Can we take this off-line?

Credentials for Web Services supported nomadic applications †

Laurent Bussard[2], Joris Claessens[2], Stefano Crosta[1], Yves Roudier[1], and Alf Zugenmaier[3]

[1]*Institut Eurécom, Sophia-Antipolis, France* {crosta, roudier}@eurecom.fr
[2]*European Microsoft Innovation Center, Aachen, Germany* {lbussard, jorisc}@microsoft.com
[3]*DoCoMo Euro-Labs, Munich, Germany* zugenmaier@docomolab-euro.com

Devices supporting nomadic applications are assumed to be able to take advantage of the capabilities of surrounding devices. This paper discusses the access control requirements of such ad-hoc federations of communicating devices, some of which may be administered by a different authority, and illustrates how such scenarios would be handled in the *Web Services Security* and the framework proposed in the *WiTness* project. The adaptation of the *WiTness* flexible attribute certificate infrastructure as tokens for the *Web Services Security* specifications suite is finally discussed as an option to support scenarios where a full-time attachment to a global network is impossible.

Key words: Web Services, attribute certificates and PKI, access control, mobile applications

1 Introduction

Nomadic applications are the tangible evidence of the development of pervasive or ubiquitous computing. Even though computers may not be everywhere, mobile users carry or wear cell phones and PDA-like devices, and encounter more and more devices in intelligent buildings or public places, all these devices interacting in short-lived federations.

The use of certificates for specifying access control rights in distributed systems has long been advocated as a measure to secure applications without connectivity to an authority. The availability of mobile networks today makes it less obvious whether such mechanisms are still in need, and may render old-fashioned Kerberos-like systems more ubiquitous. However, because of the user mobility, relying on full-time connectivity to reach a centralized authority is sometimes simply not affordable. First, communication with a base station might simply be impossible because of obstacles (buildings, tunnel, etc.) or because the communication infrastructure might have been destroyed. In a centralized architecture, server crashes or network problems, which may be caused by denial of service attacks for instance, would also block roaming users without any fallback solution. In addition, and while this may change in the future, local connectivity (e.g. Bluetooth or WLAN) is cheap while permanent mobile communications is still prohibitive today, both financially and in terms of electric consumption. Situations where a mobile device cannot achieve full-time connectivity are thus likely to occur, the prevailing communication scenario being that of today's laptop user: sometimes connected to a global network infrastructure, sometimes voluntarily isolated, sometimes in LAN with neighbors.

This paper focuses on access control for such applications and how to specify it in two security frameworks: WS-Federation, (Section 2) which is an industry specification for federating security of Web Services; and WiTness (Section 3), *Wireless Trust for mobile business*, which is a framework designed to

†This work was part of the EC WiTness Project IST-2001-32275. This paper represents the opinion of the authors

secure business applications with a PDA or mobile phone infrastructure, whose credential implementation we present. This paper also discusses how to use the complementarity of the access control features from both frameworks to provide a better support to applications without full-time connectivity (Section 4). Section 5 finally discusses other approaches to access control, especially for mobile applications.

2 Federations with Web Services

Web Services are application components designed to support interoperable machine-to-machine interaction over a network. Web Services are accessible over open protocols: they exchange XML messages through SOAP; their interface is described in detail using WSDL; they can be registered at a UDDI server, which as such makes them discoverable. In addition to these basic features, there exist specifications for secure, reliable, and transacted Web Services [IM+03a].

2.1 Web Services Security

The Web Services Security architecture and roadmap [IM+02a] proposed by IBM and Microsoft provide a comprehensive and extensible set of security specifications for SOAP-based Web Services.

WS-Security [Oas04e] provides single-message authentication and single-message confidentiality by specifying how to apply XML Signature and XML Encryption, and how to attach a security token to a SOAP message.

WS-SecureConversation [IM+05] specifies how to establish and reference a security context, and provides support to perform a secure conversation with multiple messages.

WS-Trust [IM+02b] specifies how to request, issue, validate, and exchange security tokens. Security tokens are cryptographically protected claims (e.g., identity or authorization assertion) and/or cryptographic keys. They are issued by a Security Token Service (STS), and may be forwarded, delegated, or proxied. Security token requests are typically secured using WS-Security, for instance when one security token is exchanged for another).

