A Security-aware Approach to JXTA-Overlay Primitives

Joan Arnedo-Moreno
Estudis d'Informàtica, Multimèdia i Telecomunicació
Universitat Oberta de Catalunya (UOC)
Rambla de Poblenou, 156 08018 Barcelona, Spain
Email: jarnedo@uoc.edu

Keita Matsuo
Graduate School of Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-Higashi, Higashi-Ku, 811-0295 Fukuoka, Japan
Email: bd07002@bene.fit.ac.jp

Leonard Barolli
Department of Information and Communication Engineering
Fukuoka Institute of Technology (FIT)
3-30-1 Wajiro-Higashi, Higashi-Ku, 811-0295 Fukuoka, Japan
Email: barolli@bene.fit.ac.jp

Fatos Xhafa
Department of Languages and Informatics Systems
Technical University of Catalonia (UPC)
Jordi Girona 1-3, 08034 Barcelona, Spain
Email: fatos@lsi.upc.edu

Abstract

The JXTA-Overlay project is an effort to use JXTA technology to provide a generic set of functionalities that can be used by developers to deploy P2P applications. Since its design mainly focuses on issues such as scalability or overall performance, it does not take security into account. However, as P2P applications have evolved to fulfill more complex scenarios, security has become a very important aspect to take into account when evaluating a P2P framework. This work proposes a security extension specifically suited to JXTA-Overlay's idiosyncrasies, providing an acceptable solution to some of its current shortcomings. [1]

Keywords: peer-to-peer, security, XMLdsig, JXTA, JXTA-Overlay.

1. Introduction

Peer-to-peer (P2P) applications have become highly popular in recent times due to its great potential to scale and the lack of a central point of failure. Slowly, they have evolved from simple file-sharing, such as Gnutella [1], to new environments such as e-health or e-learning [2]. As a result, the maturity of research in the field of P2P has pushed through new problems such us those related with security, becoming one of the key issues when evaluating such systems. Even at the cost of some impact on performance, a security baseline must be kept in any P2P system in order to protect it against different network vulnerabilities.

JXTA [3] (or "juxtapose") is a set of open protocols that enable the creation and deployment of P2P networks. JXTA protocols enable P2P applications to discover and observe peers, enable communication between them or offer and localize resources within the network. JXTA-Overlay [4] extends such protocols in a framework which increases the reliability of JXTA-based applications and supporting group management and file sharing. Unfortunately, the design focus on JXTA-Overlay was completely concerned with system performance, but not at all with security, a situation which may become a great constraint under today's standards.

The contribution of this paper is a security extension to JXTA-Overlay. The proposed extension fully realizes the messaging capabilities and functions of both JXTA and JXTA-Overlay and uses them in order to provide a security baseline in a transparent manner. As a result, minimum effort is necessary by application developers and end-users to deploy a secure environment. Furthermore, because of the framework's modular approach, it may be easily ported to different scenarios, according to the final application's needs.

This paper is organized as follows. Section 2 provides a general overview of JXTA and JXTA-Overlay's architecture and capabilities, as well as exposing some of the most obvious vulnerabilities in JXTA-Overlay. Section 3 presents the current related work on securing JXTA-based systems. The proposal of a basic security extension is presented in section 4. Section 5 provides a brief study on the cost of implementing security. Concluding the paper, section 6 summarizes the paper contributions and further work.

2. JXTA-Overlay Overview

JXTA-Overlay is a middleware built on top of the JXTA protocol specification. JXTA-Overlay extends JXTA protocols with the goal of overcoming some of its limitations: the need for the developer to manage the presence mechanism, peer group publication and message exchange. To achieve this end, JXTA-Overlay provides a set of basic functionalities, named *primitives* and *functions*, intended to be as

1. This work is partially supported by the Spanish Ministry of Science and Innovation and the FEDER funds under the grants TSI2007-65406-C03-03 E-AEGIS and CONSOLIDER CSD2007-00004 ARES.

complete as possible to satisfy the needs of most JXTA-based applications.

2.1. The JXTA-Overlay network

In a JXTA-Overlay network, different entities interact. *End-users* connect to the network using a *client peer*, a P2P application deployed into the network using JXTA-Overlay's libraries. End-users join the network by authenticating using a username and password. Once successfully authenticated, they are organized into different overlapping groups, so only members of the same group may interact. *Brokers* are special peers which control access to the network, taking care of end-user authentication, as well as helping client peers interact between them by propagating their related information. Brokers are very important since they exchange information about all client peers, maintaining a global index of available resources, thus allowing all peers to find network services. Brokers also act as beacons which client peers which have recently gone online use to join the network. For that reason, they usually have well-known identifiers, such as a DNS name or a static IP address.

