2012 Seventh International Conference on Availability, Reliability and Security

Enforcing Access Control in Virtual Organizations Using Hierarchical Attribute-Based Encryption

Muhammad Asim*†, Tanya Ignatenko*, Milan Petkovic*†, Daniel Trivellato† and Nicola Zannone†
*Philips Research, Eindhoven, The Netherlands
{muhammad.asim,tanya.ignatenko,milan.petkovic}@philips.com
†Eindhoven University of Technology, The Netherlands
{d.trivellato,n.zannone}@tue.nl

***Abstract*—Virtual organizations are dynamic, inter-organizational collaborations that involve systems and services belonging to different security domains. Several solutions have been proposed to guarantee the enforcement of the access control policies protecting the information exchanged in a distributed system, but none of them addresses the dynamicity characterizing virtual organizations. In this paper we propose a dynamic hierarchical attribute-based encryption (D-HABE) scheme that allows the institutions in a virtual organization to encrypt information according to a policy in such a way that only users with appropriate attributes can decrypt it. In addition, we introduce a key management scheme that determines which user is entitled to receive which attribute key from which domain authority.**

I. Introduction

The last decade has been characterized by the rise of a new operational paradigm where distributed systems and services collaborate to achieve a common goal. These collaborations, also known as *virtual organizations* [1], often consist of systems that belong to different security domains governed by different authorities, and are mostly dynamic, with systems joining and leaving the virtual organization on the fly.

While offering a high degree of operational flexibility and enabling new business models, the virtual organization paradigm has a strong impact on information security. In fact, parties in a virtual organization may be required to share a large amount of information. This information, however, might be sensitive and should only be accessed by authorized users and institutions. The access to sensitive information is usually regulated by access control policies, which specify users who can access what information. If information is confined within a single, trusted system, policy enforcement can be achieved using traditional enforcement mechanisms. When information needs to be disclosed across different security domains, guaranteeing policy enforcement becomes more challenging.

Cryptographic techniques are often deployed to enforce access control policies in a distributed system. In particular, attribute-based encryption (ABE) [2] enables the encryption of sensitive information according to a policy in such a way that only users with certain attributes (e.g., roles) can access the information. In ABE, attributes are certified by *key* (or *domain*) *authorities* (e.g., hospitals) which provide to their users an attribute (decryption) key for each attribute they possess (e.g., their role within the hospital). Besides having different roles, the users and institutions involved in a virtual organization are frequently organized in a hierarchical structure, which reflects the "chain of command" within the virtual organization. Hierarchical ABE (HABE) [3], [4] enhances ABE by reflecting the delegation mechanisms occurring in hierarchical domains, by allowing a domain authority to delegate its right to issue attribute keys to another (sub-)domain authority.

When applied to virtual organizations, the main limitation of existing HABE schemes is that they require binding the attributes in an access control policy to a specific domain authority at encryption time. Consequently, users of institutions that join a virtual organization at a later stage are not able to access previously encrypted information, even though they possess the appropriate attributes. Information need thus to be re-encrypted every time a new institution joins the virtual organization. Furthermore, HABE schemes implicitly assume existence of a mechanism that allows domain authorities to determine the attribute keys that their users are entitled to receive. In some cases this can be achieved, for instance, by simply issuing the keys according to the institution's user-role assignments. In other circumstances, however, the attributes of a user may depend on other attributes or conditions determined by third parties. As a result, current HABE schemes do not address the dynamics characterizing virtual organizations.

In this paper we propose a solution to the problem of distributed policy enforcement in virtual organizations, which combines cryptographic techniques with trust management [5]. In particular, we define:

- A dynamic HABE (D-HABE) scheme that does not require the encryptor to bind the attributes in an access control policy to a specific domain authority at the encryption time. This enables users of domain authorities that join a virtual organization at a later stage to decrypt the information that they are entitled to access, without the need of re-encrypting it.
- A key management scheme that determines which user

978-0-7695-4775-6/12 $26.00 © 2012 IEEE
DOI 10.1109/ARES.2012.78

is entitled to receive which attribute key from which domain authority.

