Regulating Access to SMIL formatted Pay-per-view Movies

Naren Kodali
Stereo Video Laboratory
George Mason University
Fairfax, VA 22030-4444,USA
001-703-371-5029
nkodali@gmu.edu

Duminda Wijesekera
Center for Secure Information Systems
George Mason University
Fairfax, VA 22030-4444,USA
001-703-993-1578
dwijesek@gmu.edu

ABSTRACT

XML [15] has become a standard format for information that moves within the World Wide Web. Previous work in securing XML documents concentrated mainly on textual documents. Those proposals are ineffective in the context of multimedia, which mostly comprises of some sensible combination of images, text, audio, and video. As multimedia constitutes a significant component of the traffic within the Internet, it requires to be secured. We propose an access control model and an encryption mechanism that enforces access control and maintains integrity of multimedia by using the Synchronized Multimedia Integration Language (SMIL)[1], thereby preventing illegal or malicious unauthorized access or modifications to the media documents in transition. We show its utility in safeguarding pay-per-view movies and multi-level secure coalitions observing unfolding scenarios.

Categories and Subject Descriptors

D.4.6 [Security]: Access controls, Authentication.

General Terms

Algorithms, Design and Security.

Keywords:

Synchronized Multimedia, Access Control, Encryption, Integrity, Pay-per-View, Smart Card, XML, and SMIL.

1. INTRODUCTION

Regulating access to sensitive data is a priority in commercial and military enterprises. Presently such sensitive information comes as text, photographs, audio recordings and video clips. As such sensitive data are used and exchanged on the Internet, their integrity and access control becomes important.

ACM Workshop on XML Security, Nov. 22, 2002, Fairfax VA, USA

The extensible markup language (XML) that is becoming the lingua franca of the Internet maintains the integrity of data by arranging them structurally. As will be discussed shortly, there are proposals to fulfill the security requirements of such documents by enhancing XML with authorization specifications, encryptions, signatures and other mechanisms. Our objective is to extend these mechanisms to synchronized multimedia documents released on the Internet. For this paper we consider releasing movies as XML documents in a pay-per-view scenario.

Synchronized multimedia documents depend upon synchronization metrics to convey their intended meaning of unfolding scenarios thereby requiring an extension to XML, referred to as the Synchronized Multimedia Integration Language (SMIL). One of the features that distinguish multimedia from other textual documents is that the Quality of Service (QoS) requirements that are essential in comprehending multimedia documents. The complexity of SMIL documents over normal XML documents derive mainly due this fact. Therefore any model or mechanism to ensure the security of multimedia documents must respect these QoS specifications.

Laws in different countries govern releasing movies and authoritative bodies enforce them. Movie rating such as R, PG13, MA in the United States are examples of such laws. Therefore any pay-per-view movie service must comply with such territorial regulations in addition to ensuring that non-paying customers are unable to view them. In this respect, the current paper provides a model that can provide an access control mechanism to broadcast SMIL documents (i.e. SMIL formatted movies) so that only authorized recipients may access parts of SMIL documents as specified by the authoritative bodies that have jurisdiction over them while honoring acceptable QoS. To enforce territorial release regulations, we seek some solutions proposed to enforce the security of XML documents.

Security of XML documents falls into three broad categories; XML access control, XML encryption and XML signatures [2]. XML Signatures are based on Public Key cryptography in conjunction with one-way hash functions. This requires the use of message digests and encryption with the private key of the signer. These recommendations cannot be directly applied in the context of multimedia due to many reasons. Firstly, XML signature does not work effectively because the digest of multimedia documents such movies require computationally intensive hash algorithms. Secondly, if used it would be server and compute intensive. A movie or a presentation contains a high volume of data and considerable amounts of meta-data. XML Encryption requires that different portions of a XML document to be encrypted with

different keys, on the basis of security policies of the source. Therefore subjects are sent all and only those keys decrypting the portions that they are allowed to access. This can be carefully extended to SMIL and different portions of the document can be encrypted with different keys. The EncryptedData element in association with the Encrypted Key element forms the crux of such a representation, which we will use in our Encryption Model in Section 5.

A typical cable or satellite based pay-per-view service provider works as follows. The service provider announces the time slots at which movies are broadcasted. Based on these announcements, subscribers pay and view movies of their choice at advertised time slots. Then the service provider broadcasts the movies during the specific timeslots and enables only those subscribers permitted to watch them. Our model considers such an environment in which access control is based on the validity and the privileges of the subscribers and upon broadcast of the movie, only authorized users are allowed to view the movie in that time window. Our model is also hardware dependent, but is scalable because one subscriber cannot subscribe to more than one movie during the same time slot. Further, our model is not dynamic and it restricts the subscribers from altering their subscription once they have committed but that limitation inherently derives from the present day commercial pay-per-view architectures.

