# A Secured Service Level Negotiation In Ubiquitous Environments

Mohamed Aymen Chalouf, Francine Krief

International Journal of Communication Networks and Information Security (IJCNIS) Vol. 1, No. 2, August 2009

# A Secured Service Level Negotiation In Ubiquitous Environments

Mohamed Aymen Chalouf[1] and Francine Krief[1]

[1]LaBRI Laboratory, University of Bordeaux,
351 cours de la Libération, F-33405 Talence Cedex, France
*{chalouf, krief}@labri.fr*

***Abstract*:** The goal of the ubiquitous connectivity is to enable mobile users to be permanently and transparently connected to the Internet. These mobile users are often connected via wireless networks like Wi-Fi or WiMax and consuming services that require a high Quality of Service (QoS) level such as video on demand or voice over IP. The wireless access to these services may make the concerned communications vulnerable to security attacks because of the open medium on which these access technologies are based. Hence, in ubiquitous environments, we need to guarantee both QoS and security for mobile users' communications. In such an environment, it becomes very difficult for service providers to satisfy these users' needs. A solution is to assign a profile to each user in order to optimize and automate the process of service level negotiation which enables guaranteeing QoS and security.

In this paper, we present a protocol for service level negotiation which uses Web Services and includes both QoS and security in its negotiation. Then, we propose to adapt it to ubiquitous environments by basing its processing on the user profile and by specifying collaboration with the IEEE 802.21 standard, which manages the mobility of users and participates in the creation of theirs profiles. After that, we provide the negotiation flow of this protocol with security features using WSS, SSL and IPSec. Since, these security protocols may have an impact on the negotiation protocol performances; we will also evaluate this impact. Test results and implementation aspects are also shown in this paper.

***Keywords*:** quality of service, security, service level negotiation, ubiquitous environment, user profile.

## 1. Introduction

To provide ubiquitous Internet, mobile terminals such as laptops, PDAs and smart-phones are equipped with many connection interfaces and wireless networks are widely deployed. Thus, mobile users will be able to connect anytime, anywhere and using different technologies especially wireless ones. In this context, the need of security is very increasing. This security could be introduced at different levels by implementing security protocols such as: RADIUS, DIAMETER, SSL/TLS, DTLS, IPSec, WEP, WPA, WPA2, etc. On the other hand, new services such as telephony over IP and video on demand require quality of service guarantees. This QoS could be enabled locally, in each domain, by the use of a QoS model such as IntServ and DiffServ, and extended to the end-to-end level. Therefore, in ubiquitous environments, communication will require both QoS and security guarantees which may depend on the used access network. In that environment, the challenge is to simultaneously provide QoS and security for communications of mobile users without compromising this mobility. This will allow users to easily change of access network for mobility reasons or because a new network, better corresponding to their needs, becomes available. One solution is to provide mobile users with capabilities of dynamic negotiation of a service level including QoS and security. In fact, a communication can involve one or more domains. So, the mobile user must initiate a service level negotiation with the different managers of the implied domains in order to establish an agreement on a service level that they will undertake to ensure it.

In this context, we had specified a negotiation protocol which allows the dynamic negotiation of a service level including simultaneously QoS and security. This negotiation protocol is based on the use of the Web Services (WS) technologies in order to provide the different negotiation parts with interoperability. Thus, the negotiation initiation can be easily based on the user profile, which will optimize and automate the negotiation process. Since this protocol can be used in order to enable service level in ubiquitous environments for critical communications, the negotiation flow can be attacked by malicious third party. For example, these attacks may aim to disable security level needed by the communication endpoints. Thus, we think that it is very important to secure the negotiation flow especially in ubiquitous environments where the negotiation can be initiated by mobile users connected via wireless access networks.

In this paper, we present a protocol which enables negotiating a service level covering both QoS and security in ubiquitous environments. Then, we study and implement the security of the negotiation ensured by this protocol. Indeed, we secure this protocol at different layers using WSS, SSL and IPSec in order to choose the most adapted solution.

The reminder of this paper is organized as follow: section 2 introduces the negotiation of service level before describing some results dealing with user profile. In section 3, the architecture of a protocol for service level negotiation in ubiquitous environments is detailed. Section 4 recalls the architecture of Web Services and the main features of the different protocols used in securing the negotiation protocol. In section 5, the implementation of the negotiation protocol is detailed. Section 6 shows test results. The last section concludes the paper and points out perspectives of this work.

## 2. General context

In this section, we introduce the service level negotiation. Then, we present some results relating to user profiles that help us in the definition of the user profile on which the negotiation in ubiquitous environments is based.

### 2.1 Negotiation of service level

The increasing need of QoS, security and mobility requires the dynamic negotiation of service level between users and service providers. In fact, service offering in IP networks is defined through a Service Level Agreement (SLA) which is a contract between the service provider and the user. The technical parameters of this SLA are grouped together in a specification called Service Level Specification (SLS). These parameters, defined in Tequila project [1] (only QoS parameters), constitute the negotiable part of the contract between a service provider and a client and can cover various aspects such as QoS, security and mobility.

To guarantee an end-to-end service level, the managers of the different domains implied in a service offer must agree on the SLS parameters. Thus, several protocols were proposed in order to provide dynamic service level negotiation such as QoS-NSLP [2], COPS-SLS [3], QoS-GSLP [4] and DSNP [5]. Generally, these protocols allow negotiation entities to establish a service level, modify or terminate it.

To allow service level negotiation in a self-management environment, we have proposed a protocol that we called SLNP (Service Level Negotiation Protocol) [6]. In such an environment, domain managers use different technologies whose integration is increasingly difficult and expensive. To overcome this problem, the SLNP protocol provides domain managers with interoperability by using Web Services technologies. This interoperability constitutes one of the major advantages of this negotiation protocol. Furthermore, the negotiated SLS using SLNP is easily extensible to new parameters because of its XML based definition.

Moreover, unlike the protocols mentioned above which negotiate only QoS, SLNP is one of a few protocols ([7] and [8]) which associates security to QoS to satisfy security needs which are increasing with the wide deployment of wireless networks. In addition, it allows SLS negotiation in ubiquitous environments by basing the definition of the SLS to negotiate on the user profile parameters and by collaborating with the IEEE 802.21 standard in order to provide users' mobility.

