X316 : $_{13}$Morph $_{1}$Access $_{16}$Pass
A New Extensible Model of Certification

Research report

Rachid Saadi[1], Jean Marc Pierson[2], and Lionel Brunie[1]

[1] LIRIS lab, INSA de Lyon, France.
{rachid.saadi,lionel.brunie}@liris.cnrs.fr,
[2] IRIT lab, Universit Paul Sabatier of Toulouse, France.
jean-marc.pierson@irit.fr

Abstract. In distributed systems, and especially in pervasive environments, the users face usually difficulties to obtain permissions to use the resources of the sites they visit. Either they have an account locally, or they can show a certificate obtained beforehand that gives them the access. Problems arise when they arrive in a unknown environment, which is likely to occur in pervasive environment. In this paper, we propose the Chameleon architecture which offers users such a possible access if they appear with trusted certificates. These certificates may be gathered during their roaming in the pervasive environment, and they offer direct or transitive access to foreign sites, based on trust relationships. We describe a new kind of certificate (the X316 certificate), to set up a contextual broader access, giving the possibility for the users to mask parts of their personal information embedded in the certificate and to show only what is mandatory for a given access.

1 Introduction

Pervasive environments are an innovative field of research which enable large access to information for any user, dynamically, with respect to different constraints of the user's context, such as position, device capabilities and network conditions. The increase in interest for the related technologies (wireless network, light devices) introduces new security challenges, where the existing security models and mechanisms are inadequate. Among existing solutions, the trust paradigm seems the most flexible approach. Indeed, in the vision of ubiquitous computing [1] especially the pervasive environment, using trust model is the most suitable solution to extend users' access scope. Furthermore, in distributed system, when a trust based approach is set up, a certification mechanism is mostly used.

In the pervasive environment, each nomadic user roams from site to site and would like to interact with surrounding users and resources. To gain an access in this environment, she must prove her rights by using credentials or

certificates. Unfortunately, all exiting models of certification must be modified and improved to satisfy the pervasive requirements such as: contextual adaptation, multi-user devices, mobility etc. In this paper we define a trust architecture called Chameleon using a new model and mechanism of certification called X316. This model gives to site administrator a general formalism to express and customize a certificate according to member's rights and user's devices capabilities. Moreover, a new method of signature is defined allowing the owner of the certificate to adapt it freely according to context by selecting only the corresponding information in the X316.

This paper is organized as follows. Section 2 presents the background of our proposal. Next, in section 3 we introduce our approach that delineates the Chameleon Architecture and show its implementation in the pervasive environment using the X316 certificate. Then we describe the X316 in section 4. In section 5 we discuss benefits and the scalability of our certification model with experiences. Finally, we conclude this paper along we suggested future directions.

2 Background

2.1 A vision of pervasive environment

The challenge of pervasive paradigm is to allow each nomadic user to roam and access inside this environment easily and transparently, by exceeding certain barriers like the heterogeneity of the different access policies. Let's consider the following use case, as illustrated in the figure 1. Let Pr Bob be a member of University A. This Professor goes to a conference in University B and goes to a meeting in University C. He communicates with the different surrounding "objects" including students, professors and resources e.g. printer, video projector etc. In fact, Bob owns a professional card or conference badge that defines his status and includes a picture or a fingerprint to identify his identity. This card or badge allows Bob an access inside these universities according to a convention or shared collaboration (the same work group). These Universities do not know the owner of the card, but trust his card.

If we map this scenario in the pervasive environment, universities correspond to sites. A certificate simulates the professional card ; the fingerprint or the picture is seen as an authentication system embedded in the certificate. In this manner, if Bob has the right to attend a conference, according to his certificate, he obtains a new temporary certificate (like a badge in a conference). This certificate allows Bob to access resources inside this new site like all other members.

In this paper we use the following terms:

- **Site:** represents an organization, domain or host that implements a local independent security policy and is limited geographically,
- **Environment:** is composed by sites like universities, restaurants, posts office, airports etc.
- **Context:** describes the user environment at a certain time : Trust, the current transaction, the entity which she interacts,...

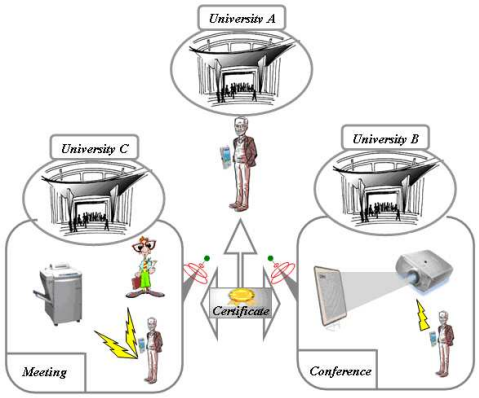

Fig. 1. Pervasive scenario

- **Profile:** each user has a profile, depending on the access policies, it can represent a role (student, doctor) or an access level (trust, distrust, confidential) etc.
- **Certificate:** it represents a digital passport of the users. one user owns some certificates (like professional cards) that prove her membership to each site.

2.2 Certification mechanism

The Certification mechanism is a service based on digital signature. It uses the concept of Public Key Infrastructure(PKI) to provide a security privilege based on the trust accorded to the signatory. This mechanism is implemented to authenticate the certificate contents and to implement a distributed system based on trust.

Two types of certificate can be considered:

1. The identity certificate: It enables to check the user identity (the public key).
2. The attribute (authorization) certificate: It enables to give a permission to an entity.

All current certification models combine these two aspects to increase its usability scope for new services and technologies requirements. In the literature, some certification models are standardized and formalized e.g. PGP(Pretty Good Privacy) [2], SPKI [3], Sygn [4], X509 [5], Akenti [6].

X509 is the most used standard. However, it has first been designed as an identity certificate, and its last extension proposed to extend its scope to attribute certificate. Unfortunately, the usability of the new extension is deemed to be too complex and requires adaptations(depending on security policies e.g. RBAC), like in PRIMA [12] and PERMIS [11] which modify the X509 attribute format to extend its capabilities.

SPKI was proposed to become an alternative to X509, SPKI focuses on authorization certificates more than identity certificates. The objective of SPKI

is simplicity. Unlike the X509, which is based on ASN 1.0 [7] format, SPKI certificate is described in XML [21] offering more flexibility and readability.

These last models of certification have some drawbacks. In fact, all of them identify one user only with her public key using a challenge-response mechanism [22]. But, each nomadic user owns multiple devices with different capacity (computing power) and capabilities (biometric identification,...). One certificate should embed more than one identification offering to user different manners to authenticate her certificates. Furthermore, on one hand, the certificate contains, more and more information (sensitive or public) and, on the other hand the context is very important in pervasive environment. Thus, the certificate contents should be adapted according to context.