All the figures used in the paper are not necessarily complete SMIL syntax, notes and other information have been used within the figure or better understanding. The outline of the paper is as follows Section 2 describes the related work. Section 3 describes SMIL and Section 4 describes our access control model. Section 5 explains the encryption process. Section 6 discusses certain application scenarios and future extensions and Section 7 concludes the paper.

2. RELATED WORK

Regulating access to XML documents of textual nature has been actively researched in the past few years after it has been accepted that XML is soon going to become the de-facto standard for the World Wide Web. Bertino et al [4,5,6], Damiani et al [9,11,12] and Gabillon et al [3] have suggested access control models for controlling access to XML documents.

Bertino et al [4,5], have developed Author-X, a Java based system to secure XML documents that supports access control policies at various granularities and user credentials. Author-X encodes security policies for a set of XML documents in an XML file referred to as the policy base. They permit both permissions and prohibitions. This feature enables the user to specify exceptions with ease as opposed to creating a set of XML documents and document type definitions (DTD's). There are conflict-resolution and default strategies to address over specification and under specification respectively.

Damiani et al [9,11] developed an access control model where the tree structure of XML documents are exploited using XPATH [13] to control access at different levels of granularity. Here, the smallest protection granularity is a node, and permissions or prohibits to a node can be defined. Fine-grained accesses are specified using XML markup where a subject is a user who is generally a member of one or more user groups, and an object is any node on a XPATH tree.

Gabillon et al [3] have suggested an alternative to Damiani et al [9,11], where authorization rules related to a specific XML document are first defined on a separate authorization sheet (style sheet). This sheet is then translated to an (eXtensible Stylesheet Language) XSL sheet. If a user requests access to the XML document then the (XSL Transforms) XSLT [14] processor uses the XSLT sheet to compute the view of the XML document with appropriate rights.

Damiani et al [10] have proposed a model for selectively controlling access to images. The XML based data model of SVG is exploited to control access to graphic information on the web. The primary focus is the vector image data, which has emerged as an alternative to contemporary raster image formats.

Kudo et al[17] have pioneered the research in provisional authorization for document security that has helped in the standardization of XACL.

But none of these proposals are completely satisfactory for multimedia. They primarily address textual documents and exploit the granular structure of XML documents. Although [10] addresses feature protection of XML format based images, its primary focus is controlled dissemination of sensitive data within an image. Multimedia for various reasons as discussed above has to be treated differently. In our framework we use an XSLT processor to deal on our authorization mechanism as well as propose a variation to the encryption model of Bertino et al [4,5].

Bertino et al. [4] introduce an encryption mechanism to achieve integrity after computing the accesses in XML documents with textual data. All elements of a XML document to which a maximal collection of users have access are encrypted with a single key. All such encrypted copies of parts of the original XML document are stored in an appropriately indexed database. When a subject requests the original document, the parts that he is authorized to obtain are encapsulated into a single package that includes an entry with appropriate keys for the recipient. The part of this package that contains the keys to decrypt portions is encrypted with the requesting subjects public key. Upon receiving the package, the subject decrypts the entry containing other keys with her private key and then uses those keys contained there to decrypt other parts of the document. This approach is beneficial as it allows shipping the same package to all subjects and protects the content at the same time because each portion can be decrypted only with the corresponding private key.

Boneh et al [7] proposed a public key encryption scheme for multimedia in which there is one encryption key and multiple decryption keys. Multimedia content is encrypted and distributed over a broadcast channel to subscribers, and the decoder box has the secret decryption key, which decrypts the public broadcast. Problems of users sharing decryption keys or groups of users colluding to create a new decryption key are addressed and a traitor-tracing scheme is implemented to identify traitors. Fiat et al [8] proposed a dynamic traitor-tracing scheme, where systems with conditional accesses are safeguarded using watermarking. The root of any potential piracy is traced and it is disconnected from the regular transmission without harming legitimate users. Using their approach it is also possible to gather forensic evidence to incriminate the abusers.

In the context of multimedia, [4] has some drawbacks. Firstly, broadcast multimedia, requires a large number of copies to be generated for each user. Secondly, each custom-made copy (view) has to be individually delivered, as the encryption method prevents multicasting. The situation where a large number of users need multiple keys would be impractical in the context of a multimedia presentation as the numbers of *elements* that need to be encrypted are large unlike in a textual document. Besides dividing a movie into small parts, as determined by the server administrator for the purpose of encryption would result in a large management overhead and metadata. Both [7] and [8] address the issue of multimedia and are very effective for commercial Cable TV applications, but cannot be adapted "as is" in the context of multimedia presentation over the internet because of the watermarking and the variations of the movies generated thus are huge and would be very server intensive, thereby could result in a bottleneck.