### 2.2 User profile

A user profile is a set of data relating to a user. This notion can be used in various contexts. For example the adaptation of media stream defined by the MPEG-21 [9] is based on the Usage Environment Description (UED) tool which offers standardized description of user characteristics and environment. This description covers four components: user, terminal, network and environment. In this case, the defined user profile is quite general, and the contained parameters could be very interesting in a service level negotiation context. Another example is the user profile defined when specifying a "smart" interface that allows users to negotiate QoS [10]. In this work, the needed QoS level is defined on the basis of application needs and user characteristics. However, SLNP combines QoS and security in its negotiation. Therefore, other parameters relating to security were specified for the SLS negotiation (Section 3.2.1).

## 3. A protocol for service level negotiation in ubiquitous environments

In this section we describe the global architecture of SLNP. Then, we detail its user profile based functioning that provide users with negotiation capability in ubiquitous environments.

### 3.1 Global architecture

SLNP was defined to guarantee an end-to-end service level negotiation in a self management environment [6]. In fact, the managers of the different domains implied in a service offer must agree on a SLS by the exchange of negotiation messages (Negotiate, Revision, Modify, Notify, Release and Response). These messages enable the establishment, the modification and the termination of a service level. Each message contains a SLS element specifying the negotiated parameters: QoS and security [8] (Figure 1).

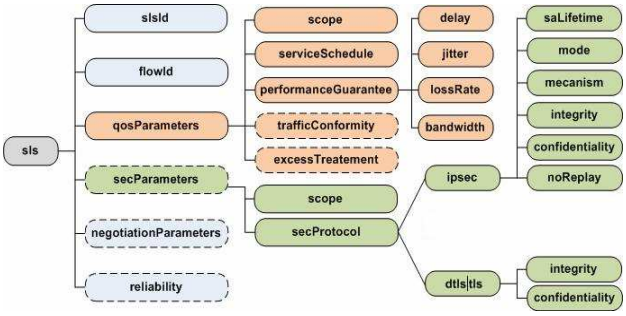


**Figure 1.** XML Schema for the negotiated SLS

The negotiation processing ensured by SLNP is the following (Figure 2). USER1, which wants to communicate with USER2, starts a negotiation by specifying the parameters of the desired SLS. This SLS is negotiated with the managers of the crossed domains (SE1 and SE2). During this negotiation, SLNP messages are exchanged between USER1 and SE2 in both directions. These messages are generally issued by the negotiation extremities (here. USER1 and SE2), but they are processed and modified, if necessary, by the intermediate entities (here. SE1). In order to process a message, an entity must interact with its Resource Management Function (RMF) that provides it with information on resources availability and requests admissibility. When negotiation entities agree on the negotiated parameters, SLS is established and recorded in SLS registries. After that, the QoS level will be guaranteed by configuring the concerned entities (e.g. Edge Routers), whereas security services will be offered at the network level using IPSec [11] or at the transport layer using TLS [12] or DTLS [13]. End-to-end security is configured by transmitting security information to the endpoints of the communication to secure [8]. Finally, the established SLS can be modified or released following USER1 request. The security impact on QoS could, in some cases, prevent the normal course of a communication. Hence, it is very important to consider it when negotiating both these two aspects [8].

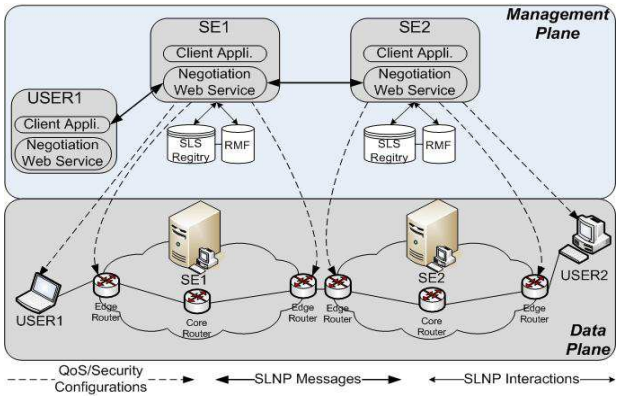


**Figure 2.** Global architecture of SLNP

To accomplish SLS negotiation, a user must have some expertise. On the other hand, in ubiquitous environments, SLNP has to manage users' mobility. Thus, to provide all users with negotiation capability in ubiquitous environments, negotiation process is based on the user profile and the user mobility is managed using the IEEE 802.21 standard [14].

### 3.2 User profile based negotiation

User profile is used to store information about the communication environment: terminal, application, access network, and user preferences. This information helps to establish a service level and to modify it if needed. In this part we detail the selected information constituting the user profile on which the negotiation is based. Then, we explain how this negotiation can be performed.

#### 3.2.1 User profile parameters

The selected information are divided into four types:

- **User preferences** contain three categories: *QoS*, *Security* and *Access network*. Regarding QoS, preferences are expressed by the desired level: *High*, *Medium* or *Low*. For security, user must specify if security is *Mandatory*, *Desired* or *Not-necessary*. In the two first cases, this user should select the needed services (*Authentication*, *Integrity*, *Confidentiality* and *No-replay*) and the level of each service (*High*, *Medium* or *Low*). Regarding access networks, user preferences are expressed by selecting a criterion for access network choice such as *Technology*, *Qos*, *Security* or *Cost*. Then, user will specify how this criterion is used in network choice.
- **Application characteristics** are essentially composed of the *Name* and the *Type* of the application that provide the negotiation layer with information on the minimal needed QoS level. Since an application can have its own security, *Security* information must be among these parameters.
- **Terminal capabilities** contain parameters such as *Screen size* and *Supported codec* providing indications on the required QoS. Moreover, these capabilities include *Performance parameters* (*CPU* and *Memory*) that give information about security impact on QoS. They also cover *Security protocols* and *Cryptographic algorithms* which are supported by the terminal which will help on defining security parameters to negotiate.
- **Access network characteristics** are composed of: an *Identifier*, an *Access technology*, a *Cost*, *Qos* and *Security* parameters. QoS parameters include *Latency, Jitter, Bandwidth* and *Loss-rate*. While security parameters specify the used *Security protocol* such as WEP, WPA or WPA2 that can secure Wi-Fi networks.

#### 3.2.2 SLS negotiation based on user profile

Since it uses Web Services, SLNP operates at the application level. Negotiation layer is therefore situated at this level, and composed of (Figure 3): Mapping and Negotiation Decision Point (MNDP), SLS Generator (SG) and SLNP Entity (SE).
The MNDP is responsible for choosing access network and for making negotiation decisions. These decisions are based on user profile parameters and changes that may occur. Then, when negotiation process should be started, the MNDP provides SG with SLS parameters defined according to user profile parameters. These parameters are used by the SG in creating the SLS element to negotiate, modify or release. Finally, the SE is composed of a client application and a negotiation Web Service (WS). The client application uses the obtained SLS to create the right message in order to start the corresponding process (Establishment, Modification or Release) by invoking the negotiation WS of the next entity.

