Institut EURECOM
Corporate Communications
2229, route des Crêtes BP 193
06904 Sophia Antipolis
(France)

Research Report[a] N^o 105 — RR-04-105

Can we take this off-line?
How to deal with credentials in federations
without global connectivity

Laurent Bussard, Joris Claessens, Stefano Crosta,
Yves Roudier, and Alf Zugenmaier

May 14, 2004

[a]This research was supported by European Union Project '*WiTness – Wireless Trust for Mobile Business*' [33], by Microsoft, and by the Institut Eurécom.

Can we take this off-line?
How to deal with credentials in federations without global connectivity

Laurent Bussard[*], Joris Claessens[†], Stefano Crosta[*],
Yves Roudier[*], Alf Zugenmaier[‡]

[*]Institut Eurécom
Sophia-Antipolis
(France)

[†]European Microsoft
Innovation Center
Aachen (Germany)

[‡]Microsoft Research
Cambridge (UK)

May 14, 2004

Abstract

In mobile and pervasive computing environments, not all devices have universal capabilities. To fulfill a certain task, it is often necessary to federate devices with specific resources. Because some devices are mobile, devices from different trust domains may have to interact with each other, and potentially sensitive data may flow from one domain into another. This interaction obviously requires access control and authorization. Achieving a secure federation of mobile devices calls for a framework and mechanisms that allow the specification and enforcement of security policies across different trust domains, even when some or all of the devices are disconnected and cannot go on-line, for example to perform an on-line verification of credentials.

The *WiTness* framework was developed in order to address authorization in device federations, particularly providing mechanisms to tackle specific off-line scenarios involving devices without global connectivity. Modes for interaction in federations of different administrative trust domains are also defined within the *Web Services Security* specifications suite. These modes are generally described assuming a global connectivity of the entities involved.

This paper shows how the experiences gained from the WiTness framework can be applied to Web Services, and investigates how the Web Services Security framework can be used to handle the off-line scenarios WiTness is optimized for.

1 Introduction

Pervasive computing envisions a world in which computers and computing capabilities are ubiquitous. Users carry devices like PDAs and mobile phones with them; in addition the environment will also be able to support the user by becoming "smart", its capabilities enriched by devices with computing power. Not all of these devices will have universal capabilities, constraining factors being notably size, power consumption, communication, and administrative boundaries. Often it would make sense to have groups of these devices collaborate to fulfill a certain task, i.e., to have a temporary *federation* of these devices.

The federated devices need not be from the same domain. This is actually the most likely case for a user who just moved to a new location federating his mobile devices with surrounding devices from the environment. Whenever the task the federated device is involved in requires the transfer of sensitive data, it is necessary to ensure the security of the federation. One example of such a task is access to corporate email from a public terminal. It is necessary to ensure that access to corporate email is only given to that terminal while an employee of that corporation is using it, and additionally it is necessary to ensure the origin of emails sent through that corporate server.

Even though such an architecture requires addressing several interesting security goals, the one we focus on in this article is access control.

Access control can be based on different factors like the identity of the person accessing the data, the role of that person, the rights that are assigned to this person, or even the computing environment. Access control may also be based on any combination of the above. A flexible way of combining these is to use credentials such as attribute certificates and security tokens [5].

These attributes can be proven by showing credentials with a challenge-response protocol. There remains the problem, though, of how does one distribute, certify, and verify these credentials, i.e., the attributes of users and devices.

Two mechanisms for federating will be compared in this paper. One of them is called WiTness for *Wireless Trust for mobile business*, the other one is the *Web Services standard for federating devices*, called WS-Federation. The problem of disconnectivity addressed by this paper is described in Section 2 and its impact on both frameworks is then analyzed in Section 3 and 4. Section 5 summarizes the features suitable to answer these specific needs available from both frameworks and suggests a synthetic solution that might address the disconnectivity issues of the scenarios drafted previously. Section 6 finally references related work about access control for mobile applications.

2 Problem statement

2.1 Disconnectivity

Is it still necessary to deal with disconnected situations? Nowadays, such a question might seem provocative since workstations and servers that are permanently online are galore, and even in the mobile context, permanent connectivity can be achieved by using local communication infrastructure (e.g., WiFi access points) and/or global cellular networks (e.g., GSM, UMTS). Devising a security infrastructure for any interaction thus might likely rely on online authorities or trusted third parties (TTP).

However, an exclusively online approach, disregarding features indispensable to disconnection, is sometimes simply not affordable for building computer systems, and more critically for securing them:

- Server and/or network problems (crash, DoS attacks, etc.) can temporarily prevent connection.
- The user may experience configuration difficulties preventing him from using local communication facilities to connect to the outside of the local network.
- There are places without communication possibilities, like some buildings, subway trains, etc.
- The price of mobile communications may deter their use at lengths.
- The round trip time to an authority may render some applications impractical because of a too slow response time. For instance, unlocking a door in a building should not take as long as paying with a credit card.
- The complexity and cost for deploying a fully connected environment might be overwhelming.
- Keeping a device disconnected from a global infrastructure like the Internet might bring the advantage of a more relaxed threat model and increased security if a sensor/appliance is only accessible by a nearby device rather than remotely and worldwide.
- In case of a disaster such as an earthquake, or worse, a terrorist attack, public authorities should get access to all necessary information and yet be protected from malicious behavior.

For all these reasons, even with the ubiquitous and cheap availability of communication channels, disconnected situations are likely to occur.

2.2 Scenarios addressed

In the rest of this paper, three scenarios will serve as the basis for evaluating the WiTness and the Web Services Federation frameworks:

- Alice is a user working at company A that wants to access her emails which are stored at the company server. She wants to use a public terminal provided by company B, which she happens to be visiting. However, she does not want to enter her password into the public terminal, so she federates her PDA with this terminal to do the authentication (cf. Figure 1). For this first scenario, we assume global connectivity, i.e., each of the entities involved can contact every other entity at any time, even though the communication bandwidths between entities might be quite different.

- In the second scenario the connectivity model is changed: we require a backup in case of network disconnection (or corporate server failure). In this case the public terminal can only be used for accessing emails that are cached on the personal device. Availability of credentials is required for getting access to local resources. However, in case of network disruption the CA may not be on-line all the time during the interaction between the federated devices, but it is on-line sometime, for instance at the beginning of the interaction of the public terminal with the personal device. This might be the case for instance if new rights must be issued by the company every time a document is accessed by the client visited by a salesperson, yet mobile phone communications should not or cannot last forever because of cost or signal propagation issues.
- The third scenario is a case where there is no availability of global network resources during the local interaction of the federated devices. This may not be particularly relevant in the case of an interaction with a fixed terminal, but may occur in an ad-hoc network application, e.g., in an emergency. Here, the CA is on-line only before interaction of the federated devices. One requirement of this scenario is, however, that the authorities of the participants involved are known in advance, like say, the police, firemen, etc.

