt(PP, j, h(j))$.

(b) HKD: $\forall$ directed edge (i,j) do: $c_2(i,j) \leftarrow E(h(i), h(j))$.

(c) $(ABE \ || \ HKD)$: $\forall$ edge (i,j) compute $(c_1(j), c_2(i,j))$ as in (a) and (b).

Derive:

Readers and writers having role $\rho \geq j$ do the steps below.

(a) ABE: Given $c_1(j)$ and $K_{ABE}(\rho)$ do: $h(j) \leftarrow Decrypt(PP, \rho, K_{ABE}(\rho), c_1(j))$.

(b) HKD: For directed edge (i,j) given $h(i)$ and $c_2(i,j)$, do: $h(j) \leftarrow D(h(i), c_2(i,j))$;

(c) $(ABE \ || \ HKD)$: Do (a) or (b) above.[12]

Remark 13 *In addition to $c_1(j)$ and $c_2(i,j)$ node j is associated with plurality of cryptograms $c_3(j)$ which are the symmetric encryptions of payload keys, whose access structure includes node j, under key $h(j)$. Another set of cryptograms is $c_4(.,.)$ which are the encryption of payload (content) of an object under the payload key whose encryption is $c_3(j)$. For lack of space we do not discuss these additional cryptograms any further in this paper.*

A Uniform Security Game (USG)

Init: Attacker picks challenged node γ.

Setup: Challenger sets up the system except for node γ: $h(\gamma)$ remain unspecified. Also:

(a) ABE: $c_1(\gamma)$ remain unspecified;

(b) HKD: $c_2(\gamma) = N(\gamma, .)$ remain unspecified[13].

(c) $(ABE \ || \ HKD)$: $c_1(\gamma)$ and $c_2(\gamma)$ remain unspecified;

Phase-1: The attacker asks the challenger to expose the secret keys of nodes in a set $V \subset X$, none of which dominates γ in $(X, \leq)$ and the challenger exposes the secret keys of V.

(a) ABE: Expose $K_{ABE}(i)$, $\forall i \in V$.

(b) HKD: Expose $h(i)$, $\forall i \in V$.

(c) $(ABE \ || \ HKD)$: Do (a) and (b) above.

Challenge: The attacker chooses a pair of equal length (m_0, m_1) and sends them to the challenger. The challenger picks a random bit b, and sets $h(\gamma) \leftarrow m_b$, he then computes:

(a) ABE: $c_{1,b}(\gamma) \leftarrow Encrypt(PP, \gamma, m_b)$,

(b) HKD: $c_{2,b}(\gamma) \leftarrow N(\gamma, m_b)$, // $N(\gamma, x)$ is defined in Notation (6).

(c) $(ABE \ || \ HKD)$: Compute $(c_{1,b}(\gamma), c_{2,b}(\gamma))$, as above.

The challenger sends the results to the attacker.

Phase-2: Same as Phase-1.

Guess: The attacker guesses bit $\widetilde{b}$. The attacker wins if $\widetilde{b} = b$.

□

Let $\mathcal{F}$ be any of ABE, HKD, or $(ABE \ || \ HKD)$. The *advantage* of attacker $\mathcal{A}$ in attacking $\mathcal{F}$ is $Adv_{USG}(\mathcal{A}, \mathcal{F}) = |\Pr[\widetilde{b} = b] - \frac{1}{2}|$. Obviously, $Adv_{USG}(\mathcal{A}, \mathcal{F}) = Adv_{SSM}(\mathcal{A}, \mathcal{F})$ when $\mathcal{F} = ABE$, and $Adv_{USG}(\mathcal{A}, \mathcal{F}) = Adv_{TSSM}(\mathcal{A}, \mathcal{F})$ when $\mathcal{F} = HKD$.

Definition 14 *System $\mathcal{F}$ is secure under USG if for all polynomial time adversary $\mathcal{A}$, $Adv_{USG}(\mathcal{A}, \mathcal{F})$ is negligible.*

[12] Each `user uses` $c_1(.)$ for her most privileged node, and continues down the poset using the much faster $c_2(.,.)$.

[13] Do not confuse the notations $c_2(i,j)$ and $c_2(j)$. The former is a single cryptogram on edge (i,j); the latter is the neighborhood of distance 1 of node j.

The Main Result: We assume an HKD built using block cipher $\mathcal{E} = (E, D)$ as defined in [7] (Def. 30), and ABE stands for any SSM secure ABE system, such as the one in [9].

Theorem 15 *If HKD is TSSM secure and ABE is SSM secure then* $(ABE \ || \ HKD)$ *is secure under the Universal Security Game (USG).*

Proof. ABE is secure under SSM ([9]) hence also under USG (Sec. 3.3.2). HKD is secure under TSSM (Sec. 10) hence also under USG. In each node γ of $(X, \leq)$ ABE and HKD encrypt the same message $h(\gamma)$, i.e. they run in parallel.

We mimic the proof of Lemma 12 (Appendix C, Sec. 8) where USG security replaces the simple definition of Appendix A (Sec. 6). This has the following consequences: (i) Within each game, setup ABE and HKD rather than F and G. (ii) In each of the three games in the proof add Phases 1,2 of (ABE || HKD) before and after the challenge-response, resp. (iii) Replace $c_{f,b}$ with $c_{1,b}(\gamma) \leftarrow Enc(PP, \gamma, m_b)$ and replace $c_{g,b}$ with $c_{2,b}(\gamma) \leftarrow N(\gamma, m_b)$).

The Hybrid Argument proof of Lemma 12 is oblivious to the distinct semantics of the cryptograms, so change (iii) does not affect the proof. As for change (ii): In equation 10 the probabilities $q_i, \quad i = 0, ..3$ are taken under the side information of Phases 1,2 of either ABE or HKD, while the probabilities $p_j, \quad j = 0, ..2$ are taken under the side information of Phases 1,2 of (ABE || HKD). All the HKD keys are chosen independently, hence the ABE and HKD side information are independent of each other. Hence the probabilities $q_i, \quad i = 0, ..3$ remain the same under the side information of Phases 1,2 of (ABE || HKD), and the probability analysis remains valid. ■

We remind the reader that if $\mathcal{E} = (E, D)$ is EMCPA secure then HKD built from $\mathcal{E}$ is TSSM secure (Theorem 10). From this and from Theorem 15 we get:

Conclusion 16 *If* $\mathcal{E} = (E, D)$ *is EMCPA secure and ABE is SSM secure then* $(ABE \ || \ HKD)$ *is secure under USG.*

4 Policy Encryption

4.1 Background

Policy Encryption (PE) can replace ABE as a component of ACE. In many cases, PE is more efficient than ABE. However PE does not prevent collusions. Instead it limits collusions. PE is useful in those cases where owners can compartmentalize objects and subjects (namely, these compartments apply to access policies and to user's roles), so that within each compartment the owners can tolerate collusions by users. Usually, in a hierarchical organization, a compartment is a connected subset of the nodes of the poset that has a greatest element g and all the nodes that are dominated by g, or a union of several such structures. PE comes at the cost of adding new dummy attributes. For m PE compartments we need $2^m - 1$ additional dummy attributes. Computational complexity is linear in the number of attributes, but has very small constants, as shown later.