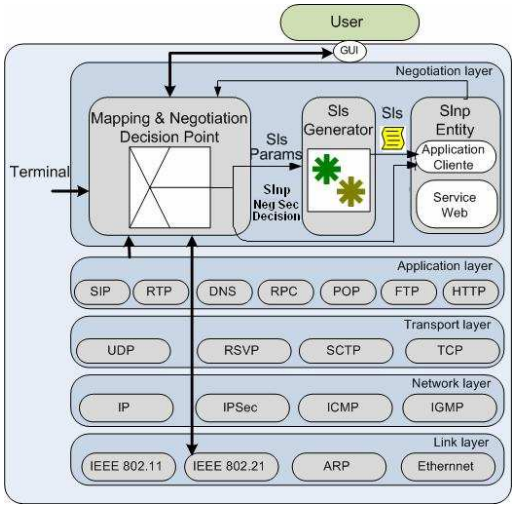


**Figure 3.** Overview of the negotiation layer

In the MNDP, the defined QoS level is adjusted taking into account security impact. Then, if the access network QoS can't ensure the required QoS, then the negotiation can not be started. In this case, if QoS level and/or security level can be degraded, then degradation is performed. The type of parameters to degrade (QoS, security, or both) depends only on the strategies implemented in the MNDP. This mechanism provides an internal negotiation, which enables avoiding the loss of time that can be caused by rounds of negotiations between the mobile user and the rest of the network. When a negotiation can be started, the MNDP transmits SLS parameters to the SG. Finally, negotiation result is returned by the SE to the MNDP that transmits it to the user.
When changes occur on user profile parameters, the entire MNDP computations are restarted which can lead to the modification of the already established SLS.
In ubiquitous environments, a user initiating a negotiation

can be connected to Internet via a non secured wireless access network. Thus, the negotiation flow can be attacked by a malicious third party in order to modify the needed service level. To overcome this problem, the SLNP signaling flow can be protected at different levels. The decision concerning this protection (SLNP Negotiation Security Decision) can also be taken by the MNDP which transmits it to the SE in order to secure the negotiation flow if required (Figure 3). This decision is also based on the user profile parameters. The security protocols that can be used in providing security services for the negotiation flow are detailed in the next section.

## 4. Security of Web Services

In this section, we introduce WS architecture and the standards on which this technology is based. After that, we describe some security protocols that can be used to provide WS based applications with security services.

### 4.1 Web Services (WS) technology

Web Services were designed to standardize exchanges on the Internet. Indeed, they allow an application to automatically find the needed service. The main characteristic of this technology is the interoperability that allows applications written in various programming languages (Java, C + +, Visual Basic, etc.) and running on various platforms (UNIX, Windows, etc.) to use WS to exchange data via Internet.

#### 4.1.1 WS architecture

The WS architecture is composed of three elements: a service provider, a service requester and a discovery mechanism. The provider creates a service and publishes its address (URI: Uniform Resource Identifier) in a WS directory. Thus, this last can provide the requester with information about the desired service (function, URI, etc.). This allows the requester to connect to the provider in order to acquire the service description and the call format.

#### 4.1.2 WS standards

Web Services are based on four standards: SOAP (Simple Object Access Protocol), WSDL (Web Service Description Language), UDDI (Universal Description Discovery and Integration) and XML (eXtensible Markup Language).

The SOAP protocol defines a set of rules for structuring the exchanged messages (Call, Response or Fault). SOAP is often associated to HTTP (Hyper Text Transfer Protocol) protocol to achieve request/response exchanges [15].

The WSDL standard allows describing the composition of a WS and how to access it. This includes details required to interact with the WS like the protocol to use, the URI, the performed operations, and the SOAP messages format [16].

To discover and locate a WS, we use a discovery mechanism like UDDI directories which contain information about WS. This will enable providers to register their services and requesters to search and locate the needed services [17].

The above-mentioned standards (SOAP, WSDL and UDDI) are based on XML. In fact, XML is used to define a language which can be used to describe all kinds of data and texts like SOAP messages, WSDL descriptions and UDDI entries [18].

### 4.2 Security of Web Services

Securing Web Services consists in providing security services (authentication, confidentiality, integrity, etc.) to the exchanged messages. This security could be introduced between two endpoints at the transport layer (SSL/TLS) or at the network level (IPSec). However, these two protocols become inappropriate to secure WS based exchanges; because these last could involve many entities where each of them may need to access some parts of the exchanged messages while access to other parts may be prohibited. Hence, security standards for WS were specified (Figure 4).

#### 4.2.1 Web Services Security (WSS) standards

WS Security (WSS) [19] allows protecting SOAP messages with XML Security. Indeed, WSS provides confidentiality using XML Encryption and integrity using XML Signature.

XML Signature [20] provides integrity, authenticity and non repudiation by enabling entities to sign an entire XML document or some parts of this document. An XML signature is an XML document containing information on the signing process (algorithm, key, etc.), references to the signed parts and the signature value. To process an XML signature, the transmitter generates a digest for each referenced part before calculating the digital signature value using the specified algorithms. Then the signed XML message is formed by incorporating the signature value, the different digests and information on used algorithms and keys. This will allow the recipient to proceed to the validation of this signature.

XML Encryption [21] provides confidentiality by allowing the encryption of XML data (document, element or content of element). The result of encryption is an XML document containing information on the encryption process (algorithm, key, etc.) and the encrypted data or references to these data. The encryption of XML data requires the selection of an algorithm and a key that will be transmitted to the pair. Then data are serialized before their encryption using the chosen algorithm and key. Finally, the message to transmit is formed by adding the encrypted data or reference(s) to these data and the information needed by the recipient for the decryption.

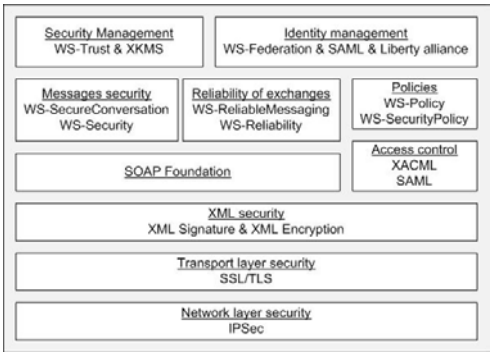


**Figure 4.** Web Services Security standards

Relying to these two standards, WSS provides SOAP messages with security. Indeed, it uses: XML Signature to sign a SOAP message and to transmit the signature, and XML Encryption to encrypt this message. WSS transmits security information in the headers of SOAP messages, such

as keys (encryption and signature) and security tokens (Kerberos tickets or X509 certificates) that represent identities and can be associated to digital signature in order to ensure authentication of the message origin.

