Institut Eurécom
Department of Corporate Communications
2229, route des Crêtes
B.P. 193
06904 Sophia-Antipolis
FRANCE

Research Report RR-08-219

A Distributed Access Control Framework For XML Document Centric Collaborations

April 25$^{\text{th}}$, 2008

Mohammad Ashiqur Rahaman, Yves Roudier, and Andreas Schaad

Tel : (+33) 4 93 00 81 00
Fax : (+33) 4 93 00 82 00
Email : {rahaman,Yves.Roudier}@eurecom.fr

[1]Institut Eurécom's research is partially supported by its industrial members: BMW Group Research & Technology - BMW Group Company, Bouygues Télécom, Cisco Systems, France Télécom, Hitachi Europe, SFR, Sharp, STMicroelectronics, Swisscom, Thales.

A Distributed Access Control Framework For XML Document Centric Collaborations

Mohammad Ashiqur Rahaman, Yves Roudier, and Andreas Schaad

Abstract

Collaboratively working on documents within a distributed context is a non-trivial task, in particular if neither a centralized access control policy enforcement platform nor a centralized document repository can be assumed to be present. Decoupling the specification of the access control policy of documents from its later autonomous enforcement can make it easier to edit documents in a decentralized yet secure fashion.

This paper introduces a distributed and fine grained access control framework for XML document centric collaborations. The framework addresses the authenticity, confidentiality, integrity, and traceability of circulated documents and their updates. It is fully distributed in that each participant can enforce and verify these security properties without relying on a central authority.

Novel aspects of the proposed framework include the adoption of a decentralized key management scheme that provides support for the cryptographic enforcement of a credential based access control policy. This scheme is driven by the access interests expressed by the participants over document parts. A protocol for the controlled edition of a document is finally introduced based on these techniques.

Index Terms

XML documents, collaborative edition, access control in distributed and mobile systems, cryptographic protocols

Contents

List of Figures

1 Introduction

Documents are an increasingly central concern in today's digital inter-organizational exchanges and collaboration processes, as illustrated by the multiplication of XML standards for instance. They have evolved from primitive forms and static collections of information, as used by humans, to complex, detailed, and ever-evolving descriptions destined to automated processing and data interchange. This increasing complexity has been accompanied by the need to handle detailed security policies defining all possible accesses to fine grained parts of documents. Such documents are composite in that they are originated by multiple responsible authorities in charge of their own portion of the document and of ruling who may edit or read it. In addition, the document edition process is becoming increasingly collaborative with participants exchanging documents arbitrarily depending on context evolution and on their mobility or churn rate. Contrary to centralized document repositories, which have been vastly studied (see for instance [4,10,11,14,16,20,23]) and which provide a central location for enforcing access control on a per request basis, such a distributed setting pleads for new models of access control in which access control specification is asynchronously decoupled from its enforcement.

Selective access to and collaborative updates of fine grained documents are challenging tasks in that context. We develop in this paper a collaboration framework and its associated protocols in order to support the cryptographic enforcement of fine grained and multi-authority access control. Our approach in particular aims at restricting document access through the encryption of its parts with keys shared by groups of participants with similar access rights. Our scheme intends at limiting the scope of document re-encryptions and key redistribution needs when participants are added or removed. Furthermore, once a document is circulated one participant should be able to verify the authenticity of the source of the document and of the structural and content-wise integrity of updates so that the contents of the documents are not altered during their transmission from one participant to the other. Finally, a participant may need to trace the updates that other participants performed over the document, in case one of them transgressed his update rights.

The rest of this paper is organized as follows. Section 2 discusses the scope of the XML document centric collaborations we address and describes the security objectives of our proposed distributed access control framework. Section 3 then introduces a motivating scenario featuring a collaborative edition of documents. Section 4 specifies the XML document model retained and in particular details the document labeling scheme. Section 5 gives an overview of the proposed distributed fine grained access control framework. The specification of the access pattern descriptions used in this framework is introduced in Section 6. Document authorization and access control policy specification are sketched in Section 7. Section 8 elaborates on the key management issues. Section 9 introduces security meta data which then encompasses the secure document envelope. Section 10 then describes the different phases and operations of controlled document edition.

Section 11 finally presents related work and points out the significant differences of our work.

2 Scope and Objectives

Assuming a distributed deployment of collaborative edition of XML documents, the system will exhibit specific properties due to the lack of a dedicated security infrastructure, as opposed to a centralized framework for XML documents:

1. *Distributed Document Sources.* Multiple participants will author documents simultaneously. This first makes it necessary for a common document schema to be agreed upon by all participants. However, documents are instantiated by the participants during collaborative work as opposed to being instantiated by a centralized source. We assume that different document parts have distinguished authorities (or *owners*) who may also participate in the collaboration.
2. *Document Distribution.* No central repository should be assumed to be available. Document editors are supposed to send new or updated documents to other participants in push mode if no storage infrastructure is available, otherwise they may store documents at a central repository devoided of any security role and independent from the owner.
3. *Distributed Document Access.* One participant may need to access document parts that are managed by an authority at any time, even if that authority is unavailable, provided the document is being circulated to another available participant.
4. *Fine Grained Document Access.* Access patterns may range from the whole document to an individual element. Conversely, documents being composite, the access rights distributed by multiple authorities will bear on a subset of the document only.

We consider that attackers might intercept communications between the participants, and also that some participants might modify parts of the document in order to gain some advantage out of the modification like rendering the document invalid, or changing its semantics or destination. However, we do not consider resilience to denial of service attacks other than through sending a document to all participants.

Compared with a centralized access control framework, one cannot rely on a central entity as authorized participants can autonomously circulate and access document parts from other participants. This makes it necessary to consider additional authorization and protection issues on documents.
Document Authorization:

1. Access Decision Delegation: For scalability reasons, access control decisions cannot be performed in centralized fashion, especially in scenarios involving virtual organizations or multiple authorities.

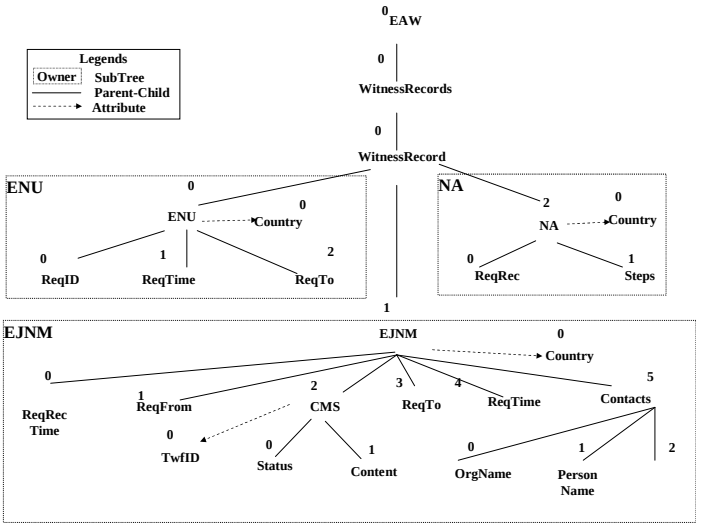

Figure 1: The agreed EAW document Schema

2. Distributed Control of Data Disclosure: Only authorized participants should get access to the protected document parts, yet access control enforcement should not rely on a central authority.

Document Protection:

1. Document Confidentiality: Selected document contents should only be disclosed to the authorized participants.
2. Document Authenticity: Upon receiving a document participants should verify that they receive the document from an authentic source.
3. Content-wise and Structural Integrity: Participants should be able to verify the content-wise and structural integrity of a received document with respect to the original document as malicious participants may circulate invalid documents which are not a part of the original document.
4. Traceability of Document Access: All access by participants should be tracked if possible to check if the past access performed were authorized. Our proposal addresses update access only, in particular to ensure that the update of a document part was performed in conformance with the authorization policy of its authority.

3 Motivating Example

This section introduces an example of the collaboration taking place between two European Union (EU) administrative bodies (Europol and Eurojust), and associated 27 member states authorities [7]. Europol and Eurojust have their representatives (Europol National Member and Eurojust National Member) for 27 member

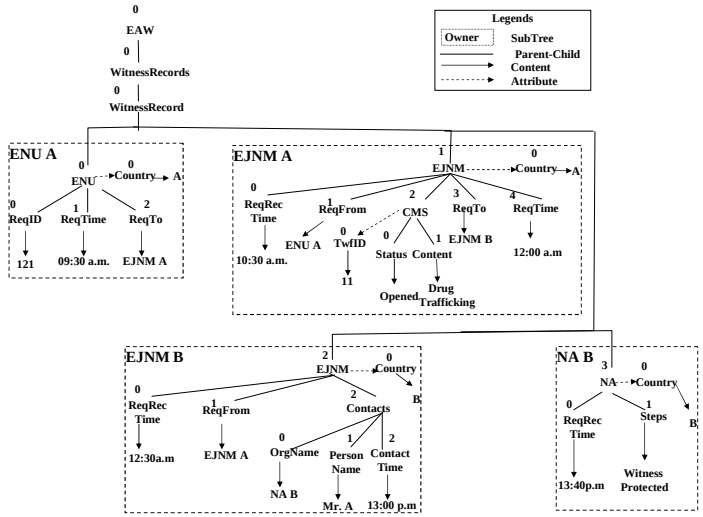

Figure 2: EAW document instance whose parts are instances of ENU A, EJNM A, EJNM B and NA B

states. Each member state has its national contact points (National Authority) for Europol and Eurojust. Europol, Eurojust, associated law enforcement authorities and their employees (hereafter called participants) collaborate whenever there is an occurrence of cross border organized crime in the EU: this entails a request for Mutual Legal Assistance (MLA).