In ACE owners decide on whether to use ABE or PE as part of the definition of each access policy. For example, within the same system, a critical high-security compartment may use ABE, and another less critical compartment may use PE. Within the ABE compartment collusions are impossible, while within each PE compartment collusions are possible. Inter-compartment collusions are not possible.

PE has also the following appealing properties: PE relies on older hence more reliable intractability assumption, the Computational Diffie-Hellman assumption, whereas ABE relies on the newer Bilinear Diffie-Hellman assumption. PE uses off-the shelf standard crypto building blocks with one small modification, with proven security. For a small number of compartments PE is much faster than ABE. For example, for 10 real attributes and 5 compartments, for reasonable security parameters, it takes about 4 millisecond (worst case on a high performance PC) to encrypt a poset node using PE, Vs. 300 ms when using ABE on comparable machines with comparable security. Above 20 compartments there is no advantage to PE.

4.2 Method

4.2.1 Coloring

The goal in ABE is to assure that no collusion of subjects can exceed the union of their legitimately accessible objects. In PE after creating m compartments we want to assure that any collusion of representative subjects from $i \leq m$ compartments cannot exceed the union of the objects legitimately accessible from those compartments. The method is to assign for each subset of i compartments a new dummy attribute to "color" the poset elements in the complement of the union of these compartments. Since members of the collusion by definition do not have that dummy attribute they cannot exceed the union of their legitimately accessible objects. However, this method explodes very quickly, since the number of new attributes is $\sum_{i=0}^{m-1} \binom{m}{i} = 2^m - 1$. For small m the PE method is practical[14].

Here in more detail: PE works for any poset but we describe it for poset (X,≤)). User roles and objects access policies are defined over the poset as before. A compartment is composed of a subset of users U and the subset of objects O that the users in U can access. The compartment is represented by a set of nodes as we define next. As in section 2.2, let $\lambda : U \rightarrow X$ be the mapping that assigns to each user her subset of attributes (i.e. her role).

Definition 17 *For a given poset* $(X, \leq)$, *the* compartment $D(U)$ *of a subset of users* U *and the objects that they can legitimately access is the subset of all the nodes of the poset* $x \leq u$, *for all* $u \in \lambda(U)$.

Consider m compartments $C_1, C_2, ...C_m$. Every subset of $i \leq m$ of compartments $C_{j_1}, C_{j_2}, ...C_{j_i}$ represents a potential collusion. It has a complement $X \backslash \{C_{j_1} \cup C_{j_2} \cup ...C_{j_i}\}$. We color this complement with a new color; i.e add a new dummy attribute to the nodes of the complement.

This assures that members of any collusion are missing at least one attribute needed to access objects that are not in the union of objects that each member can legitimately access. This coloring has no effect on the access patterns of users not in the collusion, since when a user gets a new attribute (color) all the nodes accessible to her, which are not in the collusion, also get that new attribute. Other nodes may get that attribute too, but if they were not accessible beforehand then they are still not accessible.

[14] In a system that uses both ABE in some compartments and PE in other compartments, we have to include in m only the number of compartments that use PE.

4.2.2 Key Computation

Here we assume that the owner already added the dummy attributes as described above. Let $\mathbb{G}$ be a multiplicative cyclic group of prime order p, and let $\alpha \in \mathbb{G}$ be a public generator of $\mathbb{G}$. Let $x, y, y_1, ...y_k \in \mathbb{Z}_p$ be secret keys. For each poset node the owner establishes a *node symmetric key* that is computable by any user that has the secret keys of the subset of attributes corresponding to that node. Let the owner's secret key be $x \in \mathbb{Z}_p$, and the secret key of attribute i, be $y_i \in \mathbb{Z}_p$, $i = 1, 2, ..k$. The owner's public key is $\alpha^x \in \mathbb{G}$, and the public key of attribute i is $\alpha^{y_i} \in \mathbb{G}$. The owner computes the secret symmetric key for the node as

$$w = (\prod_{i=1}^{k} \alpha^{y_i})^x \in \mathbb{G} \tag{4}$$

and a user that has all the secret keys of that node computes the same value using

$$w = (\alpha^x)^{\sum_{i=1}^{k} y_i} \in \mathbb{G} \tag{5}$$

By the claim in the next section, a user that misses even one of the secret attribute keys is cut out, and cannot compute w, assuming the intractability of CDH as shown below. From equation (5) the reader can see that the user's complexity is only slightly higher than that of one DH computation. The owner's computation (equation 4) is slightly higher, but still much faster than ABE. We use w to symmetrically encrypt the HKD key of the node, creating c_1. The rest of the cryptograms of the node (c_2, c_3, c_4) (section 2.3) are not affected by this change. We can further improve the efficiency of PE using a twist on the standard ECIES[15]. For lack of space we defer the details to the full version.

4.3 Security of Policy Encryption

The security of PE is based on a variant of the Diffie-Hellman problem. We use CDH and ADH to denote the classic Computational DH problem and the new Aggregate DH problem, resp.

Problem CDH: Let $x, y \in \mathbb{Z}_p$, and $a, b, u \in \mathbb{G}$, where $a = \alpha^x, b = \alpha^y$. Given a, b; Find $u = \alpha^{xy} \in \mathbb{G}$.

Problem ADH: Let $c = \alpha^x$, $y_i \in \mathbb{Z}_p$, $\alpha^{y_i} \in \mathbb{G}$, $i = 1, 2, ..k$, Given c, α^{y_i}, $i = 1, 2, ..k$, and all the y_i except for y_j, for some $1 \le j \le k$. Find $w = \alpha^{x \sum_{i=1}^{k} y_i} \in \mathbb{G}$.

Problem CDH is a special case of ADH with $y_i = 0$ for all $1 \le i \le k$, $i \ne j$, thus in the worst case ADH is at least as hard as CDH, which is well established (since 1976). But Discrete Log problems are self reducible, hence as hard on the average as on the worst case. Thus it seems safe to use a cryptosystem based on ADH with large enough security parameter.

From the coloring method (sec. 4.2.1) we are assured that any collusion misses at least one dummy attribute key when trying to access an object not in the union of their permissible objects. Hence the collusion faces the ADH problem when trying to violate their permissible access patterns.

[15] Standards for Efficient Cryptography,SEC 1: Elliptic Curve Cryptography, Certicom Research, May 21, 2009, Version 2.0.

5 Consolidated Audit Trail

5.1 Background

The Consolidated Audit Trail (CAT)[16] will be the world's largest data repository of securities transactions. It will ingest 58 billion records of quote, order and executions quote life-cycles for equities and options on a daily basis with additional asset classes to be added in the future. It should securely maintain data on more than 100 million customer accounts and associated unique customer information. It will grow to an estimated 21 petabyte footprint within five years of operation. It will maintain and support thousands of daily data and communication interfaces across the industry. It will require approximately 2,000 firms and 19 Self Regulatory Organizations to report data (CAT Reporters). It will have about 3,000 users ("readers" from SEC, SRO, FINRA, EXCHANGES). Naturally computationally efficient security is mandatory. We illustrate how ACE can provide efficient cryptographically enforced high granular access control and secure communication to such a vast system.

The following are example CAT queries: (i) All transactions issued by trader x in time interval T_1. (ii) All transactions with ticker MSFT issued in time interval T_1, where the seller is x and the buyer is y. The system places limitations on what queries each player can issue, based on user attributes and object access policies.