2.3 Morph mechanism

We define the morph mechanism to perform the certification contextual adaptation. It represents the ability to hide some attributes on a signed message according to context. Steinfeld and al [23] define this property as CES (Content Extraction Signature): "*A Content Extraction Signature should allow anyone, given a signed document, to extract a publicly verifiable extracted signature for a specified subdocument of a signed document, without interaction with the signer of the original document*".

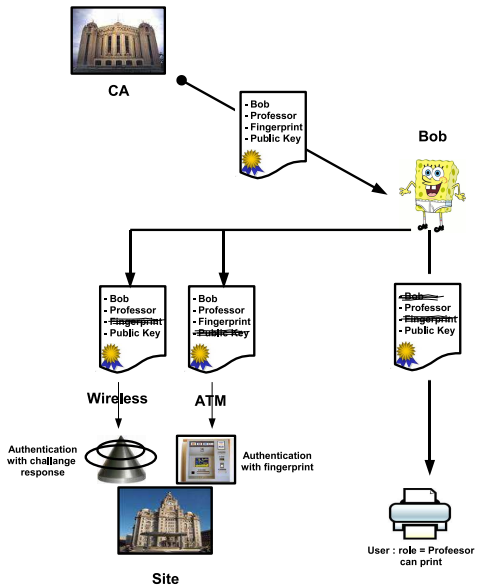

Fig. 2. Morph certificate

The most used approach divides the massages into fragments, then signs each one separately. Micaly and Rivest [8] is the first work which introduces the concept of transitive signature. In their algorithm, giving a signature on two graph edges Sig(x,y) and Sig(y,z) (where x, y and z represent subdocuments), a valid signature Sig(x,z) can be computed without access to a secrete key. Johanson and al [9] have introduced some improvements by enabling a homomorphic signature. By giving a signature Sig(x), anyone can compute a signature Sig(w) on any subpart w of x obtained by rubbing out some position of x.

[10] is the first works which use homomorphic function property to define a new signature algorithm for morphing certificates.

All previous approaches have a drawback, they define a new algorithm to perform the certificate adaptability, instead of using the existing standard.

[23] exposes a modification of the RSA computing algorithm, their approach is based on the homomorphic property of RSA, i.e. $h_1^d h_2^d mod N = (h_1 h_2)^d mod N$. This algorithm multiplies the RSA sub-messages signatures, and checks whether the result is the signature of the hash values products.

Their approaches are very useful. But, they are based on mathematical proprieties to aggregate and reduce signature length. This may decrease the PKI's reliability

The World Wide Web Consortium "W3C" standard: "XML Digital signature"(XMLDSig) [13] offers the capability to sign different parts of documents. This concept is very interesting, but it is not appropriate in a certificate model. Indeed, a credential or certificate is not composed of distinct parts, but composed of a single bloc. In fact, [24] add some elements to the XMLSignature standard to perform the certificate adaptability.

However, the last approaches treat the certificate as any documents. Each document is decomposed in several sub-documents. Consequently, the user is free to put or hide some parts according to context. However, in the certificate, we must deal with two different fields:

- **Static field** (certificate identifier, issuer identity, time of validity...). It represents the irremovable certificate data.
- **Dynamic field** (user name, user rights...) represents the removable data. It is restricted by the certificate issuer, which allows the user to remove only these data.

3 The Framework: the CHAMELEON

The main characteristic of pervasive environment is the mobility. Indeed the user roams inside the environment and tries to access or use some surrounding resources and services. The question: ***How the local site policy can control and authenticate these unknown guests?***

To extend the access scope of the users and enhance the access control policy, our architecture work as a front-end of each site which provides an authority on its resources. It is called "**Chameleon**" similarly to chameleon which has the

capability to fit in its environment. Therefore, when the user get to the proximity of each site she becomes like a local user.

To set up this architecture we define three modules:

- **TMM *Trust Manager Module:*** is applied between sites which are considered as trusted sites.
- **MAM *Mapping Access Module:*** is applied for giving an access to foreign users.
- **CIM *Credential Issuer Module:*** is used to authenticate the user's profile.

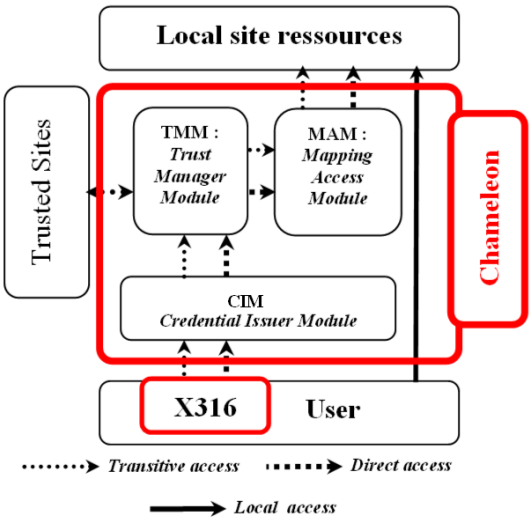

Fig. 3. Chameleon Architecture

3.1 TMM Trust Manager Module

Each mobile user belongs to one or many sites. To recognize this user anywhere, the TMM works between sites and enables them to communicate and to share some informations about their members. In fact, each site X builds its trust set. It is composed by the sites that X trusts, we call these: "trusted sites". Then, to evaluate the trusted site's trustworthiness, X gives to each of them a corresponding degree using a trust function called t^0. In this manner, the environment can be seen as a graph, and we note $T_g(S, E)$ a valued and directed graph such that:

- The nodes of the graph represent the sites of S.
- Each trust relation between two sites is represented by a directed edge e. The set of edges is consequently identified with the set of relations, E.

– Each edge is valued by the distrust degree between the sites represented by the source and destination nodes of this edge.

TMM module uses this trust graph to decide if a "foreign" user can access the local site (i.e. to decide if a user who does not own an account of the system can get logged in the system).

We will consider three different access: *local access, direct access and transitive access.*

– A local access: is provided by the home site to all registered users (i.e. where they have their accounts and can authenticate themselves).
– A direct access: is provided by the TMM module of local site to all users registered in trusted sites. This direct access is valued by the trust degree between the local and the trusted site.
– A transitive access: can be provided by a local site X to a user who does not belong to its trust set, under a condition that a valid trust chain between one of the user's home sites and trusted site exists in the graph. This transitive access is valued by the trust propagation degree between these two sites (as before, in case of the existence of several possible chains, the TMM is responsible for choosing the reference chain). To manage the users' access, each site has to define thresholds beyond which access is not allowed.

