SVG BASED SECURE UNIVERSAL MULTIMEDIA ACCESS

Ahmed Reda Kaced
GET / Télécom Paris - LTCI UMR 5141 CNRS, Computer Science and Networks Department
37-39 Rue Dareau 75014, Paris - France

Jean-Claude Moissinac
GET / Télécom Paris - LTCI UMR 5141 CNRS, Signal and Image Processing Department
37-39 Rue Dareau 75014, Paris - France

Keywords: SVG, Universal Multimedia Access, content adaptation, Merkle Hash Trees.

Abstract: In this paper, we develop and implement our Secure Universal Multimedia Access system (SUMA) for SVG content in the following three subtasks. For content adaptation, we based on XML/RDF, CC/PP and XSLT, for signing and authenticating SVG content we use Merkle hash tree technique and for content delivery, we develop a mechanism for dynamic delivery of multimedia content over wired/wireless network.
We present Signature scheme and an access control system that can be used for controlling access to SVG documents. The first part of this paper briefly describes the access control model on which the system is based. The second part of this paper presents the design and implementation of SUMA adaptation engine. SUMA aims to deliver an end-to-end authenticity of original SVG content exchanged in a heterogeneous network while allowing content adaptation by intermediary proxies between the content transmitter and the final users. Adaptation and authentication management are done by the intermediary proxies, transparently to connected hosts, which totally make abstraction of these processes.

1 INTRODUCTION

Delivering multimedia content to Web enabled mobile devices such as phones and PDA presents a challenge; different devices have different content presentation requirements. One of the most exciting developing technologies for this field is Scalable Vector Graphics (SVG) (W3C, 1998). SVG files are compact and provide high-quality graphics on the Web, in print, and on resource-limited handheld devices. SVG is exciting because it offers Web developers a method to create and animate images through an XML programming language.

SVG is a language for describing two-dimensional graphics and graphical applications in eXtensible Markup Language (XML). SVG 1.1 (W3C, 2003b) is a W3C Recommendation and forms the core of the current SVG developments. SVG 1.2 is the specification currently being developed as is available in draft form. The SVG Mobile Profiles: SVG Basic (SVGB) and SVG Tiny (SVGT) (W3C, 2003a) are targeted to resource-limited devices and are part of the 3GPP platform for third generation mobile phones.

To build a Universal Multimedia Access (UMA) system delivering SVG content in a heterogeneous network, we have to be able to adapt SVG full to SVG mobile profiles or other compliant format accepted by the handheld terminals; this is possible using adaptation techniques. There are however some drawbacks and one of the major drawbacks at the moment is the security problem. In the this area, while controlling access to text-based documents has been the focus of many research activities (Ferraiolo et al., 2003), raster graphic information has been seldom considered, mainly because of its monolithic internal structure. However, XML-based vector images present new and challenging feature protection problems, related to fine-grained access control to their internal structure. We have then defined a novel approach to fine-grained feature protection of SVG data. The proposed model allows to selectively transform SVG data according to the user's profile, thus releasing only the features that the user is entitled to see.

In this paper we present the design and implementation of an SVG based UMA system. In particular, the platform is able to allow proxy adaptation operation or third party distribution of the exchanged SVG documents.

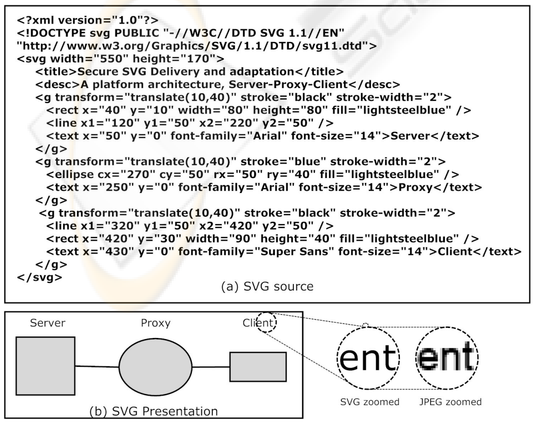
Reda Kaced A. and Moissinac J. (2007).
SVG BASED SECURE UNIVERSAL MULTIMEDIA ACCESS.
In *Proceedings of the Second International Conference on Signal Processing and Multimedia Applications*, pages 326-331
DOI: 10.5220/0002141103260331
Copyright © SciTePress