The owners decide object's access policies. An individual financial transaction has components, such as: Ticker symbol, time, ask price, offer price, trader, brokerage, seller, buyer,...The following is a proposal for definitions of user attributes, and object access policies. On each component we define intervals. On non-numerical components intervals are lexicographic. The intervals are of sizes ≥ 1. An attribute is a component together with its interval, e.g. all the ticker symbols in lexicographic interval R. In addition we allow "external" attributes, such as "manager at level h in the SEC".

5.2 Implementation

5.2.1 Attributes and intervals

Attributes An individual *attribute* is of the form (c_i, d_i), where c_i is the title of the attribute, such as ticker, time, etc. and d_i is its interval, $|d_i| \geq 1$. An *interval* is defined by a pair of values: $d_i = [\delta_i', \delta_i"]$. In the case of non-numerical variables, such as ticker symbols, the interval is lexicographic. Its size $|d_i| = 1$ if $\delta_i" = \delta_i'$. For example, when the interval of the ticker is of size one the interval includes at most one particular ticker symbol, such as MSFT. Once we agree on the meaning of each component, e.g. that c_1 is always the ticker symbol, c_2 is the time, etc. we can denote a subset of attributes as a $k-$tuple using just the intervals: $(d_1, d_2, ...d_k)$. We adopt this notation henceforth.

Intervals An interval can be represented in detail by all its components, each having interval of size 1. For example, suppose that $k = 2$, d_1 contains two elements, μ_1, μ_2 and d_2 contains five elements, $\nu_1, \nu_2, ...\nu_5$. Let "+" denote a logical OR and "$\cdot$" denote a logical AND. The elements in d_1 are connected using logical OR, $d_1 = (\mu_1 + \mu_2)$ and so do the elements contained in d_2,

[16] A search for "Summary of the Consolidated Audit Trail, January 2015" would find it. The link itself was very long and did not compile nicely into pdf.

$d_2 = (\nu_1 + ...\nu_5)$. The 2-tuple is equivalent to the expression $d_1 \cdot d_2 = (\mu_1 + \mu_2) \cdot (\nu_1 + ...\nu_5)$. Written in set theoretic terms: $d_1 = \{\mu_1, \mu_2\}$ and $d_2 = \{\nu_1, ...\nu_5\}$. Suppose we have another interval corresponding to the first item in the 2-tuple with $d_1' = \{\mu_1\}$. Then $d_1' \subset d_1$. We henceforth use this set theoretic inclusion relation in this context. A poset node is of the form $x = (d_1, d_2, ...d_k)$. To summarize, we use the following notations: For size 1 intervals inside bigger intervals small Greek letters μ, ν. For general intervals of sizes ≥ 1 : d_i. And for poset nodes the letters x, y.

An object access policy may look like: $\mathbb{A}(o) = d_1 \cdot d_2 + d_3 \cdot d_4 \cdot d_7 + d_{11}$. This expression is in DNF, but additional internal layer exists inside each general interval, as explained above. We treat each interval d_i and d_i' as distinct attribute even if $d_i' \subseteq d_i$.

5.2.2 The poset

Definition 18 Poset (L,≤) : *For a given system the poset elements are all the k-tuples of intervals of the form* $\omega = (d_1, ...d_k)$ *that are used to define user roles or objects access policies such that for every two nodes* $x = (d_1, d_2, ...d_k)$ *and* $y = (d_1', d_2', ...d_k')$, $y \leq x$, *if for all* $i = 1, 2, ...k$, $d_i' \subseteq d_i$.

The resulting poset $(L, \leq)$ is different from $(X, \leq)$. The hierarchy among nodes of $(L, \leq)$ is influenced by the hierarchy among the basic attributes. This is different from the partial order of $(X, \leq)$ which stems solely from subset inclusion relations among attributes, and where there is no hierarchy among basic attributes.

A node of $(L, \leq)$ is *usable* if it is used in some user role or in some object access policy. The owner decides which nodes are usable when constructing $(L, \leq)$.

Constructing $(L, \leq)$: Start with the top node where every component has its maximal usable interval. Below it put each of the usable nodes that differ from the father in just one element, and in this element use a maximally usable interval which is strictly included in the corresponding interval of the father. Proceed downwards iteratively. In the leaves all the intervals are of size 1. The leaves correspond to the objects, as explained below in more detail.

Truncating $(L, \leq)$: In order to make the system practical we may have to throw away a few of the lowest layers[17]. This defines the granularity with which we can express policies and queries. This is the reason that we may have overlaps among object policies.

As explained in section 30, HKD works on any poset. In particular it works on the poset $(L, \leq)$.

5.2.3 Objects

Objects are associated with subsets of attributes $\omega = (d_1, ...d_k)$, where for $i = 1...k$, $|d_i| = 1$. Each object's access policy is represented by a subset of nodes of the poset. For example, let x, y, be nodes where $x = (d_1, d_2, ...d_k)$ and $y = (d_1', d_2', ...d_k')$. If the access policy is $\mathbb{A} = \{x, y\}$ It means in DNF predicate terms $\mathbb{A} = (d_1 \cdot d_2 \cdot ...d_m) + (d_1' \cdot d_2' \cdot ...d_m')$. Remember that each interval d_i is of the form $(\mu_1 + \mu_2 + ...)$ as explained above.

[17] Didactically it is nice to keep the leaves and identify them with the protected objects, however, to keep in line with general ACE (and to save a bit on complexity) we may exclude the leaves from the poset.

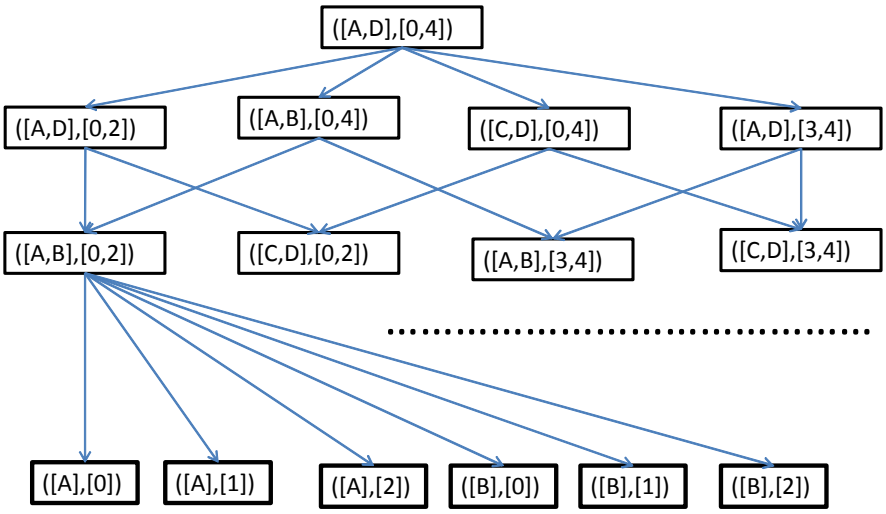

Figure 2: A partial description of the lattice (L,≤) of the CAT example.

5.2.4 Access

In CP-ABE (hence in ACE) an object's access policy is a subset of incomparable nodes of $(L, \leq)$. A user u having subset of attributes ω *can access* object o having access policy $\mathbb{A}=\{x_1, x_2, ...x_m\}$ iff there exists j, $1 \leq j \leq k$, such that $\omega \geq x_j$ in $(L, \leq)$. Recall that here each x_i is a $k-tuple$ of intervals of the form $(d_1, ...d_k)$.