All the information related to user configuration (username, password and group membership) is stored in a special single entity within the JXTA-Overlay network: a *central database*. Only brokers may access the database contents, in order to check end-user authentication attempts and organize them into groups. It is assumed that some *administrator* takes care of properly configuring the database, registering new end-users.

2.2. General architecture

The architecture of the JXTA-Overlay middleware defines three modules, which let the different entities described in section 2.1 communicate: the Client Module, the Broker Module and the Control Module. Altogether, they form an abstraction layer on top of JXTA, which may be used to develop any kind of client peer, as shown in Figure 1.

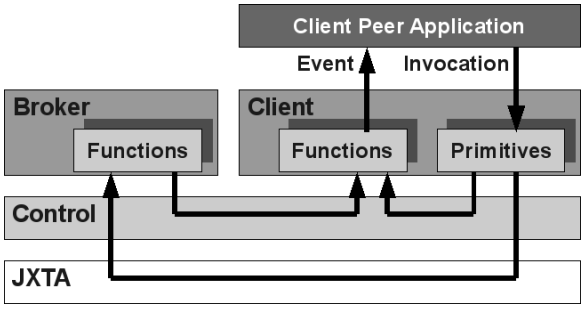

Figure 1. JXTA-Overlay architecture.

- The *Client Module* defines all necessary primitives for client peer to join a JXTA-Overlay network and interact with other peers and the broker. Applications developed on top of JXTA-Overlay are always based on the invocation of Client Module primitives and the processing of events thrown by functions, executed as a result of message reception from other peers (client peers or brokers).
- The *Broker Module* defines all the functions that client peers may call upon a broker in order to be granted access to the the network, create and publish groups or retrieve other client peers' information. Broker functions are always executed as a result of messages sent via Client Module primitives.
- The *Control Module* acts as an intermediate layer between the Broker and Client Modules, providing the generic functionalities on regards to group management and messaging.

The Control Module provides direct messaging between JXTA-Overlay entities using JXTA *pipes*, a virtual communication channel between peers. Client peers have an input pipe for each group it belongs to and other group members may send messages using the input pipe associated to that group. Brokers have a single input pipe which is shared for all incoming messaging.

Peer information is propagated across group members by brokers, which are able to distribute data beyond boundaries such as broadcast range or NAT. such information is formatted as JXTA *advertisements*, metadata documents codified using XML and transmitted using the JXTA core protocols. JXTA-Overlay has a big reliance on both JXTA-defined and custom-defined advertisements, and therefore, their data is critical for the correct operation of the network. Each client peer periodically broadcasts a set of different advertisements for each group it belongs to, each one containing data related to a specific purpose, such as available files, input pipe location, statistics or presence notifications.

2.3. JXTA-Overlay and security

As previously exposed, JXTA-Overlay's design is not concerned with security, with the only exception of end-user network access control via a username and password. As a result, its is vulnerable to different security threats which may jeopardize the network. Some of the greatest security concerns in JXTA-Overlay are the following ones:

- Transmitted data may be easily eavesdropped, since no data privacy is provided. In the case of the transmitted username and password for end-user authentication, it cannot even be argued that it is an optional feature. It should be mandatory.
- Any legitimate user may forge advertisements with no fear of reprisal. No integrity or source authenticity is maintained. Such advertisements will be distributed and

accepted by all group members, unaware of the false data.
- Client peers never check the broker legitimacy before authenticating. There is no guarantee that a brokert is a legitimate one even in the case of well-known identifiers, since it may be that traffic is being redirected to a fake one via methods such as DNS spoofing [5].

As it can be seen, some of the current JXTA-Overlay vulnerabilities are quite obvious ones, such as transmitting sensitive data with no real privacy. Therefore, it can be concluded that JXTA-Overlay does not provide an acceptable security baseline and needs serious improvement on this regard.

3. Related Work on Securing JXTA-based Frameworks

Before a security framework for JXTA-Overlay may be proposed, it is useful to review which are the current security mechanisms available to JXTA-based applications.