In this scenario, participants collaboratively define and work on a document called European Arrest Warrant (EAW). The MLA scenario is structured as follows:

1. A Europol National Unit of country A (ENU A) makes a written request of assistance (for a witness protection) to a Eurojust National Member of country A (EJNM A).
2. The EJNM A opens a Temporary Work File (Twf) in a local Case Management System (CMS).
3. The EJNM A contacts Eurojust National Member of country B (EJNM B) by forwarding the request of assistance.
4. The EJNM B contacts the responsible national authority of country B (NA B). Steps are taken by the responsible NA B to provide the requested assistance.

The MLA scenario depicts a clear scenario of collaborative activities on different parts of the EAW document. This scenario mirrors the mentioned properties and requirements of Section 2.

Distributed Sources, Autonomy, and Access Policy. The different parts of the EAW document are structured according to the local regulations and policies of the authorities of the EU. For example, one country may require to know the religious belief of a suspect. On the contrary, the disclosure of one's religious belief

may be prohibited by law in another country which compels instantiating different document instances and having different access policies over that.

Distributed and Fine Grained Access. As the crime is cross border, various sensitive information is possessed by different law enforcement authorities. EJNM B employee may need to access a deeply nested element containing useful information that is owned by ENU A.

Information Push. An information push model is observed in the MLA scenario, the EAW document being circulated among different participants during collaboration as updates are performed. Upon receipt, one participant may get access to the document parts of the EAW.

Figure 1 shows a EAW document schema that is agreed among the participants and it distinguishes the ownership of different parts of EAW schema by dotted rectangles around subtrees. ENU is the owner of the subtree rooted at `ENU`. Similarly EJNM and NA are the owners of the subtrees rooted at nodes `EJNM` and `NA` respectively. Figure 1 also shows the labels for every element and attribute of the EAW schema.

4 Modeling Document Access Patterns

In this section, we present our document labeling scheme, which is partly inspired by [21], [3], and [12], and how it can be used to specify document access patterns.

4.1 Preliminaries

Definition 1 *Document: A Document, d, is a tree of XML elements and attributes where each node represents either an element or an attribute. d is associated with a set of Owners, O_d, and each node in d is assigned a label based on a labeling scheme L_d (Section 4.2). Formally: $d = (V_d, \hat{v_d}, O_d, L_d)$, where:*
–$V_d = V_d^e \cup V_d^a$ is a set of XML elements and attributes respectively;
–$\hat{v_d}$ is the root element of the document;
–O_d represents the set of Owners;
–$L_d = \{l : l = Label(n)$, where $n \in V_d$ and $Label(n)$ is a labeling scheme$\}$;

According to Definition 1, a Document is a labeled tree where nodes represent elements and attributes, and edges (assumed) represent the hierarchical relationship among them. The tree (Figure 1) contains edges representing the element-sub element (solid lines), element-attribute (dotted arrows), element-content (solid arrows), and attribute-value (solid arrows) relationships. Each node is associated with a non-negative integer label.

Definition 2 *Participants: P_d represents a set of participants, P_i, i.e. owners of document parts and interested readers and writers who take part in the collabora-*

tive activities on a document d. Formally:

$$P_d = \{\ P_i{:}P_i \in (O_d \cup \neg O_d)\ \}$$

Here, $\neg O_d$, refers to the participants that are not owners but have certain access interest (interested participants) on a document d. Owners may also have access interests on other document parts. For the MLA scenario, the participants are all the employees of ENU A, EJNM A, EJNM B, and NA B.

Definition 3 *Document Owners: Document Owners, O_{d_i}, of a document, d, represent a set of $Participants$ who are agreed to design the document schema collaboratively, and specify access control policies on their respective $Document\ Parts$, d_i (Definition 4) autonomously.*

By definition there can be a number of owners but for simplicity we consider only one owner for each document part. In Figure 2, the EAW document instance has four owners ($O_{ENUA}, O_{EJNMA}, O_{EJNMB}, O_{NAB}$) and each of them owns $d_{ENUA}, d_{EJNMA}, d_{EJNMB}, d_{NAB}$ document parts respectively.

Definition 4 *Document Part: A Document Part, d_i, is a subtree of a Document, d, rooted at $\hat{d_i}$, and is owned by an Owner O_i. Formally: $d_i = (n_i, \hat{d_i}, O_i, L_{n_i})$, where:*
–$n_i \subseteq V_d$ is a sub set of XML elements and attributes of d;
–$O_i \in O_d$ represents the owner of d_i;
–$L_{n_i} = \{l : l = Label(n)$, where $n \in n_i\}$;

A Document part may represent a part of schema or a document instance part. In Figure 2, EAW document instance has four document part instances rooted at `ENU` (d_{ENU}), `EJNM` (d_{EJNM}), `EJNM` (d_{EJNM}), `NA` (d_{NA}) respectively.

4.2 Document Labeling

Defining a fine grained access on a XML document means performing access at the element or attribute level. We define a labeling scheme, comparable with [1]: a non negative integer is assigned to each node of the tree when the tree is traversed in breadth first fashion from the root node. For a document d with root element $\hat{v_d}$, and $n \in \mathbb{N}$:

$$Label(X) = \begin{cases} 0 \quad \text{if X = } \hat{v_d}; \\ C \quad \text{if X} \in V_d; \\ \text{where n is the number of child nodes of } V_d; \\ 0 \leq C \leq (n-1) \text{ and } \forall\ l_1, l_2 \in \mathbb{N}\ ((l_1 = Label(X_i)) \\ \wedge\ (l_2 = Label(X_{i+1})) \rightarrow l_2 > l_1); \\ \text{for } 0 < i < n \text{ and } X_{i+1} \text{ is the next pre-order} \\ \text{node of } X_i. \end{cases}$$

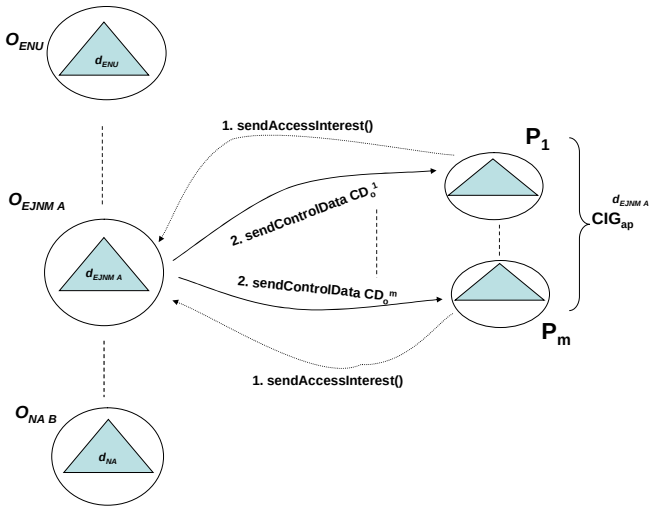

Figure 3: Message Exchange in Initiation Phase

Figure 1 and Figure 2 show the labels for the EAW document schema and an EAW document instance respectively. Each child of a node (including the root) is labeled in pre-order with C, where $0 \leq C \leq n-1$.

Several document modeling techniques have been proposed which are based on labeling schemes. [9] proposed a scheme based on a k-ary tree with no more than k children for each node. It is inefficient when k is large and [1] proposed another scheme addressing this issue. The latter scheme is based on prefix labeling, where the label of a node A is a prefix of the label of node B then A is an ancestor of B. Likewise, B is a descendant of A. However, in this approach, the labels become long because of skew [1] in the tree.

Dynamic changes in the document tree can be easily captured by the labeling scheme. Considering NAB sends the steps information taken by it to ENUA, it might add a new subtree rooted at `SendTo` under the node `NA`. In this case, `SendTo` would get a label 2 without affecting the existing labels of the tree (Figure 1).

5 Framework Overview

In this section, we present the distributed document access control framework designed to address the security requirements of Section 2. We give an overview of the framework in light of the collaborative work of EAW document. This framework features a two phase controlled edition protocol.

Based on the participants' knowledge of the schema of the EAW document, each of them can instantiate an empty document initially. We assume ENU A, EJNM A, EJNM B, NA B and their employees are the participants, P_d and EJNM A is the owner O_{EJNMA} of a part d_{EJNMA} of the EAW document d. Initially, all the interested participants express their access interests to O_{EJNMA} through the use of access primitives (see Section 6.3).

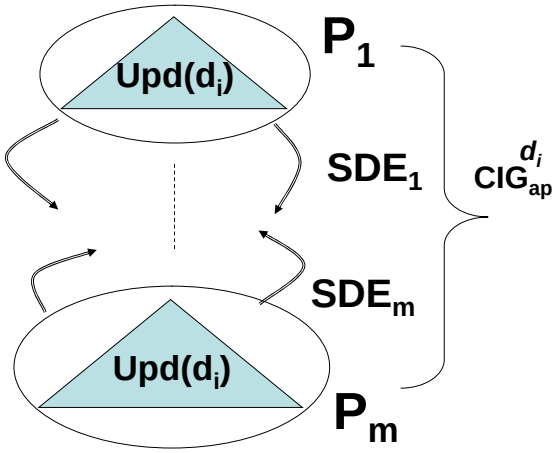

Figure 4: Message Exchange in Collaboration Phase

Upon receiving all access interests, O_{EJNMA} evaluates them with respect to its access control policy (see Section 7.1) and determines the common interest group (see Definition 6) $CIG_{ap}^{d_{i=EJNMA}}$. Figure 3 shows message exchanges in the initiation phase of the framework.

The owner O_{EJNMA} can compute a common secret key independently after it determined $CIG_{ap}^{d_i}$ based on the list of EAIs. O_{EJNMA} distributes control information to $P_i \in CIG_{ap}^{d_i} = \{$ ENU A employees, EJNM B employees, NA B employees $\}$ using which P_i can compute the common secret key (Section 8.2). This distribution ends the initiation phase of the edition protocol.

