DESIGN AND IMPLEMENTATION OF A SECURE MOBILE WIKI SYSTEM

Costantinos Kolias, Stefanos Demertzis, Georgios Kambourakis

Laboratory of Information and Communication Systems Security
Department of Information and Communication Systems Engineering
University of the Aegean, Karlovassi, GR-83200 Samos, Greece

Corresponding author email: gkamb@aegean.gr

ABSTRACT
During the last few years wikis have emerged as one of the most popular tool shells. Wikipedia has boosted their popularity, but they also keep a significant share in e-learning, intranet-based applications such as defect tracking, requirements management, test-case management, and project portals. However, existing wiki systems cannot fully support mobile clients due to several incompatibilities that exist. On the top of that, an effective secure mobile wiki system must be lightweight enough to support low-end mobile devices having several limitations. In this paper we analyze the requirements for a novel multi-platform secure wiki implementation. XML Encryption and Signature specifications are employed to realize end-to-end confidentiality and integrity services. Our scheme can be applied selectively and only to sensitive wiki content, thus diminishing by far computational resources needed at both ends; the server and the client. To address authentication of wiki clients a simple one-way authentication and session key agreement protocol is also introduced. The proposed solution can be easily applied to both centralized and forthcoming P2P wiki implementations.

KEYWORDS
Wiki; Security; XML; Mobile Systems; SSL.

1. Introduction

Wikis, also known with the term *wiki-wiki*, are nothing more than a special category of web sites that not only allow the editing of their content but also in many cases encourage it. The basic principles that wikis follow are: (a) facilitation for someone to access knowledge, (b) facilitation for someone to contribute to that knowledge. Due to these features wikis have been established as one of the most prominent collaboration platforms over the last years, especially among research groups. Indeed, in most of the existing wiki implementations access is as easy as browsing on a simple web page. Editing is also straightforward and it can be done by inserting information written in a specific (usually very simple) syntax, in the appropriate web forms that the wiki interface offers.

However, the excessive freedom that wikis support is not always desirable. In fact, the more users interact with a wiki the greater the chance that someone inserts wrong information unintentionally or insert irrelevant or offensive content, just to sabotage it. Moreover, in many cases, due to the very nature of the information that the wiki maintains (e.g. a new corporate strategic plan) it is desirable that only certain users have full access and permission to alter all of its contents, while all the rest have less privileges. Thus, the majority of existing wikis can be configured and utilized either as public or as private, i.e. corporate/enterprise, education, intranet etc [1].

On the other hand, in spite the excessive progress that have been made in mobile technology, there exits a great pluralism of standards supported by different device vendors. This fact really acts as a drawback, when comes to application portability. In most cases, a web application, say a wiki, that has been designed to run on a desktop environment will, in the best case scenario, face several problems when someone tries to access it from a mobile device. Also, until now, some efforts have been made to develop wiki systems that provide increased security in terms of access policy. Once again, the common ground of all these attempts was that they relied on technologies and protocols designed specifically for desktop systems, ignoring the idiosyncrasies of low-end mobile devices, i.e. limited processing and memory resources, battery reserves, unreliability of wireless connections etc.

In this paper we present and analyse a novel wiki engine that is appropriate for both desktop systems and mobile devices/clients. Additionally, our wiki system can be optionally configured to be secure, supporting confidentiality and integrity of wiki data in transit, and accessible only by its legitimate users. As far the portability characteristic is concerned, we have tried so the only true requirements from the mobile device to run the wiki will be to have some sort of access on the network, and a front-end application developed for device's native operating system. The application will act as the user interface for the data received from the backend wiki web application. Secondly, our goal is to minimize cryptographic operations and the associated protocol demands but without sacrificing security much at the same time.

The rest of this paper is organized as follows: next section defines and analyses the problem in further detail. Section 3 addresses previous work on the topic and examines wiki architectural issues. Our solution is presented and discussed in detail in section 4. Last section draws a conclusion and gives some pointers to future work.