3. SYNCHRONIZED MULTIMEDIA INTEGRATION LANGUAGE (SMIL)

SMIL [1] is an XML-like language developed by W3C to author multimedia presentations. The major components of multimedia such as audio, video, text and images can be integrated and synchronized to form a presentation or a movie. The distinguishing features that separate SMIL from XML are well-defined syntactic constructs for timing and synchronization of media streams that allow fine-grained synchronization. SMIL also has syntax for spatial layout including constructs for non-textual and non-image media and hyperlink support for time-based media making SMIL adaptable to varying user and system characteristics.

The important integration features offered by SMIL are the synchronization constructs such as <seq>, <excl> <repeat> and

<par>. They are used to hierarchically specify synchronized multimedia documents. The <seq> element plays the child elements one after another in sequential order. In such a hierarchical specification <excl> element plays one child at a time, but does not impose any order. The <par> element plays child elements as a group (allowing *parallel* playback). In SMIL, every element has a beginning (*begin*) and a *simple duration*. *Begin* can be specified in various ways, an element can begin at a given time, or based upon when another element begins or when some event occurs. *Simple duration* defines the basic presentation time of an element. Elements can be specified to repeat their simple duration a fixed finite number of times. The simple duration and any repetitions can be combined to determine the so-called *active duration*. An element *becomes active* when it begins its active duration, and *becomes inactive* when it ends its active duration. When an element's active duration terminates, the element can either be removed from the presentation or *frozen* meaning it is held in its terminal state: e.g., to fill dead-times in the presentation. Attributes that control these synchronization specifications can be applied not only to media elements, but to complex constructs (par, seq etc) as well. This allows, for example, an entire sequence to be repeated, and to be coordinated as a unit with other media and complex presentations such as movies. Audio and video, that constitute basic multimedia stream types can be combined to form a movie by using the synchronization constraints to specify the exact synchronization we need. SMIL has been constantly developing and becoming rich in features and is supported by many present–day commercial browsers and was developed to work with many commercial and free media players.

An example SMIL document given in Figure1 shows the syntax of movie. As the figure shows there are four forms of media video, audio, text and image that are integrated and synchronized using <seq> and <par> and duration. The primitive media entity is a frame, many of which are grouped together to form scenes. The audio, video and text frames (if any) run in parallel to combine and render a scene for a movie. An act consists of a finite number of scenes. A typical movie consists of acts, scenes, and frames in that order, which is a direct adaptation form the theatrical plays of the olden days. In our model we chose an "act" as the level of granularity on which we impose access control and encryption. As shown in the figure above, the tree structure of SMIL is depicted for a movie at the act, scene and frame level.

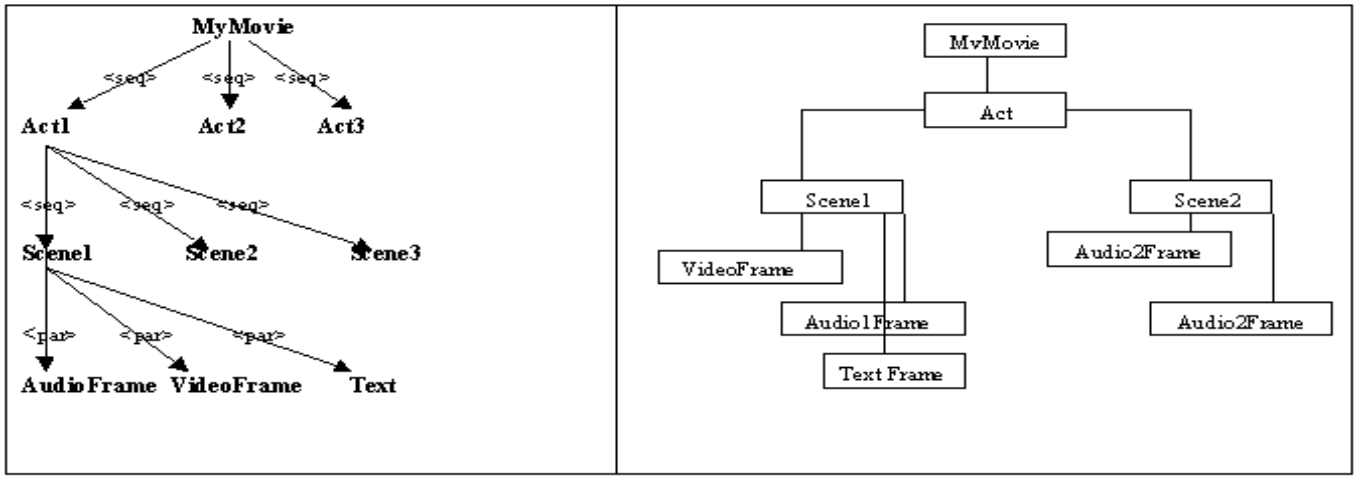

Figure 1: Structure of an Example Movie

4. ACCESS CONTROL MODEL

Figure 2 shows the proposed architecture. It consists of a server site used to encrypt and broadcast the movie and a collection of client sites to decrypt and play the movie. At the time of

registering as a client, the service provider, who is the authority in defining accesses, asks the subscriber to submit all critical information such as age, sex, country of residence and his particular likes (if any). Depending on these submitted factors the locality of the viewer, and the wishes of the authoritative body