The proposed D-HABE scheme is an extension of the CP-ABE scheme proposed by Bethencourt et al. [2]. Although subsequent CP-ABE schemes (e.g. [6]) have stronger security properties, as they are proved in the standard model, the original scheme [2], proved in generic group model and thus enjoying weaker security properties, is more efficient and expressive, since an access predicate can be expressed in terms of any monotonic formula over attributes. As in [2], we trade stronger security for efficiency and expressiveness and provide the proof in the generic group model, leaving the proof in the standard model for future work. Note also that the proposed scheme is efficient since the secret key components related to the set of attributes ω are $|\omega| + 1$, compared to $2 \cdot |\omega|$ in Bethencourt et al.'s scheme.

II. Construction of the D-HABE Scheme

In this section we present our D-HABE scheme. Before providing the formal definition, we outline its main idea. The root authority is assumed to be a trusted party that runs a setup algorithm in order to generate public parameters and a master secret key. Using these parameters and the master key, the root authority also generates secret keys for domain authorities. The level of a domain authority determines the number of parameters used to create its secret key. The secret key of a domain authority also contains the attributes for which the domain authority is entitled to issue secret keys. A domain authority generates secret keys for its users. Each user in a hierarchy is associated with an attribute set. Therefore, the secret key of a user relates to both the user's attributes and her level in the hierarchy.

An encryptor encrypts messages for users at a certain level in the hierarchy based on an access tree. The access tree is created from the policy by distributing a random secret parameter over the tree nodes that represent the attributes, using Shamir secret sharing. A user will only be able to reconstruct this parameter and thus satisfy the policy if she possesses the required attributes. Thus, a user will only be able to decrypt the ciphertext if her secret key corresponds to the correct level in the hierarchy and to the right attributes.

Note that our construction allows new domain authorities and users to join the hierarchy without any need to re-encrypt existing information, since the ciphertext is bound to a level in the hierarchy and a set of attributes, and not to a specific domain authority, user or secret key. This property is referred to as dynamic property of our HABE scheme. Note however the proposed scheme can bind ciphertext to a specific domain authority if required. In the following, we formally define the proposed D-HABE scheme.

A. D-HABE Scheme

Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \longrightarrow \mathbb{G}_1$ denote a bilinear map. A security parameter λ determines the size of the groups. We define the Lagrange coefficient $\triangle_{v,\Omega}(\kappa) = \prod_{v' \in \Omega, v' \neq v} \frac{\kappa - v'}{v - v'}$, for $\kappa, v \in \mathbb{Z}_p$ and Ω being a set of elements from $\mathbb{Z}_p$. Let $H : \{0,1\}^* \longrightarrow \mathbb{G}_0$ be a collision resistant hash function, where $\{0,1\}^*$ denotes a binary sequence of an arbitrary length. The function $H(\cdot)$ is a mapping of an attribute, described as a binary string, to a random group element in $\mathbb{G}_0$. Next, we describe the algorithms constituting the D-HABE scheme.

Setup(λ, L) This algorithm is run by the root authority to generate the system parameters for a hierarchy of depth L. We assume that at the first level of the hierarchy there are Ψ domains.[1] The algorithm selects a random generator $g \in \mathbb{G}_0$ and $\alpha, \beta \in \mathbb{Z}_p$, and sets $g_1 = g^\alpha$, $g_2 = g^\beta$, and $A = e(g,g)^{\alpha-\beta}$. In addition, it picks random elements $g_3, h_1, h_2, \ldots, h_L \in \mathbb{G}_0$ and $\mathcal{R}, y_1, y_2, \ldots, y_\Psi \in \mathbb{Z}_p$. The public key is set to be $PK = (g, g_3, h_1, h_2, \ldots, h_L, A)$ while the master secret key is set to be $MK = (g_1, g_2, \mathcal{R}, \mathcal{B} = \{y_1, y_2, \ldots, y_\Psi\})$.

Key Generation(MK, PK) This algorithm is run by the root authority to generate a secret key for a domain authority at level i $(1 \leq i \leq L)$ using the master secret key ***MK*** and public parameters ***PK***. It picks random value $r \in \mathbb{Z}_p$, $y_\psi \in \mathcal{B}$ and $y_\phi \in \mathbb{Z}_p$ that are unique for each domain authority and generates a key for this domain authority:

$$SK_i = \Big(g^\alpha \cdot \Big(g_3 \cdot \prod\nolimits_{l=1}^{i} h_l\Big)^r, g^r, h_{i+1}^r, \ldots, h_L^r, g^{\beta - \mathcal{R}y_\psi - y_\phi}, g^{y_\psi + y_\phi}, \forall j \in \Omega_{adm} : g^{\mathcal{R}y_\psi + y_\phi} H(j)^{y_\psi + y_\phi}\Big).$$

where Ω_{adm} represents the set of attributes for which a domain authority is eligible to issue secret keys. Here $y_\phi = 0$ if $i = 1$, otherwise this is just a randomly picked number.