The Web Services Security framework currently supports username/passwords [Oas04a] and X.509 certificates [Oas04b], binary security tokens, such as Kerberos tickets [Oas03], and any type of XML security token, including SAML assertions [Oas04c] and XrML tokens [Oas04d].

WS-Policy provides a framework which allows Web Services to describe and communicate (publish) their policies. WS-SecurityPolicy [IM+02c] specifies the security policy assertions that can be used in this framework.

WS-Federation [IM+03b] finally describes how to manage and broker trust relationships in a heterogeneous federated environment, including support for federated identities, attributes, and pseudonyms. A federation consists of multiple Web Services domains, each with their own STS, and with their own security policy. Among other things, WS-Federation specifies such an infrastructure using WS-Trust, for example enabling requesters from one domain to obtain security tokens in another domain and to subsequently get access to the services in the other domain. The term ‘federation’ is used here in a different manner than in the pervasive computing literature, or WiTness, where it refers to two or more devices close to each other, and whose services are jointly used; WS rather refers to it for services that are exposed across different trust domains without the necessity of physical proximity.

2.2 Back-end system queries

Alice (A) is a user working at company C_A who wants to access her emails which are stored in her company’s back-end system. She wants to use a public terminal provided by company C_B, which she happens to be visiting, which she might find using WS-Discovery [IM+04b]. Alice prefers to authenticate on her PDA rather than enter her password into the public terminal. She will then let her PDA delegate temporary rights to the terminal so she can switch off her PDA and keep its battery charged. The terminal can verify that Alice is indeed an employee at a partner of company C_B and let her access the Internet. Access control to an email requested through the public terminal is finally enforced by company C_A’s email server.

Implementing this scenario using WS is rather straightforward since the WS model essentially assumes a global connectivity, i.e., that each entity involved can contact every other entity at any time. Since the user

physically interacts with the federated terminal, the latter is the originator of the SOAP messages. Instead of considering the terminal as an active SOAP intermediary, we will closely follow the WS-Federation Active Requestor Profile [IM+03c]: the first request is initiated by the personal device and includes transferring control of the public terminal to Alice. The detailed scenario is shown in Fig. 1.

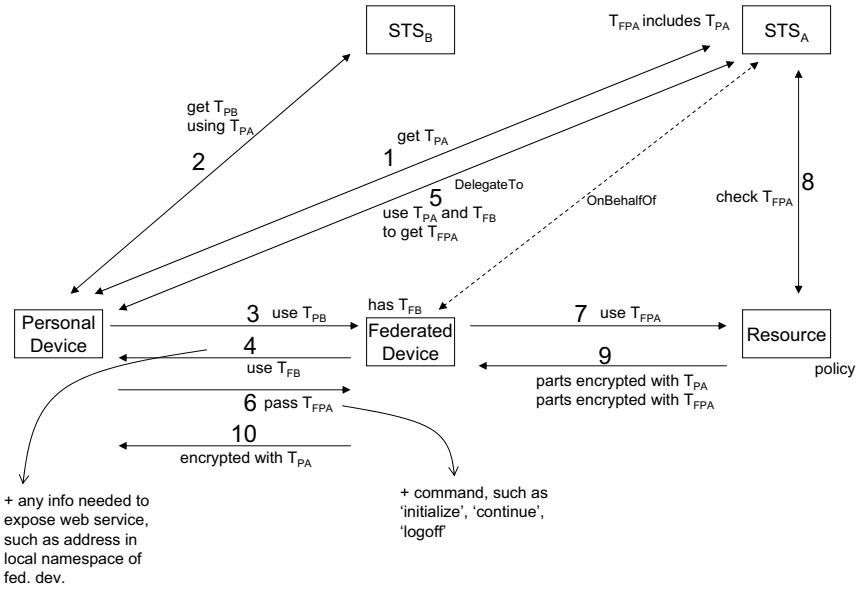

Fig. 1: The back-end system scenario with Web Services Federations.