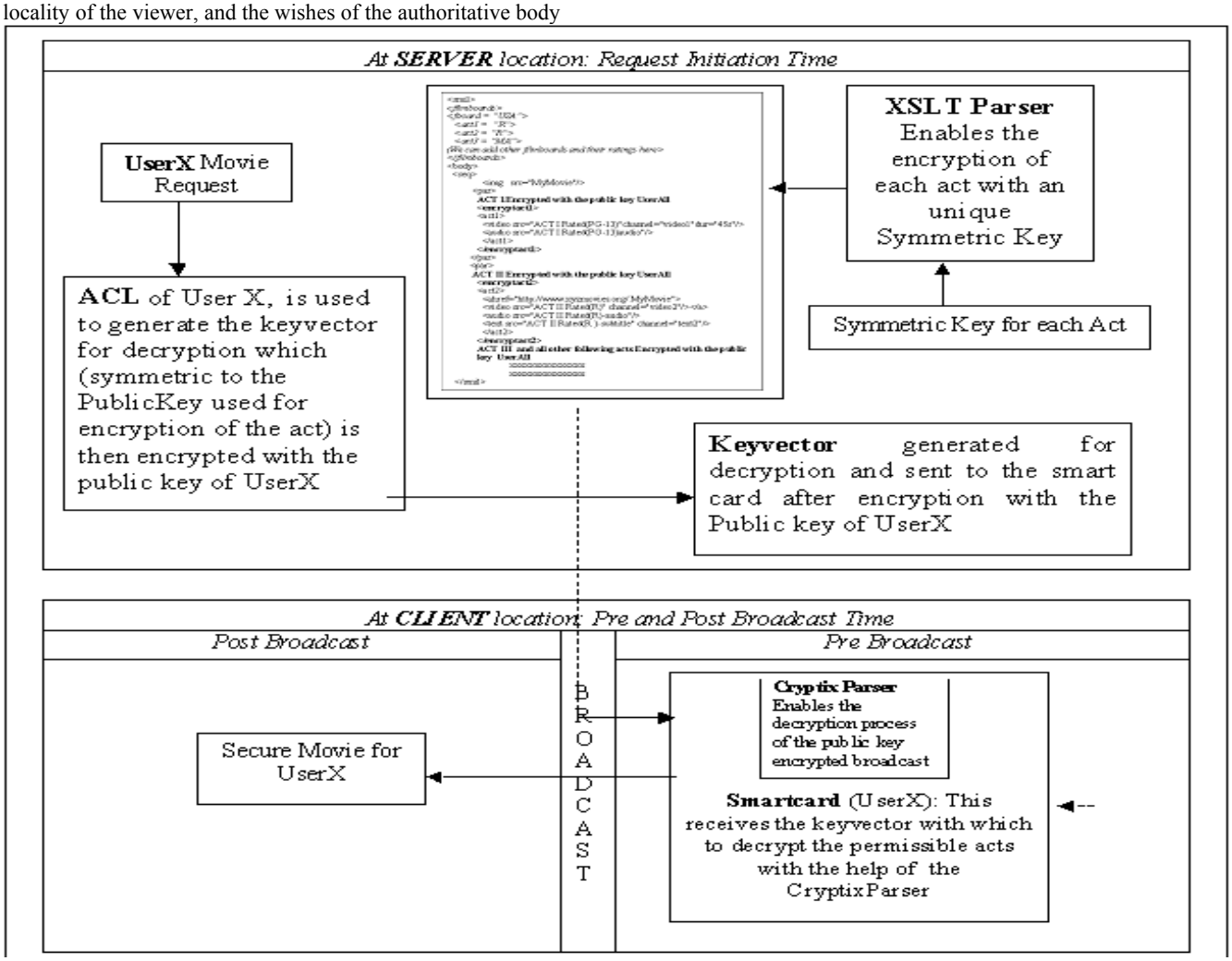

Figure 2: System Architecture

that has administrative control over that locality, and the requirements of the client an *access control list* is generated by the server . Each client site has a *smartcard* installed to enforce access control decisions made by the server on behalf of the authoritative body. A *smartcard* is a device that performs the decryption process to deliver the movie to the authorized subscribers. Before the broadcast begins the *smartcard* is given the keys necessary to decrypt the sections of the movie it is entitled to play out. The smartcard has an enhanced parser, referred to as the CryptixParser, which uses appropriate keys that are unicast to the *smartcard*, and decrypts only the associated portions of the movie thereby not displaying unwanted or unauthorized parts.