The collaboration phase of the edition protocol, which consists of the actual access to the document, can then start. Each participant P_i now computes the same common secret key $CK_{ap\in\{V,A,D,R\}}^{d_{EJNMA}}$ (see Section 8.3) after receiving the control information. After generating the encrypted document and meta data block with the common secret key, the participants can push the updated document part (see Section 10) towards other participants; this part is protected through the use of an adapted envelope SDE as described in Section 9.1. Any participant with the knowledge of the common secret key can access the received protected document part autonomously and verify that the updates performed result from a legitimate access as captured in that document envelope. Figure 4 shows message exchanges in the collaboration phase of the framework.

In the proposed framework, the access control enforcement and verification is completely decoupled from the access control policy specified by the owner O_{EJNMA}. O_{EJNMA} is only involved in determining the CIG members and in distributing the corresponding control information. The later enforcement of access control and the protection of the document part are independently handled by the participants in the collaboration.

In the sequel of the paper, $\mathcal{M}$ and $\mathcal{K}$ denote the message and key space respectively. h_1 denotes one way hash function. h_M denotes the Merkle hash function [2]. The encryption and signature of a message $m \in \mathcal{M}$ with a key $K \in \mathcal{K}$ is written as $[m]_K$ and $Sign_a(m) = [h_1(m)]_{K_a}$ (where K_a is the private key of a) respectively.

6 Access Interest Specification

This section describes how participants interested in accessing some document part can express access patterns on the data structure. To this end, the framework introduces a language for expressing access interests. The following introduces the specification of such Expressions of Access Interest (EAI), which are used to describe the targets in a document that are to be controlled, and related access primitives (View, Append, Delete, Rename).

6.1 Expression of Access Interest (EAI)

An EAI e over the document d is an expression of labels and operators that refer to a set of nodes $N \subseteq d$, defined as follows:

$$e = \begin{cases} \epsilon \mid * \mid (e_1 + e_2) \mid (e_1\ e_2) \mid e_1[e_2 \mid \rho] \mid e(); \\ \text{where } e1 \text{ and } e_2 \text{ are labels; } \rho = e_1 \oplus e_2; \\ \text{and } \oplus \in \{=, \neq, <, >, \leq, \geq, \in, \ni, \vee, \wedge, \subset, \subseteq, \supset, \supseteq\}. \end{cases}$$

ϵ and $*$ define empty and wild card expressions respectively. (e_1+e_2) and (e_1e_2) define choice and concatenation expressions respectively. Finally, $e_1[e_2 \mid \rho]$ and $e()$ define conditional expression and refers to the content of a node having the label e respectively.

Definition 5 *Valid EAI: Given a document tree d, Valid EAI is an EAI over d if it allows a traversing from the root node to a node having the final label of EAI.*

According to this definition each node $n \in d$ is referable by a Valid EAI. Formally –$\forall$n ($V_d(n) \rightarrow \exists$e $ivEAI(e, n)$); where n is any XML node and e is any Valid EAI. Here, $V_d(n)$ and $ivEAI(e, n)$ are two predicates stating $n \in V_d$ and e is a Valid EAI over n respectively.

Example 1 $0001*$*, 00012[0] are valid EAIs whereas 00015 is not a valid EAI (Figure 2). Let us consider few Access Interests of different employee participants of: EJNM A: (1) To view* `ReqID, ReqTime` *from ENU A. (2) To view* `Country` *attribute from ENU A. EJNM B: (1) To view* `TwfID, ReqTime, Status` *from EJNM A. (2) To copy* `ReqID, ReqTime` *from ENU A. (3) To copy the* `CMS` *from EJNM A. NA B: (1) To view* `ReqID, ReqTime` *from ENU A. (2) To view* `ContactTime` *from EJNM B. (3) To view* `ReqTime` *of EJNM A. ENU A: (1) To copy* `Steps` *from NA B.*

The access interests of Example 1 span from a *view access interest for an attribute to a* copy access interest over a subtree. Moreover, employees of EJNM B and NA B have similar access interest (View) over `ReqTime` of the document part d_{EJNMA}.

6.2 Advantages of EAI

The EAIs based on the labeling scheme are the means for managing access interests through access primitives. Access interests of an organization may change due to changes in regulations: for example, the new regulation of NA B requires to retrieve the `CMS` information from EJNM A instead of getting information from ENU A. The specification of an EAI allows participants to capture this regulation change. EAIs offer other advantages:

- EAIs can be realized by X-Path [8] or X-Query [5] and are implementation independent. Note that X-Path or X-Query can be used to refer to the target nodes of a document. However, they are tightly bound to an XML document instance. Two documents having a similar tree structure but different identifiers have to employ different X-Path expressions, whereas a valid EAI can be used for both documents.
- The EAI is expressive enough to specify different parts of a document from the coarser grained form (i.e. the whole document) to the finest granularity level (i.e. the deepest nested element or attribute in a document). For example, 00012 refers to the node `CMS` (Figure 1) and EAI $000*$ refers to the subtree rooted at node `WitnessRecord` (Figure 1). A flexible (using prefix, suffix) EAI is possible. For example, $0001[0]$ refers to the attribute `Country` of the node `EJNM` (Figure 1).

6.3 Access Primitives

An access interest specification for the framework is based on a set of access primitives and allows participants to specify various access interests over the different document parts. We extend [22] primitives by replacing its centralized X-Query based centralized enforcement with a distributed encryption based enforcement. Participants define unique public keys associated with its access interest, which we term access keys (Section 8.1), are passed as parameters of the various primitives and serve as principal identifiers (the use of these keys is further explained in the following sections). An access primitive takes $Valid$ EAIs ('targetEAI' and 'sourceEAI'), a propagation value, and the interested participant's access key as inputs, and outputs a subset of a *Document Part*. The 'targetEAI' and propagation are evaluated on the owner's document part. The propagation input takes a non-negative integer value, n or +. If it is $n \geq 0$ or + then the access interest is also propagated toward the n-th descendant nodes (elements and its attributes) or

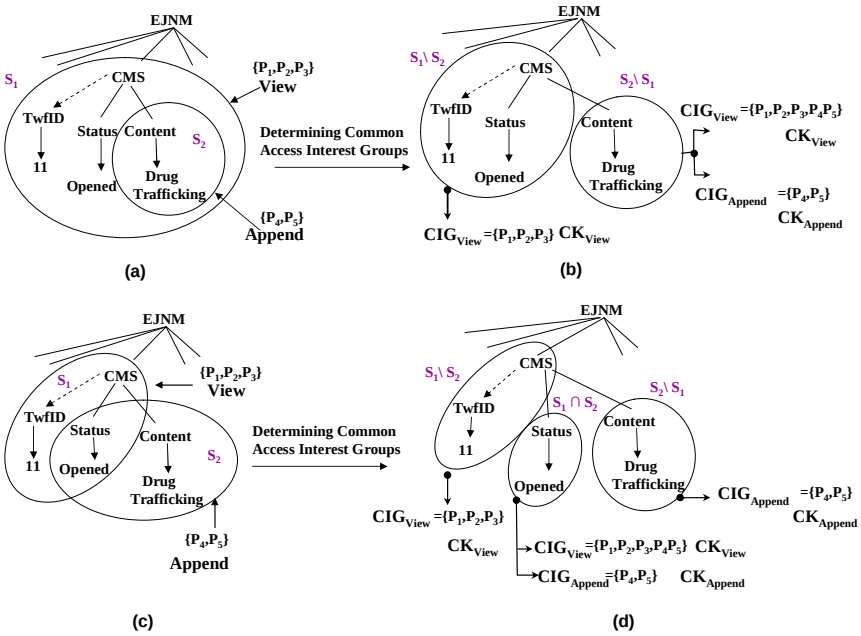

Figure 5: Determining common access interest groups considering view primitive subsumes append. (a) S_2 is subsumed by S_1. (b) Three common access interest groups are determined with two disjoint sets of nodes. (c) S_2 is partly subsumed by S_1. (d) Four common access interest groups are determined with three disjoint sets of nodes.

the whole subtree respectively. Intuitively enough, propagation 0 means that the access interest is not propagated toward the descendants.

1. **View(key, targetEAI, [Propagation])**. The **View** primitive returns the nodes of the document part that matches the valid 'targetEAI'. For a propagation value of 0, only the matching node with the '$targetEAI$' without the descendant nodes is returned.

2. **Append(key, targetEAI, newNode, [Propagation])**. The **Append** primitive creates a new node (i.e. element,attribute) with the name 'newNode' as a child node of each matching node of the valid 'targetEAI'. If the propagation value is 0 only the first matching node is considered.

3. **Delete(key, targetEAI, [Propagation])**. The **Delete** primitive deletes the nodes rooted at the matching valid 'targetEAI'. The deletion is performed either up to the n-th descendants of the matching node or the whole subtree from that node.