1. The PDA fetches a security token T_{PA} from its own security token service STS_A. T_{PA} proves that the device is operated in domain *A*, and may also indicate the owner and relevant security features of the device.

2. Using T_{PA}, the PDA then performs a WS-Trust token exchange, and gets a security token T_{PB} from STS_B.

3. The user logs on with the personal device. In particular, the functionality of the federated device is exposed via a set of Web Services. The security policy of these web services mandates that requests should be signed with tokens issued in domain *B*. The logon request by the personal device is signed with T_{PB}.

4. The response of the public terminal is signed with a token T_{FB}.

5. The personal device then submits T_{FB} signed with T_{PA} to STS_A to request a security token T_{FPA} for the federated device, making use of the of the *DelegateTo* feature of WS-Trust. T_{FPA} is a security token associated with the private key of the federated device, and asserting that the federated device is performing actions on behalf of the owner of the personal device. T_{FPA} includes T_{PA} to indicate the user on which behalf the federated device operates, and should also assert the authorization level of the federated device; this authorization level will depend on the security features that are asserted in T_{FB}.

6. T_{FPA} is passed on to the terminal, along with the command to initialize it. From then on, the PDA is not required any more, except if a secure platform is required.

7. The user can now switch to the user interface provided by the public terminal, and access the server in domain *B*. Access requests are secured with T_{FPA}.

8. The server can check the validity of T_{FPA} by contacting STS_A. The server can now enforce a policy, which may mandate that – depending on the assertions in T_{FPA} – certain parts of an email should not be available in cleartext in, nor displayed on the public terminal.

9. Depending on the policy and the sensitivity of the requested email, certain parts of the email are transmitted to the public terminal encrypted under T_{PA}, the less sensitive parts encrypted under T_{FPA}. The terminal is only able to decrypt the parts encrypted under T_{FPA}.

10. Sensitive parts of the email can be forwarded on demand by the terminal to the PDA, which may then decrypt and display them.

11. When finishing the session, the PDA sends a sign out message to STS_A, from then on all further attempts to verify the token T_{FPA} will fail.

The realization proposed above follows the Web Services Security specifications. Note that no specific format nor content of the security tokens is mandated in the specifications. The Web Services exposed by the devices and the server are of course scenario and application-specific, as well as the multipart encryption feature, and the according behavior of the federated device, which needs to be agreed upon.

2.3 Taking it "off-line"

Looking at the detailed scenario described above, the use of Web Services may be problematic if the PDA has limited connection capabilities.

Step 1 assumes the PDA to be able to contact its own STS. If not, the PDA should already possess the required T_{PA} that can be validated and trusted by an STS in another domain.

Step 2 assumes the PDA to be able to contact the STS_B of the terminal. A limited PDA may only be able to communicate with physically close federated devices. The PDA should then already have the required T_{PB} that can be validated and trusted by a federated device in another domain. In other words, federated devices must be able to directly validate T_{PA}, or the PDA must carry a set of T_{PB}s for each of the domains trusted by A.

Step 5 again assumes the PDA to be able to contact its own STS. If not, an STS_{PA} could be introduced, running on the PDA, and capable of issuing delegatable security tokens. STS_{PA} should obviously be prevented from misuse. Its private key should be stored and operated in a secure environment, like a SIM card. Moreover, it should only return delegation tokens T_{FPA} with authorization assertions that are compliant with the presented T_{PA} and T_{PB}.

Step 8 assumes the server to be able to contact its STS. Otherwise, the user cannot sign out anymore, and the lifetime of T_{FPA} should be limited.

2.4 Ad-hoc interactions

Let us consider Alice's case again and suppose the application requirements are changed: a backup is needed in case of corporate server failure at company C_A so that Alice can keep on working anyway, and in a secure manner. In that situation, the public terminal can only access emails that are cached on her PDA (a). In case of a crash, the certification authority (CA, or AA for Attribute Authority, as it is called in the Privilege Management Infrastructure) will be unreachable. Credentials describing Alice's rights must therefore be stored on her PDA so that she can work with the data stored on her PDA and prove her identity or rights locally.