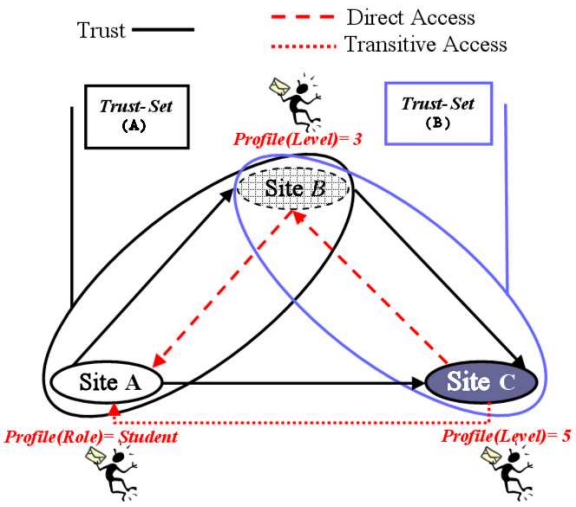

Fig. 4. The Trust Propagation

3.2 MAM: Mapping Access Module

This module defines a mapping policy which is defined between the local and the trusted users' profiles. When a user is allowed to access a site X, the MAM

module attributes to her a new profile using a mapping policy. This profile defines the user's rights inside the site X and is called : "A-profile" (Analogue profile).

A mapping policy can be simply implemented by creating a mapping table to store the correspondences between foreign profiles of trusted sites and their analogue profiles, for example(see fig 4): The user Alice has, in her home site B, a profile equal to level 3, Alice wants to access site A, this site trusts B. Then, A applies a mapping policy and maps from her original profile "Level 3" into a local profile corresponding to "STUDENT role". The mapping policy can be fulfilled between various policy models e.g. RBAC [17], MAC [16] or DAC [15]. However, it should be applied between sites which generally uses a similar policy. For example: in a medical community, it is probable that roles such as "Doctor", "Nurse" or "Patient" exist in all organizations, allowing for an easy mapping through the community. Furthermore, this mapping policy can be enhanced by looking for user's capabilities, such as delegation right and resources access, etc.

3.3 CIM: Credential Issuer Module

The main objective of our approach is to allow any user to roam and access into various sites freely from the site where she is member (home site). Consequently, each site fulfill a policy based on certification mechanism, allowing its members to prove their rights in the environment. The CIM constitutes the module which has the responsibility to manage the local certification policy, it can give and generate to its members or trust users some certificates.

All existing certification model has some drawbacks. In fact, one confronted to a new situation the user must contact her home site to have a corresponding certificate, thus she must manage several certificates.

Our approach consists of using an unique certificate. This one contains all user's privileges in her home site/trusted site. It is attractive, because the user manages only one certificate by site and adapts it according to context.

4 X316: Morph Access Pass Certificate

In the Chameleon architecture we define a new format for certificate called X316: Morph Access Pass Certificate. This format facilitates creating any sort of certificates or credentials e.g. Attribute certificate, Role certificate etc. This "X316" works as a pass, allowing its owner to roam and gain access in the environment.

This certificate mainly testifies the user's profile (status or access level) and rights in a Home/Trusted site. If the user wants to access a particular target site, her devices selects one of her certificates which is recognized by this one. Then, according to it, the target site applies a mapping policy between the its policy and the policy held in the credential.

Our contribution has an objective to define a very flexible model of certification. It is inspirited by the W3C standards: "XML Digital signature"(XMLDSig) and "XML Encryption" (XMLEnc) [14]. The X316 is designed for nomadic user. Indeed, unlike all certification system, the same X316 certificate can be used and

authenticate from various devices with different capacity and characteristics, and can be generated dynamically along to user trip. In fact, by defining specific tags to delimit the dynamic parts, this certificate acquires the capability to transform and to morph easily its content according to context, situation, and environment.

Therefor, the X316 fulfills three constraints:

- Format Flexibility.
- Multi authentication.
- Contextual adaptation.

4.1 X316: Type

X316 could be obtained by two different ways:

- Each site gives a Home Certificate or H316, to all its members (local access).
- Each site gives a Trust certificate or T316, to a guest, when it trusts her Home Site (direct/transitive access).

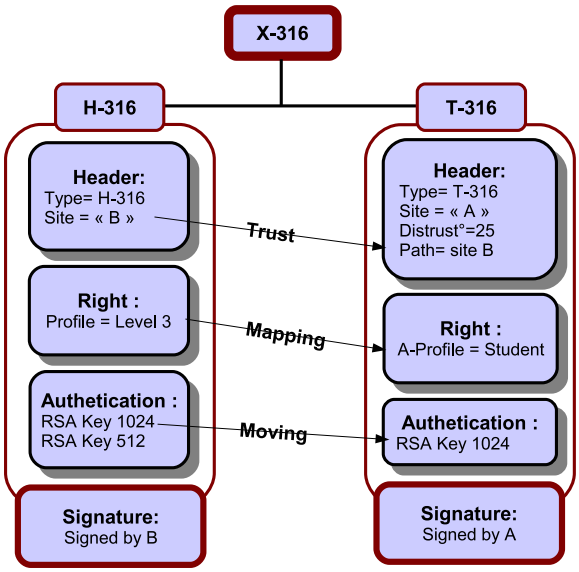

Fig. 5. X316 Type

H316: Home certificate
Each site delivers to all its members an H316 that denotes members' rights. The H316 contains four parts: ***the header, the right, the authentication, and the signature***. (see figure 5)

T316: Trusted certificate
The trust set of each site "A" contains two kinds of trusted sites (see figure 6):

- ***Near trusted sites:*** represents the trusted sites that can communicate with A (directly or indirectly).
- ***Distant Trusted sites:*** represents the trusted sites that can not communicate with A.

If a site does not recognize a user, it asks the near trusted sites about her home site. The request response could be forwarded if one site is unable to answer. This way, each user will obtain a direct access (if she belongs to a trusted site) or a transitive access (if she belongs to a community of the near trusted sites). Consequently, if she belongs to one of distant trusted sites, the user can't obtain any access.

For this reason, the access scope of the user can be increased (in another manner) by using the trusted X316 (T316). Each site delivers a T316 for the guest user if it trusts her X316. Thus, using the T316, a link between the local site and the distant trusted site is created allowing the users to gain more access.

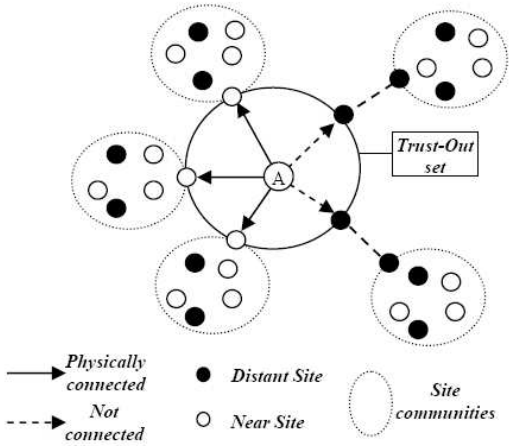

Fig. 6. Community connexion

The T316 is obtained from a main H316/T316 that is used to authenticate user on the target site. The T316 has a short period of validity, e.g. two or three days, as compared to H316 which is valid as long as it is not revoked explicitly by its issuer.