4. **Rename(key, targetEAI, newName[ ], [Propagation])**. The **Rename** primitive renames the nodes of the document parts matching the valid 'targetEAI'. It is propagated either down to the n-th descendant nodes of the matching node or down the whole subtree rooted at the matching 'targetEAI'. Each

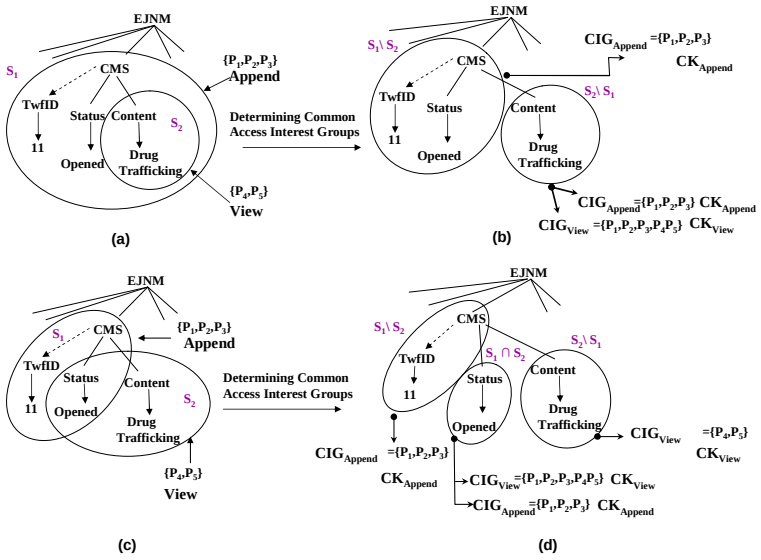

Figure 6: Determining common access interest groups considering append primitive subsumes view. (a) S_2 is subsumed by S_1. (b) Three common access interest groups are determined with two disjoint sets of nodes. (c) S_2 is partly subsumed by S_1. (d) Four common access interest groups are determined with three disjoint sets of nodes.

propagation of the access primitive renames the corresponding descendant node with a new name from the list 'newName[ ]'.

The Append, Delete, and Rename are generally called update primitives hereafter. Update primitives implicitly imply the View primitive to the nodes they apply to. Based on these primitives, there might be other composite operations like for example, Copy(key, sourceEAI, targetEAI, [Propagation]) and Move(key, sourceEAI, targetEAI, [Propagation]). Copy creates an exact subtree up to n-th descendants of the document parts rooted at the node matching the valid 'sourceEAI'. The created subtree is then appended as a child of the nodes matching the valid 'targetEAI'. Intuitively, it uses the Append primitive. Move does exactly the same operation like Copy and in addition it deletes the subtree matched by the 'sourceEAI'.

Definition 6 *Common Interest Group (CIG): Given a set of access primitives,* $ap \in \{V(iew), A(ppend), D(elete), R(ename)\}$ *on a document part* d_i*,* $CIG_{ap}^{d_i}$*, is a set of public access keys* PK_{ap}^{i}*, that defines a set of participants who have same access interest (i.e. access primitives with same '*$targetEAI$*' and Propagation) and satisfy the access policy of the authority for* d_i*. Formally:*

$$CIG_{ap}^{d_i} = \begin{cases} PK_{ap}^{i} \text{ for } i \in [1,z] \text{ } z \text{ participants; where} \\ (\forall ap_{i\in[1,z]} \exists (j,k) \in [1,z] \wedge j \neq k) \\ ap_j = ap_k \text{ for } z \text{ access primitives.} \end{cases}$$

7 Document Authorization

An owner has two roles for the authorizations on the collaborative XML documents. First, to specify the access control policy for the document part it owns. Second, to determine the common access interest group to initiate the collaboration.

7.1 Access Control Policy

Owners define access control policies to regulate access to their respective parts of a document. Access interests from other participants are evaluated against these policies. An access control policy can be complex enough, but for simplicity we chose a simple credential based model to show how policy evaluation may occur.

Definition 7 *Credential Specification: a Credential Specification, $\mathcal{CS}$, consists of any combination of the following:*
–participant identifier, P_i;
–participant access key, PK^i_{ap};
–$validEAI$ [1];
–if $cs_1, cs_2 \in \mathcal{CS}$, then $cs_1 \wedge cs_2$ and $cs_1 \vee cs_2$ are credential specifications.

In this approach, participants to which an access policy applies are implicitly referred by a set of conditions in a credential specification. Each participant may have one or more associated credentials. These credentials may be grouped within signed certificates in order to prevent malicious participants from supplying fake credentials, certificates may be used that would consist of signed credentials.

Example 2 *The following are examples of credential specifications in $\mathcal{CS}$:*
–(EJNM B employee $\vee$ NA B employee): this denotes participants who are EJNM B Employee or NA B employee;
–(NA B employee $\wedge$ $validEAI$): this denotes the NA B employee who has an access interest with $validEAI$.

Definition 8 *Policy: A Policy pol_i of the owner O_i of document part d_i is a pair consising of a credential C and of a reference N to a set of nodes in the document. Formally:*
– $pol_i = (C, N)$; where $C \in \mathcal{CS}$ and$N \subseteq d_i$

Example 3 *The following are examples of policies of O_{EJNMA} over d_{EJNMA}:*
–$(EJNM \quad B \quad employee \vee NABemployee, CMS)$: this policy states any employee of EJNM B or NA B can get an access to CMS.
–$(NA \quad B \quad employee \quad \wedge \quad validEAI)$: this policy states that any NA B employee who has an access interest described by access pattern $validEAI$ is authorized to get access to $N = validEAI$.

[1] This essentially represents nodes of a document part d_i that are requested for access.

7.2 Determining a Common Access Interest

After evaluating all the access interests, the authority determines a disjoint set of common access interest groups, $CIGs$ with respect to the 'targetEAI's of all the access primitives. We assume the authority is a member of every group it is managing. The determination of the disjoint set of $CIGs$ is described as follows:

Let us assume two access primitives $ap1$ and $ap2$ from P_1 and P_2 containing $targetEAIs$ e_1, e_2 respectively refer to the two subtrees S_1 and S_2 of the document part d_i and O_i is the authority of d_i.

If S_1 and S_2 are disjoint, meaning $S_1 \cup S_2 = null$, then e_1 and e_2 do not overlap. P_1 and P_2 are assigned to two disjoint sets of common access interest groups $CIG_{ap1} = \{P_1\}$ and $CIG_{ap2} = \{P_2\}$ respectively.

If any subtree S_2 is either (1) entirely subsumed by the other S_1, or (2) partly subsumed by the other S_1, then some overlapping occurs between e_1 and e_2. Determining the disjoint set of common access interest groups in this case proceeds as follows. Regarding case (1), two disjoint subtrees of nodes are determined: one with the subsumed subtree S_2 and the other with $S_1 \backslash S_2$. Regarding case (2), three disjoint subtrees of nodes are determined: one with $S_1 \backslash S_2$, the second with $S_1 \cap S_2$ and the last with $S_2 \backslash S_1$. Each disjoint subtree is associated with an access primitive accordingly.

Update ($U = ap \in \{A, D, R\}$) primitives generate two different groups CIG_{U_i} and CIG_{V_i} while View ($V = View$) primitives require only one group CIG_{V_i} to be formed. Note that any participant having an Update access interest may need to be a member of multiple groups because of the implicit granting of a view access and of the overlapping of different access interests:

- In case (1), assuming that e_1 and e_2 respectively refer to view and update primitives, the disjoint groups formed are: $CIG_{V_{1\backslash 2}} = \{P_1\}$, $CIG_{V_2} = \{P_2\}$ and $CIG_{U_2} = \{P_2\}$. If on the contrary e_1 and e_2 respectively refer to update and view primitives, the disjoint groups formed are now: $CIG_{V_{1\backslash 2}} = \{P_1\}$, $CIG_{U_{1\backslash 2}} = \{P_1\}$, $CIG_{V_2} = \{P_1, P_2\}$ and $CIG_{U_2} = \{P_1\}$.
- In case (2), assuming that e_1 and e_2 respectively refer to view and update primitives, the disjoint groups formed are: $CIG_{V_{1\backslash 2}} = \{P_1\}$, $CIG_{V_{1 \cap 2}} = \{P_1, P_2\}$, $CIG_{U_{1 \cap 2}} = \{P_2\}$, $CIG_{V_{2\backslash 1}} = \{P_2\}$ and $CIG_{U_{2\backslash 1}} = \{P_2\}$. In contrast, if e_1 and e_2 respectively refer to update and view primitives, the disjoint groups formed are: $CIG_{V_{1\backslash 2}} = \{P_1\}$, $CIG_{U_{1\backslash 2}} = \{P_1\}$, $CIG_{V_{1 \cap 2}} = \{P_1, P_2\}$, $CIG_{U_{1 \cap 2}} = \{P_1\}$ and $CIG_{V_{2\backslash 1}} = \{P_2\}$.

Figure 6 depicts case (1) and (2) considering view and append primitives for S_1 and S_2 respectively from P_1, P_2, P_3 and P_4, P_5.

8 Key Management

A participant may have different access interest (View, Append, Delete, Rename) depending on its collaboration needs which they convey to the authorities

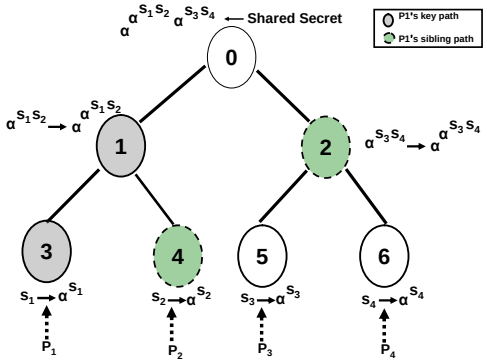

Figure 7: Tree-based Group Diffie-Hellman (DH) key agreement for 4 participants P_1, P_2, P_3, P_4. The participants host their DH values in the leaves. The notation $k \rightarrow \alpha^k$ means that participants compute the DH private value and then compute the DH public value $BK = \alpha^k$ and broadcast it.

by sending access primitives. An authority needs to identify the participants based on their signature in the access primitives. In effect the participants may not know other participants who have the same access interest yet they want to compute a common secret key keeping their privacy. In the case of unavailability of the authority, participants should be able to act as delegate. As a result participants need means using which they will be able to not only compute common secret keys of the groups they are in but also to be a delegate dynamically.