A domain authority can use its secret key to generate secret keys for domain authorities beneath its level. In particular, the secret key SK_i for a domain authority at level i $(1 < i \leq L)$ can be generated in the incremental fashion given the secret key for a parent node in the hierarchy. Let SK_{i-1} be the secret key of this parent node:

$$\begin{aligned} SK_{i-1} = \; & \Big(g^\alpha \cdot \Big(g_3 \cdot \prod\nolimits_{l=1}^{i-1} h_l\Big)^{r'}, g^{r'}, h_i^{r'}, \ldots, h_L^{r'}, \\ & g^{\beta - \mathcal{R}y_\psi - y_{\phi'}}, g^{y_\psi + y_{\phi'}}, \\ & \forall j \in \Omega_{adm} : g^{\mathcal{R}y_\psi + y_{\phi'}} H(j)^{y_\psi + y_{\phi'}}\Big) \\ = \; & \big(a_0, a_1, b_i, \ldots, b_L, c_0, c_1, \\ & \forall j \in \Omega'_{adm} : g^{\mathcal{R}y_\psi + y_{\phi'}} H(j)^{y_\psi + y_{\phi'}}\big). \end{aligned}$$

To generate SK_i, the domain authority associated with the parent node picks randomly $r'' \in \mathbb{Z}_p$ and $y_{\phi''} \in \mathbb{Z}_p$ and outputs

$$\begin{aligned} SK_i = \; & \Big(a_0 \cdot b_i \cdot \Big(g_3 \cdot \prod\nolimits_{l=1}^{i} h_l\Big)^{r''}, a_1 \cdot g^{r''}, \\ & b_{i+1} \cdot h_{i+1}^{r''}, \ldots, b_L \cdot h_L^{r''}, c_0, c_1 * g^{-y_{\phi''}}, \\ & \forall j \in \Omega''_{adm} : (g^{\mathcal{R}y_\psi} H(j)^{y_\psi}) \cdot (g^{y_{\phi''}} H(j)^{\phi''})\Big). \end{aligned}$$

[1] Note that Ψ is not fixed during the lifetime of the virtual organization.

The resulting secret key SK_i is perfectly distributed for $r = r' + r''$ and $\phi = \phi' + \phi''$.

Attribute Key Generation(SK_i, PK, ω) This algorithm is run by a domain authority at level i $(1 < i \leq L)$ to generate secret keys for its users with an attribute set ω. First, the algorithm selects a random value $x \in Z_p$ for each user. The secret key for each user is then formed as

$$\begin{aligned} SK_{i,\omega} &= \Big(g^{\alpha} \cdot \Big(g_3 \cdot \prod\nolimits_{l=1}^{i} h_l\Big)^{r}, g^{r}, g^{\beta - \mathcal{R}y_{\psi} - x - y_{\phi}}, \\ &\quad \forall j \in \omega \subset \Omega_{adm} : D_j = g^{\mathcal{R}y_{\psi}+x+y_{\phi}} H(j)^{y_{\psi}+y_{\phi}}, \\ &\quad D' = g^{y_{\psi}+y_{\phi}}\Big). \end{aligned}$$

Encryption(PK, M, τ, i) This algorithm encrypts a message $M \in \mathbb{G}_1$ for users at level i under the access control policy specified by an access tree τ. The resulting ciphertext CT can only be decrypted by users at level i whose attribute set ω satisfies the access tree τ. Conceptually, CT consists of three components corresponding to: 1) the encrypted message, 2) a level i in the hierarchy, and 3) a set of attributes ω.