The profile part contains the new analogous profile (fig 5: mapping) provided by the target site to the user, as outcome of the mapping policy. Concerning the authenticator part, the trust certificate embeds the authenticators of the original X316 (fig 5: moving).

4.2 X316: General Format

The X316 certificate is composed by four parts as follow:

1. Header : this part defines the certificate identity.
2. Right : this part defines all user's rights.
3. Authentication: it contains several items allowing user to authenticate and to prove that she is the owner of the certificate.
4. Signature : it corresponds to the signature of the certificate issuer "CA"(Certification Authority : Home/trusted site).

X316 syntax :

```
<X316 id=? xmlns="...X316MCert#"
xmlns:ds="http://www.w3.org/2000/09/xmldsig#"
xmlns:ec="http://www.w3.org/2001/04/xmlenc#">
  <HEADER>
  <RIGHT>
  <AUTHENTICATORS>
  <SIGNATURE>
</X316>
```

The X316 is represented in XML. In the rest of this paper, we use the W3C syntax definition to describe each X316 parts, where "?" denotes zero or one occurrence; "+" denotes one or more occurrences; "*" denotes zero or more occurrences; and "W3C definition" denote the same syntax of the W3C standard.

The HEADER part
This part identifies the certificate. In fact, it is composed of some fields which defines the certificate's identity, such as: The user owner, the CA (Certification Authority), The time of validity etc...

– **Header syntax:**

```
00  <HEADER>
01    <CertifiateID>
02    <TYPE>
03    <CA>
04      <Site>
05        <SiteID>
06        <SiteName>
07      </Site>
08      (<TrustPath>
09        <CertificateID>?
10        (<CA> ...</CA>)?
11      </TrustPath>)?
12      (<SignaturePath>
13        <SignatureMethod Algorithm=/>
14        <ds:SignatureValue>
15      </SignaturePath>)?
16    </CA>
17    (<CertificateDegree>
18      <ComputeMethode Algorithm=/>
19      <Values> <Value id=>+ </Values>
20    </CertificateDegree>)?
21    <Subject>
22      <SubjectId>
23      <SubjectName>
24      <SubjectPhone>?
```

```
25      ...
26    </Subject>
27    <Validity>
28      <NotAfter>
29      <NoteBefore>
30    </Validity>
31  <HEADER>
```

– **HEADER description :** The header part describes mainly four information : the identity of certificate, the certification authority, the user(subject) and the time of validity.
The certificate identity defines the Identifier of the X316 "CertificateID" (line 1) and specifies the certificate type "Type" (Home or Trusted): H316 or T316 (line 2).
The "CA" part (lines 03-16) identifies the certification authority, it is composed by two sections the site identifier "Site" (lines 04-07) and the "TrustPart" (lines 08-15). This latter marks the traceability, i.e. all sites and certificates which contribute to generate this X316. It encapsulates recursively the description of each node of the trust chain. Furthermore, when the local site searches the certificate issuer, the exploration of the trust graph is stopped if one site trusts the utilized certificate. It don't verify the rest of the chain that allows to generate this one. Thus, the TrustPart informs also the "certificateID" of the last X316.
The " SignaturePath" (lines 14-17) permits each node of the original certificate trust path to sign its link with its trusted site. To reduce the generated signature trace, recent works explore some algorithms to involve a sequential aggregation signature [20]. Therefor, unlike all certification systems path that load into the credential all certificates chain, the X316 has the capability to aggregate all chain sites' signature in only one.
Example: In this example, a RSA sequential aggregate signature algorithm [19] is performed. let A,B and C three sites where A trusts B and B trusts C. If Bob a member of C obtains a certificate from A, the corresponding CA part will be generated as follow:
1- A asks B if it recognize the bob's certificate: Cert(CertificateID="587", SiteName="C")
2- B trusts C and sends back to A this following CA signed part:

```
<CA>
  <Site>
    <SiteID>...</SiteID>
    <SiteName> Site B </SiteName>
  </Site>
  <TrustPath>
    <CertificateID> 587 </CertificateID>
    <CA>
      <Site>
        <SiteID>...</SiteID>
        <SiteName> Site C </SiteName>
      </Site>
    </CA>
  </TrustPath>
  <SignaturePath>
    <SignatureMethod Algorithm="...X316MCert/Agg#RSA-SHA1"/>
    <ds:SignatureValue>
```

```
      ASV(B)= RSASigAgg(ASV(C)=0, H(TrustPath))
    </ds:SignatureValue>
  </SignaturePath>
</CA>
```

The Aggregate Signature Value of B (ASV(B)) is the result of the RSA Signature Aggregation (RSASigAgg) algorithm with B's private key . It has as parameters:

- The last ASV(C) (equal to 0 because the original certificate doens't contain a TrustPath).
- H(TrustPath) : the corresponding digest value of the TrustPath using SHA hash algorithm.

3- A generates a trust certificate to Bob with the following CA part :

```
<CA>
  <Site>
    <SiteID>...</SiteID>
    <SiteName> Site A </SiteName>
  </Site>
  <TrustPath>
    <CA>
      <Site>
        <SiteID>...</SiteID>
        <SiteName> Site B </SiteName>
      </Site>
      <TrustPath>
        <CertificateID> 587 </CertificateID>
        <CA>
          <Site>
            <SiteID>...</SiteID>
            <SiteName> Site C </SiteName>
          </Site>
        </CA>
      </TrustPath>
    </CA>
  </TrustPath>
  <SignaturePath>
    <SignatureMethod Algorithm="...X316MCert/Agg#RSA-SHA1"/>
    <ds:SignatureValue>
       ASV(A)= RSASigAgg(ASV(B), H(TrustPath))
    </ds:SignatureValue>
  </SignaturePath>
</CA>
```

In this same manner, the ASV(A) is the result of RSASigAgg algorithm with A's private key, which has as parameters:

- The last ASV(B)
- H(TrustPath) : the hash value of the TrustPath.

Finally, by using the trusted sites (B, C) public keys, the generated signature can be checked.

As defined in the Chameleon architecture the relation between sites is valuated by a trust function. The trust value that the certificate issuer gives to the user's home site represents the certificate degree "CertificateDegree" (Lines 19-22). The defined syntax can describe different trust evaluation system e.g.

```
<CertificateDegree>
    <ComputeMethode Algorithm=....X316MCert/degree#Distrust/>
    <Values> <Value id="DistrustDegree"> 45 </Value> </Values>
</CertificateDegree>
```

Finally, the "Subject" part describes the certificate owner : user identifier, user name..., and the "Validity" part define the period whither the certificate is valid.