5.2.5 Example

Assume just two components: (ticker, time), where the ticker's maximal lexicographic interval is [A,D], and the time's maximal interval is [0,4]. The top node of the poset (L,≤) is ([A,D],[0,4]). The leaves are (i, j), where $i \in$[A,D], and $j \in$[0,4]. We assume that the usable intervals are: Tickers [A,D],[A,B], [C,D]; Time: [0,4],[0,2], [3,4]. Note that we truncated the poset by excluding one layer above the leaves, see Figure 2. In this example, each leaf (=object) has just one ancestor, and a possible object access policy could be that unique ancestor. However, if we truncate one more layer then we get 2 ancestors for each object. For example, object ([A],[0]) has the 2 ancestors ([A,D],[0,2]) and ([A,B],[0,4]). Now a policy is composed of 2 nodes, and overlaps among policies exist. ACE takes advantage of those overlaps. For example, node ([A,B],[0,4]) is both in the access policy of object (leaf) ([A],[0]) and of object ([B],[3]).

5.2.6 Auxiliary nodes

We can add auxiliary nodes such as "level h manager at SEC" and map them into subsets of nodes of (L,≤). Each such node gets an ABE and HKD keys as ordinary nodes. And HKD is used as before to create cryptograms attached to the edges going from the auxiliary node into ordinary nodes of (L,≤). We do not see a reason for having edges from ordinary (L,≤) nodes to auxiliary nodes (the auxiliary nodes are shorthand for the subsets of ordinary nodes that represent them in (L,≤).

5.2.7 Users with multiple roles

A user that has multiple roles is associated with the corresponding multiple nodes of (L,≤), and gets a CP-ABE key to each of these nodes. A user having CP-ABE keys to 2 incomparable nodes may or may not get the key to an ancestor of these nodes, defined by the union of attributes of these two nodes. Such an ancestor may or may not exist. In unpruned (X,≤) it exists. In (L,≤) it may not exist, since the union is replaced by a new attribute. For example, in Figure 2 a user having the ABE keys to nodes ([A,B],[0,2]) and to ([C,D],[0,2]) is entitled to get the key to node ([A,D],[0,2]) but this in not automatic.

5.2.8 The player's routines

The players' routines remain as before (see section 2.3.2), with the poset $(L, \leq)$ replacing the poset $(X, \leq)$. CAT is a special case, where the objects themselves can be viewed as leaves in the poset (L,≤). The object is a descendent of each of the nodes in its access policy.

5.2.9 Queries:

A query in CAT is represented as a node $Q \in L$. A user must have access to the node representing query Q in order to invoke it. The user expects to receive all the objects which fall in all the specified intervals of Q. Namely, if the user can access Q then ACE returns all the objects whose leaves are accessible in $(L, \leq)$ from the node representing query Q. Formally a query must specify all the k components, possibly some of them with maximal intervals. The meaning of a query that does not specify all the components is that the unspecified components are there and have the maximal interval. In the example above suppose that a query is $Q = ([A, D], [0, 2])$. Then a response to this query includes all the leaves shown in Figure 2 plus all the leaves coming from node $([C, D], [0, 2])$ (not shown).

Deterministic *Encrypted search* (=encrypted queries) leaks a lot of information, since it is very difficult to conceal the shape of (L,≤) and still remain practical. Non-deterministic encrypted search is too slow for practical usage. Thus we recommend to avoid query encryption. This is a practical compromise.

5.3 CAT complexity

We compare ACE for CAT over (L,≤) to a modified Crampton [7] (see TERMINOLOGY in 1) and to [12]: These other systems need extra nodes in the poset; one for each object access policy (as is done in [7] for poset (X,≤)). ACE does not need those extra nodes.

The subset of all queries leading to an object define its access policy. Its most compact definition includes just an antichain comprised of the minimal incomparable elements of the subset.

The non-asymptotic analysis in section 2.3.3 covers ACE for CAT over (L,≤) complexity as well (although, as mentioned there, the upper bounds do not apply).

Also, when assessing the size of (L,≤) the leaves can be dropped, since a leaf does not need ABE and HKD keys. All it needs is a PL key. The leaves correspond to the objects, and are helpful in explaining the system and its uniform presentation.

5.4 CAT security

This is covered in the general discussion in subsection 3.

References

[1] J. Atallah, M. Blanton, K. B. Frikken, Key Management for Non-Tree Access Hierarchies. Proceeding SACMAT '06 Proceedings of the eleventh ACM symposium on Access control models and technologies, ISBN:1-59593-353-0

[2] M. J. Atallah, M. Blanton, N. Fazio, K. B. Frikken, Dynamic and Efficient Key Management for Access Hierarchies. ACM Transactions on Information and System Security 12(3), 1–43 (2009)

[3] S. Akl, and P.D. Taylor: Cryptographic Solution to the problem of access control in a Hierarchy, 1983 ACM 0734-2071/83/0800-0239.

[4] M. Bellare, A. Boldyreva, S. Micali, Public-Key Encryption in a Multi-user Setting: Security Proofs and Improvements, Advances in Cryptology — EUROCRYPT 2000 , Lecture Notes in Computer Science Volume 1807, 2000, pp 259-274

[5] D. Boneh, M. Hamburg, Generalized Identity Based and Broadcast Encryption Schemes, J. Pieprzyk (Ed.): ASIACRYPT 2008, LNCS 5350, pp. 455–470, 2008.

[6] J. Bethencourt, A. Sahai, and B. Waters, Ciphertext-Policy Attribute-Based Encryption, 2007 IEEE Symposium on Security and Privacy(SP'07), 0-7695-2848-1/07

[7] J. Crampton, Cryptographic Enforcement of Role-Based Access Control, P. Degano, S. Etalle, and J.D. Guttman (Eds.): FAST 2010, LNCS 6561, pp. 191–205, 2011. Springer-Verlag Berlin Heidelberg 2011.

[8] V. Goyal, A. Jain, O. Pandey, A. Sahai, Bounded Ciphertext Policy Attribute Based Encryption, L. Aceto et al (Eds.): ICALP 2008, Part II, LNCS 5126, pp. 579-591, 2008. Springer Verlag 2008

[9] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data, Proceedings of the 13th ACM conference on Computer communications security, 2006, pages: 89 – 98, ISBN: 1-59593-518-5

[10] M. Ito, A. Saito, and T. Nishizeki. Secret Sharing Scheme Realizing General Access policy. In Proc. IEEE Globecom, 1987, pp. 99-102.

[11] S. Kumar N.,R. Lakshmi G V.,Balamurugan B., Enhanced Attribute Based Encryption for Cloud Computing, International Conference on Information and Communication Technologies (ICICT 2014), Procedia Computer Science 46 (2015) 689 – 696.

[12] K. E. Lauter, M. Bellare, J. Benaloh, M. E. Chase, E. J. Horvitz, C. D. Karkanias: User-specified sharing of data via policy and/or inference from a hierarchical cryptographic store, USA Patent number: 8837718, Issued: September 16, 2014.