In order to encrypt the message according to the access tree τ, the encryption algorithm first selects a random value $s \in Z_p$ and uses Shamir's secret sharing to distribute this value among the leaf nodes of τ. Specifically, the algorithm chooses a polynomial $q_z(\cdot)$ for each node z in τ in a top-down manner, starting from the root node R. First, for each node z in the tree, it sets the degree d_z of the polynomial $q_z(\cdot)$ to be one less than the threshold value T_z of that node, i.e., $d_z = T_z - 1$. Then, starting with the root node R, the algorithm sets $q_R(0) = s$ and selects at random d_R other points of the polynomial $q_R(\cdot)$ in order to define the polynomial completely. For any other node z, the algorithm sets $q_z(0) = q_{\textit{parent}(z)}(\textit{index}(z))$ and selects the rest d_z points randomly to completely define $q_z(\cdot)$. Finally now the ciphertext CT is composed as follows:

$$\begin{aligned} CT &= \Big(M \cdot A^{s}, g^{s}, \Big(g_3 \cdot \prod\nolimits_{l=1}^{i} h_l\Big)^{s}, \forall K, K \in \tau : \\ &\quad C_{att(K)} = g^{q_K(0)}, C'_{att(K)} = H(att(K))^{q_K(0)}\Big) \\ &= \Big(\hat{C}, \hat{C}_0, \hat{C}_1, \forall K, K \in \tau : C_{att(K)}, C'_{att(K)}\Big). \end{aligned}$$

Decryption$(CT, SK_{i,\omega})$ The decryption algorithm consists of two steps: the first step verifies whether a user's attribute set ω satisfies policy τ, and the second step corresponds to the message recovery. The decryption algorithm uses the recursive algorithm $DecryptNode(CT, SK_{k,\omega}, z)$ to perform the first step. We define this algorithm first for (a) leaf nodes K and then for (b) internal nodes k of τ.

(a) $DecryptNode(CT, SK_{i,\omega}, K)$: Note that each leaf node is associated with a real-valued attribute. Let $j = att(K)$. Now, if $j \in \omega$ then

$$\begin{aligned} DecryptNode(CT, SK_{i,\omega}, K) &= \frac{e(D_j, C_j)}{e(D', C'_j)} \\ &= \frac{e(g^{\mathcal{R}y_{\psi}+x+y_{\phi}} H(j)^{y_{\psi}+y_{\phi}}, g^{q_K(0)})}{e(g^{y_{\psi}+y_{\phi}}, H(j)^{q_K(0)})} \\ &= e(g,g)^{(\mathcal{R}y_{\psi}+x+y_{\phi})q_K(0)}. \end{aligned}$$

If $j \notin \omega$, then $DecryptNode(CT, SK_{i,\omega}, K) = \perp$, where $\perp$ denotes failure.

(b) $DecryptNode(CT, SK_{i,\omega}, k)$: For all nodes z that are children of k, the algorithm calls $DecryptNode(CT, SK_{i,\omega}, z)$. Its output stored as F_z is used to determine whether the user has enough attributes to satisfy the policy. Note that to satisfy the policy, there should be enough points (i.e., satisfied child nodes) to reconstruct the polynomial in node k and thus $q_k(0)$. Let Ω_k be an arbitrary T_k-sized set of child nodes z such that $F_z \neq \perp$, $\forall z \in \Omega_k$. If there exists no such a set, then node k is not satisfied and the function returns $\perp$. Otherwise, using polynomial interpolation, the algorithm evaluates the following function:

$$\begin{aligned} F_k &= \prod_{z \in \Omega_k} F_z^{\triangle_{v,\Omega_k}(0)}, \quad where \;\; v = \textit{index}(z) \\ &= \prod_{z \in \Omega_k} \Big(e(g,g)^{(\mathcal{R}y_{\psi}+x+y_{\phi}) \cdot q_z(0)}\Big)^{\triangle_{v,\Omega_k}(0)} \\ &= e(g,g)^{(\mathcal{R}y_{\psi}+x+y_{\phi}) \cdot q_k(0)} \end{aligned}$$