To secure a SOAP message, WSS defines security headers. In fact, the header of a SOAP message can contain one or more security headers where each of them provides security information (signature and/or encryption) on this message to a recipient that can be the final or an intermediary recipient.

To sign one or more elements in a SOAP message, the security header, added by the transmitter, must include a signature which conforms to that specified by XML Signature. The recipient of a SOAP message must proceed to the validation of the signature. When validation fails, a Fault message can be delivered. Otherwise, the signature is validated and a confirmation may be sent to the transmitter in the header of the Response message, when this is required.

To encrypt one or more elements of a SOAP message, the security header must include references to the encrypted elements and information on the used key. Then, each element to encrypt is replaced by the equivalent encrypted data. The recipient of a SOAP message will identify the decryption key and the elements to decrypt. Thereafter, each encrypted element will be decrypted. If decryption fails, then a Fault message will be sent to the transmitter. Encryption and decryption are performed according to XML Encryption.

#### 4.2.2 Transport Layer Security (TLS) protocol

The TLS protocol [12] provides communication with end-to-end security (confidentiality, authentication, integrity and non repudiation) at the transport layer. It enables securing TCP based applications such as Web Services because it must rely on a reliable transport protocol (e.g. TCP). It is composed of two sub-layers: the TLS Record Protocol at the lower sub-layer, and four protocols (Handshake, Alert, Change Cipher Spec and Application Data) at the upper sub-layer (Figure 5). TLS can be used with several application protocols like IMAP, POP3 and HTTP on which WS exchanges are based. To be effective, the TLS protocol aims at reducing the number of cryptographic parameters to be negotiated by the way of two concepts: session and connection [12].

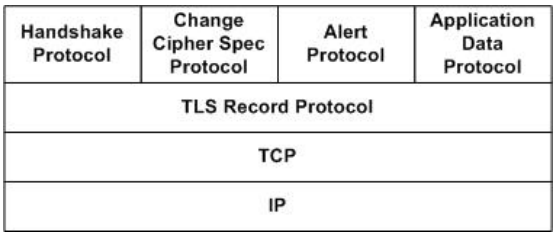


**Figure 5.** TLS protocol architecture

The record protocol provides security to the higher level protocols. Indeed, the integrity of the exchanged messages is provided by using a Message Authentication Code (MAC) which is computed with a hash function (SHA or MD5), and the confidentiality is ensured by a symmetric encryption (RC4, RC2, DES, or AES). At the higher layer, the handshake protocol allows the communication endpoints to agree on security parameters, to authenticate themselves and to exchange keys. The change cipher spec allows changing cipher specification by replacing the connection current states by some already negotiated pending states. Finally, the alert protocol allows an endpoint to detect an error and to send an informative message to the other endpoint.

#### 4.2.3 IP Security (IPSec) protocol

The IPSec protocol [11] permits protecting the traffic at the network level. It employs a Security Association (SA) in order to offer security services to the transported traffic. IPSec uses two mechanisms: Authentication Header (AH) and Encapsulating Security Payload (ESP). AH [22] provides integrity, authentication, and optionally non repudiation, whereas ESP [23] offers the same services and also confidentiality. Each mechanism supports two modes: the transport mode that protects the payload of IP datagram, and the tunnel mode where the protection covers the IP header.

An IPSec implementation may use three databases. The Security Policy Database (SPD) indicates the policies defining the treatment to apply to each traffic (Discard, Bypass or Protect), the mechanism, the mode, the options and the algorithms to use. The Security Association Database (SAD) contains the SA parameters. The Peer Authorization Database (PAD) associates the SPD to an SA management protocol like Internet Key Exchange (IKE). The IPSec processing for an outgoing packet is the following. If this packet corresponds to an already created SA, then it is treated as indicated by this SA. Otherwise, a research in the SPD is carried out (Figure 6). If the result of this research indicates a treatment of the type Discard or Bypass, then the packet is treated consequently. However, if the required treatment is Protect then the mechanism of SA management (e.g. IKEv2 [24]) is invoked to create a new SA.

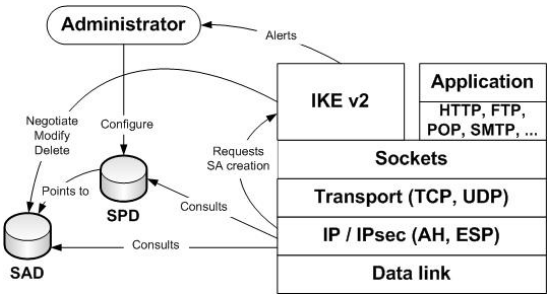


**Figure 6.** IPSec protocol processing

The security protocols described above can be used to secure the WS based negotiation protocol. In the next section, we show some implantation details as well as test results.

## 5. Implementation of the negotiation protocol

### 5.1 Components of the negotiation layer

According to its situation in a negotiation process, an entity can be initiator, intermediate and/or responder. When an entity initiates a negotiation process, it needs a Client Application (CA) that allows it to invoke the WS of the next entity on the negotiation path. In the case of an intermediate or a final entity, the client application will permit to notify an

initiator in order to inform it about changes in resources availability or about SLS violation. In addition, a negotiation entity needs a WS containing different operations (Figure 7) that will receive various request messages, treat them and return the suitable answer messages. Indeed, an intermediate or a final entity is concerned by the negotiation, modification, release and response operations, whereas an initial entity is concerned by the notification operation. Thus, a negotiation entity must contain a negotiation WS and a negotiation CA which compose what we called SE (SLNP Entity).