The RIGHT part
It is a variable part of the certificate that depends on the site's policy. This part contains information about the user's rights within each site. Indeed, unlike other systems of certification that certify an access to particular resources, this one certifies the user's rights that represent all authorized accesses to the site resources.

This part is divided into two sections:

1. **User profile:** it contains a specific profile such as the user's status or access level in her Home/Trusted Site.
2. **User capabilities:** this part is variable it mainly contains the user's authorized resources, and defines some capabilities.

– **Right syntax**

```
00  <RIGHT>
01    <PolicyType>
02    <Profiles>
03      (<Profile>
04        <ProfileID>
05        <Descritpion>
06      </Profile>)+
07    </Profiles>
08    <Attributes Format=?>
09      <Attribute Format=?>*
10    </Capabilities>
11  </RIGHT>
```

– **RIGHT description:** This part describes mainly two fields : the owner profile and the owner rights attributes.
Line 01: "PolicyType" describes the type of CA's policy e.g. RBAC.
Lines 02-07 : "Profiles" identifies and describes the user's profile e.g. Role= "Doctor".
Lines 08-10 : "Attributes" describes the user's capabilities inside her certificate's provider. This parts is very dynamic and can define any expression policy format e.g XACML.
Example : It defines the role of the user(line 04), and her capability to delegate(line 25) the access to the PrinterID(line 18)

```
00  <RIGHT>
01    <PolicyType> RBAC </PolicyType>
02    <Profiles>
03      <Profile>
04        <ProfileID> Student </ProfileID>
05        <Descritpion>...</Description>
06      </Profile>
07    </Profiles>
08    <Attributes>
09      <Attribute Format="XACML 1.0">
10        <Rule xmlns="urn:oasis:names:tc:xacml:1.0:policy"'
11                    ...RuleId="" Effect="Permit">
```

```
12        <Description>...</Description>
13        <Target>
14          <Subjects><Subject></Subject></Subjects>
15          <Resources><Resource>
16            <ResourceMatch MatchId="">
17            <AttributeValue DataType="">
18               PrinterID
19            </AttributeValue>
20            </ResourceMatch>
21          </Resource></Resources>
22          <Actions><Action>
23            <ActionMatch MatchId="">
24            <AttributeValue DataType="">
25              Delegate
26            </AttributeValue>
27            </ActionMatch>
28          </Action></Actions>
29        </Target>
30      </Rule>
31      </Capability>
32    </Capabilities>
33  </RIGHT>
```

The AUTHENTICATION part

This part permits to identify the owner of the X316. The authenticators are numerous, and related to the variety of devices used in the pervasive environment (PDA, mobile phone, terminals). Thus, facilitating certificates management could be fulfilled by embedding some authenticators according to the device's authentication capabilities and the site's security policy.

Two ways of authentication have been identified, remote and local authentication.

1) Remote authentication: this one is used with any system of communication, in general if the authentication procedure is fulfilled through a distrust communication(e.g. Wireless). The most important system is a challenge response authentication. It uses a key-pair mechanism. The X316 can embed one or several keys depending on the user's devices capabilities e.g. "512bits RSA public key" for a small device like a PDA, and "2048 RSA public" key for a laptop device.

2) Local authentication: This authentication is used if the authentication procedure is locally performed, for example: toward a terminal like the ATM (Automated Teller Machine). This authentication is mostly specified to derive with specific capabilities for example a biometric identification. Recently, some devices integrate a fingerprint identification mechanism. This one can be defined in the certificate as a parameter for an authentication protocol. The majority of the biometric identifications are standardized in XML representation called XCBF (OASIS XML common Biometric Format) [18]. Using XCBF, our certificate can describe any information to verify an identity based on human characteristics such as DNA, fingerprints, iris scans, hand geometry etc.

– **AUTHENTICATION syntax:**

```
00  <AUTHENTICATION>
01    (<Authenticator Type>
02      <AuthenticationMethod Protocol=/>
```

```
03      <Parameters>
04        (<Parameter>
05          <Description>
06          <ParameterValue>
07        </Parameter>)+
08      </Parameters>
09   </Authenticator>)+
10 </AUTHENTICATIONS>
```

– **AUTHENTICATION Description :**
Line 01 : "Authenticator type" defines the sort of Authentication : Local or remote.
Line 02 : "AuthenticationMethod Protocol" defines the protocol of authentication e.g. challenge response.
Lines 03-08 : "Parameters" defines the needed parameters to each authentication method. The definition of these parameters is expressed in standard languages namely : W3C(public key) and XCBF (fingerprint)
Example:

```
00 <AUTHENTICATION>
01  <Authenticator Type=".../X316MCert/Authentication#Remote">
02   <AuthenticationMethod Protocol=".../X316MCert/
03            .............Authentication#Challenge_Response">
04   <Parameters>
05    <Parameter>
06     <Description>"512 RSA PubKey"</Description>
07     <ParameterValue>
08      <ds:RSAKeyValue>
09       <ds:Modulus>
10          vmJ2k9b3JrDhG5LQ8uFFg9h1NO=
11       </ds:Modulus>
12       <ds:Exponent>AQAB</ds:Exponent>
13      </ds:RSAKeyValue>
14     </ParameterValue>
15    </Parameter>
16   </Parameters>
17  </Authenticator>
18 </AUTHENTICATION>
```

This example describes a remote authenticator(line 01). It defines a challenge response protocol(lines 02-03) using a 512 RSA public key(lines 05-15) as a parameter(user identifier).

The SIGNATURE part
This part is divided into two sections:

– The Float part.(see X316 Signature)
– The rest of signature.

The signature part contains the information about the used key and the result of the certificate's signature.

– **Signature syntax:**

```
00  <SIGNATURE>
01    <FloatParts>
02      <FloatPart position=>*
```

```
03    </FloatParts>
04    <Transforms>
05      <Transform Algorithm=/>+
06    </Transforms>
07    <ds:SignatureMethod Algorithm/>
08    (<KeyInfo>
09       <ds:KeyValue>
10    </KeyInfo>)?
11    <ds:SignatureValue>
12  <SIGNATURE>
```