This section introduces three different keys and their management to achieve those. First, **participant key** pair (PK_z, SK_z) associated with each participant P_z to relate them with credentials as described in Section 7.1 and is not discussed further. Second, **access keys** associated with each access primitive identifies particular access interest of a participant. Finally, a **common secret key** defines each common access interest group.

8.1 Access Key Generation

Each participant P_i possesses a set of **access key** pairs: $(SK^i_{ap},\ PK^i_{ap})$ for each access primitive. Based on a new unique private access key SK^i_{ap}, the participant generates his corresponding public access key using DH [13] protocol where arithmetics are performed in a group of prime order p with generator α.

$$PK^i_{ap} = \alpha^{SK^i_{ap}} \mod p \tag{1}$$

Participants P_i send signed (using the private key SK_i) access primitives including the corresponding public access key PK^i_{ap} as a parameter (see Section 6.3) to the authority A_i.

$$P_{i\in[1,n]} \xrightarrow{Sign_{P_i}(AccessPrimitive())} A_i$$

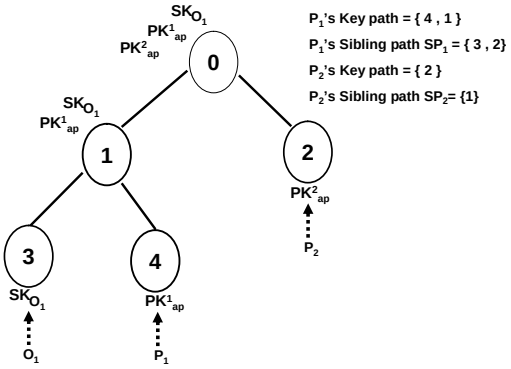

Figure 8: Owner O_i's key tree with two participants P_1 and P_2. O_i computes the sibling paths for P_1 and P_2.

8.2 Control Data Block Distribution

Participants do not know others with the same access interest yet they have to compute a common secret key. After determining the common access interest groups the authority takes the charge of building a control data block CD_{A_i} containing information for common secret key computation for each member of a group it manages. This block consists of a set of individual blocks $CD_{A_i}^{z\in[1,m]}$ destined to the m members of group $CIG_{ap}^{d_i}$ and defined as follows:

$$CD_{A_i}^{z\in[1,m]} = [SP_z]_{PK_z}$$

SP_z is the sibling path containing a list of public DH values which participant P_z requires to compute its key-path. The number of such required values being variable with the participant but always smaller or equal to number m of participants in that group and larger than or equal to $log(m)$ in case of a balanced tree. This also prohibits one participant to identify other members in the group and thus they remain anonymous to him.

The authority finally encrypts each individual block $CD_{A_i}^z$ with the public key PK_z of every participant P_z as determined from the submitted credential and signed requested access pattern and sends it afterwards.

$$A_i \xrightarrow{CD_{A_i}^z} P_{z\in CIG_{ap}^{d_i}}$$

Knowledge of the control data block enables each participant to compute the common secret key of its respective groups and act as a delegate afterwards. Such a message cannot be intercepted to gain access to protected documents since each individual block is encrypted with authorized member's public key.

8.3 Common Secret Key Management

In a distributed environment like the EAW scenario (Section 3), one cannot assume the presence of any centralized entity for computing and distributing the

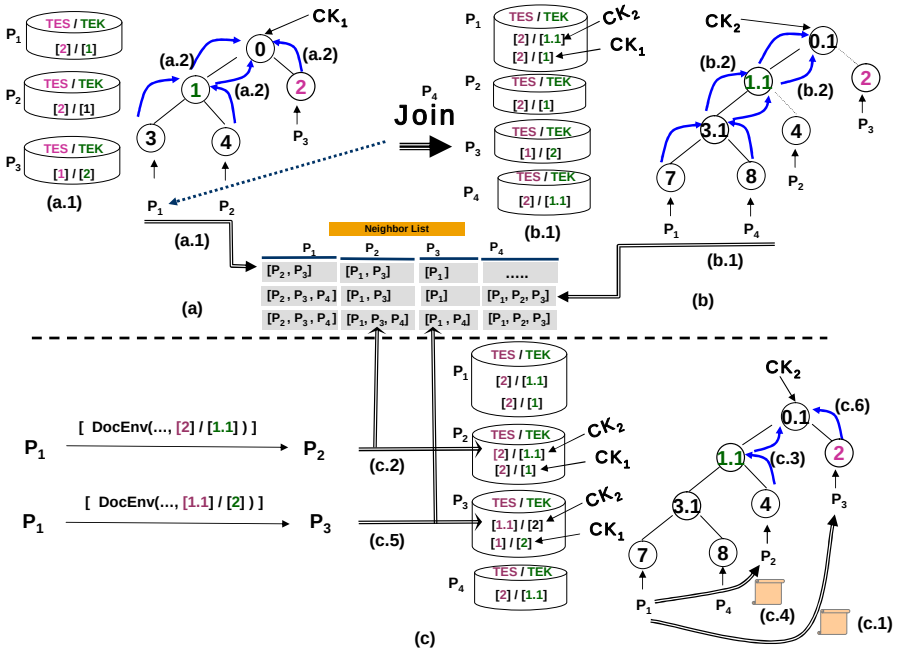

Figure 9: Lazy rekeying. (a) P_1, P_2 and P_3's key-paths with two data structure (i.e. TES/TEK, Neighbor List). (b) after P_4 joins to P_1. (c) Lazy rekeying of P_2 and P_3 after receiving document envelope from P_1. DocEnv(..) is the secure document envelope (Section 9.1).

common secret key. Even if such an entity were available, it would constitute a single point of failure thereby rendering the system vulnerable. In contrast, the owner (original authority) takes the charge of initializing the group collaboration by exploiting the key tree structure of TGDH. In particular, the owner generates a key tree by providing its DH private value in one leaf node and taking other participants' DH public values (i.e Public access keys (PK^i_{ap})) one by one as other leaf nodes. In the process of such bottom-up computation of the DH private values in its key-path, a common secret is computed for the root node.

In effect, Every node in the key tree is assigned a unique number v, starting with the root node that is assigned 0: the two child nodes of a non-leaf node v are set to $2v + 1$, and $2v + 2$ respectively. Each node v is associated with a key pair consisting of a DH private value K_v and of a DH public value BK_v, relying on the hardness of solving the discrete logarithm. For every node v, K_v is computed recursively as follows:

$$K_v = \begin{cases} \text{if } v \text{ is a non-leaf node;} \\ (BK_{2v+1})^{K_{2v+2}} \mod p \\ = (BK_{2v+2})^{K_{2v+1}} \mod p \\ = \alpha^{K_{2v+1}K_{2v+2}} \mod p \\ \text{if } v \text{ is a leaf node;} \\ SK^i_{ap}. \end{cases}$$

In short, computing the DH private value K_v of a non-leaf node requires the knowledge of the DH private value of one of the two child nodes and the DH public value of the other child node. In effect, one participant only needs to compute the DH private values along its key-path. In other words, one participant only needs to know the DH public values of the siblings of the nodes of its key-path (sibling path). Therefore, the value K_0 computed for the root is the secret for all the participants (including owner). At this point, the common secret key is derived from the shared secret as follows: $CK^{d_i}_{ap} = h_1(K_0)$.

To illustrate this, taking two participants' $PK^{i\in[1,2]}_{ap}$ the owner O_1 builds the key tree in Figure 8. Once the key-tree is generated the owner can determine the sibling path values $SP_{z\in[1,2]}$ required for each participant in the group which it sends to them as part of control data block.

The owner being initializer of a group collaboration does not make the framework a centralized one as all the participants need to compute the common secret key along their key-paths by themselves. Moreover, the computation of the shared secret is still contributory [15] in nature as the owner takes public access keys of all participants as the leaf nodes of the logical key tree to compute the common secret key and thereby to compute the SP_z. Furthermore, this adopted scheme has a two fold advantage. First, participants can compute the common secret key neither generating the complete key tree nor identifying other participants in the group, which is essential with respect to document centric exchanges. Second, group membership scales as described in [15, 17].

Definition 9 *Protected Document Part d^e_i: Given a common interest group $CIG^{d_i}_{ap}$ with a 'targetEAI' e over a document part d_i, any participant in the group can build a protected document part d^e_i by encrypting all nodes $N \in e$ with the common secret key $CK^{d_i}_{ap}$ while other nodes of d_i (i.e. $d_i \backslash N$) are left unchanged.*

According to the definition any participant having the common secret key $CK^{d_i}_{ap}$ is able to get $ap \in \{V, A, D, R\}$ access to $N \in e$ of d^e_i. The document owner is assumed to originally distribute a protected document in which all subtrees are protected with appropriate common secret keys as determined by the set of access interests it received and in which nodes without any interest expressed may either be left unencrypted or encrypted with the owner's access key depending on its access control policy.

8.4 Lazy Rekeying

Managing dynamically joining and leaving participants requires updating the common secret key. Lazy rekeying refers to the act of re computation of a new common secret key by a participant only when it requires to do so. This will take place when interacting with a participant that knows about a different version of the group, which may happen upon receiving a document envelope.

Definition 10 *Neighbors: A participant P_i's neighbors is a list of participants who provide their DH public values in order to compute the DH private values along the key path of P_i.*

Definition 11 *Top End Key-path Value (TEK) and Top End Sibling-path Value (TES): A participant P_i's TEK is the computed DH private value associated with the top most node along its key path and TES is the received DH public value associated with the top most node along its sibling path.*

According to these definitions, in Figure 7 P_1 and P_2 are neighbors to each other and so do P_3 and P_4. The DH values of nodes 1 and 2 are the TEK and TES for P_1, P_2 respectively and the DH values of nodes 2 and 1 are the TEK and TES of P_3, P_4 respectively. In other words, neighbors have exactly the same TEK and TES for a common access interest group.