The main difference between these scenarios that is of interest for comparing both frameworks is the connectivity model. The following section will in particular focus on how the distribution and validation of credentials is done in each of these solutions, and their complementarity because of the different implicit assumptions they make.

3 An Offline-Oriented Approach: WiTness

This section describes the general characteristics of the WiTness framework and how the scenarios outlined previously can be securely implemented using WiTness.

The WiTness project [2] sets out to define a framework for the easy development of secure business to employee (B2E) applications in nomadic environments. Employees are remotely accessing their corporate application server from their personal device, be it a laptop, a personal digital assistant (PDA), or a cell phone. In nomadic context, the operator's smart card (SIM or USIM card) is a ubiquitous security module that can be used to ensure business application security. In WiTness, each employee has a SIM card that acts as a security module hosting the cryptographic secrets necessary to authenticate this employee and to ensure the integrity and confidentiality of data access.

This framework is dedicated to mobile and pervasive computing and thus assumes 1) restricted computational power and 2) restricted communication channels. The former imposes that creation and validation of credentials remain simple in terms of cryptography and XML parsing. The latter requires that the

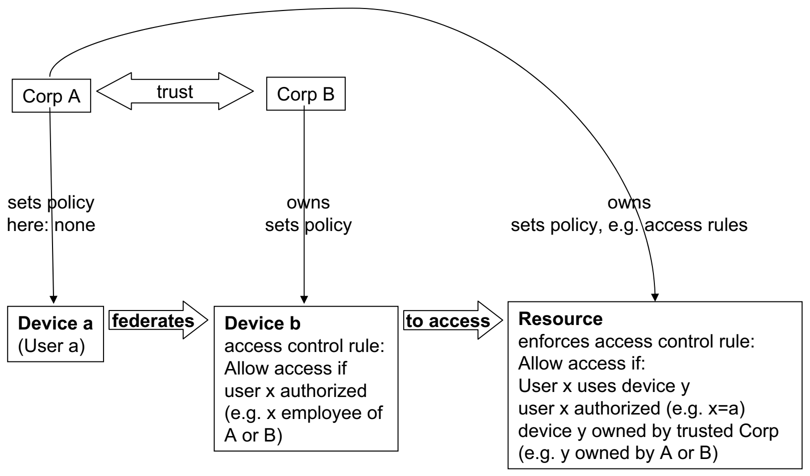


Figure 1: scenario 1 – the device is used to access a corporate resource through a public terminal

scheme does not rely on remote third parties during access control, i.e., can be used off-line. We will therefore focus on an off-line scenario and then show how connectivity makes the scheme more flexible.

3.1 Security in the WiTness framework

Federations for B2E/B2B applications The personal device embedding the user security module makes it possible to access corporate services remotely. However, the personal device can also request local services from the environment, thereby creating a local federation with devices in its vicinity with which it collaborates.

Even if federations are an extension of the nomadic access to corporate servers, we assume that federations are self-standing associations that make it possible for several devices to communicate. They also extend the nomadic model of access to fixed corporate servers by making it possible to have a temporary access to these servers through members of the federation or to perform disconnected-mode or *off-line* operations using prefetched data.

Federations generally associate devices from different trust domains. For instance, a personal PDA can be federated with a public terminal in order to get a more convenient display for reading corporate e-mails. To tackle different security-levels of involved devices, each device holds its own asymmetric key pair so that it can be certified by its owner.

WiTness is especially aimed at securing federation based applications and focuses on providing adaptable credentials that may be used off-line. The following applications are especially targeted as typical B2E and B2B security objectives:

- Artifact based corporate applications where communicating with devices from the environment in a building are the only way to access to pervasive services, like opening the lock of a "smart door" or obtaining a coffee from a "smart vending machine".
- Extension of personal devices that are often limited in terms of computational power, memory, communication bandwidth, user interface (display, keyboard), etc. with better appliances.
- Sharing of documents in a P2P groupware fashion between users from multiple companies.
- Liability for operations that must be controlled with respect to the business partners involved, like for instance a cooperative decision, or the responsibility of a part of a document during the collaborative edition of a document. These applications generally involve signature mechanisms.

WiTness Security: off-line Certificates Access control is based on the rights (authorizations or role) of an employee and on the security level of the device from where he performs the access to a given resource (also referred to as Trust level). These rights and security levels are derived from chains of certificates from the different corporate authorities involved.

For dealing with authorizations, roles, id, security levels, and policy certificates in an integrated way, a proprietary XML-based certificate format has been defined.

Existing works prove the interest for an XML format for a certificate structure. A proposal for an XML format of SPKI certificates was taken into a Draft [32] for IETF, now expired. ITU-T group is working on specifications X.6xx [34] to translate ASN.1 into an XML format. Liberty Alliance has chosen SAML [22] (XML language for authorization assertions) for its Identity Management infrastructure; the Web Services Security standards fully support XML tokens, particularly SAML and XrML [23].

In order to satisfy the named requirements due to the mobile environment, and because many standards are not – or were not at the start of the project – mature, a new format and a new concept of certificates has been developed. WiTness certificates are in between SPKI and PKI certificates, and are neither pure Authorization Certificates, nor Identity Certificates. Figure 2 compares the X.509, SPKI and WiTness Certificates concepts.

WiTness certificates have been designed as a very flexible data structure oriented to distributed credentials management easy to deploy and use.

XML Attribute Certificates The WiTness Certificate data structure is constituted of these sections (quite conventional in a 'certificate' semantic):

- **Certificate Type**, containing application-level information such as version, content type, serial number

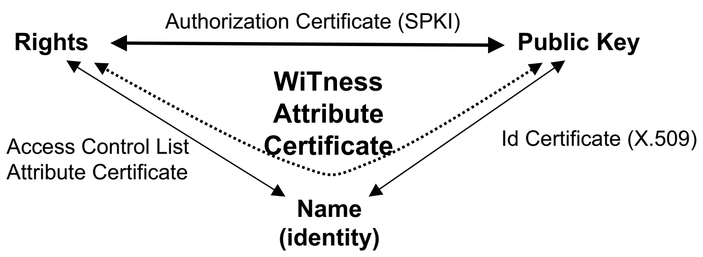

Authorization Certificate (SPKI)
Rights
Public Key
WiTness Attribute Certificate
Access Control List Attribute Certificate
Id Certificate (X.509)
Name (identity)

Figure 2: WiTness Certificate compared to SPKI and X509

- **Holder** of the certificate
- **Issuer** of the certificate (the signature on the certificate must correspond to the Issuer)
- **Attributes** associated to the holder
- **Duration** temporal validity for the certificate, in terms of initial and final validity date.
- **Signature** over the whole certificate - except the signature itself - made by the issuer.