To decrypt the ciphertext CT, the decryption algorithm first checks if the user satisfies the access control policy. This is done by evaluating the $DecryptNode(\cdot)$ function on the root node R of the access tree τ. If $DecryptNode(CT, SK_{i,\omega}, R)$ returns $\perp$, then τ is not satisfied by the attribute set ω of the key $SK_{i,\omega}$. In this case, decryption is impossible and the function returns $\perp$. Otherwise τ is satisfied, and the decryption algorithm performs the following steps. First, it computes the following intermediate steps

$$\begin{aligned} Z^{(1)} &= DecryptNode(CT, SK_{i,\omega}, R) \\ &= e(g,g)^{(\mathcal{R}y_{\psi}+x+y_{\phi})s}, \\ Z^{(2)} &= e\left(g^{\beta - \mathcal{R}y_{\psi} - x - y_{\phi}}, \hat{C}_0\right) \cdot Z^{(1)} \\ &= e(g,g)^{\beta s}, \\ Z^{(3)} &= \frac{e\left(g^{s}, g^{\alpha} \cdot \left(g_3 \cdot \prod_{l=1}^{i} h_l\right)^{r}\right)}{e\left(g^{r}, \left(g_3 \cdot \prod_{l=1}^{i} h_l\right)^{s}\right)} \\ &= e(g,g)^{\alpha s}. \end{aligned}$$

Note that the correct value of $Z^{(3)}$ can only be recovered by users at the right level in the hierarchy.

Finally (assuming the user's key satisfied τ and corresponds to the right level of hierarchy, otherwise the decryption algorithm returns $\perp$), the message M is recovered as:

$$\begin{aligned}\hat{C} \cdot \frac{Z^{(2)}}{Z^{(3)}} &= M \cdot A^s \cdot \frac{e(g,g)^{\beta s}}{e(g,g)^{\alpha s}} \\ &= M \cdot e(g,g)^{(\alpha-\beta)s} \cdot e(g,g)^{-(\alpha-\beta)s} \\ &= M\end{aligned}$$

III. Security Proof

A. Security Model

In this section we define the security game for D-HABE between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$. We refer to this game as the D-HABE security game.

Setup: The challenger $\mathcal{C}$ runs the *Setup* algorithm and gives adversary $\mathcal{A}$ the public parameters, while keeping the master secret key to itself.

Phase 1: $\mathcal{A}$ performs a polynomially bounded number of queries of the following types:

- *Type 1:* $\mathcal{A}$ asks for a user secret key from a domain authority at level i for attribute set $\omega_1, \omega_2, \cdots, \omega_Q$. The challenger returns secret keys $\mathsf{SK}_{i,\omega_\gamma}, \forall \gamma \in \{1, 2, \cdots, Q\}$ to $\mathcal{A}$.
- *Type 2:* $\mathcal{A}$ asks for a user secret key from a domain authority at level $\hat{i}, \hat{i} \neq i$, for attribute set $\omega_1, \omega_2, \cdots, \omega_Q$. The challenger returns $\mathsf{SK}_{\hat{i},\omega_\gamma}, \forall \gamma \in \{1, 2, \cdots, Q\}$ to $\mathcal{A}$.

Challenge: In this phase the adversary $\mathcal{A}$ submits two equal length plaintexts M_0 and M_1 from a message space, on which $\mathcal{A}$ wants to be challenged. Moreover, $\mathcal{A}$ also gives the challenger an access structure $\mathbb{A}^*$ such that the queried secret keys from Phase 1 do not satisfy $\mathbb{A}^*$. The access structure encompasses both part of the hierarchy up to a certain level and the access tree τ over ω. The challenger flips a random coin $b \in \{0, 1\}$ and returns the encryption of M_b under $\mathbb{A}^*$ to the adversary $\mathcal{A}$.

Phase 2: Repeat Phase 1 querying for the secret keys that do not satisfy $\mathbb{A}^*$ and that have not already been queried for in Phase 1.

Guess: In this phase, $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b' = b$. The advantage of the adversary in attacking the scheme is $|Pr[b' = b] - \frac{1}{2}|$.

Definition 1: The D-HABE scheme is secure if all polynomial time adversaries have at most negligible advantage in the D-HABE security game.

B. Security Proof

The security of the D-HABE scheme can be proved using arguments similar to those in [7], [8], [2]. We use the generic group model and the random oracle model to argue that there is no efficient adversary who can break the security of our scheme with non-negligible probability if the adversary acts generically on the groups used in our scheme. This means that, if there are any vulnerabilities in the scheme, then they are due to specific mathematical properties of elliptic curve groups or cryptographic hash functions used in our constructions. In the generic group model, group elements are encoded into unique random strings, in such a way that the adversary $\mathcal{A}$ can manipulate group elements using canonical group operations in $\mathbb{G}_0$ and $\mathbb{G}_1$ and cannot test any property other than equality. The following theorem gives a lower bound on the advantage of a generic adversary $\mathcal{A}$ in breaking our scheme.