– **SIGNATURE description :**
Lines 01-06 : "FloatParts" and "Transforms" are needed to perform the morph capability (see next section)
Line 07 : "SignatureMethod Algorithm" defines the algorithms to compute the signature namely the hash algorithm (e.g. SHA) and the public key algorithm (e.g. RSA), it uses the W3C notation.
Lines 09-11 : "KeyInfo" defines the public key value as W3C key definition.
Example: In this example, the signature result(line16-19) is computed by using SHA1 as Hash function and RSA as a public key algorithm. (lines 01-04 see next section)

```
00 <SIGNATURE>
01   <FloatPart> </FloatPart>
02   <Transforms>
03     <Transform Algorithm="...X316MCert#Morph_Body"/>
04   </Transforms>
05   <ds:SignatureMethod Algorithm="..xmldsig#rsa-sha1"/>
06   <KeyInfo>
07     <ds:KeyValue>
08       <ds:RSAKeyValue>
09         <ds:Modulus>
10           vmJ2k9b3JrDhG5...hhyYOLlISgizAHG2L3K=
11         </ds:Modulus>
12         <ds:Exponent>AQAB</ds:Exponent>
13       </ds:RSAKeyValue>
14     </ds:KeyValue>
15   </KeyInfo>
16   <ds:SignatureValue>
17     HO1pw9Ia9MoEOat+UifKlJ7w1v47dazGddE+ckJNkh
18       ....pBmESETXExWgUZXf+dQqw/CvKknsnip1vpSg=
19   </ds:SignatureValue>
20 <SIGNATURE>
```

4.3 X316: The morph capability

All standards e.g X509, PGP use a hash algorithm to obtain a residual value from the certificate data. This value is signed by a private key of the certification authority. Consequently if the content of the certificate is modified, the residual result will be erroneous. In this case, the users can't adapt her certificate by masking any information inside.

In our solution, we use a single certificate that mainly contains the user profile, all user access rights and some authentication systems. Yet we define in this model a specific signature method, using specific tags. Thus, The user can manage and morph her certificates according to the specific transaction or

context. However, some authorized information can be freely masked by the certificate owner far from her home site. In this manner each user extracts a sub-certificate from the original one which only contains needed information for each specific situation.

The problem of the traditional method of certification is the non flexibility of the certificate structure.

Each certificate "Cert" is composed by a body B and a signature S: Cert(B,S). The procedure to compute the signature is the following:

1. Apply a hash function to the body B and obtain a residue R.
2. The residue R is encrypted (by the private key of signatory) to obtain the signature of the documents S.

Consequently, no information can be moved or masked from the certificate.

Thus, the challenge is: how each user can customize her static certificates according to a contextual situation? To solve this problem, we must distinguish The Dynamic Part from the Static Part.

- ***The Static Parts:*** is composed of mandatory and non changeable data (ex: the ID of the certificate, the time of validity). These data set up the identity of the certificate.
- ***The Dynamic Part:*** provides sensitive information (e.g. the user name profile, telephone...) and a contextual information (e.g. the device capability, security context...).
  Our proposal allows each user to freely removes some information from the dynamic part. Therefore, we enrich the traditional model of certification by new Tags. In this manner, the certificate computes differently the dynamic part which gives to a certificate a morph capability.

Some scenarios can exploit the advantage of this type of certificate, by embedding or masking some information, for example:

Community access: when the user is roaming inside different sites she obtains some trusted certificates. For example, given two colleges Alice and Bob that have the same profile in their home site. They roam differently in the environment. Consequently, they collect some trusted certificates. When Alice and Bob would access a target site, by using one of her trusted certificates, Alice obtain an access , but any Bob certificates are recognized by the target site, because Bob is took a different way than Alice and gain different credential. In this case Alice can help bob to access by giving him a morph certificate that contains only the static part, her original profile and its analogue. Thus, the local site can decide to gives Bob the same Alice access profile.

Delegation access: the delegation mechanism allows the user Bob to provide an access to a specific resources. For example: Bob would gives a printer access to his friend Alice who is a guest to the local site. Thus, he gives a morph certificate to prove his right to delegate an access to a printer. This morph certificate contains only the indispensable information to this used context, namely the static part and only the right to delegate the printer ID in the RIGHT part.

X316 signature

To perform the X316 signature algorithm, all dynamic parts in the certificate must be delimited.

Dynamic Part: syntax definition

$$\overbrace{\left|\begin{array}{l} < DP\ id =?\ Description =? > \\ ....(... < DP\ id =?\ Description =? > ...)* \\ < /DP > \end{array}\right|}^{< DPDigest > digest\ value\ of\ DP < DPDigest >}$$

We define two types of tags:

- DP tag : delimits the dynamic part, each dynamic part can also contain another DPs;
- DPDigest tag: delimits the corresponding digest value of the DP part. The signature hash algorithm is used to compute the digest value.

We apply a new Algorithm (Morph_Body) to compute the X316 certificate as follow:

1. Transform the source Body B to a Morph Body MB, by replacing all dynamics parts "DP" with the corresponding digests values (DPDigest).
2. Apply a hash function to the Morph Body MB to obtain a Digest D.
3. This residue D is encrypted (by the private key of signatory) to obtain the signature of the document S.
4. Finally, according to context, the dynamic parts can be put or masked(replaced by its corresponding digest value) then moved to signature certification float part.

In this manner the user is allowed to have two kinds of certificates:

- The Source certificate Cert(B,S)
- The morph certificates MCert(MB(C),S).

The source certificate: is composed of the source body and the signature.

The morph certificates: the user is allowed to create some versions of her certificate. she should only put the required information for a specific context (C) and replace all other information (in the dynamic parts) with corresponding hash residues. Then, she obtains a new version (morph body MB(C)) of the original body. Therefore, the morph certificate MCert is composed of the morph body MB(C) and the signature S. To verify the authenticity of this certificate, the remaining dynamics parts are replaced by their corresponding hash values before checking signature.

When the corresponding all DPDigest are replaced, they are moved to the float part in the signature. This helps to keep clarity in the certificate (see figure 7 step 4) The FloatPart contains all DPDigest parts and their positions in the

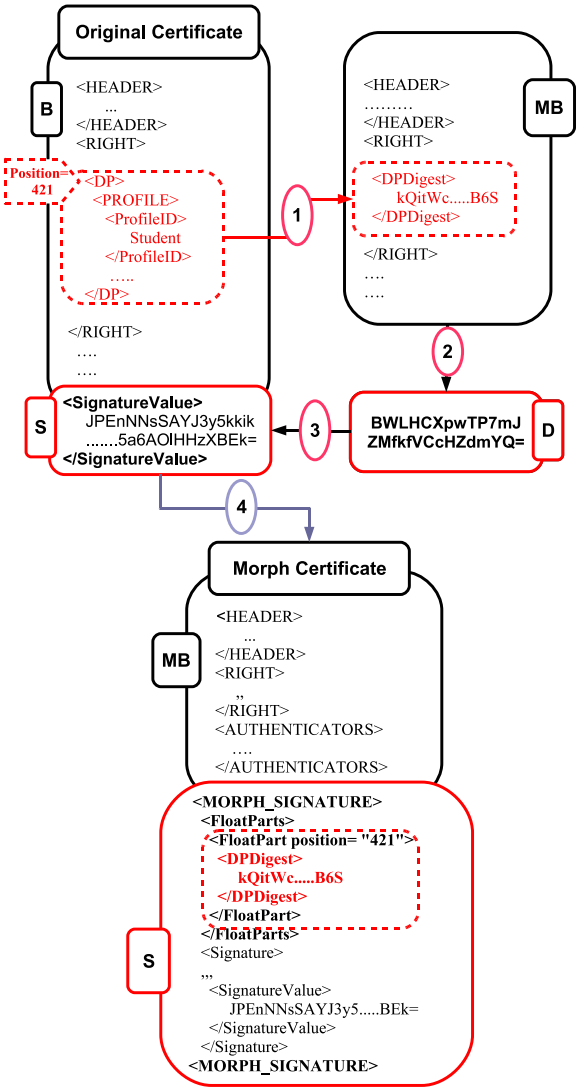

Fig. 7. X316 Signature

original certificate. The position field is mandatory to rebuild the morph body for checking the signature.