As far as network access control is concerned, a specific trust model is proposed in [6]. Rendezvous peers provide credentials to peers, that can be used to prove network membership. Such credentials are issued only to those peers which are authorized via an LDAP (Lightweight Directory Access Protocol) directory [7]. Rendezvous peers use a secure connection to the LDAP service check peer authorization. This proposal is extended in [8], moving to a centralized Public Key Infrastructure (PKI) approach and providing a protocol to authenticate the group itself. More elaborated proposals are presented in [9], [10], based on joint authorization by multiple peers under voting schemes in order to maximize decentralization. Credentials are issued by a CA (Certification Authority), however access is based on an agreement reached between several group members. The main difference between both proposals is that [10] includes a rank system, where peers may only rise to higher positions by contributing to the group.

On regards to message security, JXTA has two available mechanisms: TLS (Transport Layer Security) [11] and CBJX (Crypto-Based JXTA Transfer) [12]. On one hand, JXTA provides its own definition of standard TLS as a transport protocol, providing private, mutually authenticated, reliable streaming communications using symmetric cryptography for data encryption and a keyed Message Authentication Code (MAC) [13] for message integrity. On the other hand, CBJX is a JXTA-specific security layer which pre-processes messages to provide an additional secure encapsulation, creating a new message that is then relayed to an underlying transport protocol. The original message's is signed, and an additional information block, is also added to the secured message.

JXTA also provides some degree of advertisement security with signed advertisements. A signed advertisement encapsulates the original XML advertisement as plain text encoded using the Base64 algorithm [14]. The original advertisement type cannot be recognized without processing the signature. An alternative method is proposed in [15], based on the XMLdsig [16] standard. In contrast, in this proposal, the resulting secure advertisement maintains its original type.

Unfortunately, in order to use all the secure mechanisms provided by JXTA, it is mandatory to use a specific implementation of the JXTA Memberhip Service, the Personal Secure Environment (PSE). The membership service one of JXTA's core services which takes care of group membership and identity management. Being limited to PSE is a great constraint, since the choice of membership service has strong implications on a system's architecture, and thus it is difficult to apply on an existing framework such as JXTA-Overlay. Furthermore, PSE has a limited range of cryptographic module support, solely supporting Java keystore files [17] and X509 certificates [18] as credentials.

4. An Approach for JXTA-Overlay Security

In this section we present an extension to the original JXTA-Overlay primitives, so end user data is protected against the vulnerabilities exposed in section 2.3. In this proposal, we combine several methods of those previously described in section 3, adapting them to JXTA-Overlay's specific architecture and network setup. Client peers are protected against impersonation by using broker-issued credentials in a similar way to the approach in [6]. However, we further extend this approach to the broker, so a legitimate one may be told apart from malicious ones. Advertisement integrity and authenticity, as well as a transparent method for authentic key transport is provided by adapting the method defined in [15]. Finally, data privacy is used to protect message exchanges against eavesdroppers.

The following notation will be used from now on to describe the secure framework:

- SK_i: Peer i's secret key.
- PK_i: Peer i's public key.
- $Cred_i^j$: Peer i's credential, issued by j. It is assumed that the credential contains P_i's public key.
- $E_{PK_i}(x)$: A string x encrypted using the public key of peer i by means of a wrapped key encryption scheme (such as the one defined in [19]).
- $S_{SK_i}(x)$: A string x signed using the private key of peer i.
- $i \longrightarrow j : \{m_1, \cdots, m_n\}$: A message sent from peer i to peer j, with the content $m_1, \cdots, m_n$.

4.1. System setup

An initial network setup is assumed at deployment and peer boot time. During this setup, JXTA-Overlay entities are

provided with the necessary cryptographic data to properly execute the secure extended primitives.

The JXTA-Overlay administrator, Adm, generates a key pair PK_{Adm} and SK_{Adm} and a self-signed credential $Cred_{Adm}^{Adm}$, thus acting as trusted party by all peers. This is a sensible stance, since the system administrator is the entity that grants access to the JXTA-Overlay network by creating legitimate usernames and passwords into the database.

Each broker, Br_i, generates a key pair PK_{Br_i} and SK_{Br_i}. From PK_{Br_i}, the administrator will provide Br_i with a credential $Cred_{Br_i}^{Adm}$. Therefore, only a legitimate broker may be able to prove the ownership of such credential.

Each client peer, Cl_i, is provided with a copy of $Cred_{Adm}^{Adm}$. At boot time, a key pair PK_{Cl_i} and SK_{Cl_i} are created. Credential generation will be performed as part of secure primitive execution, as will be shown in section 4.2.2.