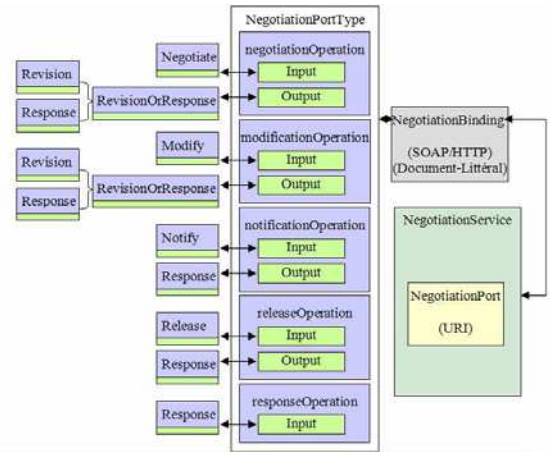


**Figure 7.** A WSDL representation of the negotiation WS

The negotiation WS has been defined through its WSDL description represented in Figure 7. This definition covers the various operations and the types of input and output messages associated to each of these operations. Indeed, the structure of each message type has been defined through an XML schema using the XSD (XML Schema Definition) language [25]-[26]. Since each message contains a SLS that specifies the negotiated service level, the generic structure of this SLS has also been defined using XSD (Figure 1).

The negotiation CA is used by the initiator to request the establishment or the modification of a service level by taking the SLS element as argument. In addition, it enables the SLS cancellation by taking its identifier as argument.

In ubiquitous environment, the negotiation process is initiated by the user. Thus, this initial negotiation entity will also include a MNDP and a SG which enable defining SLS parameters and generating the corresponding SLS element.

### 5.2 Implementation of the negotiation layer

In our SLNP implementation, we chose to use Tomcat of Apache as an application server and Axis as SOAP protocol implementation. The treatments included in: the MNDP, the SG, the various operations of the negotiation WS and the negotiation CA are written in Java programming language. This choice is explained by the fact that Tomcat and Axis are two open source projects. In addition, we dispose of two interesting tools. The first is WSDL2Java that permits to generate Java classes from WSDL description and the WSSD (Web Services Deployment Descriptor) file that facilitates the deployment of WS. The second interesting option is the ability to view the exchanged SOAP messages between the negotiation entities using SOAP Monitor or TCP Monitor.

In the case of the establishment of a SLS in ubiquitous environment, the MNDP is responsible for defining SLS parameters on the base of the user profile parameters, when a negotiation can be started (Figure 8). Indeed, the included treatments are based on the user profile parameters and cover: the access network choice, the definition of QoS parameters, the definition of security parameters and the impact estimation when security services are required, and finally the definition of the needed SLS parameters when negotiation can be started. Then the SG (SLS Generator) is in charge of creating the corresponding SLS element.

After that, the negotiation CA of the mobile user creates a Negotiate message with the already generated SLS. This message is used to invoke the negotiation operation of the next entity negotiation WS (Figure 9). After that, according to the returned message, the mobile user will: record the established SLS locally if the requested SLS is accepted (Response-Ack), accept or refuse the proposed alternative (Revision), or end the negotiation process (Response-Nack).

The SLS establishment initiated by a mobile user involves all the negotiation parts. Indeed, the sent Negotiate message will transit through all the intermediate entities and reach the final entity using a recursive call to theirs negotiation operations.

To optimize the implementation of the negotiation protocol, a negotiation entity must contain a single negotiation operation that may be invoked when it is an intermediate or a final entity. Thus, the processing of this operation must cover both these two use cases (Figure 10).

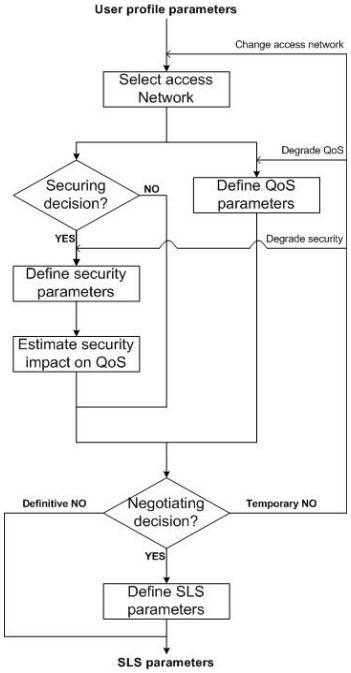


**Figure 8.** A general diagram of the MNDP processing

In the case of an intermediate entity, the received Negotiate message is modified according to QoS and security information provided by the RMF. Then this message is transmitted in the direction of the final entity by calling the negotiation operation of the next entity WS. The result of this invocation (Response or Revision) will be returned to

the caller. When it is a final entity, a decision must be taken (accept, reject or propose an alternative) and the corresponding message (Response-Ack, Response-Nack or Revision) must be returned to the caller in order to transmit it to the negotiation initiator. When the SLS is accepted, it is recorded locally by each negotiation entity.

### 5.3 Implementation of the negotiation flow security

The security services that we want to ensure for the SLNP negotiation flow can be provided by various security protocols at different levels of the TCP/IP protocol stack.

First, the SOAP messages exchanged during SLS negotiation are secured using WSS which operates at the application layer. So, we use the WSS4J library that implements the WSS specification in the Tomcat-Axis environment. In fact, the WSS4J API allows us to define different security operations to apply to a SOAP message using handlers implemented in Java. These handlers are transparent to the WS based negotiation protocol and control the creation and the use of secured SOAP requests and responses. The calls to these handlers must be placed in the deployment descriptors of WSS4J on each negotiation entity to describe the security measures to be applied. It is very important to note that an initial or a final entity is concerned with one set of security properties associated to the negotiation exchange with the next or with the previous entity, while an intermediate entity is concerned by two sets of security properties: one for the negotiation exchange with the previous entity and another one for that with the next entity.

Then, to secure the negotiation flow with SSL/TLS, we chose HTTPS because it is easily usable with the Tomcat server.

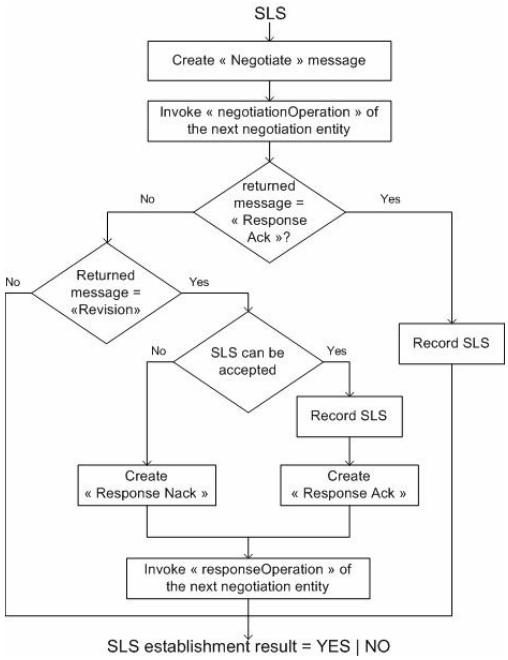


**Figure 9.** A general diagram of the negotiation CA processing

In fact, we created a connector which allows the application server of a negotiation entity to support SSL. This requires changing the Tomcat server configuration (server.xml) by adding a connector that associates SSL to a port (eg. 8443) in order to exchange securely the negotiation messages. Then, the URI addresses, allowing each entity to call the next one, are modified to enable the negotiation WS invocation through the ports secured with SSL.