Figure 3 provides a graphical description of the XML schema for WiTness certificates.

The WiTness Certificate has been designed to be generic, extensible and flexible. The certificate associates to the Holder some Content with the tamper-proof guarantee of an Issuer through its signature. The Holder proves ownership of the certificate through the classic challenge-response protocol demonstrating he knows the private key associated to the public key contained in the public certificate.

Holder and Issuer The Holder and the Issuer of the certificate can be either a simple public key, when the certificate is associated to a new principal, or the existing issuer certificate from which the new one is generated from. This allows to have fully contained chains of certificates, since a delegated certificate can contain the full certificate of its issuer and so on, up to the root certificate which vouches for all.

In the off-line scenario this permits to validate the origin of a certificate without accessing any server, in a simple way: no need to rebuild and reduce SPKI chains; all the needed information is included in the certificate itself. The drawback of this method, is to have certificates which grows in size as the delegation levels increase. A reference to a certificate instead of the full XML certificate can be used for the issuer, if we are sure we can provide the full

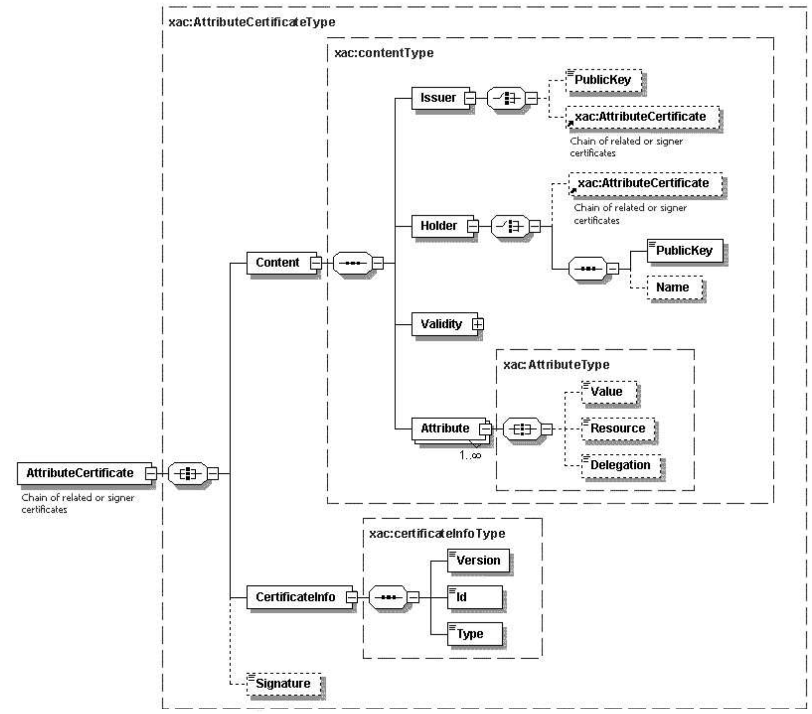


Figure 3: WiTness Certificate XML Structure

chain in another (off-line) method; the proposed inline method guarantees that all the information is always accessible from one single data structure. In a practical scenario, delegation levels should not grow much. The use of logical chains inside the same domain can help keeping the chain short.

The Holder can be a full certificate too. This permits to stick credentials to existing certificates, for instance certificates containing identity information, when required.

Attributes Most of the flexibility of this data structure comes from the very open Attribute format. An Attribute is defined by its *name*, *value*, *resource*, and *delegation*. The content of the attribute must be evaluated at application level, though some simple rules apply at library level to validate the certificate. The Name of the attribute, associated with the application-level certificate type, allows to apply semantic aware validation to the Value for the attribute, and the optional related Resource.

For instance, an attribute used for Identity information will be in an 'Identity' certificate, and will be named 'Last Name', and no related resource. Or an attribute used for file access rights will be called 'File Access', Value: 'RW', Resource: 'validFileName' or 'URL'.

The Delegation value indicates how many delegation levels are permitted for this attribute.

Delegation The WiTness Certificates are designed for a mixed off-line/on-line use. In order to achieve the goal of dynamic roles and rights attribution in an off-line scenario, particular care has been put in defining the delegation mechanism.

Any entity can act as an authority, creating new attributes or delegating owned attributes, and distributing them to other entities. The value of the Delegation field can be zero when the Issuer intends to give no right to delegate, any positive number to indicate the number of possible delegation levels (1 means the attribute can be delegated one time only, 2 means two times, etc.), and -1 means no limit. In a chain of certificates, Delegation must be coherent, and must always decrease (with the exception of the right to unlimited delegation).

Certificate Chains and Validation The crucial point is being able to validate the origin of the Attributes of the certificate, i.e., the entity which signed the certificate, and then recursively go up in the chain validating all levels up to a *Trusted (root) certificate*. At the beginning, the only trusted certificate is the company certificate used to sign the owned certificates.

This chain is called a 'physical chain': the initial root certificate to sign a delegation certificate to an entity, which then delegated this certificate to another entity and so on, and since all certificates are generated one after the other they can be included in the final certificate in a nested structure.

A different kind of chain, called 'logical chain' is used to create inter-domain delegation, and correspond to an agreement between two companies. With a cross-delegation, we create a logical chain which adds the subject certificate to the trust certificates set of the entity which trust the signing cross-certificate. Figure 4 depicts such a scenario where employee α_1 of company A uses a device b_1 of company B to access some corporate resource. Both kinds of chains are necessary: the dashed arrow line represents the cross-certificate which creates a logical chain from company A to device b_1; the continuous lines represent two physical chains between respectively company A and user α_1 and company B and device b_1. In this way, an existing certification tree structure is added to the valid certificate domain.

Signature The signature section contains the value of a digital signature calculated on the full XML structure, except the signature itself. It guarantees the integrity of the certificate in respect with the specified Issuer. The signature is calculated by the issuer with its private key, and can be verified with the issuer public key contained in the issuer field of the certificate itself.

The WiTness framework has been designed to be run primarily on PDAs running PocketPC 2002. For limitations mainly due to the computational power and the Java virtual machine version (the Personal Edition [27], basically version 1.1.8), some development constraints have limited the features implemented in

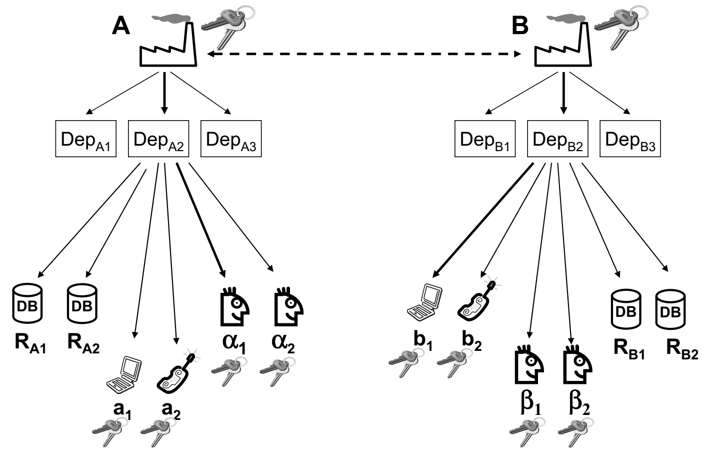


Figure 4: Physical and logical certificate chains

the WiTness Certificates Management Library. Implementing the XML DSIG standard is a desired improvement of the schema.