Theorem 1: Let q be an upper bound on the total number of group elements that an adversary $\mathcal{A}$ can receive from queries she makes to the challenger $\mathcal{C}$ for elements from the hash function $H(\cdot)$, groups $\mathbb{G}_0$, $\mathbb{G}_1$, bilinear map $e(\cdot, \cdot)$, and from his interaction in the D-HABE security game. The advantage of the adversary in the security game is $O\left(q^2/p\right)$. The proof of Theorem 1 is presented in [9].

IV. Augmenting D-HABE with Attribute Key Management

The access trees used to protect the information exchanged in a virtual organization are determined by the information' access control policies. In the next two subsections we address the issues of (a) how to derive the access tree corresponding to an access control policy (Section IV-A), and (b) how to determine which user is entitled to receive which attribute key (Section IV-B). Then, we discuss how to integrate these solutions with the proposed D-HABE scheme (Section IV-C).

A. From Access Control Policies to Access Trees

Existing (H)ABE schemes implicitly assume access control policies to be specified in the format required to encrypt information (e.g., access trees). In the access control frameworks for distributed systems proposed in the literature (e.g., [10], [11]), however, policies are mostly specified in logic programming-based languages. Here, we show how to translate such access control policies into the corresponding access trees. Conceptually, the translation of the access control policy of a data object o into the access tree of o is performed in two steps:

1) The rules in the access control policy are transformed into (conjunctions of) *attribute certification chains*. An attribute certification chain consists of an attribute that the user needs to possess to access o, followed by a sequence of domain authorities (and domain authority "classes") that denotes a path in the hierarchy of a virtual organization.
2) The attributes identified in step 1 are combined into a logical formula reflecting the original access control policy. The formula represents the access tree of o.

Before formalizing the translation process, we present a formalization of access control policies. An access control policy is a set of rules of the following form

canRead(U,O) $\leftarrow$
certifies(RA,C_{11},DA_{11}), …, **certifies**(DA_{1n_1},A_1,U),
…
certifies(RA,C_{m1},DA_{m1}), …, **certifies**(DA_{mn_m},A_m,U)

where U can be a specific user or a variable representing any user satisfying the policy conditions, O is the data object that the access control policy protects, each $A_1, \ldots, A_m$ are attributes, each $C_{\ell\dot{p}}$ (with $1 \leq \ell \leq m$ and $1 \leq \dot{p} \leq n_\ell$) is a domain authority class (e.g., hospital, clinic), RA is the root authority and $DA_{\ell\dot{p}}$ is a domain authority or a variable representing any domain authority of class $C_{\ell\dot{p}}$. Intuitively, a rule states that a user U can read object O if he has attributes $A_1, \ldots, A_m$; each attribute A_ℓ is derived from a certification chain involving domain authorities $RA, DA_{\ell 1}, \ldots, DA_{\ell n_\ell}$ of class $C_{\ell 1}, \ldots, C_{\ell n_\ell}$. Since we are dealing with hierarchical domains, the first domain authority RA is always the root authority (e.g., VWS in our scenario).

We are now ready to formalize the translation from access control policies to access trees which consists of two steps:

1) Let $AR_o = \{AR_1, \ldots, AR_n\}$ be the set of rules protecting a data object O. From AR_o we derive the corresponding set of (conjunctions of) attribute certification chains $ACC_o = \{ACC_1, \ldots, ACC_n\}$ as follows. Let AR_ℓ ($1 \leq \ell \leq n$) be

   canRead(U,O) $\leftarrow$
   $\bigwedge_{1 \leq \dot{p} \leq m} \Big($**certifies**($RA$,$C_{\dot{p}1}$,$DA_{\dot{p}1}$), …,
   certifies($DA_{\dot{p}n_{\dot{p}}}$,$A_{\dot{p}}$,U)$\Big)$

   Then ACC_ℓ is

   $\bigwedge_{1 \leq \dot{p} \leq m} \Big(A_{\dot{p}}$ @ $RA \rightarrow C_{\dot{p}1} = DA_{\dot{p}1} \rightarrow \cdots$
   $\rightarrow C_{\dot{p}n_{\dot{p}}} = DA_{\dot{p}n_{\dot{p}}}\Big)$