[13] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption, Advances in Cryptology, in H. Gilbert (Ed.): Eurocrypt 2010, Springer Verlag Lecture Notes in Computer Science Volume 6110, 2010, pp 62-91

[14] A. Lewko, B. Waters, New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts, Theory of Cryptography, Lecture Notes in Computer Science Volume 5978, 2010, pp 455-479

[15] A. Lewko, B. Waters, Decentralized Attribute-Based Encryption, Advances in Cryptology–EUROCRYPT 2011, Lecture Notes in Computer Science Volume 6632, 2011, pp 568-588

[16] A.J. Menezes, P.C. van Oorshcot, S. A. Vanstone, Handbook of Applied Cryptography, CRC, ISBN 0-8493-8523-7.

[17] R. Ostrovsky, A. Sahai, B. Waters, Attribute Based Encryption with Non-Monotonic Access Structures, Proc. CCS'07, Proceedings of the 14th ACM Conf. on Computer and Communications Security, Pp. 195-203, ISBN: 978-1-59593-703-2

[18] T. Okamoto, D. Pointcheval, REACT: Rapid Enhanced-security Asymmetric Cryptosystem Transform, CT-RSA 2001, D. Naccache Ed., LNCS 2020, pages 159-175, Springer-Verlag, 2001.

[19] B. Waters, Ciphertext Policy Attribute Based Encryption: An Expressive, efficient, and provably secure realization, Proceedings, Public Key Cryptography - PKC 2011, 2011 - Springer

[20] A.C. Yao, "Theory and applications of trapdoor functions," in FOCS' 82: Proceedings of the 23rd IEEE Annual Symposium on Foundations of Computer Science, pp. 80-91, 1982.

6 APPENDIX A: Symmetric Encryption

The following definitions of semantic security is taken from [18].

Definition 19 *(Symmetric Encryption Scheme). A symmetric encryption scheme with a key-length k, on messages of length l, consists of 2 algorithms $\mathcal{E}=(E_\kappa^{sym}, D_\kappa^{sym})$ which depends on the k-bit string κ, the secret key: the encryption algorithm $E_\kappa^{sym}(m)$ outputs a ciphertext c corresponding to the plaintext $m \in \{0,1\}^l$ in a deterministic way; the decryption algorithm $D_\kappa^{sym}(c)$ gives back the plaintext m associated with the ciphertext c.*

Definition 20 *(Semantic Security). A symmetric encryption scheme is said to be semantically secure if no polynomial-time attacker can learn any bit of information about the plaintext from the ciphertext, excepted the length. More formally, a symmetric encryption scheme is said $(t;\epsilon)$-IND if for any adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ with running time bounded by t, $Adv(A, \mathcal{E}) < \epsilon$; where*

$$Adv(\mathcal{A}, \mathcal{E}) = 2 \cdot \Pr_{\substack{\kappa \in_R \{0,1\}^k \\ b \in_R \{0,1\}}} [(m_0, m_1, s) \leftarrow \mathcal{A}_1(1^k),\ c \leftarrow E_\kappa^{sym}(m_b) : \mathcal{A}_2(c, s) = b] - 1 \qquad (6)$$

in which the probability is also taken over the random coins of the adversary, and m_0, m_1 are two identical-length plaintexts chosen by the adversary in the message-space $\{0,1\}^l$ (there is also an asymptotically unimportant factor 2 difference between this definition and our standard $|\Pr[b' = b] - \frac{1}{2}|$).

7 Appendix B: Proof of Security for HKD

Notation 21 *($\mathcal{C} \leftrightarrow \mathcal{A}$) denotes a security game, where $\mathcal{C}$ is a challenger, and $\mathcal{A}$ is an attacker. The challenger is always on the left side and the attacker is on the right side.*

Theorem 10 *Let $\mathcal{E} = (E, D)$ be EMCPA secure (Sec. 3.2.2). Then HKD built using $\mathcal{E}$ (Def. 30) over any poset $(X, \leq)$ is TSSM secure (Sec. 3.2.1).*

Proof. We use a proof technique which is similar to Hybrid Argument ([20]). Let $\mathcal{A}$ be any efficient TSSM-attacker of HKD, $C_\mathcal{E}$ be an $\mathcal{E}-$Challenger in an EMCPA game, and $\mathcal{B}$ be an EMCPA attacker of $\mathcal{E}$ and a TSSM challenger of $\mathcal{A}$. We show a reduction from EMCPA of $\mathcal{E}$ to TSSM of HKD built from $\mathcal{E}$, proving that the existence of efficient attacker $\mathcal{A}$ of HKD implies the existence of efficient attacker $\mathcal{B}$ of $\mathcal{E}$ that uses $\mathcal{A}$ as an oracle.

It is sufficient to prove that all the leaves of $(X, \leq)$ are secure since if a non-leaf node is not secure then all its descendents including leaves are not secure. We can always add dummy leaves to any node, and if the system with dummy leaves is secure then so is the system without the dummy leaves (by a trivial reduction). Let $r \in X$ denote the root of $(X, \leq)$ (we can always complete the poset to have a unique root even if it does not represent a usable role). W.l.g. we assume that $X \backslash r$ is not empty. A node of $(X, \leq)$ that is already assigned an HKD key is called *assigned node*. Node names are arranged in some linear order and serve as key selectors so that node s is associated with HKD key $h(s)$. The reduction consists of the following steps **(a)**$\rightarrow$ (**h**)

(a) $\mathcal{A}$ picks a leaf $\gamma \in X \backslash r$ to be attacked. Let u be the direct ancestor of γ in $(X, \leq)$.

(b) $\mathcal{B}$ acting as an attacker of $\mathcal{E}$ issues type (ii) EMCPA queries to $\mathcal{C}_{\mathcal{E}}$ to determine edge cryptograms on each of the edges on all the paths leading to u. //The corresponding node keys are implicitly assigned by $C_{\mathcal{E}}$.

(c) $\mathcal{B}$ acting as an attacker of $\mathcal{E}$ issues type (i) EMCPA queries to $\mathcal{C}_{\mathcal{E}}$ using identical twin messages (the query is of the form (s, m, m)[18]) for each unassigned descendent s' of each node s that was assigned in step **(b)**. // $\mathcal{B}$ uses it to assign HKD key m to s' and get the (s, s') edge-cryptogram. Note that by definition s' does not dominate γ in $(X, \leq)$.

(d) $\mathcal{B}$ picks random node keys for all the yet unassigned nodes and computes all the remaining edge cryptograms, except for neighborhood $N(\gamma, .)$. // They are all descendents of nodes that were assigned in step **(c)** above.

(e) $\mathcal{A}$ does phase-1 of a TSSM game. $\mathcal{B}$ knows the keys of those nodes from **(c,d)** above and can expose them. // Every node that does not dominate γ was assigned by $\mathcal{B}$.

(f) $\mathcal{A}$ picks a challenge of equal length message pair (m_0, m_1) against leaf γ. $\mathcal{B}$ transfers (u, m_0, m_1) to $C_{\mathcal{E}}$ & Gets $c_b \leftarrow E(h(u), m_b)$. //This is a standard type (i) EMCPA query.

(g) $\mathcal{A}$ does phase-2 of a TSSM game, which is handled like Phase-1 above.

(h) $\mathcal{A}$ guesses a bit $\widetilde{b}$ & sends it to $\mathcal{B}$ who passes it to $C_{\mathcal{E}}$ as his guess in the EMCPA attack on $\mathcal{E}$.