Off-line Revocation Revocation of certificates is a difficult problem in disconnected contexts. *Certificate Revocation Lists* (CRL) and *Online Certificate Status Protocols* (OCSP) enable to check which certificates have been invalidated. OCSP assume a way to contact this service: this approach cannot be deployed in off-line context. CRL can be used in off-line context as long as lists can be regularly updated. Unfortunately, the disadvantage of CRL is their potential large size, which makes them inappropriate for mobile devices. Another way to control the rights delegated to a third party is to rely on *short-term* [5] or even *one-time* certificates [3]. In this case, a mechanism for periodically renewing rights is necessary. It has already been shown [4] that revocation lists are not a optimum solution of the problem and thus an hybrid approach was chosen in WiTness: long-term certificates are used to define the roles, rights, or security levels of employees and devices and short-term certificates are created when rights are delegated. As in SPKI, the 'Validity' field of the certificate indicates the temporary validity of the credentials. A possible extension of this scheme is to use certificates for updating policies that are enforced by the security module. For instance, it could be possible to define a shorter validity time to be used when delegating rights so that when a user leaves (no more in Bluetooth range), the terminal looses all delegated rights that are no more renewed.

3.2 Addressing the off-line scenario

One scenario implemented within the WiTness project is the use of a local terminal for dealing in a user-friendly way with corporate data like e-mails (see Figure 4). The scenario is as follows: Alice (α_1) is an employee of company A and travels from place to place in a pervasive environment. Alice only carries a small personal device (a_2) that is trusted but has strong limitations (cannot print, small display, small keyboard, etc.). A PDA was used in a prototype but the concept can be used for cell-phones, watch, or even e-ring. Due to those limitations, discovered local facilities are federated to extend the personal device.

For instance, Alice will print some corporate e-mail on the printer in an airport lounge or edit some files (R_A) by using a public terminal or the laptop (b_1) of Bob (β_1) working in another company (B). It is assumed that edited and printed resources are stored in the personal devices of Alice. The following access control requirements have been defined:

1. Is employee α_1 authorized to access resource R_A?
2. Is device b_1 trusted enough to edit resource R_A?
3. Is α_1 authorized to use device b_1?

Each question has to be solved locally even when Alice uses the terminal b_1 spontaneously, i.e., for the first time and without having planned to do so. The WiTness framework relies on the credentials defined in section 3.1:

1. Employee α_1 has a pair of public and private keys and is certified by her employer $\mathrm{CERT}_A(\alpha_1, \mathit{attributes})$ where attributes can be authorizations or roles. This authorization certificate is used to access resources.
2. Device b_1 has its own pair of public and private keys and is certified by its owner $\mathrm{CERT}_B(b_1, \mathit{attributes})$ where attributes define ownership and optionally security-levels. 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. This agreement is another credential: $\mathrm{CERT}_A(B, \mathit{attributes})$ where attributes can define that devices of B are trustworthy enough to deal with confidential documents but cannot be used to deal with secret documents.
3. Agreements between corporations can also define rights. For instance, company B can authorize employees of company A to use public terminal or company B can authorize any visitor, who went to a front desk, to use local printers. To use b_1, Alice has to prove that she is authorized. For instance, she could prove that she works for company A and that any employee of this company is authorized by B to use b_1.

To summarize, Alice has to travel with authorization or role certificates that define her rights to access data, trust root certificates that define which devices can be trusted, and agreements that enable access to services of other companies. In other words, Alice has to cache a large amount of credentials or to upload credentials relative to the companies she plans to visit during her trip. Protocols for automatically uploading credentials on Alice's PDA according to corporate security policies have not been developed, yet.

3.3 Going online, or the benefits of transient connectivity

Transient connectivity is obviously a realistic option for many situations and can improve the usability and in some cases the security of a WiTness secured application. When some credential is missing and the user's device is temporarily online, it becomes possible to establish a connection to the user's authority or any other authority and get this credential. For instance, if Alice meets Bob and does not have the appropriate credentials to work with him, she could initiate a communication with her corporation in order to retrieve agreements between her company and Bob's. This is clearly a possible enhancement of the initial framework that requires a protocol for negotiating and exchanging credentials. Three different situations may occur:

- Initial behavior: all required credentials are available locally and the interaction is done off-line.
- Credentials are missing and a connection is established to get the required credential.
- Credentials are missing and a connection cannot be established for physical or policy-related reasons. In this case, either the service access fails or the service is downgraded, for instance secret data might not be accessible until a connection is indeed available.

4 An Online-Oriented Approach: Web Services Security

In this section we show how to implement access control and the exchange of credentials using the Web Services Security specifications, in particular WS-Federation and WS-Trust.

Web Services bring the paradigm of service-oriented architecture in practice. They offer an interoperable framework for stateless, message-based and loosely-coupled interaction between software components. These components can be spread across different companies and organizations, can be implemented on different platforms, and can reside in different computing infrastructures.

There are many and broad definitions of the *Web Services* concept. In the scope of this paper, we refer to Web Services as services that expose useful functionality on the Internet via XML messages which are exchanged through a standard protocol, called SOAP [10]. The interface of a Web Service is described in detail in an XML document using WSDL [11]. Finally, a Web Service is registered at a UDDI [12] server, and is as such discoverable. In addition to these basic features of Web Services, it is essential that Web Services are provided in a secure and reliable way, and that they support transactions. Specifications for secure, reliable, and transacted web services have been and are being developed by IBM, Microsoft, and others [7].

Web Services are currently standardized in different bodies. The core underlying messaging related technologies (such as XML, SOAP, WSDL) are standardized within the W3C. Other mechanisms (such as UDDI and WS-Security) are standardized within OASIS. The more recent specifications are proposed as industry initiatives by IBM, Microsoft, and others.

An important implicit assumption taken in the typical scenarios leveraging the Web Services Security specifications, is the permanent availability of all required services. As a natural starting point, we will therefore build up the email scenario with global connectivity. Subsequently, we will try to transform the on-line solution into an off-line one, which is still using the Web Services Security specifications.