The scenario so far assumed that the email server was consulted on-line. Alice's PDA, which is trusted, has to cache these emails to let her read on, but it has strong limitations: it cannot print, has a small display, small keyboard, etc. Alice may want to print a corporate email on a printer in the airport lounge where she is waiting for her plane and edit a contract draft (R_A) with the laptop (b) of Bob, an employee at company C_B.

Alice's PDA must thus be able to decide whether the printer with which it federates can be trusted to print a confidential messages. In that respect, the PDA can be considered as a mobile extension of the corporate server. The security policy should thus be stored together with emails, and access control enforced by the

PDA. In conclusion, enabling more ad-hoc interactions is possible by not only working with a local copy of a resource accessed, but also with a local security token service, and/or one or more locally cached security tokens. Access control for such a pervasive application will result in the following verifications being enforced by the PDA:

1. *Is employee A authorized to access resource R_A?* Employee *A* has a pair of public and private keys and is certified by the employer C_A, i.e. roles or authorizations certificates.

2. *Is device b trusted enough to edit resource R_A?* Device *b* has its own pair of public and private keys and is certified by its owner C_B, i.e. ownership or security-level certificates. To evaluate the security level of the terminal, the user needs to know whether there is an agreement between her company and the company owning the terminal.

3. *Is A authorized to use device b?* Agreements between corporations can also define rights. For instance, company C_B can authorize employees of company C_A to use local printers. To use *b*, *A* has to prove that she is authorized to.

Kerberos tokens are connected credentials. The security relies on a permanent access to the ticket granting server. WS-Federation extends this model and relies on a permanent connection with some STS during delegation and validation phases. We will show how to define disconnected instantiations of this model. No such type of credential can support federation scenarios as depicted above in a convenient manner.

3 Credentials for Federations: WiTness

The WiTness project [BRKC03] sets out to define a framework for the development of secure business to employee (B2E) or business to business (B2B) applications in nomadic environments that let employees access corporate information on the road from their personal laptop, PDA, or cell phone. The WiTness framework intends to promote the development of multi-application SIM or USIM card as a security anchor, thanks to the ubiquity of cell phones, each employee's smartcard hosting the cryptographic secrets necessary to authenticate this employee and performing critical functions as a personal security module.

3.1 Access Control for Ad Hoc interactions

WiTness is dedicated to mobile and pervasive computing and thus assumes 1) restricted computational power and 2) restricted communication channels, especially that short-range connectivity is assumed to be the standard. The following applications are especially targeted as typical security objectives: (1) interaction with artifacts (smart door, smart vending machine); (2) extension of personal devices, e.g. larger display, keyboard; (3) document sharing during a meeting; (4) voting, contract signature.

WiTness developed adaptable credentials that may be used for "off-lineable" ad-hoc interactions. Access control is based on the rights (or roles, etc.) of an employee and on the trust level of the device from where he performs the access to a given resource. These rights and trust levels are derived from chains of certificates coming from the different corporate authorities involved.

3.2 Witness Attribute Certificates

Certificates are proven to be the solution for distributed access control and identity management. WiTness applications had special requirements because authorization rules are defined in terms of multiple attributes. The decentralized management of mobile devices, need for flexible delegation, and limited computational capabilities were additional requirements.

Neither X.509v3 Attribute Certificate (AC), introduced to solve limitations of X.509 Identity Certificate with a Privilege Management Infrastructure (PMI), nor any other solution (SPKI [spki99], Akenti [Akenti], ICare [Icare] to cite a few) was found to completely satisfy these requirements. In fact, X.509v3 ACs rely too strongly on an existing X.509 structure for authentication. SPKI certificates have proven very complex to use mainly due to the chain discovery and tuples reduction and because of their restrictive definition of authorization. Figure 2 shows the bindings that are authenticated with SPKI, X.509 and with attribute

certificates. The WiTness attribute certificates place identity and attributes on the same level, and also make it possible to assess platform trust. All attributes are defined in a uniform and extensible manner, identity being just like any other attribute.