1.1 Overview of SVG

SVG supports scripting and animation, so is ideal for interactive, data-driven, personalized graphics. Animation techniques can range from a simple linear movement to 3D double helix morphing effects. Web developers, once they are more aware of the possibilities, can find unprecedented levels of control. The broad support behind SVG comes from its many advantages as the following short feature list demonstrates:

- SVG files are pure XML and can be read and modified by a large range of tools (e.g. notepad)
- SVG files are smaller and more compressible than JPEG or GIF images
- SVG images are scalable
- SVG images are zoomable without degradation
- Text in SVG is selectable and searchable

In SVG two-dimensional graphics are described using XML. Therefore a set of basic shape elements is defined e.g. lines, rectangles, circles, ellipses, polygons, paths, etc. The position, size, color, etc. of each shape is defined by attributes. Additionally there are special tags to group shapes <g> and reference self-defined symbols <use>. SVG provides a possibility to add alternative descriptions of the content. Therefore each element can include a title and a description tag (<title> and <desc>). The SVG source code example in Figure 1 describes a server-proxy-client architecture. Each component (that we call *object* in the rest of this paper) in the path is represented in a group (<g>) that contains the component itself, the text, and the link paths associated. Our solution can also use finer granularity where objects are straightforwardly basics shape elements.



```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg width="550" height="170">
   <title>Secure SVG Delivery and adaptation</title>
   <desc>A platform architecture, Server-Proxy-Client</desc>
   <g transform="translate(10,40)" stroke="black" stroke-width="2">
     <rect x="40" y="10" width="80" height="80" fill="lightsteelblue" />
     <line x1="120" y1="50" x2="220" y2="50" />
     <text x="50" y="0" font-family="Arial" font-size="14">Server</text>
   </g>
   <g transform="translate(10,40)" stroke="blue" stroke-width="2">
     <ellipse cx="270" cy="50" rx="50" ry="40" fill="lightsteelblue" />
     <text x="250" y="0" font-family="Arial" font-size="14">Proxy</text>
   </g>
    <g transform="translate(10,40)" stroke="black" stroke-width="2">
     <line x1="320" y1="50" x2="420" y2="50" />
     <rect x="420" y="30" width="90" height="40" fill="lightsteelblue" />
     <text x="430" y="0" font-family="Super Sans" font-size="14">Client</text>
   </g>
</svg>
```

Figure 1: Example of SVG document.

1.2 Universal Multimedia Access

Universal Multimedia Access (UMA) means the access to multimedia information by any terminal through any network (Mohan et al., 1999). The objective of UMA systems is to make available different presentations of the same information, suiting different terminals, networks and user preferences (Butler et al., 2002), (Kirda et al., 2001). This can be achieved through customizing the content to the environment where it shall be consumed.

This content customization (adaptation) can be implemented in three different places: 1) at the content server, 2) at a proxy side, and 3) at the user terminal.

The proxy-based adaptation is the most suitable for the context change support (Wijnants et al., 2005), (Endler et al., 2005), (Palit et al., 2006). In this case, the proxy is the entity responsible of retrieving client contexts and looking for the eventual changes that may occur. These changes are therefore transparent for the content server. The proxy can transform existing SVG content and thus the content server is not directly involved in the adaptation.

Diagram in Figure 2 illustrates the UMA system and its major elements. The developed system implements the customization engine at the proxy side. Content server needs to consider adaptation *a posteriori* by taking into account user's profile and device capability.

Server this dimension applies to the users that is the content source, and do not especially consume content as end-users but perform any action regarding the content, in principle easing the consumption task for an end-user. A server can be a user that creates content, authenticates content, distributes content and so on.

Proxy this dimension is introduced as an intermediary between the content provider and the client device. It acts as transcoding proxy and/or cache proxy to deliver the right (adapted) version of an original content to the client.

Client this dimension applies to the user that receives content for consumption and wishes to have the content tailored to its user environment, the so-called end-users.