4.1 Web Services Security

The Web Services Security architecture and roadmap [13] was proposed by IBM and Microsoft in April 2002 and provides a comprehensive security framework for SOAP-based Web Services. The framework supports and integrates various security models, mechanisms, and technologies in a way that enables a variety of systems to securely interoperate in a platform- and language-neutral manner. The Web Services Security roadmap consists of different specifications, each addressing specific parts of the security framework. The framework and the specifications are flexible and extensible.

The architecture essentially supports the secure exchange of SOAP messages in the following way:

- WS-Security [14] specifies how to apply XML Signature [8] and XML Encryption [9] within a SOAP message in order to provide single-message authentication (origin authentication and integrity) and single-message confidentiality. It also specifies how to attach, or refer to, the associated security tokens in a SOAP message.

- WS-SecureConversation [15] specifies how to establish, and how to reference to, a security context, and supports a secure conversation with multiple messages.

- WS-Trust [16] 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. Security tokens may be forwardable, delegatable, or proxiable. Security tokens are issued by a Security Token Service (STS). Security token requests may be secured using WS-Security (when exchanging one security token for another), or are secured with an explicit challenge/response or other negotiation protocol (when the requestor does not have a WS security token yet).

  The Web Services Security framework can in fact support any type of security token. However, security tokens that are to be used across different domains should be interoperable. The following tokens are currently standardized and explicitly supported: username/passwords [20] and X.509 certificates [21]; binary security tokens, such as Kerberos tickets [24]; and XML security tokens, such as SAML assertions [22] and XrML tokens [23].

- WS-Policy provides a framework which allows Web Services to describe and communicate (publish) their policies. WS-SecurityPolicy [17] specifies the security policy assertions that can be used in this framework. The security policy assertions state requirements on the kind of security tokens used, whether or not a message should be signed or encrypted.

- WS-Federation [18] 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. WS-Federation specifies scenarios using WS-Trust for example to allow requesters from the one domain to obtain security tokens in the other domain and subsequently get access to the services in the other domain. Additionally, mechanisms are defined for single sign-in and sign-out, sharing of attributes based on authorization and privacy policies, and integrated processing of pseudonyms (aliases used at different sites/federations).

One implementation of these specification is in the Web Services Enhancements toolkit (WSE) [26]. The current version supports WS-Security, WS-SecureConversation, WS-Trust, WS-SecurityPolicy, and Kerberos tokens.

4.2 Web Services and WiTness federation scenarios

Web Services are an ideal framework for service-oriented business processing between organizations and individuals. Web Services are obviously also an alternative and standardized approach to enable federations of devices to interact with each other, within the different WiTness scenarios discussed in this paper.

Note that the term ‘federation’ has a different meaning in the context of WiTness and Web Services. Federation in WiTness primarily refers to the notion of two or more devices that are close to each other, and whose features and services are jointly used to achieve a single purpose. Federation in Web Services refers to the fact that services are exposed and accessed across different trust domains. The Web Services notion of federation is clearly also present in WiTness when the federated devices belong to different trust domains with different security policies.

A very recent Web Services specification is particularly relevant to WiTness federations, as it defines a multicast discovery protocol to locate services that are exposed by nearby devices [25]. In the remainder of the paper however, we assume that devices already know which nearby services are exposed, and how they can be contacted. We focus on the security and authorization issues, and investigate how the Web Services Security specifications can support the WiTness scenarios.

4.3 Securing the on-line scenario

When implementing the email scenario using Web Services, a first idea may be to use the federated device as an active SOAP message intermediary, which may have the additional functionality of displaying parts of the messages that it relays. Unfortunately, this approach cannot be used because the user interacts with the federated device, so the user input actually goes into the federated device which therefore becomes an originator of SOAP messages.

Therefore it seems natural to closely follow the WS-Federation Active Requestor Profile [19]: the first request is initiated by the personal device and includes transferring control of the public terminal to the owner of the personal device. The detailed scenario is shown in Fig. 5:

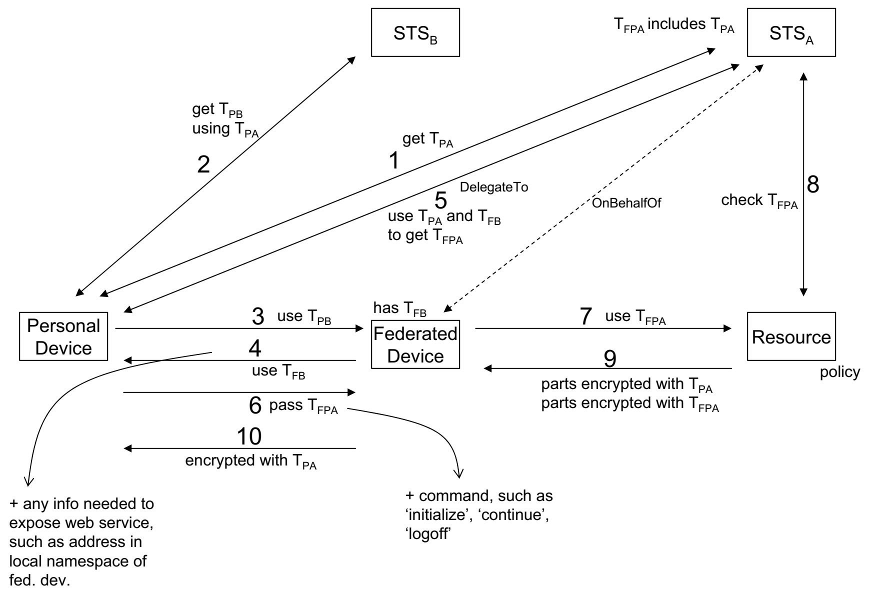

STS$_B$
T_{FPA} includes T_{PA}
STS$_A$
get T_{PB} using T_{PA}
2
get T_{PA}
1
DelegateTo
5
use T_{PA} and T_{FB} to get T_{FPA}
OnBehalfOf
check T_{FPA}
8
Personal Device
3 use T_{PB}
has T_{FB}
Federated Device
7 use T_{FPA}
Resource
4
use T_{FB}
9
parts encrypted with T_{PA}
parts encrypted with T_{FPA}
policy
6 pass T_{FPA}
10
encrypted with T_{PA}
+ any info needed to expose web service, such as address in local namespace of fed. dev.
+ command, such as 'initialize', 'continue', 'logoff'

Figure 5: possible realisation of the on-line scenario with Web Services.

1. The personal device fetches a security token T_{PA} from its own security token service STS_A. T_{PA} is fetched using WS-Trust, for example with

a username and password in domain 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 personal device then performs a WS-Trust token exchange, and gets a security token T_{PB} from the security token service STS_B of domain B.