2. Problem Statement

Today, mobile devices such as Personal Digital Assistance (PDA) or cell phones are widely proliferous. Their capabilities have been dramatically upgraded, i.e. they have greater computational power and memory, bigger screen with more colors, and most of them have access to the Internet. Of course, their employment for quick communication and collaboration is almost synonymous to their name. As already mentioned, the main target of a wiki should be the facilitation of the user for accessing and altering information preferably at any time and place. Thus, whatever its usage, the possibility of accessing a wiki through mobile devices is considered nowadays more than ever necessary. Nevertheless, there are several obstacles that must be carefully considered.

The first problem is actually the number of different standards and technologies supported by today's mobile devices. It seems that mobile device vendors cannot easily agree on certain standards and each one has tried to develop their own. This situation has led to serious compatibility issues among mobile devices of different vendors. Secondly, the vast majority of wikis today have been designed for access from desktop systems. They usually employ standards such as the Hyper Text Markup Language (HTML) and technologies such as JavaScript for enabling the users to access and alter their pages. While these technologies and standards have proven their value on desktop systems, yet the majority of mobile devices do not fully support them. For instance, certain HTML content will not be displayed properly on mobile devices designed to support Compact HTML (C-HTML) [2] or i-mode HTML, and probably will not be displayed at all in mobile devices that incorporate browsers based on Wireless Markup Language (WML) [3]. Even worse, most wikis have evolved from simple script based implementations written in Perl, thus making them difficult to extend. Thus, one goal is the implementation of a truly portable wiki.

On the other hand, although the true spirit of the "wiki way" is to allow for the online collaboration of documents for visitors or contributors to be able to create their own pages or to edit existing pages, security is often essential for many wiki implementation scenarios. Actually, as already mentioned the majority of existing wikis can be configured and utilized either as public or as private [1]. For example, consider a software organization that is currently working on a new project. A good collaboration practice would be the utilization of a wiki among the developers so that they add / amend documentation or general information for the project on each step of the development process. But should all contributors have the same amount of privileges? Probably the employees that are not immediately involved with the development process should not have the right to alter or add content. Another example is the one of a university wiki. Consider the hypothetical scenario where a faculty member wants to be able to provide general information for her lesson and students or others would be welcome to contribute to that knowledge. But what if the instructor would also want to provide the semester grades for this lesson through the wiki? Obviously, students must not have the right to alter them. Moreover, how the instructor must be rest assure that her grades were not altered when in transit from her desktop (or mobile device) to the wiki server. It is thus obvious that the need for a wiki engine that suffices the basic security principles is of great importance. For these cases, a mechanism for authentication and authorization is mandatory; also confidentiality and integrity for the transferred data might be desirable.

Until now, very few works try to intertwine wiki webs with security [4],[5],[6]. Moreover, to the best of our knowledge, none of them explicitly focus on mobile wikis, that is, wikis that can be accessed from low-end mobile devices. So, the following question arises: Is it possible to implement security on a wiki that among others will be appropriate for mobile devices with very limited resources? It is obvious that if security is desirable then it is certain that we will have to execute some sort of functions (e.g. data encryption) which will be very demanding in computational power affecting also battery reserves. Thus, a second goal is the implementation of a wiki that will provide security in the environment of mobile devices with as fewer requirements in computational power as possible.

3. General Issues

Currently many implementations of wiki engines [1] include a set of static web pages placed on a folder of a web server. The client receives a specific page upon request. Any attempt for securing such a system is basically rely on: (a) user's authentication process, (b) the use of ACLs supported by the operating system running on the web server for authorization, and (c) Secure Socket Layer (SSL) protocol for providing confidentiality.

Nevertheless, there are serious doubts concerning the efficiency of such an ACL approach in large wikis where hundreds or even thousands of users access the wiki simultaneously. According to this scenario, when a client queries for a topic each static HTML page would have to be opened and searched for that specific term. Probably, this approach might still be useful only in environments with very few clients and very limited wiki topics.