It can be observed from a participant's point of view that any dynamic change in its neighbors incurs an update in its key-path and similarly any dynamic change in its non-neighbors incurs an update in its sibling path. In particular, incurred dynamic changes cause new DH values to be computed in corresponding key paths and sibling paths that are accumulated in TEK and TES respectively.

Lazy rekeying relies on the usage of Neighbor List and the pair TES/TEK maintained by each participant in a group where TES/TEK values are piggybacked with the secure document envelope (further described in Section 9.1). The usage of Neighbor List and TES/TEK is as follows:

- Neighbor list is a history of neighboring participants with which P_i is collaborating.
- P_i updates its neighbor list and TEK only when acting as a delegate for a joining/leaving event or receiving a secure document envelope containing a new TEK value indicating there has been a change in its neighbor list. P_i updates its TES only when it receives a document envelope containing a new TES value meaning there is a dynamic change in the key-paths associated with it's TES. The TES/TEK being piggybacked merely adds simple value in the envelope that makes it suitable for a scalable system .

As group membership changes (further described in Section 8.5), initial re computation is performed only for the key-paths associated with the current authority and the participant that is subject to join or leave. As such the authority

and the subject participant can immediately compute the new common secret key along their key-paths. The pair TES/TEK also contains the subject participant's key (not shown in the Figure 9) so that recipient can update its neighbor list accordingly. At this point both can either exchange previous document updates to the existing group members or perform new updates in documents and then exchange new document updates to the current group members including/excluding the subject participant. For the former, the secure document envelope is piggybacked with previous TES/TEK whereas for the latter the new TES/TEK is piggybacked.

To illustrate this in Figure 9, P_1, P_2 and P_3 having their same original authority O_i (not in the figure) maintain their initial Neighbor List as (P_2, P_3), (P_1, P_3) and (P_1) respectively (a.1). P_1 and P_2's TES/TEK as [2]/[1][2] and P_3's TES/TEK as [1]/[2] (a.1). All of them have computed the common secret key CK_1 (a.2).

Now P_4 joins with the delegate authority P_1 and P_1, P_4 can compute their new key-paths and update their corresponding Neighbor List as (P_2, P_3, P_4) and (P_1, P_2, P_3) (b.1). P_1 and P_4 update their TES/TEK with the new value of [2]/[1.1] (b.1). At this point P_1 and P_4 can compute the new common secret key CK_2 (b.2). However, P_2 and P_3 are unaware about this joining event and thus do not know the identity of P_4 at all.

Now if P_1 being a neighbor sends a secure document envelope piggybacked with updated TES/TEK [2]/[1.1] to P_2 which will then notice that there has been a change in its neighbor and thus in TEK by comparing its TES/TEK with the received one (c.1). P_2 then updates its TES/TEK (c.2) and computes the new common secret key CK_2 in order to decrypt the document envelope (c.3). Similarly P_3 can update its neighbor and TES/TEK and compute the key CK_2 if P_1 sends a secure document envelope piggybacked with updated TES/TEK [1.1]/[2] to P_3 (c.4,c.5,c.6). Note that the sent TES/TEK is inversed for P_3 with compare to P_2 as P_3's TEK is P_2's TES in the key tree.

Other members still can collaborate with their previous knowledge of common secret keys without stopping the collaboration as they do not even notice the changes occurred by the join/leave event. However, they will be noticed when they receive a secure document envelope containing new piggybacked information and can re compute the new common secret key utilizing the received TES/TEK. Note that the dynamic changes in the group is neither broadcasted nor requested yet available participants continue collaboration with previous common secret key.

8.5 Joining and Leaving

The authority may delegate its access decision among the participants it is managing so a group of participants with the same access interest might be updated dynamically even when the owner (initial authority) is unavailable. While the participants may only know their closest authority they do not know other participants who have similar access interests yet they want to compute a common

[2]The labeled integer value of a key tree node represents the corresponding DH values.

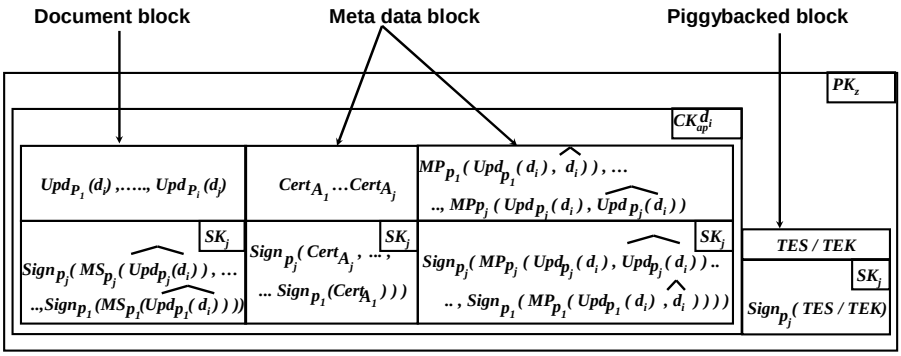

Figure 10: Secure Document Envelope.

secret key. We now assume that the control data block CD_{A_i} introduced in Section 8.2 contains additional information, in particular the description of access decision delegations and a description of the access control policy rules that apply to the document part whose access is granted to group members, as follows:

$$CD_{A_i}^{z\in[1,m]} = [SP_z, Cert_{A_1}..Cert_{A_i}, SecObj]_{PK_z}$$

($Cert_{A_1}..Cert_{A_i}$) denotes a chain of certificates originated from the owner O_i (i.e. A_1) to the participant P_z for an access primitive $ap \in \{V, A, D, R\}$. Each certificate $Cert_{A_i} = Sign_{A_i}(PK_z, PK_{ap}^z)$ asserts that the authority A_i authorizes P_z to perform the access ap over the document nodes of d_i (i.e. 'targetEAI'$\in d_i$). The first certificate in the chain being from the owner enables a participant to be a delegate which then also may add its certificate in the chain delegating further. This certificate then can be used as a proof to other participants of $CIG_{ap}^{d_i}$ that P_z was entitled to access d_i. This can be also used to trace that P_z has performed the updates on d_i.

'SecObj' defines security objectives, i.e., access control policy rules relevant for the 'targetEAI', like for instance the fact that the data referred to in the 'targetEAI' should be reserved to the German police. Based on the chain of certificates and the objectives, the participant, acting as a delegate for the owner, take over the access decision related tasks of the owner in the interest specification phase, and can evaluate the requested access.

A new participant sends its access primitives to an authority P_r it knows just as described in Section 8.1. P_r being a delegate evaluates the new participant's request and determines its eligibility to becoming a new member of an existing group (or to create a new group). P_r re-computes its key path taking the new member's access key into account and sends control information to the new members as described in Section 8.2.

The access control policy might additionally specify whether backward secrecy applies to the new participant, which should be described in 'SecObj'. If it does, the new member can only start document exchanges using the new common secret key from that point on. Otherwise, the authority sends the previous *n* common

secret keys to the new member P_j so that it can observe the previous updates and collaborate on these if possible.

$$P_r \xrightarrow{[CK_1 ... CK_i .. CK_n]_{PK_j}} P_j$$

Note, that the other members may not collaborate on document updates performed with the new key right way after new members join and thus do not re compute the new common secret key. This means that available members recompute the corresponding common secret key in lazy fashion as described in Section 8.4 and depicted in Figure 9.

In case of a voluntary leave, the participant sends its associated certificate $Cert_{A_i} = Sign_{A_i}(PK_z, PK^z_{ap})$ in similar fashion to the direct authority P_r it joined before.

$$P_{i \in [1,m]} \xrightarrow{[Cert_{A_i}]_{PK_j}} P_{j \neq i \in [1,m]}$$

P_r deletes the leaving member node from its key path and recomputes its new key-path. More often, the participant's authority will decide on his group members' leave. If forward secrecy applies P_r immediately sends a secure document envelope piggybacked with new TES/TEK values to the available members of the groups werein the leaving participant was a member of so that they can re compute the new common secret key. Then available members recompute the new common secret key and collaborate further using it. Otherwise, the other members recompute only when it receives a secure document envelope from other participants in the group. In case the direct authority is unavailable the subject participant may inform to the next indirect authority accordingly that it knows from the certificate chain of the received control data block.

9 Document-Related Security Metadata

As mentioned earlier, the scheme described above makes no special assumption regarding how participants interact. In particular, we target scenarios in which only documents would be exchanged, possibly only on top of an asynchronous messaging scheme like email for instance. In that context, a document should piggyback all security metadata related to its content and data structure as well as to the correctness of its updates so far. It should also carry the necessary security metadata making it possible for the receiving participant to decide whether to rekey as explained above in Section 8.4 and which key to use to decrypt the various document parts. This section describes the secure document envelope data structure that carries such security metadata, their use for document protection and how to use them for a navigation through the protected document.