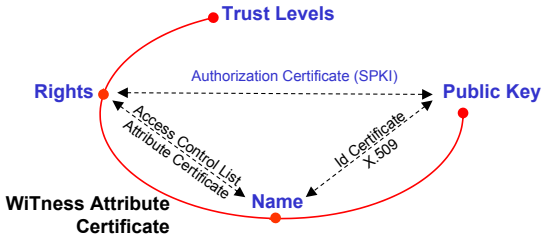

Fig. 2: WiTness, SPKI, and X509 Certificates

WiTness Certificate Basics WiTness certificates are a flexible data structure for the management of distributed credentials. They were designed based on the experience acquired with ICare to exploit XML strengths, most specifically extensibility and readability for the programmer. A certificate associates attributes to a principal, the holder, with a data structure signed by the certificate issuer; Certificate holder and issuer can either be a public key, like in SPKI, or a referenced WiTness certificate. The attribute data structure is a tuple of a *name*, *value*, and optional *resource* and *delegation*. For instance, the following example illustrates what might be an attribute issued by company C_A's authority to Alice to endorse her as one of its employees working on Project X. This endorsement is not delegatable by Alice to anyone.

```
<Attribute Name="Company-A Corporate Position">
  <Value>Employee</Value>
  <Resource>Project X</Resource>
  <Delegation>0</Delegation>
</Attribute>
```

The same attribute structure holds for Identity information or to specify file access rights but with different values, for instance Name being 'File Access', Value 'RW', and Resource 'validFileNameURL'.

The content of the attribute must be evaluated at application level, because its semantics cannot be known at the certificate API level. Nevertheless, some basic rules are implemented in the library to perform a first delegation validation check; moreover, the library exposes a plug-in based architecture, which permits to easily develop new attribute types along with their validators. These validators can be passed to the library for an automatic validation.

The signature makes the certificate tamper-proof and builds a trust chain. The holder proves the ownership of the certificate through a classic challenge-response protocol demonstrating private key ownership. Using a public key as a holder creates a new, independent attribute certificate from the issuer for the holder; using a reference to an existing certificate permits to associate attributes to existing entities, and can be used to build cross-domain trust chains, as described below. Using a public key as an issuer is typically for self-signed root certificates only. Any certificate further issued in the chain contains the full issuer certificate, exploiting recursive XML schema capabilities, thereby yielding self-contained certificates with an explicit trust chain. Validation of the trust chain and delegation is made easier by the full inclusion of the chain into the issued final certificate, at the sole expense of an increased size of certificates and chains.

WiTness Certificates are designed to allow dynamic roles and rights distribution and delegation in mixed ad-hoc/back-end server scenarios. Any entity can act as a certification authority, creating new attributes or delegating existing attributes, and distributing them to other entities. The Issuer may intend no right to delegate, any number of successive delegation levels, or no limit to delegation. In a chain of certificates, the library verifies that the delegation field is regularly applied, and monotonically more restrictive (with the exception of unlimited delegation).

Certificate Chains and Validation The chain from the root to the leaf delegated certificate is called a *physical chain*. All parent certificates in a physical chain can be included in the final certificate for ad-hoc validation, since they are generated in sequence. *Logical chains* are used to create inter-domain delegation, and correspond to an agreement between two companies, linking existing physical chains from different authorities. A logical chain is created through cross-delegation, that is by assigning an existing attribute certificate as an holder.

Certificate validation proceeds as follows: the certificate must be valid (integrity, XML validation); the root of the physical chain must be a *trusted (root) certificate* (as in X.509), or must be trusted through an inter-domain delegation signed by a trusted certificate; delegation must be coherent; finally, attributes synthesized by the chain are validated at application level using a plug-in architecture.

4 Putting it all together

4.1 Credentials and Transient Connectivity

As illustrated in Section 2, supporting ad-hoc interactions is an interesting option to improve the usability of a nomadic application. This does not preclude the transient availability of the CA/AA to the user PDA at some point in time, before or after the interaction with the terminal. In a back-end authority based model, credentials are obtained on demand, per request. In an ad-hoc model however, credentials can be anticipated as required and requested from an authority before going offline, or the lack thereof may be logged. For instance, if Alice plans to meet Bob, she might request the appropriate credentials to work with him before leaving her company. Otherwise, if Alice meets Bob for the first time and knows she will meet him again, she can register his identity and request proper credentials to grant Bob's device access to her resources the next time her PDA gets in touch with her authority. Conversely, Bob might want to delegate some rights to Alice, without knowing her identity, when he meets her.