Today, the majority of wiki engines are based on a centralized, multi-tier architecture which typically includes presentation, application, and data tiers. For example, wikipedia has a set of PHP scripts which reside in the logic tier that access a MySQL database (the data tier). After that the data may be delivered to the client for presentation. In wikipedia for example, editing a page does not require registration, but the IP address of every unregistered user is recorded. In case that an unregistered user is deliberately trying to undermine the content of a page then his IP is inserted into a black list. However, an adversary could easily change his IP and continue causing damage. Usually implementations that follow this particular model employ the SSL protocol in order to provide a higher level of security.

The drawbacks of such an approach are many. First of all, the data transferred between the client and the server is in HTML format. As known, an HTML file consists of two parts; data and markup information. The latter determines the exact way the actual data must be displayed in the client's browser. Whilst HTML is a de-facto standard for desktop machine browsers the overwhelming majority of mobile devices today support only a subset of it or other standards solely. As a result, some of the information (markup) being transferred to the mobile device is useless, while at the same time the data is not displayed properly. Secondly, utilizing SSL means that all HTML files being transferred are scrambled without exceptions. In practice however, only some sensitive or private portions of the actual data must be cryptographically protected; not all. This unfortunately means that the mobile device side, where processing power matters, must decrypt / encrypt more information (markup, non sensitive data) than it actually needs to.

An additional implementation that appeared recently, wants wikis to be based on a decentralized architecture [7],[8]. Peer-to-Peer (P2P) approach attempts to deal with the problem of huge amounts of information stored and administered in a single database. This means increased cost for physical means, e.g. backup, decreased performance etc. On the contrary, the P2P approach is based on hosting only part of data on clients and not on a centralized database. The amount of data that a client can host depends on its power (many data on server, less data on desktops etc). Thus, the more nodes that are connected, the more the total capacity of the system becomes. Although still immature, this model seems to be based on solid theoretical foundations and justify its *raison d'etre*. Despite the fact that a secure P2P implementation does not yet exist, it is obvious that the use of SSL protocol could not be the best approach. SSL provides point-to-point security; not end-to-end. Data are encrypted at one end and it is securely transferred at the other end, where it is decrypted. Information in this way remain secure during transfer but in no way within clients. Thus, if one wants to transfer some data securely, from A to C through B then she should accept that B would have access to those data.

4. The Proposed Solution

The proposed model is mainly based on the centralized three-tier architecture that was described earlier and has been adopted by most wiki implementations nowadays. The changes we propose focus on: (a) *portability*, thus making it appropriate for devices with limited resources, (b) *higher level of security*, thus making it appropriate for closed or confined environments. Moreover, as explained further down, the theoretical model in which our solution is based on fits well to P2P wiki architectures too. We relied on existing technologies and standards for our implementation. To achieve the portability goal we relied on XML language. One the other hand, to satisfy security needs we used XML security, i.e. XML Encryption and XML Signatures as well as on a custom authentication and key establishment protocol described in the following.

4.1. Overview and technologies employed

XML is a markup meta-language, i.e. a language that is used for the creation of other languages. Languages that are based in XML do not carry information related to the presentation of data, as in the case of HTML, but only for the data itself. This feature makes XML an ideal choice for our case since no unnecessary presentation information, part of which might not be needed at all in various platforms, is transferred. More specifically, it is adequate to define a very simple XML-based language which is able to describe wiki data. Naturally, the exact way the wiki data is organized on databases is specific to each implementation per se and its special needs. However, the language should have the following characteristics: (a) be simple enough for someone to recognise the representational structure of the data to be transferred, (b) include no unnecessary tags, (c) provide information about which elements have been chosen for encryption or signature and (d) fully conform with the XML Security and XML Signature specifications. A generic example of such a language is illustrated in figure 1.

```
<Message>
  <Title>
        Xml Encyrption
  </Title>
  <Description ToEncrypt='true'>
        Xml Encryption is a specification that defines...
  </Description>
  <Title>
        Xml Signature
  </Title>
  <Description ToSign='true'>
        Xml Signature is a specification that defines...
  </Description>
```

Figure 1. An example of a wiki XML message

Upon receiving a wiki request the server will query for the matching data stored in the database. After that, it will transform the data back to XML form (which the client can also understand) and forward it to the client. The exact way the data should be presented on the client's screen is up to the client itself. On the other hand, the client must also understand this custom tailored XML-based language and fabricate its queries accordingly, i.e. building the appropriate XML file before transmitting it to the server.