4.1 Access Control Policies

The model requires that we specify access control rules made based on the *access control list* on a XPAS (XML Policy Authorization Sheet). This is a typical XML style sheet that contains all the information about the user (age, jurisdiction, and other access rights). A copy of this access control list is also maintained in the smart card that resides on the subscriber machine. This access list is instrumental in designing the decryption key for that particular user (e.g. UserX). Figure3 gives the SMIL specification of the movie in Figure 1.The layout section represents the various channels that can be created for use for different parts of the movie depending on the conditional access. The time-container tags encompass the frames and scenes and the level of granularity is the act, which means that we would divide the entire document into a sequence of individual acts upon which conditional and privileged access and encryption can be implemented to enforce security. Any SMIL document can be represented as a tree with “n” number of nodes. XPATH [15] is a regular language for denoting nodes of an XML document.

Since the syntactic structure of SMIL is similar to XML syntax we use XPATH to address individual nodes. The authorization rules specify access privileges granted to the user when he

subscribes to the service. In these rules, *subjects* are single users of SMIL documents and he/she is uniquely identified by some attributes such as user-id, age, sex, country of domicile etc. An *object* is any node in a XPATH tree. The most helpful feature of XPATH it facilitates pattern matching using regular expressions. This is helpful in identifying objects that meet and adhere to the specified policy.

```
<smil>
  <head>
    <layout type="text/smil-basic">
    <channel id="video1" left="20" top="50" z-index="1"/>
    <channel id="text1" left="20" top="120" z-index="1"/>
    <channel id="video2" left="150" top="50" z-index="1"/>
    <channel id="text2" left="40" top="70" z-index="1"/>
    <channel id="video3" left="60" top="120" z-index="1"/>
    <channel id="text3" left="70" top="110" z-index="1"/>
    </layout>
  </head>
 <body>
 <seq>
        <img  src="MyMovie"/>
      <par>
        <act1>
        <video src="ACT I video" channel="video1" dur="45s"/>
        <audio src="ACT I audio"/>
        </act1>
      </par>
      <par>
        <act2>
        <ahref="http://www.xyzmovies.org/ MyMovie">
        <video src="ACT II " channel="video2"/></a>
        <audio src="ACT II audio"/>
        <text src="ACT II subtitle" channel="text2"/>
        </act2>
      </par>
      <par>
        <act 3>
        <video src="ACT III video" channel="video3"/>
        <audio src="ACT III audio"/>
        <text  src="ACT III subtitle" channel="text3"/>
        </act3>
      </par>
 </seq>
 </body>
</smil>
```

Figure 3: SMIL Representation of the Movies in Figure 2

A pattern search is run on the XPATH tree to yield what acts he/she can possibly see. An XSL Transform [XSLT] [16] is used to convey the Authorization rules to the SMIL document and extract the nodes that obey the authorization rules. The transformation is the actual interface that relates the authorization stylesheet to the SMIL specification of the movie. XSLT understands the rules as patterns and using XPATH parses through the SMIL document and retrieves the appropriate acts. The SMIL document in the Figure 4 below shows how we could incorporate the above specification in SMIL.

In our example we consider three types of classifications based on subscriber age **RatedPG-18** for ages 13 to 18, **RatedR** for ages 18-25 and **Rated MA** for ages 25 and above. These ratings and classifications of the acts are not unique all throughout for all subscribers and that depends on the geographical location of the subscriber. We consider that we have two movie control boards viz. the USA filmboard and the French film board. The French filmboard could be less lenient and allow young people to view portions of the movie that its American counterpart thinks is inappropriate for that particular age group. All such controls that are decided based on the access policies and security considerations can be very precisely represented using SMIL and thereby essentially capture even minute detail that is necessary.

```
<smil>
<accesspolicy>
<fboard = "USA">
  <act1 = "MA">
  <act2 = "R">
  <act3 = "MA">
<fboard = "France">
  <act1 = "R">
  <act2 = "R">
  <act 3 = "R">
(We can add other filmboards and their ratings here>
</accesspolicy>
<body>
 <seq>
        <img  src="MyMovie:Vinerva"/>
      <par>
        <act1>
        <video src="ACT I video)"channel="video1"dur="45s"/>
        <audio src="ACT I audio"/>
        </act1>
      </par>
      <par>
        <act2>
        <ahref="http://www.xyzmovies.org/ MyMovie">
        <video src="ACT II" channel="video2"/></a>
        <audio src="ACT II audio"/>
        <text src="ACT II subtitle" channel="text2"/>
        </act2>
              xxxxxxxxxxxxxxx
              xxxxxxxxxxxxxxx
              xxxxxxxxxxxxxxx
</body>
</smil>
```