The bits $\widetilde{b}$ and b are common to the above two games: $(C_{\mathcal{E}} \longleftrightarrow \mathcal{B})$ and $(\mathcal{B} \longleftrightarrow \mathcal{A})$. Therefore

$$\Pr[\mathcal{A} \text{ returns } \widetilde{b} = b \text{ in TSSM of HKD}] = \Pr[\mathcal{B} \text{ returns } \widetilde{b} = b \text{ in the EMCPA game of } \mathcal{E}] \quad (7)$$

implying

$$Adv_{TSSM}(\mathcal{A}, HKD) = Adv_{EMCPA}[\mathcal{B}, \mathcal{E}]. \quad (8)$$

Therefore if for all efficient $\mathcal{B}$, $Adv_{EMCPA}[\mathcal{B}, \mathcal{E}]$ is negligible then for all efficient $\mathcal{A}$, $Adv_{TSSM}(\mathcal{A}, HKD)$ is negligible. ■

[18] These are "trick" queries, where the value of the "guessed" bit is unimportant.

8 Appendix C: Proof of Security of Parallel Connection of Simple Block Ciphers

Lemma 12: *Let F and G be semantically secure block ciphers (see Def. in Appendix A Sec. 6) having the same message space. Then $(F \,||\, G)$ is semantically secure.*

Proof. We use a Hybrid Argument[19] ([20]). Let $\mathcal{A}$ be an efficient attacker of $(F \,||\, G)$. And let $C_{(F||G)}$ denote a challenger in the following game:

Game ($\mathcal{C}_{(F||G)} \leftrightarrow \mathcal{A}$):
· $C_{(F||G)}$ sets F and G using secret keys $k_f, k_g \in_R \mathcal{K}$,
· $\mathcal{A}$ picks equal length messages $m_0, m_1 \in \mathcal{M}$ and sends them to $C_{(F||G)}$.
· $C_{(F||G)}$ picks secret $b \in_R \{0,1\}$, and returns to $\mathcal{A}$ the pair $(\ c_{f,b},\ c_{g,b})$,
where $c_{f,b} \leftarrow F(k_f, m_b), \quad c_{g,b} \leftarrow G(k_g, m_b)$.
· $\mathcal{A}$ outputs $\widetilde{b} \in \{0,1\}$, his guess of b.

Let $p_0 = \Pr[\mathcal{A} \text{ returns } \widetilde{b} = 1 \mid \ b = 0]$ and $p_2 = \Pr[\mathcal{A} \text{ returns } \widetilde{b} = 1 \mid b = 1]$ in the above game. The advantage of attacker $\mathcal{A}$ in the above game is

$$Adv(\mathcal{A}, (F \,||G)) = |p_2 - p_0| \tag{9}$$

System $(F \,||G)$ is semantically secure if for all efficient attackers $\mathcal{A}$, $Adv(\mathcal{A}, (F \,||G))$ is negligible. In addition we consider a case where $\mathcal{A}$ is presented with a "hybrid" pair $(\ c_{f,1},\ c_{g,0})$. This event is impossible for $C_{(F||G)}$ but can happen in the hybrid experiments that we describe next. Let $p_1 = \Pr[\mathcal{A} \text{ returns } \widetilde{b} = 1$ when presented with a "hybrid" pair $(\ c_{f,1},\ c_{g,0})]$. Let $\mathcal{B}_F$ denote an attacker of F and a challenger of $\mathcal{A}$. We denote these two simultaneous games as $(\mathcal{C}_F \leftrightarrow \mathcal{B}_F \leftrightarrow \mathcal{A})$ and define them as follows:

Game ($\mathcal{C}_F \leftrightarrow \mathcal{B}_F \leftrightarrow \mathcal{A}$) :
· C_F sets F using secret key $k_f \in_R \mathcal{K}$,
· $\mathcal{B}_F$ sets G using $k_g \in_R \mathcal{K}$,
· $\mathcal{A}$ picks equal length messages $m_0, m_1 \in \mathcal{M}$ and sends them to $\mathcal{B}_F$ who sends them to C_F.
· C_F picks secret $b \in_R \{0,1\}$, and returns to $\mathcal{B}_F$ $c_{f,b} \leftarrow F(k_f, m_b)$.
· $\mathcal{B}_F$ computes $c_{g,0} \leftarrow G(k_g, m_0)$ and sends to $\mathcal{A}$ the pair $(\ c_{f,b},\ c_{g,0})$.
· $\mathcal{A}$ outputs $\widetilde{b} \in \{0,1\}$, his guess of b and $\mathcal{B}_F$ transfers it to C_F.

Likewise, let $\mathcal{B}_G$ be an attacker of G and a challenger of $\mathcal{A}$ defined as follows:

Game ($\mathcal{C}_G \leftrightarrow \mathcal{B}_G \leftrightarrow \mathcal{A}$) :
· C_G sets G using secret key $k_g \in_R \mathcal{K}$,
· $\mathcal{A}$ picks equal length messages $m_0, m_1 \in \mathcal{M}$ and sends them to $\mathcal{B}_G$ who sends them to C_G.

[19] http://cs.nyu.edu/courses/fall08/G22.3210-001/lect/lecture5.pdf

· C_G picks secret $b \in_R \{0,1\}$, and returns to $\mathcal{B}_G$ $c_{g,b} \leftarrow G(k_g, m_b)$.
· $\mathcal{B}_G$ sets F using $k_f \in_R \mathcal{K}$, computes $c_{f,1} \leftarrow F(k_f, m_1)$
and sends to $\mathcal{A}$ the pair $(c_{f,1}, c_{g,b})$.
· $\mathcal{A}$ outputs $\widetilde{b} \in \{0,1\}$, his guess of b and $\mathcal{B}_G$ transfers it to C_G.

The essence of the reduction is showing that an efficient $\mathcal{A}$ that breaks (F||G) with non negligible probability can be used as an oracle to efficiently break at least one of F or G with non-negligible probability. Clearly $\mathcal{B}_F$ and $\mathcal{B}_G$ are efficient. In the reduction we flips a fair coin $w \in \{F, G\}$ and play game $(\mathcal{C}_w \leftrightarrow \mathcal{B}_w \leftrightarrow \mathcal{A})$ as defined above. So with probability $\frac{1}{2}$ we play $(\mathcal{C}_G \leftrightarrow \mathcal{B}_G \leftrightarrow \mathcal{A})$ and with probability $\frac{1}{2}$ we play $(\mathcal{C}_F \leftrightarrow \mathcal{B}_F \leftrightarrow \mathcal{A})$.