In both cases, in the WS-Federation architecture, this means that each mobile device should be offering a local and miniature embedded Security Token Service able to generate new tokens that would be adapted to a given situation. The delegation mechanism presented in Section 3 seems a natural way to assure that such an embedded STS will provide enough access without giving more rights than what is allowed by the corporate policy.

The storage of credentials is therefore necessary in an ad-hoc interaction model, yet the capacity means of a mobile device like a PDA or cell phone does not make it possible to cache all credentials representing rights, roles, or security-levels of an entity. The pre-installation of credentials of authorities that can be known in advance like the police, fire service, or a business partner might be considered in the first place. WS-Federation offer protocols that may serve as the vector for negotiating such prefetching of credentials in advance, or as a regular requirement, and between several STSes. WS-Federation protocols may also be used to precise the granularity of these credentials, in particular by restricting the scope of delegated rights that may be issued. Certificates [Itu88, spki99] are generally understood as "offlineable"' or distributed credentials and would thus fulfill the need to represent such agreements and rights. The Web Services Security specifications do not specify tokens equivalent to WiTness certificates, but in principle support any type of XML tokens. WiTness credentials might thus potentially be embedded within the Web Services Security architecture.

Degraded mode is the last option making it possible for a nomadic user to work on the road and is clearly easy to specify with the trust level functionality of WiTness certificates. Some functionalities will not be available except in a trusted environment, like email decryption in Alice's PDA in our last example.

The right part of Figure 3 presents the construction of a chain of certificates in WiTness: Alice A is authorized by her company C_A to access some resource R and can delegate this right. In an ad-hoc context, A authorizes a federated device FD to access R. The left part of Figure 3 shows how those credentials could be integrated in a WS-Federation of STS: the first credential defines the trust relationship between STS_{C_A} and STS_A and the whole chain is the security token provided to FD, i.e. T_{FPA}.

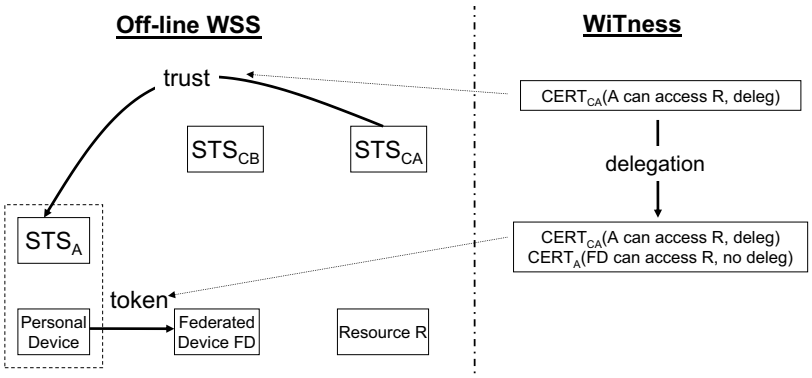

Fig. 3: WiTness credentials as WS security tokens.

4.2 Ad-hoc Credential Revocation

The necessity of prefetching credentials brings up the difficult issue that credentials used in ad-hoc federative situations are validated and delegated locally only, i.e., without any communication with a centralized authority such as a certification authority (CA) or a central Security Token Service (STS). How to check whether a certificate is still valid when used?

Certificate invalidation is a difficult problem in distributed systems in general. While *Online Certificate Status Protocols* (OCSP) is excluded in our case, *Certificate Revocation Lists* (CRLs), although awkwardly large for mobile devices, can be used in ad-hoc scenarios as long as they can be updated on a regular basis. A different way to control the validity of certificates is to rely on *short-term* [spki99] or even *one-time* certificates [Cha88]. In this case, a mechanism for periodically renewing rights is necessary, and finding the right delegation granularity is not easy. Electronic cash [Cha88] may even be seen as an ad-hoc enabled credential dedicated to specific type of rights. A hybrid approach was chosen in WiTness that is well adapted to transient connectivity: long-term certificates are used to define the roles, rights, or trust levels of employees and devices, and are updated along with revocation information when connected to an authority; short-term certificates are created when rights are delegated in ad-hoc mode.