Figure 4: SMIL Specification of a Movie with Embedded Access Control Policies

5. ENCRYPTION MODEL

The encryption model is analogous to the real world methodology [10] adopted by the Cable companies when broadcasting movies to paying subscribers. A *smartcard*, which contains the authorization rules based on the *access control list*, is installed on the all the clients. It could be a hardware device or a software program to achieve the same result. In our model we current have a hardware box to accommodate buffering and storage at the client location. It has an inbuilt Cryptix Parser that is programmed in firmware to handle the decryption process that is described below. Decrypting the associated acts in a broadcasted environment is the primary functionality of the intelligent Cryptix parser within the *smartcard*. The encryption model uses both symmetric encryption and Public Key encryption for two mutually diverse events to serve varying purposes. . XML Encryption can be used for this purpose; Figure 5 shows that each "act" needs to be encrypted with a unique symmetric key.

Step1:

The SMIL format of the movie is divided into ***acts*** as discussed earlier, and since our element of granularity is an "act"; we

encrypt each act with a unique Symmetric Key. In the SMIL document act1 is encrypted with **SymmetricKey**$_{act1}$ and act2 with **SymmetricKey**$_{act2}$ and the process is repeated for all the acts within the document.

```
<smil>
<filmboards>
<fboard = "USA">
  <act1 = "R">
  <act2 = "R">
  <act3 = "MA">
(We can add other filmboards and their ratings here>
</filmboards>
<body>
 <seq>
        <img  src="MyMovie:Vinerva"/>
      <par>
       ACT1 is encrypted with the SymmetricKey Act1
       <encryptact1>
       <act1>
         <video src="ACT I video "channel="video1"dur="45s"/>
         <audio src="ACT I  audio"/>
         </act1>
       </encryptact1>
      </par>
      <par>
      ACT2 Encrypted with the SymmetricKey Act2
       <encryptact2>
       <act2>
         <ahref="http://www.xyzmovies.org/ MyMovie">
         <video src="ACT II video" channel="video2"/></a>
         <audio src="ACT II audio"/>
         <text src="ACT II subtitle" channel="text2"/>
         </act2>
       </encryptact2>
       ACT3  and all other following acts up to ActN are encrypted with
       Appropriate  SymmetricKey Act(3-N)
              xxxxxxxxxxxxxxx
              xxxxxxxxxxxxxxx
 </smil>
```

Figure 5: SMIL Specification of the Movie with all acts encrypted.

All the encrypted acts have a corresponding symmetric decryption key (which is the same as the encryption key). Based on the subscriber authorization rules a finite number of symmetric decryption keys strictly corresponding to the allowed acts are grouped together as a vector of keys (***keyvector)***. This distinct keyvector for a particular client would be able to release all or a subset of acts depending on their nature and the authorization rules.

Step 2:

When information is sent to a client a keyvector is generated and is then encrypted with the **PublicKey** of the particular client. The format of the keyvector is as follows:

PublicKey of User ((SymmetricDecryptionKeyNumber (Act Number)…...…SymmetricDecryptionKeyN (ActN))).

Figure 6 shows how we could use XML encryption for achieving what has been described in detail above. It show the syntax to use a symmetric key to encrypt an "act" and to embed a *keyvector* within the document The *keyvector* (a finite set of valid symmetric decryption keys) encrypted with the PublicKey of the requesting user (UserX) is unicast to the client prior to the actual broadcast of the media clip or it could be embedded into the broadcast media document itself as shown in Figure 6. Eventually when the movie arrives, the decryption process is initiated and completed by the CryptixParser.

We do not encrypt the entire movie with a Public Key, because all the acts are encrypted, any unintended or malicious recipient cannot view the movie even though they have access to it, thereby eliminating the need for such a methodology. There are unique decryption *key vectors* specific to each individual transmission that encompass the diverse privileges so we transfer the load from the server onto remote client.

The server at the Administrative the following steps:

1. Create the *keyvector*, encrypt with the PublicKey of the user (client) and unicast it to the *smartcard* on the client machine.(OR) embed the keyvector in the SMIL media document
2. Broadcast the encrypted movie

Each client on receiving a key vector takes the following steps:

1. Decrypt the received *keyvector* (either from unicast or from within the media broadcast) and install on the *smartcard*.
2. Decrypt the encrypted parts of the media clip with available symmetric decryption keys.