As far as the confidentiality and integrity of the data in transit are concerned, we utilize the XML Encryption [9] and XML Signature specifications [10]. XML Encryption is a specification recommended by W3C that defines a process for encrypting data and representing the result in XML. In XML Encryption the data that may be encrypted can be a XML Document, a XML Element or the contents of a XML Element [9]. This characteristic proves to be a great advantage, since it provides the possibility of encrypting part or the whole of the data to be transferred. This advantage is much appreciated especially on a secure wiki environment where most of the time there is need for securing only a small chunk of the information to be transmitted. Thus, no unnecessary data go through the resource consuming process of encryption.

XML-signature is a specification recommended by W3C that defines processing rules and syntax for digitally signing an XML Document, an element, or just the contents of an element. XML Signature has many similarities with PKCS #7 [11] but is far more extensible for signing XML documents. It is also used by various Web technologies such as Simple Object Access Protocol (SOAP) [12] and Security Assertion Markup Language (SAML) [13].

Since the SSL/TLS protocol [14] is currently the first choice for secure data transport in web applications, a comparison between SSL and XML is necessary. In contrast to XML Encryption, the following are two important areas not addressed by SSL: (a) encrypting part of the data being exchanged, (b) secure sessions between more than two parties. As already said, if the application requires that the whole communication be secure, then SSL is the proper choice. On the other hand, XML Encryption is an excellent choice if the application requires a combination of secure and insecure communication; which means that some of the data will be securely exchanged and the rest will be exchanged as plaintext. This feature fits best in our case ensuring maximum portability and improved performance for low-end mobile devices. Also, the same feature makes XML Encryption appropriate for future secure P2P wiki implementations as well.

4.2. Architecture

Our implementation follows the centralized multi-tier model described earlier in section 3 (see also figure 2). More specifically it comprises from the data tier, the logic tier and the presentation tier. The data tier hosts a database in which all information about the topics of the wiki is stored, like titles and contents of topics, pictures etc. The retrieval of data is done with simple SQL select commands, while topics are updated by SQL update and insert commands. Supposing that the database is secured (from outsiders), data can be stored in it in unencrypted form. Upon registration to the wiki system a new user is created to the database using the credentials {username, password} of his choice. Particular rights are assigned to her by utilizing a role-based system. There might be roles that allow a user to execute select queries on the database, roles that allow select and update, and roles that provide full privileges.

In the logic tier lies our application that is in charge of the following operations: (a) retrieval of data from the database after a corresponding client's request, (b) transformation of the data into XML files (c) forwarding the produced files (containing the data) to the client (d) parsing the data contained in the files and (e) storing data to the database. Where necessary, the application performs encryption of data. The application retrieves from the database the sensitive data (e.g. a paragraph of a topic that its content has to remain confidential), it symmetrically encrypts it and finally places it on the corresponding XML file appropriately.

In the client side resides a wiki application too. The client application receives XML files from the server and parses them to extract the data from the rest of the XML tags. In case the XML file contains some cryptographically protected sensitive fields then the application deciphers them by utilizing the session key (see next section). After that it displays the data on appropriate controls (labels, text fields etc) of the application. The procedure of decryption follows the reverse path. Ciphering/Deciphering procedure on the client is based on a symmetric session key and it is flexible by the means that it can be optionally applied only to some certain sensitive fields.