4.3 Security = Application + Middleware

The WiTness framework is a programmatic framework for securing applications, and it offers a perfect interface to applications for accessing certificates content. The Web Services approach, on the other hand, take a middleware approach of introducing security functionalities transparently into applications, only relying on a policy based configuration of security.

We suggest that securing federative nomadic applications requires combining these two approaches to get the best of both worlds. The application designer might choose whether to rely on the transparent security mechanisms provided by the current Web Services Security specifications, or to implement security mechanisms using an application layer framework, when the security requirements cannot be handled at message level.

In a WiTness typical scenario [BRKC03], users access corporate information through a public terminal, which they federate with their personal security device. The corporate information sensitivity may be used to decide on running the application in a degraded mode, for example not displaying sensitive emails outside of the employee PDA. However, determining the sensitivity of such data can only be done at the application level.

WiTness actually focuses on providing a flexible application-layer security framework enabling the development of such fully customized, scenario specific solutions, in particular through the provision of an adapted certificate based authorization framework. It would thus perfectly complement the standardized secure messaging infrastructure provided by Web Services, which is a guarantee for interoperability, and which makes application development with such an infrastructure easier for non-security experts.

5 Related Work

XACML [Oas05b] defines an XML schema for an extensible access control language. It provides a common language for defining security policies, and in particular doing so consistently across different applications and components within the enterprise. While XACML focuses more on policy enforcement across technologies (e.g., firewalls, email servers, etc) within an enterprise, WS-(Security) Policy deals with describing the security policy associated with Web Services endpoints, and with communicating this policy across enterprise boundaries.

The Liberty Alliance [Liberty] specification aims at providing a federated network identity management infrastructure. While it focuses on a dedicated and specific identity infrastructure, the WS-Federation and the Web Services Security specifications constitute a generic security framework, enabling authentication and authorization for Web Services across organizations, enterprises and individuals.

The Security Assertion Markup Language (SAML) [Oas05a] is an XML-based framework designed to solve three main use cases: single sign-on, distributed transactions (dealing with transport of cross-domain authorization), and authorization service (dealing with authorization). It handles the exchange of security information like XML-encoded security assertions, XML-encoded request/response protocol, and rules on using assertions with standard transport and messaging frameworks. SAML supports three kind of assertions: authentication, attribute, and authorization decision. SAML protocols and the Liberty Alliance provide similar functionality as the WS-Trust. However, SAML protocols are only used in combination with SAML tokens while WS-Trust support numerous tokens like username-password, X.509, SAML, etc. and can be extended to support others.

WiTness defines a proprietary XML-based certificate format to deal with all kinds of attributes in an integrated way. Existing works prove the interest for an XML format for a certificate structure: [Paa00] is an IETF Draft for an XML format of SPKI certificates; an ITU-T group works on specifications X.6xx (e.g. X.693) [Itu01] to translate ASN.1 into an XML format. The XML SPKI draft and similar attempts failed mainly because they directly translated existing formats into XML structures, without fully respecting the XML philosophy of extensibility, thus not exploiting XML features fully.

Other ongoing research in mobile environments and even in pervasive computing environments show that Web Services seem promising in that area. Mobile Web Services address the access to Web Services from mobile environments and pervasive web services focus on the discovery of services. [RH03] suggests that a reference to a dedicated web service be attached to physical objects (train ticket, room, etc.) using barcodes; RFID tags or infrared beacons may also be envisioned. This goes one step further to the PANs discussed in this paper since physical artifacts would then all be references to web servers responsible for handling ad-hoc interactions.

6 Conclusions

Today's business environments deploy systems of mobile devices that must interact together in increasingly ad-hoc federations. However security tokens available in the most prevalent security infrastructures, like for instance Web Services or Liberty Alliance, fall short of supporting such applications in the case where the mobile user is not permanently connected to his authority. We described scenarios illustrating these issues, and their implications in the WS-Federation architecture.