In a broadcast environment where there is more than one user, which is generally the case, there is a significant delay between acts in the movie. This is because the encrypted acts arrive sequentially and there is a delay associated with the decryption process and the sequential order of the viewable acts. There is a significant delay between acts, if an intermediate act is not valid. In our current model a hard disk, which can buffer acts, is assumed to present on the hardware device on the receiving end at the client location. Real time system issues such as the ability of the client to download the keys as well as the encrypted movie well in advance, so as to decrypt and buffer the valid acts in a sequential order and the parameters (e.g. how many acts in advance, for what time) to minimize any significant delay is an open issue that we intend to address in very soon

6. APPLICATIONS AND EXTENSIONS

This model can be used in the pay-per-view scenario, where paying customers obtain the rights to watch a movie. One of the advantages of using our model as opposed to using the existing method is that the legitimate customer is able to keep an encrypted copy that she can share with others who are in the exact same category as her. In addition keeping an encrypted movie reduces the piracy, provided that the key sequence is not stolen.

In the event that a legitimate customer later decides to become a traitor, then we can use a watermarking similar to those described in [7,8] to trace the traitor. The traitor tracing can be used even in the military domains to trace traitor cells that intentionally sell out or inadvertently leak information to unauthorized persons.

```
EncryptedData with Symmetric Key  (SymmetricKey Act1)
  <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
      Type='http://www.w3.org/2001/04/xmlenc#Element'/>
        <EncryptionMethod
        Algorithm='xmlenc#3des-cbc'/>
          <ds:KeyInfo xmlns:ds='keyvector'>
        <ds:KeyName>My Movie</ds:SymmetricKey Act1>
   </ds:KeyInfo>
<CipherData><CipherReference>VINERVA</CipherReference></CipherData>
  </EncryptedData>

Notes:
1)We encrypt an Element, in our case it is an individual ACT.
2)The  symmetric key cipher is  (3DES CBC)
3)The symmetric key has an associated movie name "My Movie".
4)CipherData  contains a CipherReference, VINERVA(which is the keyvector)
which is a  reference  with transforms necessary to obtain the encrypted data.
```

Figure 6: Example using XML Encryption

Pseudo Code	Related Activity
procedure **EncryptAct** divide movie into "acts" encrypt each "act" with unique Symmetric Key *SymmetricKeyA1(act1),SymmetricKeyA2(act2)…..* *SymmetricKeyAN (actN)* end;	The entire SMIL specification of the movie is divided into "acts". Each of these acts is encrypted with a unique Symmetric Key. The collection of encrypted acts conforming to the timing specifications is created to form the encrypted movie that will be broadcast.
procedure **CreateKeyVector** UserX requests movie; retrieve allowed acts; group symmetricdecryption keys for selected acts ***(SymDecryptKeyA1(act1),SymDecryptKeyA2 (act2)…);*** Encrypt with PublicKey of UserX $PublicKey_{UserX}$***(SymDecryptKeyA1(act1), SymDecryptKeyA2 (act2)…);*** end;	A keyvector is to be generated for the decryption process that and would be unicast to the CryptixParser in the smartcard of the client of the subscriber. After obtaining relevant acts based on the authorization rules for UserX requests a movie, the decryption key vector is created for that particular user for the acts that he is allowed to see. The keyvector is then encrypted with the PublicKey of UserX.
procedure **UnicastKeyVector** unicast to remote subscribers; $PublicKey_{UserX}$***(SymDecryptKeyA1(act1),SymDecryptKeyA2(act2)…));*** end;	This keyvector is unicast to the smartcard of the subscriber client station, and the CryptixParser within the smartcard knows how to decrypt the movie that is going to be broadcast later using this decryption keyvector.
procedure **Showmovie** intiate CryptixParser *input:* **EncryptAct** Movie decrypt using $keyvector_{UserX}$ *output:* permissible acts of the movie in accordance to access control list for UserX	This process is the heart of the encryption model after receiving the keyvector and eventually the movie that is broadcast, the Cryptix parser decrypts the acts that correspond to the keyvector at hand. This ensures that UserX sees only what he is exactly permitted to see according to his privileges during subscription.

Figure 7: Phases of Encryption and Decryption

Another potential area of usage is combat coordination, briefing of sensitive information and other scenarios that require multi-level access control for unfolding military situations or emergency management. In a sense our model for multimedia and its delivery mechanism can be considered *view-based access control* where each view is custom encrypted for a group of users. Access control for multi-level security by generating secure views from XML documents [16] while retaining the their semantic properties is possible with sufficient enhancements in our model. In order to use our model for the stipulated situation, the film board ratings must be changed to the clearance levels given by authoritative military organizations. Furthermore by having multiple such bodies, we can accommodate military coalitions such as NATO sharing sensitive live information. One of the concerns of applying our mechanisms in this given scenario is that because such conflicts run for a considerable duration, then keys could be stolen, thereby revealing the secrets encrypted in SMIL documents. Proposed future work will address this situation.