Float part syntax:

```
<FloatParts>
  (<FloatPart position=>
    <DPDigest>...</DPDigest>
  </FloatPart>)*
</FloatParts>
```

Each DP can contain some DP, in this manner the user has the possibility to mask all the DP parts or some sub-parts inside the DP part. Consequently before computing the global DP part, the digest of all sub DP parts must be computed recursively.

X316 encryption

X316 certificate contains several information. Each user travel between different sites, some sites are trusted and others are not.

Alice want to access to target site. For her request she must send some sensitive information, held in her certificate, to site administrator. Unfortunately, she consider the communication protocols not safe.

One solution consists to encrypt these informations. If one section in the certificate is ciphered, the scope of the certificate will be limited, allowing only the site that has the cipher key to check the signature.

The morph certificate can solve this problem, allowing each user to make safe any information as long as it is delimited by a DP tag. If the certificate is to be transferred to several sites, Alice ciphers the different sensitive information with different keys.

Therefor, the X316 encryption is defined to allow user to make confidential any dynamic part inside the certificate. X316 encryption operates like XMLEncryption, with differences. Indeed, the certificate must be cheeked even if some parts are ciphered.

Each user can manage her certificate freely; she can cipher any dynamic part, if this one is considered as sensitive.

The X316 encryption allows to cipher only the dynamic part, the idea is to put, in the cipher part, the corresponding hash value of the plain text part. This hash value allows anyone to check the validity of the certificate without need to know the actual content. In fact, the morph transform algorithm replaces also all EncryptedDP parts by the corresponding DPDigest of its plain text.

X316 encryption syntax:

```
00  <EncryptedDP>
01    <DPDigest>
02    <ec:EncryptionMethod Algorithm=/>
03    <KeyInfo>
04    <ec:CipherValue>
05  </EncryptedDP>
```

EncryptedDP description

"DPDigest" (Line 01) contains the digest value of the plain text part. When the X316 encryption is used, DPDigest must appear. Indeed, the certificate must be checkable by anyone not by only the recipient.

"EncryptionMethod Algorithm" (Line 02) defines the used algorithm to perform the encryption task, it is defined as W3C recommendation.

The X316 encryption allows using the symmetric and the asymmetric encryption. The difference between these modes is in the "KeyInfo" parameter.

KeyInfo syntax

```
00  <KeyInfo>
01    <ds:KeyID id= />?
02    <ds:KeyValue>?
03    (<EncryptedKey>
04      <ec:EncryptionMethod Algorithm=/>
05      <KeyInfo>
06      <ec:CipherValue>
07    </EncryptedKey>)?
08  </KeyInfo>
```

KeyInfo description The keyInfo contains the description of key that is used to cipher the DP part. We define two types of encryption:

- Symmetric encryption: In this case, only the KeyId (Line 01) is informed as W3C definition. This Id allows to identify and retrieve the used key.
- Asymmetric encryption: uses two keys, a public key and a session key (symmetric key). Indeed, this method of encryption ciphers the plain text with a session key. Then, it is ciphered with the public key. Thus, "EncryptedKey" (lines 03-07) are required to inform the ciphered session key ; the line 04 defines the Asymmetric encryption algorithm as W3C recommendation ; "KeyInfo" in the line 05 defines the public key that is used to cipher the session key. it contains the KeyID (line 01) or the KeyValue (line 02) ; the line 05 contains the encrypted session key.

In the next example, we describe two sorts of encryption:

- Symmetric encryption using AES in CBC mode(line 04).
- Asymmetric encryption using RSA(line 19) with AES(line 16).

Example

Symmetric encryption :

```
00  <EncryptedDP>
01    <DPDigest>
02      kQitWcHqiq6rcZopVVpmm/bB6S=
03    </DPDigest>
04    <ec:EncryptionMethod Algorithm="..xmlenc#aes128-cbc"/>
05    <KeyInfo>
06      <ds:KeyID id="JMP_IRIT" />
07    </KeyInfo>
08    <ec:CipherValue>
09      A4Fed78TBWdfcg54rTf
10    </ec:CipherValue>
11  </EncryptedDP>
```

Asymmetric encryption:

```
<EncryptedDP>
  <DPDigest>
    kQitWcHqiq6rcZopVVpmm/bB6S=
  </DPDigest>
  <ec:EncryptionMethod  Algorithm="..#aes128-cbc"/>
  <KeyInfo>
    <EncryptedKey>
      <ec:EncryptionMethod Algorithm="..xmlenc#rsa-1_5"/>
      <KeyInfo>
        <ds:KeyId Id="YrQkh1zr.2SsoKE1M="/>
      </KeyInfo>
      <ec:CipherValue>xizrbc</ec:CipherValue>
    </EncryptedKey>
  </KeyInfo>
  <ec:CipherValue>
    G5LyRhgvjChfo0SYiPGWxwPW2
  </ec:CipherValue>
</EncryptedDP>
```

X316 context

In fact, the user is allowed to manage its certificate. This procedure is difficult because she must manually choose the corresponding dynamic parts in function of context . To help her, we introduce the concept of X316 context. It defines the context profile e.g. delegation, mobility, security etc.

Each profile defines its corresponding parts and index the essential parts in the source certificate.

The context is defines by the user or by default. The home site can define with the certificate some contexts, in the same manner the user can create her specific context and define the essential parts for each situation. A context can be composed from others using: addition, subtraction ...