As shown in Figure 2, our UMA mechanism considers three types of actors and two kinds of access channels between these actors. The three actors are devices, servers, and proxies; the two access channels are a network access between device and proxy, and a content access between proxy and server. From the

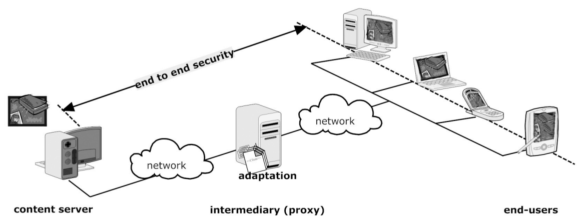


Figure 2: UMA infrastructure in SUMA.

users' point of view, an adaptive and seamless connection of the two access channels results in a universal access channel to the content. From the content point of view, a dynamic delivery of adaptive content results in a universal content presentation to the users.

1.3 Motivation

Let us consider for example a tele-health multimedia application using Internet to distribute SVG recordings (images, videos, and sounds) of doctors and confidential data. Multimedia objects accessed by the application users are content unaware and annotations emptied. Users are equipped with different devices (PC, Laptop, PDA, Smartphone, etc.). Exchanged contents have to be adapted to each device profile using adaptation proxies (they can also add content like advertisements...). For security and confidentiality reasons, exchanged contents must be authenticated. Moreover some parts of the SVG presentation appearing in these recordings is highly classified and should not be changed by unauthorized users (including proxies). So the problem here is how can we allow adaptation operations by intermediaries keeping the authentication and confidentiality of the exchanged contents?

Solving such issue using current security techniques and access control models is very hard as the content is not already defined. In reality, the authorization manager should be able to specify authorization rules and policies by stating the features of such intermediaries.

2 SYSTEM ARCHITECTURE

In this section we describe SUMA (*Secure Universal Multimedia Access*) our proxy-based content adaptation framework. The main architecture type of SUMA system is shown in Figure 3. In this approach, the user takes the UMA Engine as its proxy, which will then make the request to the content server on behalf of this user. The content server response is then received by the UMA Engine, which afterwards decides and performs the adaptation (if needed), and then sends the transformed content back to the client.

The idea is based on the concept of create content once and used with various presentation. Instead of preparing different formats for the different devices, the server use standard SVG files, XML and XSL to describe Web content and presentation, respectively. In the proxy side, we find an *adaptation engine*, implemented with XSLT, and a *security engine* which is used to ensure security and authentication of the exchanged content. To ensure signature validity, the client has an *authenticity verifier* which tests the correctness of the received signature; it is also equipped with an *SVG reader* or in the worst case a "adapted format" reader. In the following we describe each one of these components.

2.1 Content Authentication Engine

There are three phases available in order to build a secure SVG content to send from the server to a client. Once a client logs in the system and requests the multimedia data, the server by the *Access Control Policy Generator* (ACPG) verifies user's identification and the related permission, and generates an adaptive version which specifies what parts of the SVG can be adapted by the intermediary proxy. After that, the service provider signs this content using technique described in 2.1.2 and sends it to the client passing by the proxy.

2.1.1 ACPG

In SUMA, access control services simultaneously support a Role-Based Access Control (RBAC) model (Ferraiolo et al., 1995) and an access control model based on Access Control Lists (ACL) (Barkley, 1997). The RBAC model incorporates a user-role assignment relation, a role hierarchy and a permission-role assignment relation. Three user-roles are defined *owner*, *adapter* and *consumer*. The ACL model provides a straightforward way of granting or denying access to a given resources for specified users.