2) Given the set of attribute certification chains $\{ACC_1, \ldots, ACC_n\}$, the D-HABE access tree HAT_o of O is constructed as follows:

   $$\bigvee_{1 \leq \ell \leq n} \Big(T_{1n_1} \wedge A_1 \wedge \ldots \wedge T_{mn_m} \wedge A_m\Big)$$

   where $T_{\dot{p}n_{\dot{p}}}$ (with $1 \leq \dot{p} \leq m$) is $C_{\dot{p}n_{\dot{p}}}$ if $DA_{\dot{p}n_{\dot{p}}}$ is $*$, and $T_{\dot{p}n_{\dot{p}}}$ is $DA_{\dot{p}n_{\dot{p}}}$ if $DA_{\dot{p}n_{\dot{p}}}$ is a domain authority.

Consider the following example access control policy protecting the Electronic Health Record (EHR) of a patient John: *John's EHR may be accessed by: (a) the doctor of GP practice gp_2 (John's family doctor); (b) any hospital doctor who has a treatment relationship with John; (c) any clinic doctor who has a treatment relationship with John; and (d) first-aiders of any hospital recognized by the Dutch Ministry of Health (VWS).* This policy can be seen as the disjunction of four rules, denoted by (a), (b), (c), and (d). Fig. 1 shows the attribute certification chains of the policy. Then, Step 2 transforms the attribute certification chains into the formula "*(gp_2 $\wedge$ doctor) $\vee$ (hospital $\wedge$ doctor $\wedge$ treating doctor) $\vee$ (clinic $\wedge$ doctor $\wedge$ treating doctor) $\vee$ (hospital $\wedge$ first-aider)*", which corresponds to the access tree of John's EHR. Notice that the attributes appearing in an access control policy and in the corresponding access tree must be elements of the set of attributes included in the public parameter PK of the D-HABE scheme.

(a)	*doctor @ VWS $\rightarrow$ GP practice $= gp_2$*
(b)	*doctor @ VWS $\rightarrow$ hospital $= *$ $\wedge$*
	treating-doctor @ VWS $\rightarrow$ hospital $= *$
(c)	*doctor @ VWS $\rightarrow$ clinic $= *$ $\wedge$*
	treating-doctor @ VWS $\rightarrow$ clinic $= *$
(d)	*first-aider @ VWS $\rightarrow$ hospital $= *$*

Figure 1. Attribute Certification Chains Derived from the Example Policy

B. Issuing Attribute Keys

This section presents an attribute key management scheme that relies on trust management techniques [5] for determining the attribute decryption keys that a user is entitled to receive. Trust management is an approach to attribute-based access control in distributed systems. A trust management policy specifies in which conditions a domain authority certifies that a given user or domain authority has a certain attribute, where policy conditions are in turn represented in terms of attributes certified by (possibly different) domain authorities. Formally, the trust management policy of a domain authority DA consists of a set of rules of the form

certifies$(DA, A, E) \leftarrow$ **certifies**(DA_1, A_1, E_1), …,
certifies(DA_n, A_n, E_n)

where DA, DA_ℓ (with $1 \leq \ell \leq n$) are domain authorities, E, E_ℓ are users or domain authorities, and each A_ℓ is an attribute. Policy rules may also have an empty conditions set (i.e., $n = 0$, in which case the keyword **if** is omitted); we refer to these rules as *credentials*. The following are examples of policy rules and credentials defined by a hospital $h1$ and the hospital's cardiology department:

certifies$(h1, doctor, X) \leftarrow$ **certifies**$(h_1, department, Y)$,
certifies$(Y, doctor, X)$
certifies$(h1, department, cardiology)$
certifies$(cardiology, doctor, Alice)$

Intuitively, the first rule states that $h1$ certifies as doctor any doctor working in its departments. The other two rules state respectively that $cardiology$ is a department of hospital $h1$, and that $Alice$ is a doctor working in that department.