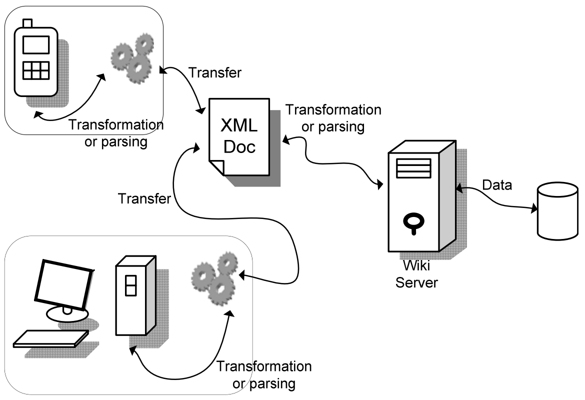
Transfer
Transformation or parsing
XML Doc
Transformation or parsing
Data
Transfer
Wiki Server
Transformation or parsing

Figure 2. The proposed architecture

4.3. Authentication and Key Agreement Protocol

In this section we describe a simple lightweight Authentication and Key Agreement protocol which enables a user to securely authenticate himself to the wiki server. The protocol also produces a 256 bits length key to serve as the session key. Our protocol utilizes both symmetric and asymmetric cryptographic operations. The well-known Advanced Encryption Standard (AES) with a key length of 256 bits is used for symmetric ciphering / deciphering, while the Rivest Shamir Adleman (RSA) algorithm with a key length of 1024 bits for public key operations. It is also assumed that all clients hold a copy of the server's public key in the form of a base-64 encoded X.509 certificate issued by a Certification Authority (CA). Note that a base-64 encoded certificate is very easy to manage and transfer to virtually every mobile device as it is in plain text. The way the client receives the certificate is out of the scope of this paper, thought it can be installed manually on the client application or received via e-mail.

The authentication protocol is one-way. This means that it is used to authenticate the client to the server but not the opposite. Actually, mutual authentication is not necessary here since fake or rogue wiki servers cannot harm clients by any means, except causing Denial of Service (DoS). That is, the client repeatedly attempts to authenticate to a fake server unsuccessfully. The server is not in a position to eavesdrop on any valuable information transmitted, so this attack is considered harmless. Upon authentication the following steps are performed: (a) The client produces a random 256 bits session key, (b) the client encrypts the session key with the server's public key retrieved from the server's certificate, (c) Upon receiving the encrypted session key the sever decrypts it using its own private key, (e) the client encrypts user's credentials, i.e. {username, password} with the session key and sends them towards the server, (f) as soon as the encrypted credentials of the user are received the server decrypts them with the session key.

Up to this point the protocol is very simple and lightweight but has the disadvantage that is extremely weak in man in the middle / replay attacks. More specifically, assuming that the attacker eavesdrops on the link between the server and the client he can record all the messages transmitted. After that, at a later time, he can replay them towards the server and become successfully authenticated Thus, some more steps must be executed in order to guarantee that the client attempting to authenticate is the legitimate one, i.e. does have the proper session key: (h) the server generates a random number and sends it to client, (i) the client is required to encrypt it with the session key (j) server receives the encrypted session key and decrypts it with the session key (k) if the result matches with the original random number the client is considered authenticated. Figure 3 depicts the overall protocol messages flow.

4.4. Confidentiality

Confidentiality of the data is achieved through a simple procedure of symmetric encryption. The mechanism assumes that both the client and the server hold the session key, which in turn means that the procedure of authentication and key establishment has completed successfully. The one end encrypts the data and inserts them into the XML file following the syntax described by the XML encryption standard. After that the XML file is transmitted towards the other end which follows the opposite procedure to acquire the original content.