5 Discussion & Experiments

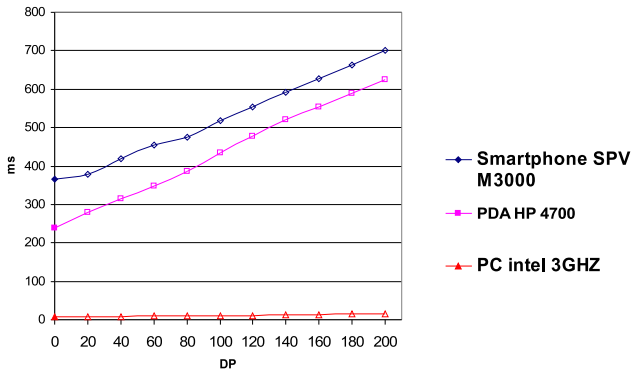

Fig. 8. Test and evaluation

The Chameleon architecture allows the user to roam transparently in the pervasive environment simply by using her X316s. The Chameleon using the X316 presents a number of advantages. It is a decentralized architecture since each site knowing only its neighbors can perform a large but controlled access to some user communities. Chameleon reduces the human interaction where most security management functions can be processed dynamically. In addition, Chameleon does not modify the local site policy, increases the user rights along her trip, having a unique certificate to manage many devices. Finally, the new computing signature algorithm and the morph characteristic gives to the certificate the ability to be adapted to context.

The X316 chain might be seen as a problem due to the chained of certificates. Nevertheless, the X316 contains a traceability from the trust chain (TrustPath) and a new mechanism of aggregation signature to minimize the certificate size. The target site may checks the Trust path, and denies access if it does not trust any site in this chain, or if the degree becomes greater than a given threshold (leading to high distrust in the guest).

Some tests were implemented to verify the scalability of the X316 morph characteristic. We used an XML file of 20KByte, and computed the elapsed time to verify the signature by varying the number of dynamic parts (DPs) from 0 to 200. For these tests we have used three devices: a smartphone : SPV m3000 (195MHZ CPU), PDA HP4700 (624MHZ), and a PC (3GHZ). As shown in the Fig 8, even the SPV M3000 can compute the X316 signature within less than 1 second

6 Conclusion & Future work

The certification model is the basis of the authentication in ubiquitous computing. In this paper, we define a new model of certification(X316) which allows a broad user access when this latter is roaming. Besides that, the user can enter unknown sites with various user interfaces (devices) using the same certificate. We have also introduce a new computing method signature to enrich the certificate adaptability.

In fact, when the user wants to access a target site, her device should perform two actions :

- Select the corresponding certificate which helps user to gain a maximum access in the target site.
- Select the corresponding certificate subparts which are essential for this access, according to context, and hide others.

One of pervasive environment challenge is the fluency of the interaction between the environment and the user. As future work, we will integrate the description of context to X316 giving the user device the capacity to manage and adapt the certificate dynamically with respect to context without soliciting any user intervention.

References

1. N. Shankar, W. Arbaugh. *On Trust for Ubiquitous Computing.* Workshop on Security in Ubiquitous Computing, Sep 2004.
2. P. R. Zimmermann. *The Official PGP User's Guide.* MIT Press, Cambridge, MA, USA, 1995.
3. *ITU-T Simple public key infrastructure* (SPKI) charter, http://www.ietf.org/html.charters/OLD/spki-charter.html.
4. L. Seitz, J.M. Pierson and L. Brunie. *Semantic Access Control for Medical Applications in Grid Environments.* A International Conference on Parallel and Distributed Computing, pp374-383, Aug 2003
5. *ITU-T Rec. X.509 (2000).* ISO/IEC 9594-8 The Directory: Authentication Framework
6. M.R. Thompson, A. Essiari, and S. Mudumbai 2003. *Certificate-based authorization policy in a PKI environment.* ACM Trans. Inf. Syst. Secur. 6, 4, pp 566-588, Nov 2003.
7. *ITU-T Rec. X.680 (2002)* ISO/IEC 8824-1:2002, http://asn1.elibel.tm.fr/en/standards/index.htm
8. S. Micali,and R. Rivest L. 2002. *Transitive Signature Schemes.* In Proceedings of the the Cryptographer's Track At the RSA Conference on Topics in Cryptology, Computer Science, vol. 2271. pp 236-243, Feb 2003.
9. R. Johnson, D. Molnar, D. Song and D. Wagner, *Homomorphic signature schemes*, Proceeding in Cryptology - CT-RSA 2002, ed. B. Preneel, LNCS 2271, pp. 244-262, 2002.
10. Stefan Brands. *A technical Overview of Digital Credentials.* Research Report, Feb 2002.
11. D. Chadwick and A. Otenko. *The PERMIS X.509 Role Based Privilege Management Infrastructure.* In Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, pages 135140, Jun 2002.
12. M. Lorch, D. Adams, D. Kafura, and al. *The PRIMA System for Privilege Management, Authorization and Enforcement.* In Proceedings of the 4th International Workshop on Grid Computing, Nov 2003.
13. M. Bartel, J. Boyer, B. Fox, B. LaMacchia, and E. Simon. *XML-encryption syntax and processing.* In W3C Recommendation. Feb 2002. http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/
14. T. Imamura, B. Dillaway and E. Simon. *XML-signature syntax and processing.* In W3C Recommendation. Dec 2002. http://www.w3.org/TR/2002/REC-xmlenc-core-20021210/
15. M. H. Harrison, W. L. Ruzzo, and J. D. Ullman. *Protection in Operating Systems.* Communications of the ACM, 19(8):461-471, 1976.
16. D. E. Bell. *A Refinement of the Mathematical Model.* Technical Report ESD-TR-278 vol. 3, The Mitre Corp., Bedford, MA, 1973.
17. R. Sandhu, E. J. Coyne, H. L. Feinstein, and al. *Role-Based Access Control Models.* IEEE Computer, 29(2):38-47, 1996.
18. XCBF 1.1, OASIS Standard, approved Aug 2003. www.oasis-open.org/committees/xcbf/
19. A. Lysyanskaya, S. Micali, L. Reyzin, and H. Shacham. *Sequential Aggregate Signatures from Trapdoor permutations.* In Eurocrypt 2004, volume 3027 of LNCS, pages 7490. Springer-Verlag, 2004.

20. M. Zhao, S.W. Smith, and D.M Nicol. *Aggregated path authentication for efficient BGP security*. In Proceedings of the 12th ACM Conference on Computer and Communications Security. CCS '05. pages 128-138. Nov 2005.
21. X. Orri, J. M. Mas, SPKI-XML Certificate Structure Internet-Draft, Octalis SA, Nov 2001. http://www.ietf.org/internetdrafts/draft-orri-spki-xml-cert-struc-00.txt
22. Challenge-response authentication From Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/Challenge-response_authentication
23. R. Steinfeld, L. Bull and Y. Zheng; *Content Extraction Signatures*. In Proceedings of 4th International Conference of Information Security and Cryptology. pages 285-2004. Dec 2001.
24. L. Bull,P. Stanski, and D. M. Squire. *Content extraction signatures using XML digital signatures and custom transforms on-demand*. In Proceedings of the 12th international Conference on World Wide Web pages 170-177. May 2003.