The problem of determining which user is entitled to which attribute keys can thus be reduced to the problem of determining which credentials can be derived from the trust management policy of the domain authorities in a virtual organization. *Credential chain discovery algorithms* [10] provide a solution to this problem. Given a query $\dot{q}$ of the form **certifies**(DA, A, E)? and a set of trust management policy rules TMR, credential chain discovery algorithms compute the answers of $\dot{q}$ that satisfy TMR. For instance, given the policy rules above, the answer returned

by a credential chain discovery algorithm to the query **certifies**$(h1, doctor, X)$? would be **certifies**$(h1, doctor, Alice)$. Domain authorities can therefore rely on credential chain discovery algorithms to derive the set of attributes that the users within their institution possess, and release the corresponding attribute keys.

C. Unified Scheme

The proposed key management scheme can be easily integrated with the D-HABE scheme to form a complete framework for the enforcement of access control policies in virtual organizations. The integration can be achieved in three steps:

1) *Setup of the D-HABE infrastructure*: the root authority of the virtual organization initiates the D-HABE scheme by running the setup algorithm and releasing the secret keys to the domain authorities at level 1 of the hierarchy. Then, in turn, each domain authority releases a secret key to the domain authorities beneath its level.
2) *Translating access control policies into attribute trees*: the access control policies protecting the data objects that need to be exchanged in the virtual organization are translated into the corresponding attribute trees. Each entity can perform this process independently. As mentioned in Section IV-A, the attributes appearing in access trees must be a subset of the attributes included in the public parameter PK of the D-HABE scheme.
3) *Issuing the attribute keys*: each domain authority runs the credential chain discovery algorithm to determine the attributes of its users. The domain authority then issues the attribute keys of its users accordingly, using the attribute key generation algorithm.

Note that step 2 can be executed independently from steps 1 and 3, while step 3 must be executed after step 1, since the attribute key generation algorithm depends on the setup and key generation algorithms. Once these three steps are executed, the users and institutions in the virtual organization can exchange information (encrypted with the corresponding access control tree) with the guarantee that only authorized users at a specific level in hierarchy can access it.

V. Conclusions and Future Work

This paper presents a solution to the problem of distributed policy enforcement in virtual organizations. In particular, it presents a new dynamic HABE (D-HABE) scheme that addresses the dynamics of these collaborations. Furthermore, the paper makes an important link between attribute-based encryption schemes and trust management, which is proposed as a mean to determine the attribute keys to be issued.

The work presented in this paper suggests some interesting directions for future research. First of all, the proposed scheme does not address the problem of accountability for key disclosure. More precisely, a domain authority may create another domain authority at the same level in the hierarchy through the re-randomization of its secret key. In addition, a user may disclose her keys (e.g., by publishing them on the Internet) without fear of being caught as there is no linkability established between the key and the user. As a future work we would also like to provide the security proof of the proposed scheme in the standard model where the problem of breaking the scheme is reduced to a well-studied complexity-theoretic problem.

Acknowledgments

This work is funded by the Dutch national program COMMIT through the THeCS project and by the European Commission through the FP7 universAAL project (nr. 247950).

References

[1] C. Handy, *Trust and the virtual organization*. Harvard Business School Press, 1999, pp. 107–120.

[2] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. of S&P'07*. IEEE Computer Society, 2007, pp. 321–334.

[3] J. Li, Q. Wang, C. Wang, and K. Ren, "Enhancing attribute-based encryption with attribute hierarchy," *MONET*, vol. 16, no. 5, pp. 553–561, 2011.

[4] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. of CCS'10*. ACM, 2010, pp. 735–737.

[5] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized Trust Management," in *Proc. of S&P'96*. IEEE Computer Society, 1996, pp. 164–173.

[6] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography*, 2011, pp. 53–70.

[7] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. of EUROCRYPT'97*, 1997, pp. 256–266.

[8] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proc. of EUROCRYPT'05*. LNCS 3494, Springer, 2005, pp. 440–456.

[9] M. Asim, T. Ignatenko, M. Petkovic, D. Trivellato, and N. Zannone, "Enforcing access control in virtual organizations using hierarchical attribute-based encryption," arXiv: 1205.5757, 2012.

[10] N. Li, W. H. Winsborough, and J. C. Mitchell, "Distributed credential chain discovery in trust management," *Journal of Computer Security*, vol. 11, no. 1, pp. 35–86, 2003.

[11] M. Y. Becker, "Cassandra: flexible trust management and its application to electronic health records," Ph.D. dissertation, Computer Laboratory, University of Cambridge, UK, 2005.