This paper described in what respect WiTness attribute certificates might ideally complement the WS-Federation architecture to support various types of federative nomadic applications. Web Services specifications in turn provide a standardized and secure messaging framework on top of which secure and interoperable applications can be built in a transparent way. Federation capabilities, understood both in terms of personal area federations of devices, and in terms of federations of authorities, are in any case an essential requirement for the development of nomadic applications. We plan to develop such features in the IST Mosquito project [Mosquito], a follow-up of the WiTness project which has just started recently.

The emergence of context awareness in the field of pervasive computing security illustrates the current trend towards the distribution of trust assessment functions. We believe that the integration of context-sensitive security parameters within an organized security architecture requires the exchange of ad-hoc

enabled credentials like the ones described in this paper. These topics represent a promising area of investigation in Web Services security for the future.

References

[Akenti] Akenti Project. http://www-itg.lbl.gov/Akenti

[BRKC03] L. Bussard, Y. Roudier, R. Kilian Kehr, and S. Crosta, *Trust and authorization in pervasive B2E scenarios*, in Proceedings of the 6th Information Security Conference (ISC03), LNCS, Springer, 2003.

[Cha88] D. Chaum, A. Fiat, M. Naor, *Untraceable Electronic Cash*, Proceedings of Crypto'88, LNCS 403, Springer Verlag, pp. 319–327, 1988.

[Icare] ICare Project. http://www.cert-i-care.org

[IM+05] Web Services Secure Conversation Language 1.2 (WS-SecureConversation). February 2005.

[IM+04b] Web Services Dynamic Discovery (WS-Discovery). February 2004.

[IM+03a] Secure, Reliable, Transacted Web Services: Architecture and Composition. September 2003.

[IM+03b] Web Services Federation Language (WS-Federation). July 2003.

[IM+03c] WS-Federation: Active Requestor Profile. July 2003.

[IM+02a] Security in a Web Services World: A Proposed Architecture and Roadmap. April 2002.

[IM+02b] Web Services Trust Language 1.2 (WS-Trust). February 2005.

[IM+02c] Web Services Security Policy Language (WS-SecurityPolicy). December 2002.

[Itu01] ITU-T *ASN.1 encoding rules: XML encoding rules*, ITU-T standard, December 2001.

[Itu88] ITU-T. Recommendation X.509: The Directory - Authentication Framework, 1988.

[Liberty] Liberty Alliance. http://www.projectliberty.org/

[Mosquito] Mosquito, Mobile Workers' Secure Business Applications in Ubiquitous Environments. IST-004636. https://www.mosquito-online.org

[Oas05a] OASIS. Security Assertion Markup Language 2.0 (SAML). March 2005.

[Oas05b] OASIS. Extensible Access Control Markup Language 2.0 (XACML). February 2005.

[Oas04a] OASIS. Web Services Security: UsernameToken Profile 1.0. March 2004.

[Oas04b] OASIS. Web Services Security: X.509 Certificate Token Profile. March 2004.

[Oas04c] OASIS. Web Services Security: SAML Token. December 2004.

[Oas04d] OASIS. Web Service Security REL Token. December 2004.

[Oas04e] OASIS. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). March 2004.

[Oas03] OASIS. Web Services Security Kerberos Token. July 2004.

[Paa00] Paajarvi *XML format for SPKI certificates* IETF Draft 2000, expired.

[RH03] P. Robinson and S. Hild, *Controlled Availability of Pervasive Web Services* In proceedings of 23rd International Conference on Distributed Computing Systems Workshops (ICDCSW'03), 2003.

[spki99] Network Working Group, Request for Comments 2693: *SPKI Certificate Theory*, September 1999.

[Witness] WiTness, Wireless Trust for Mobile Business, IST-2001-32275. http://www.wireless-trust.org

[Woh00] Petra Wohlmacher, *Digital certificates: a survey of revocation methods*, in Proceedings of the 2000 ACM workshops on Multimedia (International Multimedia Conference), pp. 111–114, 2000.

[Wse] Microsoft. Web Services Enhancements for Microsoft .NET (WSE 2.0).