The implementation for the prototype of ACPG has been done using the JAVA Standard Edition. Therefore, the XACML 2.0 (Anderson, 2004) is chosen to be implemented in the authorization service using sunxacml1.2 (SUN, 2003). The RBAC policies and ACL policies have been built using sun's XACML APIs. The generated RBAC policies and ACL policies are shown in Figure 4. An RBAC XACML *PolicySet* contain *role <PolicySet>* (RPS) and *permission <PolicySet>* (PPS). The RPS associates holders of a given role attribute and value with a PPS that contains the actual permissions associated

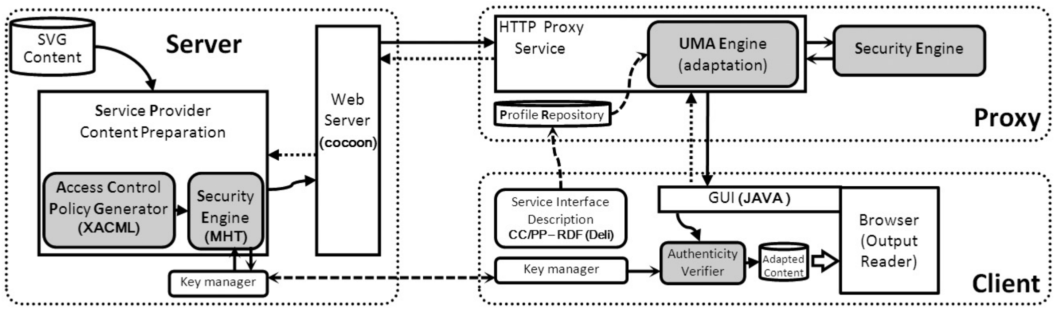


Figure 3: SUMA architecture.

with the given role. This means each RPS references a single corresponding PPS.

The SVG document is parsed by the ACPG, and generate a secure version where every adaptive object is tagged with the access policy associated with. This version is transmitted to the Security Engine (SE) to be signed.

2.1.2 Signature Scheme

To ensure the authenticity of the exchanged SVG document, we use AMCA, a Merkle Hash Tree (MHT) based signature technique which we have described in (Kaced and Moissinac, 2006) adapting it to SVG content. The method we propose allows a subject to prove source authenticity as well as the authenticity of the content of a document.

To accomplish this goal, the idea is to associate a hash value with each node in the graph representation of an XML document. More precisely, the hash value associated with an attribute is obtained by applying a hash function over the concatenation of the attribute value and the attribute name. By contrast, the hash value associated with an element is the result of the same hash function computed over the concatenation of the element content, the element tag name, and the hash values associated with its children nodes, both attributes and elements. Hash values associated with the nodes of an SVG document are computed by the Merkle hash function.

Concretely, the SE starts by adding new empty objects (called *freeleaves*) to the SVG, each one is associated to an adaptive object. After, it uses AMCA to sign the SVG resulted and sends it to the proxy. This last proceeds to the adaptation operations putting the adapted objects in the *freeleaves* signing them itself. The end-user when receives the adapted version, can recompute the signature of the sender using AMCA techniques. Please refer to (Kaced and Moissinac, 2006) for more details of these techniques.

2.2 Content Adaptation Engine

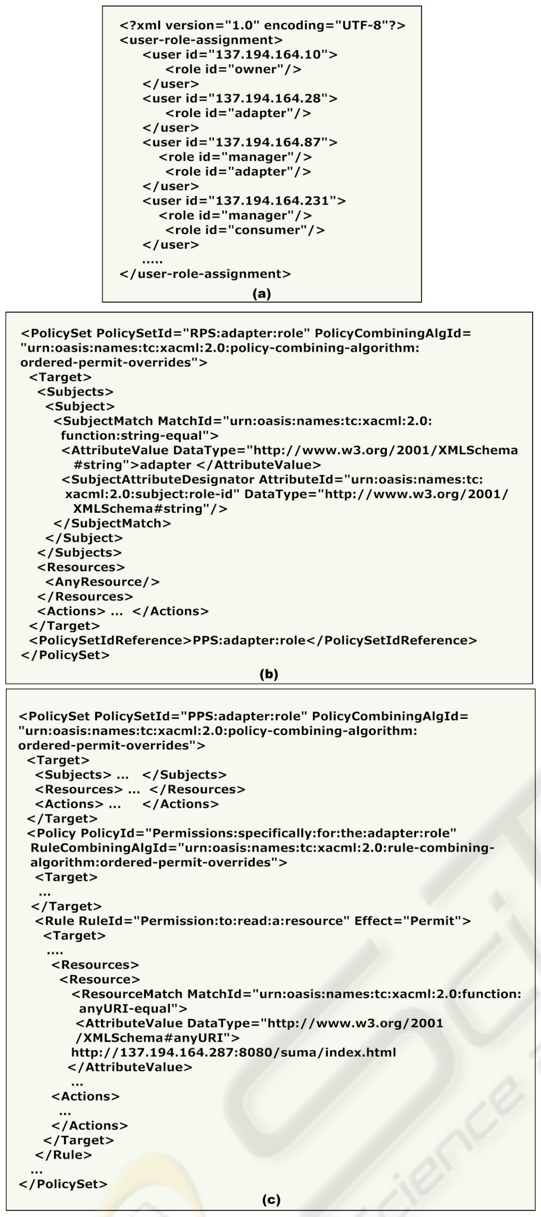
An adaptation method $\mathcal{D}$ is applied on a content $\mathcal{M}$, if the profile description of the receiver $\mathcal{R}$ matches the profiles input requirement of $\mathcal{M}$, and the profile output description of $\mathcal{M}$ matches the client requirements. To avoid developing static adaptation methods for each kind of resources adaptation, its preferable that the server has methods that provides many outputs depending to the context of their application. This is called dynamic adaptation. A dynamic adaptation is the one that interacts with the current client context. Taking into account the client context by an adaptation method can be achieved at three locations: the client, the server, or the proxy.