9.1 Secure Document Envelope

Document parts may be arbitrarily exchanged between and modified by an authority/editor of any authorized participant. This requires ensuring the authenticity and integrity of the data exchanged, even though the documents may be passed through third-parties like unauthorized participants or a node on the communication network. The Merkle Tree authentication mechanism [19] used for instance in [2] to produce a Merkle Signature out of a static XML document addresses such issues. A unique digital signature can be applied at the root node of the document to ensure both its authenticity and integrity as a whole. The collaborative edition process iteratively modifies document fields, therefore this technique alone is not enough. The following therefore introduces a document containment property similar to the one discussed in [2] that addresses such concerns:

Definition 12 *Document Containment: Given a set of updated nodes $N \subseteq d_i$ of a document part d_i, a Merkle signature [2] of $MS_i(\hat{d}_i)$ and the Merkle hash path* [3] *$MP(N, \hat{d}_i)$: N is said to be contained in d_i if the locally computed Merkle hash value of $\hat{d}_i$ from the received N and $MP(N, \hat{d}_i)$ is equal to the verified signature value of $MS(\hat{d}_i)$, where $\hat{d}_i$ is the root node of d_i.*

A secure document envelope SDE_j consists of document block, meta data block and piggybacked block. The document block, $(Upd_{P_1}(d_i), ..., Upd_{P_j}(d_i))$ is the updated document parts of d_i after the authorized access ap performed by the participants $P_1, ..., P_j$ respectively. Each participant P_j computes a Merkle signature over the root node of $Upd_{P_j}(d_i)$ which it signs together with the received Merkle signatures from the previous editors using its private key SK_j. The computed signature yields the value of

$$Sign_{P_j}(MS_{P_j}(\widehat{Upd_{P_j}(d_i)}), Sign_{P_{j-1}}$$
$$(MS_{P_{j-1}}(\widehat{Upd_{P_{j-1}}(d_i)}), ..., Sign_{P_1}(MS_{P_1}(\widehat{Upd_{P_1}(d_i)}))).$$

The meta data block consists of a certificate chain and Merkle hash paths blocks. Each certificate in the chain, $(Cert_{A_1}..Cert_{A_i})$ is formed as described in Section 8.5. Each participant P_j signs its certificate $Cert_{A_j}$ received from its authority together with the received certificate chain with its private key SK_j yielding the signature as

$$Sign_{P_j}(Cert_{A_j}, Sign_{P_{j-1}}(Cert_{A_{j-1}}, ..., Sign_{P_1}(Cert_{A_1}))).$$

The Merkle hash path blocks, $(MP_{P_1}(Upd_{P_1}(d_i), \hat{d}_i), ...,$
$MP_{P_j}(Upd_{P_j}(d_i), \widehat{Upd_{P_j}(d_i)}))$ is a list of Merkle hash paths of the nodes of d_i that are required for the recipient to compute the corresponding Merkle signatures locally. Each participant P_j signs its Merkle hash path $MP_{P_j}(Upd_{P_j}(d_i),$
$\widehat{Upd_{P_j}(d_i)})$ together with the received Merkle hash paths starting from the owner (i.e P_1) with its private key SK_j yielding the signature as

[3] a list of nodes' hash values required to compute the root's hash value.

$$Sign_{P_j}(MP_{P_j}(Upd_{P_j}(d_i), \widehat{Upd_{P_1}(d_i)}), Sign_{P_{j-1}}$$
$$(MP_{P_{j-1}}(Upd_{P_{j-1}}(d_i), \widehat{Upd_{P_{j-1}}(d_i)}), ..., Sign_{P_1}(MP_{P_1}(Upd_{P_1}(d_i), \hat{d}_i))).$$

The document and meta data blocks are bundled together and encrypted by the common secret key $CK_{ap}^{d_i}$. The final block contains TES/TEK as being piggybacked (see Section 8.4) with the above mentioned encrypted block and signed by P_j that resulted in $Sign_{P_j}(TES/TEK)$. Finally, the encrypted data block with the piggybacked block is encrypted by the public key PK_z of other interested participant P_z which only $P_{z\neq j} \in CIG_{ap}^{d_i}$ can decrypt.

$$P_j \xrightarrow{SDE_j} P_{z\in CIG_{ap}^{d_i}}$$

9.2 Document Protection Proof

A secure document envelope SDE_j which is built and initially sent by P_j can be forwarded by any participants in the group during collaboration. As the document and meta data blocks are only be disclosed to a participant having the common secret key and thus remain confidential to others unless they are able to re-compute the required key in lazy fashion (see Section 8.4). The signature over every block prevents any malicious participant in the group to include or exclude any fake data block (i.e. fake document parts, certificates and Merkle hash paths).

Moreover, upon decrypting the encrypted data block any participant in the group can verify the received document part's authenticity and integrity as a whole. It is important to note that an authority initially is assumed to send a secure document envelope containing its current document updates (possibly empty) with its Merkle signature to all the participants it manages so that participants can verify the integrity and authenticity of the document part they are going to collaborate with. To illustrate this, when P_j receives the initial secure document envelope containing $Upd_{A_1}(d_i)$ it can verify $Upd_{A_1}(d_i)$'s containment in the original document d_i (i.e. authenticity and integrity) by computing a Merkle signature out of the received Merkle hash path $MP_{A_1}(Upd_{A_1}(d_i), \widehat{d_i})$ and locally computed hash values of $Upd_{A_1}(d_i)$. The computed Merkle signature should match with the verified signature value of $Sign_{A_1}(MS_{A_i}(\hat{d}_i))$.

Any participant P_j upon receipt of a secure document envelope from P_{j-1} can verify the document integrity and authenticity by computing the Merkle signatures out of the received Merkle hash paths $MP_{P_1}(..), .., MP_{P_{j-1}}(..)$ and locally computed hash values of corresponding document part updates $Upd_{P_1}(d_i), .., Upd_{P_{j-1}}(d_i)$. Each locally computed Merkle signature should match with the corresponding verified signature values of $Sign_{P_{j-1}}(MS_{P_{j-1}}(\widehat{Upd_{P_{j-1}}(d_i)}), ...,$
$Sign_{P_1}(\widehat{Upd_{P_1}(d_i)}))$.

Finally, as each participant signs its document updates along with the previous series of updates performed by previous editors the recipient can trace everyone's updates by simply verifying the signatures iteratively. It can also verify the eligi-

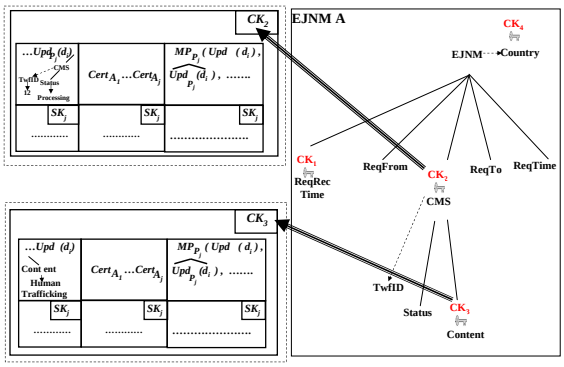

Figure 11: $EJNMA$ document part schema nodes annotation with common secret keys and corresponding navigation before sending of secure document envelopes.

bility of previous editor's authorization by iterative verification of the certificate chain.

9.3 Document Navigation

As described in Section 7.2, an accessible target document part can be divided into several disjoint fine grained target nodes, thus the participants are assigned to several common access interest groups with disjoint set of document nodes. This also means that, in the collaboration phase, a document node will be encrypted by a unique common secret key, which might change when participants join or leave as described in Sections 8.4 and 8.5. However, participants possess the knowledge of the document schema and thus know each document part's structure. Given this fact each participant annotates the document part schema nodes with the associated common secret keys that it computes and uses those as encryption/decryption keys for corresponding document envelopes. As such participants can determine which key to use to encrypt/decrypt for which document part nodes while they are collaborating, in particular encrypting before sending and decrypting after the reception of secured document envelopes.

- Before sending an updated document envelope participants parse the schema to find the annotated common secret key associated with the updated document part.
- After receiving a document envelope participants can determine the required decryption key by observing the piggybacked TES/TEK value. If a re computation of a new common secret key is performed as a result of new TES/TEK, participants update their corresponding annotation in the schema with the associated new common secret key.

Figure 11 shows a $EJNMA$ document part schema nodes: `ReqRecTime`, `CMS`, `Content`, `Country` are annotated with CK_1, CK_2, CK_3, CK_4 respectively. Before sending two se-

cure document envelopes SDE_2 and SDE_3 the corresponding common secret keys CK_2, CK_3 can be determined by parsing the annotated schema.

10 Controlled Document Edition

This section describes how the mechanisms presented so far are combined to support the distributed execution of XML document centric collaborations. The collaboration is initiated by the receipt of first wave of access interests of the participants to the owners (original authorities) in an interest specification phase. The collaboration phase, during which the document is actually edited and exchanged, involves document viewing, update, and verification.

10.1 Interest Specification Phase

The interest specification phase consists of the following tasks with respect to an authority A_i.

1. Reception of access interests from n other participants.
2. Evaluation of Access Control Policy: While owners will directly evaluate the participants with its policy, the delegate can evaluate with the security objective that it receives from its direct authority.
3. Determination of Common access interest groups.
4. Control data blocks: Generation of encrypted control data blocks $CD_{A_i}^{z\in[1,m]}$ and distribution of them.

In this phase, the authority decides of a fixed list of members of a common access interest groups, for which access control is enforced later on by participants themselves using encryption. This scheme allows any participant having computed the common secret key at this stage to access and update document parts at any time after the interest specification phase.

Upon receipt of $CD_{A_i}^{z}$ an authorized participant P_z performs following steps:

1. Retrieval of control data block: Decrypts the secure document envelope using SK_z and retrieves the required sibling path values SP_z. It also retrieves the certificate chain associated with the access primitive ap and 'SecObj' that enable P_z to manifest its eligibility to perform the corresponding access ap and to be a delegate afterwards.
2. Common secret key computation: compute the common secret key $CK_{ap}^{d_i}$ using the received sibling path values.