Finally, we secure the negotiation at the network level by configuring IPSec on each entity. To do this, we opted for an IPSec implementation situated at the OS kernel. Indeed, on each entity, we installed two packages: the ipsec-tools package is used to manage the SPD and the SAD, whereas the racoon package implements the key management protocol IKE which permits establishing SA between negotiation entities. Each interaction between two adjacent negotiation entities is secured in the two directions using two SA.

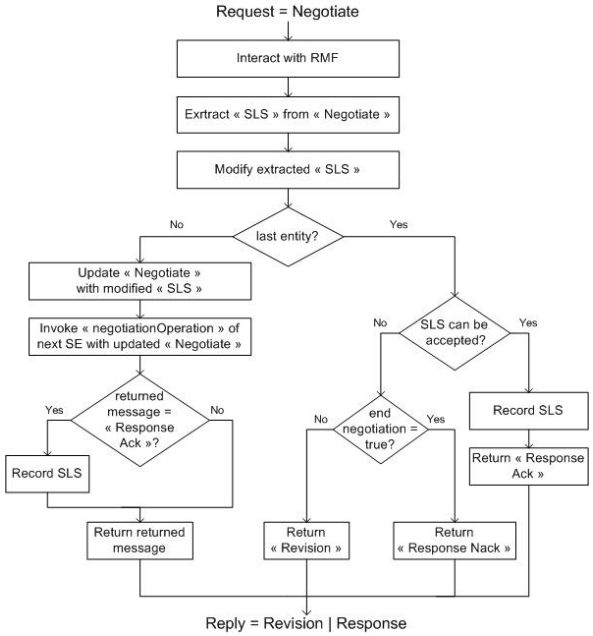


**Figure 10.** A general diagram of the negotiation operation processing

## 6. Tests and results

### 6.1 Testbed architecture

To perform local tests, we use an IBM system equipped with a Pentium IV, 2.5 GHZ processor and a 1GB RAM. On this system, we configure three negotiation entities. Each entity is composed of a negotiation WS deployed on a Tomcat server and a negotiation CA. The interactions of each entity with its RMF are simulated with a MySQL database. We note that for WSS and SSL tests, the three negotiation WS can be deployed on the same Tomcat server because each one is identified through its URI. Whereas to test IPSec security we have to create a virtual system for each entity because we need IP addresses to distinguish them and to create SA.

For each implemented security, we visualize the exchanged messages to check its good functioning. Then, we conduct a set of measurements of the two performances parameters of the protocol: the messages size and the negotiation time.

### 6.2 Test and evaluation of WSS security

The WSS security impact on SLNP performances is measured based on several scenarios that may correspond to real needs. The identified five scenarios are represented in Table 1.

| Scenario | WSS security features |
|---|---|
| *S0* | No security |
| *S1* | Simple authentication (username token and password) |
| *S2* | Strong authentication (encrypted username token and timestamp) |
| *S3* | Strong authentication and integrity (message signature) |
| *S4* | Strong authentication and confidentiality (message encryption) |
| *S5* | Strong authentication, integrity and confidentiality |

*Table 1.* Various WSS security scenarios

Tests are conducted for a single round negotiation process with well-defined SLS parameters for the request message which is presented in Figure 11 and Figure 12 respectively in the case of a non secured exchange and in the case of a secured exchange following the security policy used in S3.

```
- <soapenv:Envelope>
  - <soapenv:Body>
    - <Negotiate>
        <negotiationId>155</negotiationId>
        <endNegotiation>true</endNegotiation>
      - <sls>
          <slsId>13</slsId>
        + <flowId>...</flowId>
        - <qosParameters>
          + <scope>...</scope>
          + <serviceSchedule>...</serviceSchedule>
          + <performanceGuarantee>...</performanceGuarantee>
          + <trafficConformity/>
          + <excessTreatement>...</excessTreatement>
          <qosParameters>
        - <secParameters>
          + <scope>...</scope>
          + <secProtocol>...</secProtocol>
          <secParameters>
        + <negotiationParameters>...</negotiationParameters>
        + <reliability>...</reliability>
        </sls>
    </Negotiate>
  </soapenv:Body>
</soapenv:Envelope>
```

**Figure 11.** A SOAP message exchanged in the scenario S0

```
- <soapenv:Envelope>
- <soapenv:Envelope>
  - <soapenv:Header>
    - <wsse:Security soapenv:mustUnderstand="1">
      + <xenc:EncryptedKey Id="EncKeyId-12245160">...</xenc:EncryptedKey>
      - <ds:Signature Id="Signature-15055830">
        - <ds:SignedInfo>
            <ds:CanonicalizationMethod Algorithm="http://...xml-exc-c14n#"/>
            <ds:SignatureMethod Algorithm="http://...xmldsig#rsa-sha1"/>
          + <ds:Reference URI="#id-33431531">...</ds:Reference>
          </ds:SignedInfo>
        - <ds:SignatureValue>YA7J9itDce0wP5L7ZvX+ ... </ds:SignatureValue>
        + <ds:KeyInfo Id="KeyId-1696092">...</ds:KeyInfo>
        </ds:Signature>
      - <xenc:EncryptedData Id="EncDataId-20079748" Type="http://...xmlenc#Element">
          <xenc:EncryptionMethod Algorithm="http://...xmlenc#aes128-cbc"/>
        + <ds:KeyInfo>...</ds:KeyInfo>
        - <xenc:CipherData>
            - <xenc:CipherValue>gfRjYkMp0jCyere1a+jn...</xenc:CipherValue>
          </xenc:CipherData>
        </xenc:EncryptedData>
    </wsse:Security>
  </soapenv:Header>
- <soapenv:Body>
    - <Negotiate>
        <negotiationId>156</negotiationId>
        <endNegotiation>true</endNegotiation>
      + <sls>...</sls>
      </Negotiate>
  </soapenv:Body>
</soapenv:Envelope>
```

**Figure 12.** A SOAP message exchanged in the scenario S3

The size of the exchanged messages is evaluated using TCP Monitor, while the mean negotiation time is measured on a sample of 1000 measurements (Table 2).

| Scenario | Message size (Bytes) | Mean negotiation time (ms) |
|---|---|---|
| *S0* | 3749 | 66 |
| *S1* | 4427 | 82 |
| *S2* | 7300 | 151 |
| *S3* | 8710 | 205 |
| *S4* | 9235 | 164 |
| *S5* | 10623 | 217 |

*Table 2.* Impact of WSS on the protocol performances

We note that the size of SOAP messages increases when the negotiation exchanges are secured with WSS (Figure 13.a). This is explained by the introduction of security headers whose size depends on the provided security services and the mechanisms put in place. Indeed, a simple authentication increases slightly the message size because security header contains only the username token. However, the strong authentication increases almost twice the size of the same message (95%) because the security header includes all the information required for the encryption of the username token and the timestamp such as algorithms and keys. This message size increases by 183% when confidentiality and integrity are provided in addition to the strong authentication.