In the general case, the client content adaptation is achieved using scripts and rendering styles which are sent inside the content and evaluated during the rendering according to the capability of the user device. Unfortunately, this kind of adaptation has many disadvantages and depends widely on the client processing power.

Dynamic adaptation on the proxy represents a best alternative to deal with the variety of client contexts. It allows the delivery of adapted content directly. In our system, we consider two kinds of adaptations: the structural adaptation (transformation) and the media adaptation. Structural transformations are based on XSLT (Clark et al., 1999) which allows transforming SVG document into other XML-based documents. Media adaptations use specific plug-ins applications.

We developed then, an SVG template for SUMA server, and several transcoding plug-ins for the adaptation proxy:

XML to SVG transcoder. We developed an SVG template which is composed of three nested sets: object templates. background image. and foreground objects. Object templates are used to define specific types of objects and their shapes. Foreground objects use a group of global parameters of translation



```
<?xml version="1.0" encoding="UTF-8"?>
<user-role-assignment>
    <user id="137.194.164.10">
        <role id="owner"/>
    </user>
    <user id="137.194.164.28">
        <role id="adapter"/>
    </user>
    <user id="137.194.164.87">
       <role id="manager"/>
        <role id="adapter"/>
    </user>
    <user id="137.194.164.231">
       <role id="manager"/>
        <role id="consumer"/>
    </user>
    .....
</user-role-assignment>
```

(a)

```
<PolicySet PolicySetId="RPS:adapter:role" PolicyCombiningAlgId=
"urn:oasis:names:tc:xacml:2.0:policy-combining-algorithm:
ordered-permit-overrides">
 <Target>
  <Subjects>
   <Subject>
    <SubjectMatch MatchId="urn:oasis:names:tc:xacml:2.0:
     function:string-equal">
     <AttributeValue DataType="http://www.w3.org/2001/XMLSchema
       #string">adapter </AttributeValue>
     <SubjectAttributeDesignator AttributeId="urn:oasis:names:tc:
      xacml:2.0:subject:role-id" DataType="http://www.w3.org/2001/
       XMLSchema#string"/>
    </SubjectMatch>
   </Subject>
  </Subjects>
  <Resources>
   <AnyResource/>
  </Resources>
  <Actions> ...  </Actions>
 </Target>
 <PolicySetIdReference>PPS:adapter:role</PolicySetIdReference>
</PolicySet>
```