Recall the definitions of p_i, $i = 0, 1, 2.$ of game $(\mathcal{C}_{(F||G)} \leftrightarrow \mathcal{A})$. The values of $\widetilde{b}$ and b of the two simultaneous games within $(\mathcal{C}_F \leftrightarrow \mathcal{B}_F \leftrightarrow \mathcal{A})$ are the same. Therefore in this game $\Pr[(\mathcal{B}_F$ returns $\widetilde{b} = 1) \ \Lambda \ (C_F$ chooses $b = 0) \ \Lambda \ (w = F)] = p_0$. Let $q_0 = \Pr[(\mathcal{B}_F$ returns $\widetilde{b} = 1) \ \Lambda \ (C_F$ chooses $b = 0)]$. Then $q_0 \cdot \Pr[w = F] = p_0$, hence $q_0 = 2 \cdot p_0$. We define and analyze q_i, $i = 1, 2, 3$ likewise:

$$\begin{aligned}
q_0 &= \Pr[(\mathcal{B}_F \text{ returns } \widetilde{b} = 1) \ \Lambda \ (C_F \text{ chooses } b = 0)] = 2 \cdot p_0 \\
q_1 &= \Pr[(\mathcal{B}_F \text{ returns } \widetilde{b} = 1) \ \Lambda \ (C_F \text{ chooses } b = 1)] = 2 \cdot p_1 \\
q_2 &= \Pr[(\mathcal{B}_G \text{ returns } \widetilde{b} = 1) \ \Lambda \ (C_G \text{ chooses } b = 0)] = 2 \cdot p_1 \\
q_3 &= \Pr[(\mathcal{B}_G \text{ returns } \widetilde{b} = 1) \ \Lambda \ (C_G \text{ chooses } b = 1)] = 2 \cdot p_2
\end{aligned} \tag{10}$$

Given that each of F and G is semantically secure $Adv(\mathcal{B}_F, F) = |q_1 - q_0| = 2 \cdot |p_0 - p_1|$ is negligible and $Adv(\mathcal{B}_G, G) = |q_3 - q_2| = 2 \cdot |p_2 - p_1|$ is negligible. Let $p_1 = p_0 + \epsilon_1$, and $p_2 = p_1 + \epsilon_2$ where ϵ_1, ϵ_2 are negligible. Then $p_2 = p_0 + \epsilon_1 + \epsilon_2$. $Adv(\mathcal{A}, (F \ || \ G)) = |p_2 - p_0|$ (Eq. 9). Therefore $Adv(\mathcal{A}, (F \ || \ G)) = |\epsilon_1 + \epsilon_2|$ is also negligible.

■

9 Appendix D: Monotone Access policies and their upper bound

We consider here w.l.g. Monotone Access Structures (aka monotone access policies). A MAS is a collection of non-empty subsets of the set of attributes, such that for every element in the MAS all its supersets are also elements of the MAS, implying that the whole set of attributes is an element of every MAS. The following definition is due to Ito, Saito and Nishizaki ([10]). This appendix is confined to poset $(X, \leq)$.

Definition 22 *Let Att denote a set of binary attributes, and let* $X = 2^{Att}$. *A* `Monotone Access Structure` *is a collection* $\mathbb{S} \subseteq 2^{Att} \backslash \phi$, *s.t. for* $A, B \subseteq 2^{Att}$, *if* $A \in \mathbb{S}$, *and* $A \subseteq B$ *then* $B \in \mathbb{S}$.

An element of a MAS, $\beta \in \mathbb{A}$, is *minimal*, if there is no other element $\alpha \in \mathbb{A}$ s.t. $\alpha \subseteq \beta$ (where inclusion is over the set of attributes *Att*). Every MAS $\mathbb{A}$ is completely defined by its minimal elements. This description is an anti-chain in the poset $(X, \leq)$. This anti-chain corresponds to a minimized predicate $P()$ in Disjunctive Normal Form, such that for every $\omega \subseteq Att$, $P(\omega) =$ True iff $\omega \in \mathbb{A}$.

For example, let $Att = \{x, y, z\}$, and let $\mathbb{A} = \{\{x, y\}, \{y, z\}, \{x, y, z\}\}$. One can easily verify that $\mathbb{A}$ includes every superset of each of its elements, and therefore $\mathbb{A}$ is a MAS. Like every MAS it includes the full set of attributes, which in this case is $Att = \{x, y, z\}$. However some elements of 2^{Att} are missing, e.g. $\{x, z\} \notin \mathbb{A}$. The corresponding DNF predicate $P = xy + yz + xyz$ can be minimized into $P = xy + yz$. Now the clause xyz which corresponds to element $\{x, y, z\}$ doesn't show. Yet, for every element $\omega \subseteq Att$, $P(\omega) = \text{True}$ iff $\omega \in \mathbb{A}$.

10 APPENDIX E: Comparison to Crampton

10.1 A short survey of Crampton's relevant definitions and results

We compare ACE to [7]. Let Att denote the set of attributes. A role r is isomorphic to a subset of attributes, we view a role and the corresponding subset of attributes synonymously and write $r \subseteq Att$. The set of roles is X. We use $(X, \leq)$ to denote a poset were partial orders are dictated by the natural subset inclusion of subsets of attributes. Directed edges represent these relations. The sets U and O represent users and objects, resp. We next define Role Based Access Control (RBAC).

Definition 23 *A core RBAC policy is represented by a function $\phi : U \cup O \to 2^X$. A user $u \in U$ is authorized to access object $o \in O$ iff $\phi(u) \cap \phi(o)$ is not empty.*

When $\phi(u) = \{r_1, r_2\}$ Crampton denotes it as a conjunction $r_1 \wedge r_2$, and when $\phi(o) = \{r_1, r_2\}$ he denotes it as a disjunction $r_1 \vee r_2$. This reflects a semantics that when $\phi(u) = \{r_1, r_2\}$ user u has both roles, and when an object o has $\phi(o) = \{r_1, r_2\}$ it means that it is sufficient for users to have at least one of the roles to access o. Let $r = \{r_1, r_2, ...r_k\}$. Crampton uses the notation $\vee r$ to denote $(r_1 \vee r_2 \vee ... \vee r_k)$, and likewise $\wedge r$ denotes $(r_1 \wedge r_2 \wedge ... \wedge r_k)$.

For every non-empty chain $C \subseteq X$, $\vee C$ is the minimum element in C and $\wedge C$ is the maximum element in C. Therefore the most compact representation of the set of roles authorized for a user and the set of roles authorized for a permission ("access policies" or "access structures") are antichains over X. W.l.g. we focus in this paper on Monotone Access Structures (MAS). The following definition is due to Ito, Saito and Nishizaki ([10]).

Definition 24 *Let Att denote the set of attributes, and let $X = 2^{Att}$. A Monotone Access Structure (MAS) is a collection $S \subseteq 2^{Att}$, s.t. for $A, B \subseteq 2^{Att}$, if $A \in S$, and $A \subseteq B$ then $B \in S$.*

The notation $(\forall b \in B)(\exists a \in A)[a \leq b]$ is the usual shorthand for "for all b in B there exists a in A such that $a \leq b$."

Lemma 25 *Let $\mathcal{A}^X$ denote the set of antichains in poset $(X, \leq)$. For $A, B \in \mathcal{A}^X$ define:*
- *$A \preceq_1 B$ iff $(\forall b \in B)(\exists a \in A)[a \leq b]$, and*
- *$A \preceq_2 B$ iff $(\forall a \in A)(\exists b \in B)[a \leq b]$,*

Then $\preceq_1$ and $\preceq_2$ define partial orders on $\mathcal{A}^X$.

Definition 26 *Given a poset of roles $(X, \leq)$ and a hierarchical RBAC policy $\phi : U \cup O \to \mathcal{A}^X$, define $(Auth(X), \sqsubseteq)$, the authorization poset induced by $(X, \leq)$ and ϕ, as follows:*

– $\forall x \in X, \quad x \in Auth(X)$;
– $A = \phi(u)$ *for some* $A \in \mathcal{A}^X \;\Rightarrow\; \wedge A \in \mathcal{A}^X$;
– $B = \phi(o)$ *for some* $B \in \mathcal{A}^X \;\Rightarrow\; \vee B \in \mathcal{A}^X$;
– $\vee A \sqsubseteq \vee B$ *iff* $A \preceq_1 B$;
– $\wedge A \sqsubseteq \wedge B$ *iff* $A \preceq_2 B$;
– $\vee A \sqsubseteq \wedge B$ *iff* $(\exists x \in X)[A \preceq_1 \{x\}\ AND\ \{x\} \preceq_2 B]$;
– *It is never the case that* $\wedge A \sqsubseteq \vee B$.