The negotiation time also increases when security is provided (Figure 13.b). This is due to security treatments performed by the different entities such as encryption, decryption, performing signature, validating signature, etc. Indeed, a simple authentication increases by 30% the negotiation time, while a strong authentication increases by 180% this time because it requires the encryption of the username token and the timestamp elements of the security header.

We note that the periodic peaks that can be observed on the curves of the negotiation time measurements (Figure 13.b and Figure 13.d) are explained by the memory management inside the Java Virtual Machine (the garbage collector).

### 6.3 WSS Security Vs SSL and IPSec Securities

In this part we compare the impact of different security protocols on the performances of the negotiation protocol.

To check the SSL security implementation, the exchanged messages can be viewed using the debug mode of the Tomcat server (SOAP Monitor and TCP Monitor cannot be used). Whereas IPSec security can be verified using TCPdump. To measure the impact of these two security protocol on the SLNP performances, we use the same evaluation criteria (i.e. size of the exchanged messages and the time spent in the negotiation) under the same conditions (same request message, one round negotiation, system characteristics, CPU and RAM consumption, sample of 1000 measurements, etc.).

In order to compare performances of WSS, SSL and IPSec (Table 3), we test security implementations with very close security levels. That means that, with all security protocols, we choose very similar algorithms

for each security service.

| Security type | Security features | Impact on performances |
|---|---|---|
| *No security* | Authentication : Null<br>Integrity : Null<br>Confidentiality : Null | Message size: 4560 octets<br>Mean negotiation time: 238 ms |
| *WSS* | Authentication : certificates<br>Integrity : SHA1<br>Confidentiality : AES-128-CBC | Message size: 11846 octets<br>Mean negotiation time: 457 ms |
| *SSL* | Authentication : certificates<br>Integrity : SHA1<br>Confidentiality : AES-128-CBC | Message size: 4929 octets<br>Mean negotiation time: 243 ms |
| *IPSec* | Authentication : certificates<br>Integrity : SHA1<br>Confidentiality : AES-128 | Message size: 5040 octets<br>Mean negotiation time: 267 ms |

*Table 3.* Comparative impact of security on performances

Figure 13.c shows that the size of the exchanged packets secured with SSL (4929 bytes) and IPSec (5040 bytes) is very close to that of an unsecured packet (4446 bytes), and is much less than that measured for a WSS security (11846 bytes). This can be explained by the fact that the IPSec and SSL protocol overheads are less important than those introduced by WSS. In addition, IPSec and SSL securities require a negotiation phase, during which IPSec associations and SSL connections are established, that allows the two communication endpoints to configure security parameters such as algorithms and keys. Whereas these security parameters are usually transmitted or referenced in the exchanged messages if WSS is employed.

Concerning the negotiation time (Figure 13.d), we note that time measured for SSL (243 ms) and IPSec (267 ms) secured negotiations is also very close to that of an unsecured one (238 ms). However, for the same security level, this time is equal to 457 ms when security is ensured by WSS. In fact, the treatments required for security services (encryption, decryption, signature computation, etc.) need more time when they are executed at application layer than that when they are performed at network or transport level.

For these tests, the configured IPSec security is characterized by the use of the AH mechanism to offer integrity and the ESP mechanism to provide confidentiality. In fact, we could use ESP in offering these two services, but we opted for this configuration because it is more robust. Indeed, the integrity offered by ESP has a slightly inferior quality than provided using AH, since it takes into account less IP header fields. Then, it can be very interesting to measure separately the impact of the AH and ESP mechanisms on the performances of SLNP. The results show that, for the same integrity level, the performances are almost similar. For example, the mean time of negotiation secured with IPSec is equal to: 242 ms when AH is used and 254 ms when ESP is employed.

### 6.4 Negotiation time in function of entities number

The above-shown negotiation time mesurements are performed using only three entities: initial, intermediate, and final. Without security, these measurements show a mean negotiation time equal to 66 ms for a single round. Since this time will vary in function of the number of the implied negotiation entities, we tried to evaluate this time according to the number of negotiation entities. We found that time spent in negotiation is proportional to the number of negotiation entities with a factor of 35.5 (Figure 14).

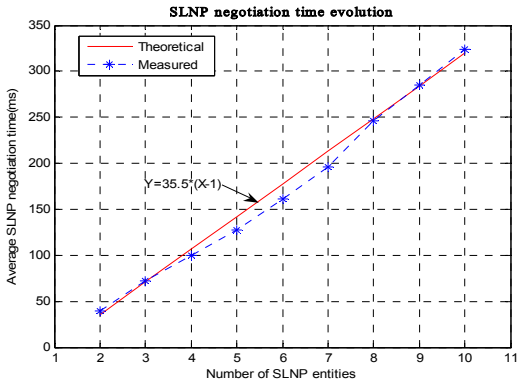


**Figure 14.** Evolution of the negotiation time

## 7. Conclusion

In this paper, we have described the architecture and the functioning of a negotiation protocol enabling QoS and security guaranties for mobile users' communications in ubiquitous environments. Then we have presented some protocols which can be used to secure WS based applications. These protocols are then used in securing the already introduced WS based negotiation protocol (SLNP).