(b)

```
<PolicySet PolicySetId="PPS:adapter:role" PolicyCombiningAlgId=
"urn:oasis:names:tc:xacml:2.0:policy-combining-algorithm:
ordered-permit-overrides">
 <Target>
  <Subjects> ...   </Subjects>
  <Resources> ...  </Resources>
  <Actions> ...    </Actions>
 </Target>
 <Policy PolicyId="Permissions:specifically:for:the:adapter:role"
  RuleCombiningAlgId="urn:oasis:names:tc:xacml:2.0:rule-combining-
  algorithm:ordered-permit-overrides">
  <Target>
   ...
  </Target>
  <Rule RuleId="Permission:to:read:a:resource" Effect="Permit">
   <Target>
    ....
    <Resources>
     <Resource>
       <ResourceMatch MatchId="urn:oasis:names:tc:xacml:2.0:function:
        anyURI-equal">
        <AttributeValue DataType="http://www.w3.org/2001
        /XMLSchema#anyURI">
       http://137.194.164.287:8080/suma/index.html
       </AttributeValue>
        ...
    <Actions>
     ...
    </Actions>
   </Target>
  </Rule>
 ...
</PolicySet>
```

(c)

Figure 4: ACPG role-based access control, (a) a user-role assignment configuration file, (b) Role <*PolicySet*>. for the adapter role, (c) Permission <*PolicySet*> for the adapter role.

and scaling, that can he changed while the SVGs are cropped or scaled. Individual object has its translations and scaling parameters relative to a group origin point.

SVG to SVG transcoder. Based on the client device profile, SUMA transcodes an SVG based on the viewbox, global shift of the foreground objects and cropping/scaling of the background image in SVG. Also, we embed the manipulated background image into the SVG to reduce the work and transmission load for proxy load.

SVG to Mobile SVG transcoder. Mobile SVG includes two profiles - SVG Basic and SVG Tiny. We develop this transcoder for converting general SVGs into the format that is compliant to the SVG profiles used by the Pocket SVG Viewer. Some transcoding is needed such as the font style of text and strict requirements of DTDs in the SVG headers.

SVG to HTML transcoder. This transcoder is based on an XSLT stylesheet. For a non-SVG rendering devices, general browsers that reads HTMLs can be used to displayed a transcoded SVG files that have been converted to HTML and an image. The hyperlink properties of the original SVGs are utilized by the image maps inside the HTML.

SVG to Image transcoder. Batik (Apache, 2004) is a Java-based toolkit for applications or applets that want to use images in the SVG format for various purposes. Batik provides building blocks that developers can assemble in various ways in their Java technology applications or applets to generate, parse, view or convert SVG contents. Batik can convert SVG content into other formats such as JPEG, PNG or Tiff or other formats (transcoder API). We use Batik to generate a plug-in which converts the original SVG images to either JPEG or GIF.

3 EXPERIMENTS

SUMA performs two preliminary operations: content signature and content adaptation. We deployed SUMA over a wireless LAN 802.11g environment. Any browser of devices can set the URL of the adapter server as its HTTP proxy. We have tested the system on desktops, laptops, HP Pocket PC (Win-mobile device), and Sony Mobile Phone Browser. Figure 5 shows an example of the SUMA content triggered by different transcoding profile. It shows the transcoded content using the Pocket IE browser on a PocketPC device. It also shows the intelligence information associated with the objects on the SVG. we can see also the transcoded result on the Sony w900i Mobile phones. All figures and property sheets are transcoded to HTML and JPEG images.

An advantage of the proposed system was observed. After our presentation at the debut demo, an audience challenged whether this system can let his Sony Clie, a Palm-OS device which was not originally supported, access the SUMA content. Within 10 minutes, we added this device to the repository profiles, helped the device accessing the wireless LAN, and set the HTTP proxy. Afterwards, it could access the customized SUMA content right away. This example

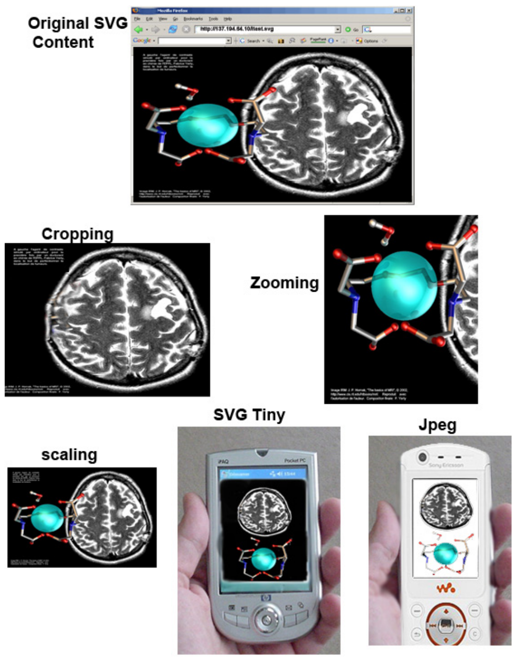


Figure 5: Example of the SUMA content triggered by different transcoding profile.