10.2 Collaboration Phase

Upon receipt of a secure document envelope SDE_j from P_i, $P_{j\neq i}$ which is in the same group performs the following tasks:

1. Retrieval of piggybacked block: Decrypt secure document envelope SDE_j with SK_j and retrieve the TES/TEK value. Verify the integrity of TES/TEK using PK_i and checks whether it has the same TES/TEK value in its possession. If it possesses the value then it determines the required common secret key as described in Section 9.3 otherwise re computes the key in lazy fashion (see Section 8.4).
2. Retrieval of document and meta data blocks: P_j decrypts the encrypted block of components with the common secret key.
3. Meta data integrity: Verify the integrity of certificate chain and Merkle hash paths using PK_i.
4. Authorization verification: Verify P_i's eligibility of the performed access by checking that the certificate chain contains a certificate $Cert_{A_i}(PK_i, PK^i_{ap})$ for P_i.
5. Document authenticity, integrity and containment check: Verify the received document parts authenticity, integrity and containment as described in Section 9.2.

After performing the authorized access ap over d_i a participant P_i can build the secure document envelope SDE_i incrementally as follows:

1. Document block: Compute a Merkle signature over the root node of $Upd_{P_i}(d_i)$ and sign it using SK_i together with the received series of updates and add the signature and updated document parts.
2. Meta data block: Compute a signature over its certificate together with the chain of certificates using SK_i and add the signature and the certificate chain. Compute a signature of its Merkle hash path together with the previous editors' Merkle hash paths, sign and add the signature and the Merkle hash paths.
3. Determine the required common secret key to encrypt the document and meta data blocks as described in Section 9.3 and encrypt those blocks with the common secret key.
4. Piggybacked block: Compute signature of TES/TEK using SK_i and add the signature and the TES/TEK value.
5. Secure document envelope: Encrypt the whole block using recipient's public key.

11 Related Work

There has been a quite remarkable progress in the area of XML access control in recent years. Significant improvements in the area of fine grained access control on XML documents has been made in [11]. It depicts a client-server centralized framework. Clients request the server for accessing a document. The server, which

is responsible for designing the document schema, decides about the authorizations and at the same time enforces access control on the document. In [10] the authors describe a fine grained access control technique for SOAP based communication among web services. However, the aforementioned work does not address the particular requirements of collaborative XML document edition as described in this paper.

From the enforcement perspective, these approaches are known as view based XML access control. However, the view based approach inherently contains two significant limitations [4]: scalability and storage. As an increasingly large number of requesters is involved, the management of views does not scale up and the increasing number of documents and clients demands more storage and cost on the server side. The view based approach also does not consider the issue of document updates where documents are dynamically exchanged among several participants and in particular document protection aspects.

The most dominating assumption of those approaches is the fact that all the security specifications are specified and enforced by a centralized entity (e.g. DBA) of the XML data sources.

In [18] some mechanisms and algorithms for cooperative updates of XML documents in a distributed environment are provided. While this work is similar with ours regarding the use of cryptography to support controlled document edition, this does not consider distributed sources of documents and their ownership. This approach is more tailored to a posteriori verification of the correct execution of a document edition process.

Encryption as an enforcement mechanism for access control decisions made at a server has been discussed in the literature for a while [3, 20]: the server encrypts the data it stores with secret keys; the client can access these only provided it possesses the right decryption keys. This technique supports dynamic change only through the use of the server as a centralized point of enforcement that computes and distributes keys and therefore constitutes a single point of failure. Scalability and performance are central issues growing with the number of clients accesses, notably regarding the need for partial reencryption of data because of changes in the access control rules. This technique also does not address the need for distributed sources of data. It should also be mentioned that these papers altogether do not address traceability issues with respect to document updates.

The use of tamper-resistant modules as described in [6] however makes it possible to alleviate the limitation of the latest approach regarding policy dynamicity, both in terms of access control decision and enforcement. This approach however requires the difficult and expensive deployment of a trusted infrastructure. We instead think delegation might be enough to adapt the access control policy in most scenarios.

Our distributed access control solution is fundamentally different compared to these approaches:

- Our solution is distributed as opposed to a centralized framework of [11].

- It provides an access interest specification based on primitives which allows each participant to specify its fine grained access interests on the document parts owned by other participants.
- Instead of a purely server based approach, it makes it possible to follow indiscriminately a push or pull based approach since access control enforcement does not rely on a central authority.
- While controlled by the owner, the edition of documents can be initiated and run by all participants interacting autonomously.

12 Conclusion and Future Work

We proposed a distributed and fine grained access control framework for XML document centric collaboration. This framework perfectly fits document edition scenarios in which multiple organizations are involved and manage a part of a composite document at their own discretion, as illustrated by the Europol-Eurojust case of mutual legal assistance. To the best of our knowledge, this framework is the very first work of its kind.

Our distributed access control framework addresses situations in which authorization granting authorities may be offline or unavailable and may not be directly in touch with collaboration participants. Access control is enforced thanks to the combination of document authorization and document protection. Document authorization relies on a cryptographic enforcement: only participants that have received proper encryption keys can view or update a document part. Document protection is necessary to address the fact that, compared with a centralized access control scheme, a document is not stored in a safe repository but exchanged between participants, and thus subject to attacks by malicious participants or by outsiders. Document protection also encompasses verifying the conformance of the update of a document part with the authorization policy of its authority. Access control relies on two asynchronously decoupled phases: the initiation phase addresses authorization decisions and preliminary key distribution concerns, while the collaboration phase essentially focuses on enforcing the authorization policy and document protection.

We are currently working on the design of an architecture to actually deploy the framework and on synchronization and document reconciliation issues. Such a system might for instance be run on top of a workflow. We are also considering the handling of schema changes at runtime and of corresponding policy updates as future work.

References

[1] S. Abiteboul, H. Kaplan, and T. Milo. Compact labeling schemes for ancestor queries. In *SODA '01: Proceedings of the twelfth annual ACM-SIAM sympo-*

sium on Discrete algorithms, pages 547–556, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[2] E. Bertino, B. Carminati, and E. Ferrari. Merkle Tree Authentication in UDDI Registries. Idea Group Inc, International Journal of Web Services Research, 1(2):37-57, 2004.

[3] E. Bertino and E. Ferrari. Secure and Selective Dissemination of XML Documents. *ACM Trans. Inf. Syst. Secur.*, 5(3):290–331, 2002.

[4] W.-C. L. Bo Luo, Dongwon Lee and P. Liu. *A Flexible Framework for Architecting XML Access Control Enforcement Mechanisms*, volume Volume 3178/2004 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, December 2004.

[5] S. Boag, D. Chamberlin, M. F. Fernndez, D. Florescu, J. Robie, and J. Simon. XQuery 1.0: An XML Query Language. Technical report.

[6] L. Bouganim, F. D. Ngoc, and P. Pucheral. Dynamic access-control policies on xml encrypted data. *ACM Trans. Inf. Syst. Secur.*, 10(4):1–37, 2008.

[7] A. D. A. Boujraf and M. Noble. Towards e-Administration in the Large (R4eGov). Deliverable WP3-D7, 2007. EU IP R4eGov.

[8] J. Clark and S. DeRose. XML Path Language (XPath), http://www.w3.org/tr/xpath.

[9] E. Cohen, H. Kaplan, and T. Milo. Labeling Dynamic XML Trees. In *Symposium on Principles of Database Systems*, pages 271–281, 2002.

[10] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. Fine Grained Access Control for Soap E-services. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 504–513, New York, NY, USA, 2001. ACM.

[11] E. Damiani, S. D. C. di Vimercati, S. Paraboschi, and P. Samarati. A Fine-grained Access Control System for XML Documents. *ACM Trans. Inf. Syst. Secur.*, 5(2):169–202, 2002.

[12] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. Securing XML Documents. In *International Conference on World Wide Web*, 1999.

[13] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, 1976.

[14] W. Fan, C.-Y. Chan, and M. Garofalakis. Secure XML Querying With Security Views. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 587–598, New York, NY, USA, 2004. ACM Press.

[15] Y. Kim, A. Perrig, and G. Tsudik. Simple and Fault-Tolerant Key Agreement for Dynamic Collaborative Groups. In *CCS '00: Proceedings of the 7th ACM conference on Computer and communications security*, pages 235–244, New York, NY, USA, 2000. ACM Press.

[16] G. Kuper, F. Massacci, and N. Rassadko. Generalized XML Security Views. In *SACMAT '05: Proceedings of the tenth ACM symposium on Access control models and technologies*, pages 77–84, New York, NY, USA, 2005. ACM Press.

[17] P. Lee, J. Lui, and D. Yau. Distributed Collaborative Key Agreement Protocols for Dynamic Peer Groups. In *Network Protocols, 2002. Proceedings. 10th IEEE International Conference.*, pages 322– 331, New York, NY, USA, 12-15 Nov. 2002. ACM Press.

[18] G. Mella, E. Ferrari, E. Bertino, and Y. Koglin. Controlled and Cooperative Updates of XML Documents in Byzantine and Failure-Prone Distributed Systems. *ACM Trans. Inf. Syst. Secur.*, 9(4):421–460, 2006.

[19] R. Merkle. A certified digital signature. *In Advances in Cryptology - CRYPTO '89 Proceedings, Lecture Notes in Computer Science*, 435:218–238, 1989.

[20] G. Miklau and D. Suciu. Controlling Access to Published Data Using Cryptography. In *VLDB*, pages 898–909, 2003.

[21] T. Milo and S. Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *VLDB '98: Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 122–133, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[22] S. Mohan, J. Klinginsmith, A. Sengupta, and Y. Wu. Acxess - access control for xml with enhanced security specifications. In *ICDE '06: Proceedings of the 22nd International Conference on Data Engineering*, page 171, Washington, DC, USA, 2006. IEEE Computer Society.

[23] M. Murata, A. Tozawa, M. Kudo, and S. Hada. XML Access Control Using Static Analysis. In *CCS '03: Proceedings of the 10th ACM conference on Computer and communications security*, pages 73–84, New York, NY, USA, 2003. ACM Press.