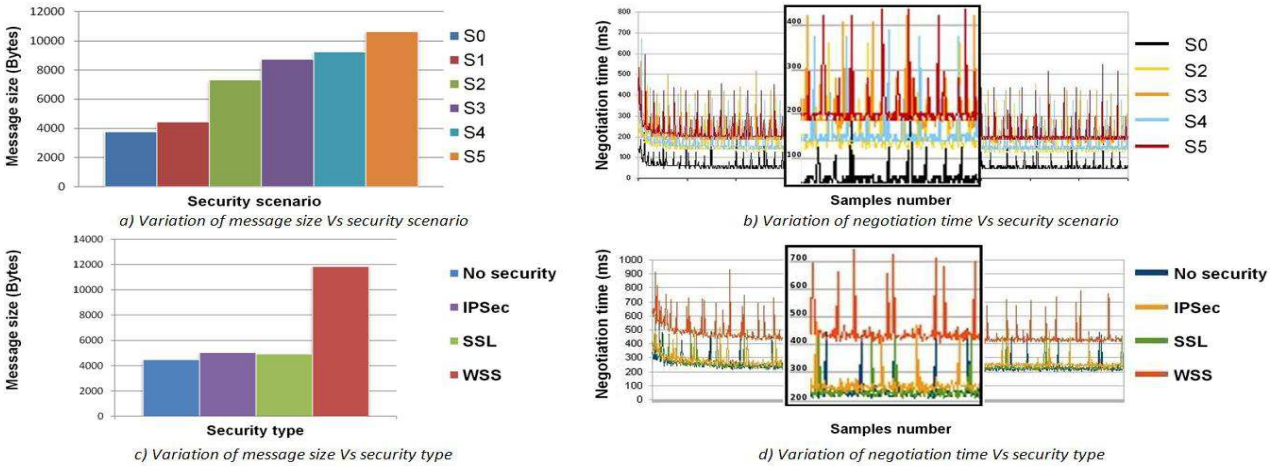


**Figure 13.** Impact of security on the performances of the negotiation protocol

After that, these security types were evaluated in terms of negotiation performances (size of exchanged messages and time spent in a negotiation process). From these tests, we conclude that for equivalent security levels, performance of the negotiation protocol secured with SSL or IPSec are significantly better than those secured using WSS.

In fact, one of the main benefits of WSS is the high degree provided granularity. Indeed, if we need to secure a SOAP message transiting through several Web Services by keeping confidential for some of these WS but not for others, WSS is the ideal solution; because SSL and IPSec provide end-to-end security. In the case of SLNP, messages are transmitted from one entity to another and there is no need to the granularity which can be provided by WSS. Thus, we can use quite SSL or IPSec to secure SLNP. The use of WSS will be allowed if the negotiation protocol performance is not considered or if the use of SSL or IPSec is impossible.

The results presented in this paper are obtained by conducting tests on entities involved in a single negotiation (i.e. at a given moment, each negotiation entity has only one request to treat). Thus, it would be very interesting, in the future, to evaluate the scalability of our negotiation protocol.

## References

[1] D. Goderis, and D Griffin, "Attributes of a service level specification template," IETF, draft-tequila, October 2003.

[2] S.V. Den Bosh, G. Karagiannis, and A. McDonald, "NSLP for Quality of Service Signaling," IETF Internet draft, draft-ietfnsis-qos-nslp-06, February 2005.

[3] T. M. T. Nguyen, and al., "COPS-SLS: A Service Level Negotiation Protocol for the Internet," IEEE Communication Magazine, pp. 158-165, May 2002

[4] Ambient Networks Consortium, "Connecting Ambient Networks – Architecture and Protocol Design (Release 1)," Del. D 3.2, March 2005.

[5] J. C. Chen, and al., "Dynamic Service Negotiation Protocol (DSNP) and Wireless Diffserv," Proc. ICC, New York, NY, pp. 1033-1038, April 2002.

[6] N. Mbarek, F. Krief, and M. A. Chalouf, "A negotiation Protocol Using Web Services in a Self-Management Framework," Global Information Infrastructure Symposium, GIIS 2007, Moroco, pp. 93-98, July 2007.

[7] S. Duflos, and al., "Considering Security and Quality of Service in SLS to improve Policy-based Mabagement of Multimedia services," ICN-07, Martinique, April 2007.

[8] M. A. Chalouf, X. Delord, and F. Krief, "Introduction of Security in the Service Level Negotiated with SLNP Protocol," Second IFIP International Conference on New Technologies, Mobility ans Security, NTMS 2008, Morocco, November 2008.

[9] ISO/IEC TR21000-7:2007, "Information technology – Multimedia Framework (MPEG-21) – Part 7: Digital Item Adaptation", 2007.

[10] Z. Jrad, and al., "A user assistant for QoS negotiation in a dynamic environment using agent technology", Proc. WOCN-05, UAE, Mars 2005.

[11] S. Kent, and K. Seo, "RFC: Security Architecture for Internet Protocol," Request For Comments 4301, December 2005.

[12] T. Dierks, and E. Rescola, "RFC: The Transport Layer Security (TLS) Protocol Version 1.1," Request For Comments 4346, April 2006.

[13] E. Rescola, and N. Modadugu, "RFC: Datagram Transport Layer security (DTLS)", Request For Comments 4347, April 2006.

[14] IEEE P802.21/D10.0, "Draft Standard for Local and Metropolitan Area Networks: Media Independent Handover Services", LAN MAN Standards Committee of the IEEE Computer Society, April 2008.

[15] D. Box, and al., "Simple Object Access Protocol (SOAP) 1.2," W3C Note, April 2007.

[16] E. Christensen, and al., "Web Services Description Language (WSDL) 1.1," W3C Note, March 2001.

[17] T. Bellwood, and al., "Universal Description, Discovery and Integration (UDDI) specification," Technical report, OASIS Committee, July 2002.

[18] T. Bray, and al., "Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, November 2008.

[19] A. Nadalin, and al., "Web Services Security Specification 1.1," OASIS Standard Specification, OASIS Committee, February 2006.

[20] M. Bartel, and Al., "XML Signature Syntax and Processing (Second Edition)," W3C Recommendation, June 2008.

[21] T. Imamura, and Al., "XML Encryption Syntax and Processing," W3C Recommendation, December 2002.

[22] S. Kent, "RFC: IP Authentication Header," Request For Comments 4302, December 2005.

[23] S. Kent, "RFC: IP Encapsulating Security Payload," Request For Comments 4303, December 2005.

[24] C. Kaufman, "RFC: Internet Key Exchange IKEv2 Protocol," Request For Comments 4306, December 2005.

[25] H. S. Thompson, and Al., "XML Schema Part 1: Structures Second Edition," W3C Recommendation, October 2004.

[26] P. V. Biron, and Al, "XML Schema Part 2: Datatypes second Edition," W3C Recommendation, October 2004.