3. The user steps in front of the public terminal and uses the personal device to log on. In particular, the functionality of the public terminal (federated device) is exposed via a set of web services (we assume that these have been discovered by the personal device). 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 federated device is signed with a token T_{FB}. The response indicates success or failure of the logon attempt, and may include any info needed for the personal device to expose web services to the federated device.

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}.

   Note: alternatively, the federated device could request a security token *OnBehalfOf* the personal device; the disadvantage would then be that the personal device does not authenticate itself to STS_A; the advantage would however be that the federated device authenticates itself; in the proposed scenario with *DelegateTo*, the token T_{FB} is validated by STS_A, but STS_A cannot check if the federated device claiming to be T_{FB} is really present.

6. T_{FPA} is passed on to the federated device, along with the command to initialize the federated device, and start working.

7. The user can now switch to the user interface provided by the public terminal, and access the email server in domain B. The email server is exposed by a web service, and access requests are secured with T_{FPA}.

8. The email server can check the validity of T_{FPA} by contacting STS_A (using WS-Trust validation). The server can now enforce a policy, which may for example mandate that – depending on the authorization assertions in T_{FPA} – certain parts of an email should not be available in cleartext in, and 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}, as it only has that corresponding private key.

10. The sensitive parts of the email can be forwarded by the federated device to, and decrypted and displayed by, the personal device. Forwarding is done upon request by the user through the user interface of the federated device. Forwarding is done as a request to a web service exposed by the personal device.

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

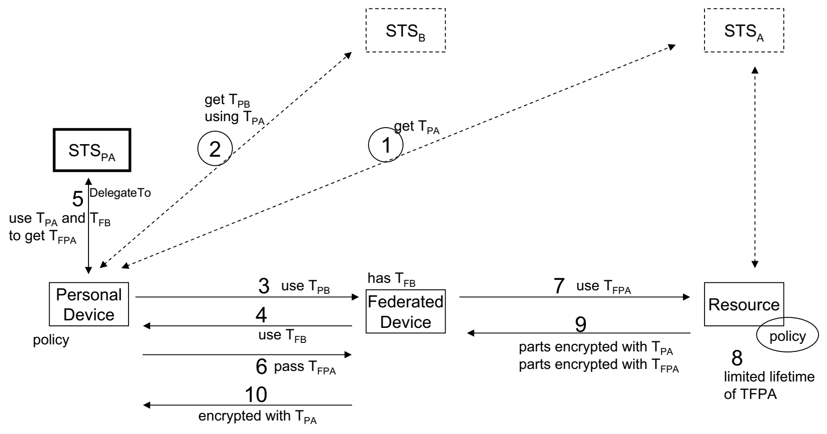
The realization proposed above very closely follows the Web Services Security specifications in the standard. Exchange of security tokens and communication of policy is performed as specified in WS-Federation, WS-Trust, and WS-Policy. The web services themselves that must be exposed by the federated device, personal device, and the email server, are of course scenario and application-specific, and their interface needs to be agreed upon. The feature of multipart encryption of emails, and the implementation and behavior of the federated device accordingly is also scenario and application-specific. Last but not least, it is important to note that the Web Services Security specifications do not mandate the format and contents of the security tokens, particularly T_{FPA}.

4.4 Taking it off-line

As in Section 3, the solution for the case with temporary unavailability of connection to the corporate servers can be solved with the same architecture as the case with complete unavailability. Looking at the detailed scenario described above, we can identify the following steps which may be problematic in an off-line scenario:

- Step 1 assumes the personal device to be able to contact its own STS. In an off-line scenario, the personal device may not be able to communicate with a service in a remote domain. The personal device should therefore already possess the required T_{PA} that can be validated and trusted by an STS in another domain.

- Step 2 assumes the personal device to be able to contact the STS_B of the federated device. While the personal device will be "closer" to the STS_B of the visiting domain than to its own remote STS_A, a limited personal device may only be able to communicate with the physically nearby federated device. This is more problematic: it assumes that the personal device has already the required T_{PB} that can be validated and



STS$_B$
STS$_A$
get T$_{PB}$ using T$_{PA}$
2
get T$_{PA}$
1
STS$_{PA}$
5
DelegateTo
use T$_{PA}$ and T$_{FB}$ to get T$_{FPA}$
Personal Device
policy
3 use T$_{PB}$
4 use T$_{FB}$
6 pass T$_{FPA}$
10 encrypted with T$_{PA}$
has T$_{FB}$
Federated Device
7 use T$_{FPA}$
9 parts encrypted with T$_{PA}$ parts encrypted with T$_{FPA}$
Resource
policy
8 limited lifetime of TFPA

Figure 6: possible realization of the off-line scenarios with Web Services. The grayed out boxes are the security token servers that are unavailable. The resource is now a locally cached copy of the emails.

trusted by a federated device in another domain. In other words, federated devices must be able to directly validate T_{PA}, or the personal device must carry a set of T_{PB}s for each of the domains trusted by A.

- Step 5 assumes the personal device again to be able to contact its own STS, which may not be possible in an off-line scenario. This could be solved by introducing an STS_{PA} running on the personal device, and capable of issuing delegatable security tokens.

  Obviously, STS_{PA} should be prevented from misuse, for example by the user of the personal device. Its private key should be stored and operated in a secure environment (e.g., smart card or trusted platform), and moreover it should only return delegation tokens T_{FPA} with authorization assertions that are compliant to the presented T_{PA} and T_{PB}. In addition, STS_{PA} should issue tokens that can be linked up to STS_A but that are specific to the personal device. We will discuss in the next section how the WiTness framework can provide a solution in this context.

- Step 8 assumes the email server to be able to contact its STS, which may not be possible in the off-line case. A solution would be to significantly limit the lifetime of T_{FPA}. The user cannot sign out anymore in an off-line scenario, and a limited lifetime of T_{FPA} prevents a federated device from keeping on using the valid T_{FPA} to access a resource on behalf of the user.

- The scenario assumes that the email server is consulted on-line. Emails or other information may be stored off-line on the personal device, and the user may want to view thes

- Step 2 assumes the personal device to be able to contact thee cached files on the federated device. This would require the security policy description be embedded in, or attached to, the files, or be present within the personal device. It also requires the security policy enforcement to be performed within the personal device.

In conclusion, enabling off-line applications is possible by not only working with a local copy of the corporate resource, but also with a local security token service, and/or one or more locally cached security tokens. In the next section we will discuss the drawbacks and trade-offs that this implies, and in particular we will show how the WiTness framework can be of benefit here.

5 Synthesis: Combining WS Federations and a WiTness approach

When looking at the proposed solutions in the two frameworks, the most apparent difference are the connectivity models. This is reflected in the much more complex structure of the certificates used in the WiTness framework. The Web Services Security specifications do not explicitly support tokens which are equivalent to WiTness certificates. However, the Web Services Security specifications in principle support any type of XML tokens, so using WiTness credentials within Web Services should be straightforward.