We next define Ciphertext Policy ABE (CP ABE).

Definition 27 *An instance of CP ABE is defined by a function* $\psi : U \cup O \rightarrow 2^{2^{Att}}$, *where*
- $\psi(u), u \in U$, *is equal to* $\{A\}$ *for some* $A \subseteq Att$;
- $\psi(o), o \in O$, *is equal to some monotone access structure* S *defined over* Att.
- *A user* u *is authorized for object* o *iff* $\psi(u) \in \psi(o)$.

Definition 28 *An* information flow policy *specifies which flows of information are authorized [14]. To specify such a policy, we define:*
– *a partially ordered set (poset) of security labels* $(L, \leq)$;
– *a set of users* U *and a set of objects* O;
– *a security function* $\lambda : U \cup O \rightarrow L$, *where* $\lambda(x)$ *denotes the security label of entity* x.
A user u *is authorized to read* o *if and only if* $\lambda(u) \geq \lambda(o)$.

Definition 29 *Generic* Cryptographic Enforcement of Information Flow Policies:
Given a set of security labels L *and a function* λ:
1. *associate a cryptographic key* $\kappa(x)$ *with each* $x \in L$,
2. *encrypt every object* o *such that* $\lambda(o) = x$ *with* $\kappa(x)$, *and*
3. *for every user* u *such that* $\lambda(u) = x$, *ensure that* u *can derive key* $\kappa(y)$ *iff* $y \leq x$.

Then user u can decrypt any object o with security label $y \leq x$, but cannot decrypt any other object. There are several generic techniques that can be used to realize such cryptographic enforcement schemes for information flow (called key assignment, key allocation, key distribution, or key agreement schemes). In this paper, we focus on key assignment schemes that use iterative key derivation (which is defined by the traversal of some path in an appropriate directed, acyclic graph, in our case the Hasse-Diagram representing $(L, \leq)$). In such schemes, each user is given a single key and the scheme administrator publishes additional information that enables the derivation of other keys. We use the name *Hierarchical Key Derivation* (HKD) for such a scheme. We build HKD around symmetric-key encryption function $\mathcal{E} = (E, D)$ defined over spaces $(\mathcal{K}, \mathcal{M}, \mathcal{C})$; the key, message and cryptogram spaces, resp. Let $E(\kappa, m)$ denotes the symmetric encryption of message m with key κ.

Definition 30 *(A concrete cryptographic enforcement of information flow policy): Given a directed graph* $G = (V, A)$ *and a function* $\lambda : U \cup O \rightarrow V$, *assign a key* $h(x)$ *to each* $x \in V$, *and for each edge* $(x, y) \in A$ *include* $E(h(x), h(y))$ *in the public information.*

A user u such that $\lambda(u) = x$ can derive the key for any $y \in V$ if there exists a directed path from x to y in G. Clearly, such a scheme can be used to enforce an information flow policy, where G is the Hasse diagram [12] of $(L, \leq)$.

Crampton showed how to implement RBAC using Cryptographic Enforcement of Information Flow Policies, and how to use RBAC to implement CP ABE (hence use Cryptographic Enforcement of Information Flow Policies to implement CP ABE). Implying that we can use symmetric key cryptography to implement CP ABE. Crampton proved his claim in detail for CP ABE, however the proof applies to KP ABE as well (Key Policy ABE; a detailed definition appears in sec. 2.2). See a comment on pp. 201 (11) in [7]. In this paper when we write ABE we mean KP ABE.

None of these methods explains how users get their top keys. It could of course come from a trusted Certification Authority. However, when roles are equated with subsets of attributes, ABE is the natural tool to deliver the top keys. This is especially true in the case of multiple uncoordinated CAs. We consider a design where each user gets her top key using ABE and only then proceeds down $(Auth(X), \sqsubseteq)$ using symmetric key functions.

The size of $(Auth(X), \sqsubseteq)$ is upper bounded by the Dedekind number of $m = |Att|$, which is almost double exponential in m (see section 2.2.1), while the size of $(X, \leq)$ is upper bounded by a single exponential in m. Next we show that in fact it is sufficient to work with $(X, \leq)$ and there is no need to work with the much larger $(Auth(X), \sqsubseteq)$. The significance of this point will become apparent in section 2.2.2, where we explain how and why we run ABE and HKD over the same poset of roles. For owners the total number of ABE encryption operations is upper bounded by the sizes of those posets. For example for $m = 8$ the upper bound on the size of $(Auth(X), \sqsubseteq)$ is already impractical, while the upper bound on the size of $(X, \leq)$ is still very practical (2^{72} Vs. 2^8, resp. see section 2.2.1).

10.2 Small Posets are Sufficient

Logically this sub-section does not belong under the title "basic," however, it is easier to read right after seeing Crampton's definitions.

Our improved efficiency compared with [7] relies on the following Lemma 31. As before, let Att, denote the set of attributes. Roles correspond to subsets of attributes. Let the set of roles be $X \subseteq 2^{Att}$. Let $\mathcal{A}^{Att}$ denote the set of antichains over Att, and let $\psi : U \times O \rightarrow \mathcal{A}^{Att}$ be a function describing user roles and objects MAS'. In CP-ABE $\psi(u)$ is a singleton in $\mathcal{A}^{Att}$ ($\psi(u) \subseteq Att$) and $\psi(o)$ can be a general antichain ($\psi(o) \in \mathcal{A}^{Att}$).

The poset $(X, \leq)$ depicts role hierarchy, i.e. the usual subset (of attributes) inclusion relations. A much larger poset where each MAS is associated with a node is $(Auth, \sqsubseteq)$, see Def. 26. It includes the nodes of $(X, \leq)$ as a subset and maintains their poset relations. In addition it includes a node for each MAS that the system uses. In $(Auth, \sqsubseteq)$ a user u can access object o iff $\psi(u)$ dominates $\psi(o)$. In $(X, \leq)$, $\psi(o)$ is represented by a subset of nodes.

Lemma 31 *For any user u and object o, $\psi(u)$ dominates $\psi(o)$ in $(Auth, \sqsubseteq)$ iff $\psi(u)$ dominates at least one element of $\psi(o)$ in poset $(X, \leq)$.*

Proof. From the definition of the notation $\preceq_1$inside Lemma 25, and Definition 26, a singleton element s in $(Auth, \sqsubseteq)$ dominates a node $\vee r$ for $r = \{r_1, r_2, ...r_k\}$ (i.e. $\vee r \sqsubseteq s$) iff it dominates at least one element r_i in $(X, \leq)$ (i.e. $(\exists i,\ 1 \leq i \leq k)[r_i \leq s]$). ■

This Lemma means that we can work with poset $(X, \leq)$ instead of $(Auth, \sqsubseteq)$. The size of poset $(X, \leq)$ is upper bounded by a single exponential in m, the size of the attribute set, and the upper bound on the size of $(Auth, \sqsubseteq)$ is almost double exponential in m (see Sec. 2.2.1).