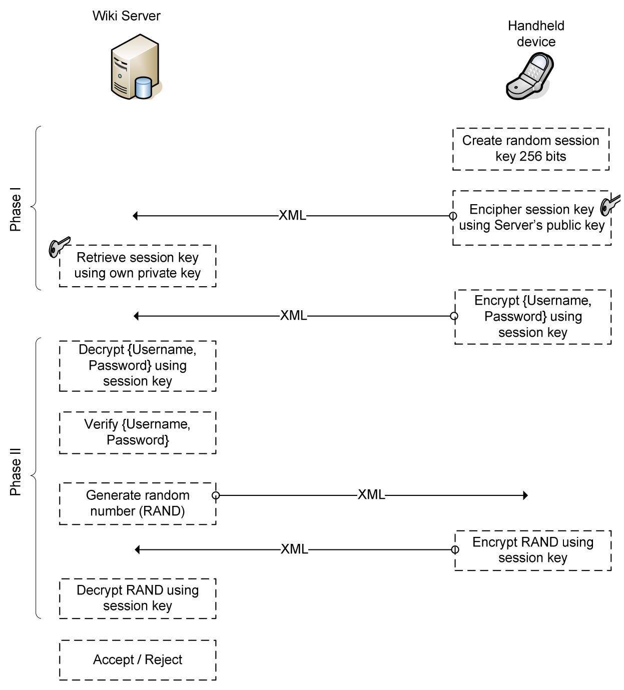
Wiki Server
Handheld device
Phase I
Create random session key 256 bits
XML
Encipher session key using Server's public key
Retrieve session key using own private key
XML
Encrypt {Username, Password} using session key
Phase II
Decrypt {Username, Password} using session key
Verify {Username, Password}
Generate random number (RAND)
XML
XML
Encrypt RAND using session key
Decrypt RAND using session key
Accept / Reject

Figure 3. The authentication and key agreement protocol

4.5. Integrity – Non repudiation

Our system offers the possibility for signing data (XML files, elements, contents of elements) for ensuring the integrity of the data. It can also provide non repudiation services to the user, if desirable. In the following, when referring to hash calculation the use of SHA-1 [15] is implied. If the client wishes to sign a XML document then the procedure is as follows: (a) a hash of the document to be sent is calculated, (b) the client encrypts the hash with the session key established during authentication, (c) the client sends the encrypted hash within the XML file as a special tag according to the XML Signature specifications, (d) The other part verifies the acquired signature by recalculating the hash of the file and comparing it with the received one after decryption.

Contrariwise to all know implementations where the hash gets encrypted with the private key of the entity who wishes to sign, here there is no reason to subject the client to the process of acquiring its own public-private key pairs as well as the demanding process of asymmetric encryption. In our case there is only one entity (the server) that will ever wish to confirm the signature. The server has already acquired the (symmetric) session key - through the previously discussed process of authentication - and since both parties share a unique secret key, they can use it to sign the message (client-side) and then verify it (server-side). The overall integrity procedure is illustrated hereunder in figure 4.

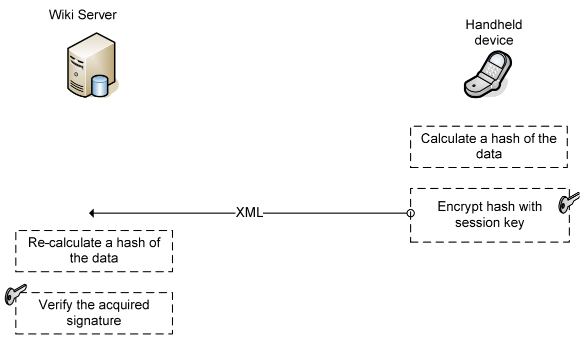
Wiki Server
Handheld device
Calculate a hash of the data
XML
Encrypt hash with session key
Re-calculate a hash of the data
Verify the acquired signature

Figure 4. The integrity service

4.6. Experimental Results

Preliminary measurements showed that the wiki authentication and key agreement protocol service time span between 0.9 and 1.25 seconds. Also, measurements were gathered during repeated wiki-topic secure transactions, between the wiki server and a PDA client. Each wiki topic was encrypted, decrypted and signed fifteen times and only average values were recorded. Specifically, encryption, decryption and signature mean times for a 0.5 Mbytes file were 2, 2.32 and 0.44 seconds correspondingly. Tests were conducted using as client device a Fujitsu – Siemens Loox N560 Pocket PC, which incorporates a 624 MHz Intel X-scale PXA270 processor, and an IEEE 802.11b/g wireless connection. The operating system running on the device was Windows Mobile in version 5.0. A snapshot of the wiki client application interface is depicted in figure 5. As already mentioned and it is demonstrated in figure 5, confidentiality and integrity services can be applied selectively; either to the wiki topic as a whole or to some sensitive parts of it.