There is also a more fundamental difference. The WiTness framework is much more a development framework for implementing security within the application, offering modules and interfaces for security, while the approach that Web Services take is the one of a middleware: the security functionality should be transparent to the application. A policy is required to make the application secure.

It seems to be desirable to see if it is possible to combine these approaches to get the best of both worlds: the application designer can choose whether to rely on the transparent security mechanisms as provided by the current Web Services Security specifications, or whether to implement security mechanisms using an application layer framework, if a seperate security policy seems not expressive enough or too cumbersome.

5.1 Disconnected vs Connected Access Control

In general, disconnected access control schemes are more difficult to manage than connected models. Indeed, the fact that disconnected security cannot rely on some trusted third party when creating new credentials (e.g., rights delegation) and when validating credentials makes security more complex.

In this paper we defined credentials that can be validated and delegated locally, i.e., without requiring any communication with a centralized authority such as a Certification Authority (CA) or a central Security Token Service (STS).

Topic	**On-line**	**Off-line**
Cache	one authentication credential.	All right, role, or security level credentials.
Delegation	Tokens are always signed by a remote STS.	Any entity should be able to act as a STS.
Granularity	Get the expected token.	Delegate the appropriate subset of a cached credential.
Management	Centralized, up to date.	Distributed, no direct revocation.
Depreciated Mode	Always get appropriate credentials.	Cannot always get appropriate credentials.

Table 1: Comparison between on-line and offline models

Certificates [28, 5] are generally thought as disconnected credentials. However, to check whether a certificate has been revocated a connection is often needed [1] and thus it becomes a connected model. Electronic cash [3] may also be seen as a disconnected credential dedicated to specific type of rights. In comparison, Kerberos' tokens are connected credentials. The security relies on a permanent way to access the ticket granting server. Web services federations (WS-Federation) extend this model and generally rely on a permanent connection with some STS during delegation and validation phases. We will show how it is possible to define disconnected instantiations of this model.

Section 4.4 shows what is required to offer web services in a disconnected scenario. The main difference is that any device requires to be able to generate new security tokens. Indeed, when disconnected, it is no more possible to request a specific security tokens when required. Moreover, it is not possible to request in advance and store all possible tokens because spontaneous interactions cannot be planned. Like this, each device has to offer a Security Token Service in order to create appropriate tokens on demand. The delegation mechanism presented in Section 3 seems a natural way to assure that such an embedded STS will not provide more rights than what it is allowed.

The main differences between on-line and off-line approaches in terms of rights delegation management are described below and summarized in Table 1.

- The storage of credentials is necessary in an offline model, the number of credentials that have to be available on the personal device can be important when there are numerous inter-corporation agreements and when users have multiple roles and rights. All credentials representing rights, roles, or security-levels of an entity have to be cached by this entity. On the contrary, in an online model, only one credential is required for authentication of requests in order to obtain a given credential.

- Spontaneous collaboration with unknown entities implies some form of delegation. In offline schemes, any entity has to act as a CA to delegate a subset of its rights. Only required rights should be delegated and thus the granularity in terms of validity time and delegated attributes has to be as precise as possible. Online scenarios make it possible to request the STS to create a new credential for another entity.

- Rights management is more difficult in a disconnected mode because policies are distributed and certificate revocation lists cannot be used. Each time an entity is online, it is necessary to synchronize its rights with corporate security policies and to update cached credentials.
- Because it is not always possible to have the required credential locally and because an online connection can be impossible (or not desired), some otherwise authorized interactions may fail or be done in a depreciated mode.

The right part of Figure 7 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 off-line context, A authorizes a federated device FD to access R. The left part of Figure 7 shows how those credentials could be integrated in Web Services: 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} in Figures 5 and 6.

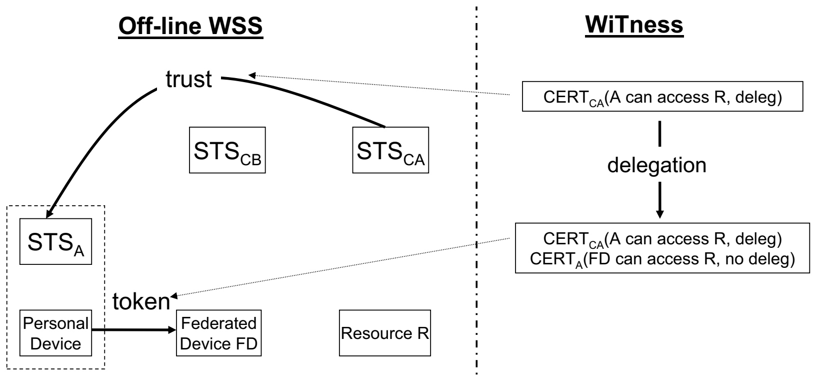


Figure 7: Basic example showing how WiTness credentials could be used as WS security tokens.

5.2 Application and Middleware Approach

For instance, in the WiTness scenario [2], users access information through a public terminal, which they federate with their personal security device. As parts of the information they want to access are sensitive, these parts will be encrypted such that the public terminal cannot display them. They can only be decrypted and displayed on the personal secure device. For this scenario, WiTness provides the authorization framework that focuses on certificates. It also offers the reuse of these certificates in a natural way for the encryption and decryption of the multipart encrypted email. WiTness provides a very flexible application-layer security framework which enables the development of fully customized, scenario specific, solutions. It may however be difficult to guarantee interoperability, particularly across different platforms, as there is for example no standardized secure messaging infrastructure which can be relied upon. Developing new applications may also be difficult for non-security experts.

This paper shows that it is possible to build the same federated scenario, whereby all of the messaging (interfaces) is done over Web services using the Web services security specifications. This should in principle guarantee interoperability. Web services security implementations also intend to make development of secure applications as transparent as possible. However, the specification of the multipart encryption is an example that shows that part of the security must still be handled at the application-level, on top of the web services. Web services security tokens can give input to that, but the real decision on what to do with the data is done within the application. It may not be feasible to realize security within the application, purely by relying on the interaction of the application with the security services through a configured security policy. Direct and customized invocation of security mechanisms from within the application may still be required.

6 Related Work

This section describes some related work which is relevant in the context of this paper.

XACML [29] defines an XML schema for an extensible access control language. It provides a common language for expressing security policies within an enterprise, and allows a consistent security policy 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 Next-Generation Secure Computing Base (NGSCB) [30] combines computer hardware platform enhancements with trustworthy computing capabilities and services. NGSCB provides strong process isolation, sealed storage, attestation, and secure i/o paths to the user. Within the WiTness federated scenarios, NGSCB could provide an increased trust assurance, and the security policy could grant NGSCB-enabled devices a higher security level.