showed the flexibility and usability of the proposed platform.

4 CONCLUSION

In this paper, we have highlighted the challenges for SVG delivery access control and presented SUMA framework, a Universal Multimedia Access system which ensure end-to-end data authentication.

SUMA provide a proxy-based adaptation for SVG content. It is based on two major parts, a Content Authentication Engine which uses Merkle Hash Tree signature technique to allow adaptation operations keeping possibility of signature verification in the client side, and a Content Adaptation Engine, which implement some plug-ins to convert standard SVG content to transcoded format for different profile devices.

The major advantages of SUMA is the ease of maintenance and the end-to-end security of the exchanged content. However SUMA does not implement content Encryption which is a big drawback for the moment. This is our next challenge.

REFERENCES

Anderson, A. (2004). XACML Profile for Role Based Access Control (RBAC). *OASIS Access Control TC Committee Draft*, 1:13.

Apache (2004). Batik, a java-based toolkit for scalable vector graphics (svg). http://xml.apache.org/batik/.

Barkley, J. (1997). Comparing simple role based access control models and access control lists. *Proceedings of the second ACM workshop on Role-based access control*, pages 127–132.

Butler, M., Giannetti, F., Gimson, R., and Wiley, T. (2002). Device independence and the Web. *Internet Computing, IEEE*, 6(5):81–86.

Clark, J. et al. (1999). XSL Transformations (XSLT) Version 1.0. *W3C Recommendation*, 16(11).

Endler, M., Rubinsztejn, H., da Rocha, R., and do Sacramento Rodrigues, V. (2005). *Proxy-based Adaptation for Mobile Computing*. PUC.

Ferraiolo, D., Cugini, J., and Kuhn, D. (1995). Role-Based Access Control (RBAC): Features and Motivations. *Proceedings of 11th Annual Computer Security Application Conference*, pages 11–15.

Ferraiolo, D., Kuhn, D., and Chandramouli, R. (2003). *Role-based access controls*. Artech House Boston.

Kaced, A. R. and Moissinac, J.-C. (2006). Multimedia content authentication for proxy-side adaptation. In *Proceedings of the IEEE International Conference on Digital Telecommunications, ICDT 06*.

Kirda, E., Kerer, C., Jazayeri, M., and Kruegel, C. (2001). Supporting multi-device enabled Web services: challenges and openproblems. *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001. WET ICE 2001. Proceedings. Tenth IEEE International Workshops on*, pages 49–54.

Mohan, R., Smith, J., and Li, C. (1999). Adapting Multimedia Internet Content for Universal Access. *IEEE TRANSACTIONS ON MULTIMEDIA*, 1(1).

Palit, H. N., Chi, C.-H., and Liu, L. (2006). Proxy-based pervasive multimedia content delivery. *compsac*, 1:255–264.

SUN, M. (2003). Sun's XACML implementation. *For information about this implementation see web site http://sunxacml.sourceforge.net*.

W3C (1998). Scalable vector graphics (svg). http://www.w3.org/Graphics/SVG/.

W3C (2003a). Mobile svg profiles: Svg tiny and svg basic. http://www.w3.org/TR/2003/REC-SVGMobile-20030114/.

W3C (2003b). Scalable vector graphics (svg) 1.1 specification, w3c recommendation 14 january 2003. http://www.w3.org/TR/SVG11/.

Wijnants, M., Monsieurs, P., Quax, P., and Lamotte, W. (2005). Exploiting proxy-based transcoding to increase the user quality of experience in networked applications. *aaa-idea*, 0:73–80.