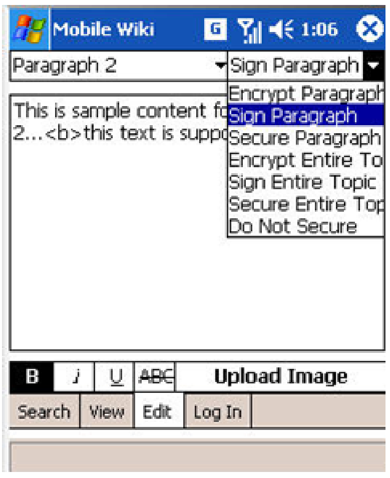
This is sample content fo
2...<b>this text is suppo

Figure 5. A snapshot of the client wiki application ("Secure" means encrypt + sign)

5. Conclusions

In our opinion the ideal wiki should be accessible to an anywhere, anytime basis and optionally secure to its legitimate users. The former requirement makes wiki accessible from virtually any mobile device affords a web browser, while the latter ensures that when optionally extra security is needed the wiki will be able to support it. In this paper we investigated both aforesaid requirements discussing ways that can be realizable. Furthermore, a novel wiki multiplatform prototype implementation was presented and its major components were analyzed and shortly tested. Since Ajax technology utilizes mainly XML (among other choices) for sending and receiving data it would be desirable to embed Ajax in our implementation sometime in the future. Taking into account that Ajax runs from within web browsers (javascript) this would actually eliminate the need for separate client applications to be developed and installed on wiki target devices.

References

[1] Wikipedia, *Comparison of wiki software*, http://en.wikipedia.org/wiki/Comparison_of_wiki_software
[2] W3C, *Compact HTML for Small Information Appliances*, available at: http://www.w3.org/TR/1998/NOTE-compactHTML-19980209/
[3] OMA, *Wireless Markup Language version 2 Specification*, available at: www.openmobilealliance.org/tech/affiliates/wap/wap-238-wml-20010911-a.pdf.
[4] Mason, R. and Roe, P., "RikWik: An Extensible XML Based Wiki", *proceedings of Symposium on Collaborative Technologies and Systems*, pp. 267-273, 2005.
[5] Dondio, P., Barrett, S., Weber, S., Seigneur, J. M., "Extracting trust from domain analysis: A case study on the wikipedia project", *LNCS 4158*, pp. 362-373, 2006.
[6] Raitman, R., Ngo, L., Augar, N., Zhou, W., "Security in the online E-learning Environment", proceedings of the *5th IEEE ICALT*, pp. 702-706, 2005.
[7] Morris, J., Lüer, C., "DistriWiki: A Distributed Peer-to-Peer Wiki", 2007, submitted for publication, http://www.cs.bsu. edu/homepages/chl/P2PWiki/.
[8] Urdaneta, G., Pierre, G. and Van Steen, M., "A Decentralized Wiki Engine for Collaborative Wikipedia Hosting", *In Proceedings of the WEBIST* 2007.
[9] W3C, *XML Encryption Syntax and Processing*, available at: http://www.w3.org/TR/xmlenc-core/.
[10] W3C, *XML-Signature Syntax and Processing*, available at: http://www.w3.org/TR/xmldsig-core/.
[11] Kaliski, B., *PKCS #7: Cryptographic Message Syntax*, Version 1.5, RFC 2315, RSA Laboratories, March 1998.
[12] W3C, *SOAP Version 1.2 Part 0: Primer (Second Edition)*, W3C Recommendation 27 April 2007.
[13] Ragouzis N. et al., "Security Assertion Markup Language (SAML) V2.0 Technical Overview, Feb. 2007, available at: http://www.oasisopen.org/committees/download.php/22553/sstc-saml-tech-overview-2%200-draft-13.pdf.
[14] Frier A., Karlton P. & Kocher P., *The SSL 3.0 Protocol*, http://home.netscape.com/eng/ ssl3/draft302.txt.
[15] W3C, *SHA1 Secure Hash Algorithm - Version 1.0*, http://www.w3.org/PICS/DSig/SHA1_1_0.html.