XrML (eXtensible rights Markup Language)[17] provides a universal method for securely specifying and managing rights and conditions with all kinds of resources including digital content and other pertinent services. It supports a wide variety of business models and in the world of multimedia MPEG-21 has selected XrML as a basis for its rights expression language. We could extend our model to accommodate XrML and provide an XML-based grammar that can securely specify and assign rights and conditions to the digital content at varying levels of granularity. It is a right expression language adopted by MPEG (Moving Pictures Expert Group) and allows users to issue permissions on actual usages and activities allowed over assets, which in our case is regulated multimedia.

ODRL (Open Digital Rights Language is based on the XML Schema unlike XrML and is pluggable to other DRM architectures, integrating it with SMIL for the purpose of regulating access in the commercial pay-per-view sector is possible.

7. CONCLUSION

We have presented a framework for regulating access to a synchronized multimedia presentation by extending standard access control model defined for text documents in [4,5,6] and [3]. As a mechanism for access control, we have provided an encryption model based on [7]. Our ongoing work addresses stronger encryption models and video-on-demand type of services. These can be applied to military coalitions that view unfolding situations in multi-level secure environments and pay-per-view kind of services in the entertainment industry.

8. REFERENCES

[1] J.Ayers et al. "Synchronized Multimedia Integration Language (SMIL 2.0)". World Wide Web Consortium (W3C). http://www.w3.org/TR/smil20/ (August 2001).

[2] E. Bertino, B. Carminati, E. Ferrari. "XML Security" in Information Security Technical Report, Vol 6,No 2(2001) Pages 44-58.

[3] A. Gabillon, E. Bruno. Regulating Access to XML documents. in *Proc. IFIP WG11.3 Working Conference on Database Security*, Niagara on the Lake, Ontario, Canada, July 15-18, 2001.

[4] E. Bertino, S. Casatano, E. Ferrari "Securing XML Documents with Author-X" in IEEE Internet Computing, vol 5,no3 May/June 2001

[5] E. Bertino, M. Braun, S. Castano, E. Ferrari, M. Mesiti. "AuthorX: A Java-Based System for XML Data Protection". In Proc. of the 14th Annual IFIP WG 11.3 Working Conference on Database Security, Schoorl, The Netherlands, August 2000

[6] E. Bertino, S. Castano, E. Ferrari and M. Mesiti. "Specifying and Enforcing Access Control Policies for XML Document Sources". World Wide Web Journal, vol. 3, n. 3, Baltzer Science Publishers.

[7] Dan Boneh, Matthew Franklin "An Efficient Public Key Traitor Tracing Scheme" in Eurocrypt 99.

[8] Amos Fiat, Tamir Tassa "Dynamic Traitor Tracing" Pages 211-223The Journal of Cryptography, April 2001.

[9] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "Securing XML Documents," in *Proc. of the 2000 International Conference on Extending Database Technology (EDBT2000),* Konstanz, Germany, March 27-31, 2000.

[10] E. Damiani, S. De Capitani di Vimercati, E. Fernandez-Medina, P. Samarati "An Access Control System for SVG Documents" in *Proc. IFIP WG11.3 Working Conference on Database Security*, King's College, and Cambridge, England

[11] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati, "Controlling Access to XML Documents," in IEEE Internet Computing, vol. 5, n. 6, November/December 2001, pp. 18-28.

[12] E. Damiani, S. De Capitani di Vimercati, S. Paraboschi, P. Samarati "XML Access Control Systems: A Component-Based Approach" in *Proc. IFIP WG11.3 Working Conference on Database Security*, Schoorl, The Netherlands, August 21-23, 2000.

[13] J. Clark et al.. "XML Path Language (XPath) Version 1.0". World Wide Web Consortium (W3C). http://www.w3c.org/TR/xpath (November 1999).

[14] J. Clark et al. "XSL Transformations (XSLT) Version 1.0". World Wide Web Consortium (W3C). http://www.w3c.org/TR/xslt (November 1999).

[15] T. Bray et al. "Extensible Markup Language (XML) 1.0". World Wide Web Consortium (W3C). http://www.w3c.org/TR/REC-xml (October 2000).

[16] A.G. Stoica and Csilla Farkas. "Secure XML Views" in *Proc. IFIP WG11.3 Working Conference on Database Security*, King's College, Cambridge, England.

[17] Michiharu Kudo and Satoshi Hada, "XML Document Security based on Provisional Authorization", 7th ACM Conference on Computer and Communication Security, pages 87-96, Nov. 2000

[18] ContentGaurd " eXtensible rights Markup Language(XrML)"

http://www.xrml.org