The Liberty Alliance [31] specifications intend to provide a federated network identity management infrastructure. While Liberty Alliance 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) is an XML-based framework designed to solve three main use cases: Single sign-on (SSO), distributed transaction (dealing with transport of cross-domain authorization), and authorization service (dealing with authorization). It exchanges security information: XML-encoded security assertions, XML-encoded request/response protocol, and rules on using assertions with standard transport and messaging frameworks. There are three kind of assertions: authentication, attribute, and authorization decision. Different kind of assertions can be defined.

Other ongoing research work shows that Web services seem promising in mobile environments and even in pervasive computing environments. Mobile web services address the access to web services from mobile environments and pervasive web services focus on the discovery of services. Authors of [6] propose that a reference to a dedicated web service be attached to physical objects (train ticket, room, etc.). Barcodes are proposed but RFID tags or infrared beacons could also be envisioned. Once a user get the address of the service he can access it by establishing a connection with the server. Our work does not focus on RFID tags but on Personal Area Networks and thus it goes one step farther by proposing that physical objects could embed web servers in order to suit disconnected interactions.

7 Conclusions

The main starting point of the research presented in this paper is the comparison of the WiTness framework and the Web Services Security specifications in the context of offline federations. We took example scenarios illustrating situations ranging from an environment in which authorities are ubiquitously reachable to an environment in which they are most of the time unreachable. We then evaluated how these scenarios could be realized with Web Services and within the WiTness framework. We showed that the scenarios can be realized with the WiTness framework and with the Web Services specifications. Both approaches have different interesting characteristics, and can learn from each other. The Web Services Security specifications may particularly benefit from supporting offline usable credentials as implemented in WiTness. The WiTness framework may benefit from the Web Services specifications as a standardized, secure messaging framework.

Federation capabilities, understood both in terms of personal area federations of devices as in the WiTness framework, and in terms of federations of authorities as in WS-Federation, are an essential feature for future developments of mobile and ubiquitous computing. From the assumption made in this paper, that offline situations will remain to occur in the future, we can conclude that the development and support of disconnected, i.e., offline usable, credentials and associated protocols is an important field of investigation in this context. This should be studied more, in particular also taking into account as much contextual information as possible, as is increasingly available in mobile and ubiquitous computing environments.

The Web Services specifications provide a standardized, secure messaging framework on top of which secure and interoperable applications can be build in a transparent way. However, the WiTness scenarios show that a large amount of security functionality remains to be provided within the application, and cannot be handled by the Web Services specifications as such in a straightforward manner. This suggests that further standardization may be needed at the Web Service application level. For example, specific security-related Web Services may need to be developed.

In summary, a trend that seems to be emerging from our comparison of both frameworks is that application level security will develop on top of a generic model for negotiating trust between authorities and will make it possible to adapt the credentials used to specific applications.

References

[1] 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.

[2] 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.

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

[4] R.L. Rivest, *Can We Eliminate Certicate Revocation Lists?* in proceedings of Financial Cryptography '98.

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

[6] 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.

[7] IBM and Microsoft. Secure, Reliable, Transacted Web Services: Architecture and Composition. September 2003. `http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnwebsrv/html/wsoverview.asp`

[8] IETF/W3C. XML Signature Syntax and Processing. W3C Recommendation 12 February 2002. `http://www.w3.org/TR/xmldsig-core/`. IETF RFC 3275. March 2002. `http://www.ietf.org/rfc/rfc3275.txt`

[9] W3C. XML Encryption Syntax and Processing. W3C Recommendation 10 December 2002. `http://www.w3.org/TR/xmlenc-core/`

[10] W3C. SOAP Version 1.2. W3C Recommendation 24 June 2003. http://www.w3.org/TR/soap/

[11] W3C. Web Services Description Language (WSDL) 1.1. W3C Note 15 March 2001. http://www.w3.org/TR/wsdl

[12] OASIS. Universal Description, Discovery and Integration. UDDI Version 3.0.1. 14 October 2003. http://uddi.org/pubs/uddi_v3.htm

[13] IBM and Microsoft. Security in a Web Services World: A Proposed Architecture and Roadmap. April 2002. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnwssecur/html/securitywhitepaper.asp

[14] OASIS. Web Services Security: SOAP Message Security 1.0 (WS-Security 2004). March 2004. http://www.oasis-open.org/committees/download.php/5941/oasis-200401-wss-soap-message-security-1.0.pdf

[15] IBM, Microsoft, RSA Security and VeriSign. Web Services Secure Conversation Language (WS-SecureConversation). December 2002. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-secureconversation.asp

[16] IBM, Microsoft, RSA Security and VeriSign. Web Services Trust Language (WS-Trust). December 2002. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-trust.asp

[17] IBM, Microsoft, RSA Security and VeriSign. Web Services Security Policy Language (WS-SecurityPolicy). December 2002. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-securitypolicy.asp

[18] BEA, IBM, Microsoft, RSA Security and VeriSign Web Services Federation Language (WS-Federation). July 2003. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-federation.asp

[19] BEA, IBM, Microsoft, RSA Security and VeriSign WS-Federation: Active Requestor Profile. July 2003. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/grfWS-FederationActiveRequestorProfile.asp

[20] OASIS. Web Services Security: UsernameToken Profile 1.0. March 2004. http://www.oasis-open.org/committees/download.php/5942/oasis-200401-wss-username-token-profile-1.0.pdf

[21] OASIS. Web Services Security: X.509 Certificate Token Profile. March 2004. http://www.oasis-open.org/committees/download.php/5943/oasis-200401-wss-x509-token-profile-1.0.pdf

[22] OASIS. Security Assertion Markup Language (SAML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[23] OASIS. based on the assumptions made in this paper. eXtensible Rights Markup Languaderatingge (XrML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=rights

[24] IBM and Microsoft. Web Services Security Kerberos Binding. December 2003. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-security-kerberos.asp

[25] BEA, Canon, Intel and Microsoft. Web Services Dynamic Discovery (WS-Discovery). February 2004. http://msdn.microsoft.com/webservices/?pull=/library/en-us/dnglobspec/html/ws-discovery.aspbased on the assumptions made in this paper.

[26] Microsoft. Web Services Enhancements for Microsoft .NET (WSE). http://msdn.microsoft.com/webservices/building/wse/

[27] SUN. Java Personal Edition http://java.sun.com/products/personaljava/ Java 2 Micro Edition http://java.sun.com/j2me/

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

[29] OASIS. eXtensible Access Control Markup Language (XACML). http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[30] Microsoft. Next-Generation Secure Computing Base (NGSCB). http://www.microsoft.com/resources/ngscb/.

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

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

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

[34] ITU-T *ASN.1 encoding rules: XML encoding rules*, ITU-T standard, December 2001, http://www.itu.int/rec/recommendation.asp?type=folders&lang=e&parent=T-REC-X.693