Once each a client peer or a broker has established its credential, it is distributed to other group members using the approach in [16]. This grants an authentic credential distribution mechanism based on Crypto Based IDentifiers (CBIDs) [20], which is invisible to both JXTA-Overlay and JXTA. In addition, pipe advertisement integrity and source authenticity is also provided. However, it must be taken into account that JXTA-Overlay only allows advertisement exchanges once a peer has joined the network. This has some implications on those secure primitives related to network connection, as will be shown in section 4.2.

4.2. Secure broker connection

In order to join a JXTA-Overlay network, a client peer must first locate a broker which will authenticate the end user by checking a username and password. This process is defined into the JXTA-Overlay set of *discovery* primitives. As far as the discovery primitives are concerned, even though they also encompass those related to network information, such as peer status retrieval, the proposed secure extension exclusively on focuses the initial interactions with the broker in order to join the JXTA-Overlay network.

Joining the network via the broker is divided into two distinct parts, each one in the form of a particular primitive:

- *connect*: Locates a broker and waits for a connection to open.
- *login*: Authenticates the current client peer end-user by sending a username and password that will be checked against the system database.

The secure extension maintains the separation in two parts, also relating each one to a particular JXTA-Overlay primitive: the *secureConnection* and *secureLogin* primitives. In contrast with the original discovery primitives, additional steps are taken in each one to ensure a secure message exchange and issue s credential to client peers.

4.2.1. secureConnection. The *secureConnection* primitive uses a challenge-response [21] approach to authenticate the broker and check its legitimacy. The description of the connection process, initiated by the primitive invocation, follows:

1) The client Cl waits for a broker Br to become available and initiates the connection attempt.
2) Cl chooses a byte array, $chall$, as a random challenge.
3) $Cl \longrightarrow Br$: $\{chall\}$
4) Br generates a sufficiently long random session identifier sid, and stores it.
5) $Cl \longleftarrow Br$: $\{sid, S_{SK_{Br}}(chall), Cred_{Br}^{Adm}\}$
6) Cl checks the authenticity of $Cred_{Br}^{Adm}$ by verifying its signature using PK_{Adm} (contained in the administrator's credential, $Cred_{Adm}^{Adm}$).
   - If its is not authentic, it can be concluded that Br is not a legitimate Broker.
7) Cl checks $S_{SK_{Br}}(chall)$ using PK_{Br} (which is contained within $Cred_{Br}^{Adm}$).
   - If signature validation fails, it can be concluded that Br does not possess SK_{Br}, and thus is an impersonator.
8) If both checks succeed, it can be concluded that Br is a legitimate Broker.
9) Cl stores sid and $Cred_{Br}^{Adm}$.

4.2.2. secureLogin. Once the Broker Br's credential has been retrieved, and its authenticity established, it is possible to use the *secureLogin* primitive to actually join the JXTA-Overlay network. Just as it is the case for the original *login* primitive, a username and password are provided by the client application's end user. The session identifier, sid, generated in the *secureConnect* primitive is used at this stage to avoid replay attacks at the authentication attempt. Otherwise, an attacker can reuse any authentication attempt from other client peers to impersonate them. The attacker need not know the content of the encrypted message to perform this kind of attack, it is enough that it contains a valid username and password that will be accepted by the broker. The description of the underlying protocol follows:

1) Cl generates the login request $req = S_{SK_{Cl}}(username, password, PK_{Cl})$
2) Cl retrieves PK_{Br} and sid, obtained during the *secureConnection* primitive call.
3) $Cl \longrightarrow Br$: $\{E_{PK_{Br}}(req, sid)\}$
4) Br decrypts the message using SK_{Br}.
5) Br checks if sid is currently stored. If that is not the case, login is aborted. Otherwise, Br no longer stores sid and the login process continues.

6) Br checks username and password matching according to JXTA-Overlay's standard procedures (for example, a secure backend database connection).
   - If they do not match, it can be assumed that Cl's end user is an impersonator and login is aborted.
7) Br checks key authenticity against the claimed client peer identifier according to the mechanism described in [15].
   - If the check fails, it can be concluded that the request was not received from a client peer with the claimed identifier. The login attempt is aborted by Br.
8) If both checks were correct, Br generates a credential $cr = Cred^{Br}_{Cl}$, containing PK_{Cl} and Cl's current end user's username.
9) $Cl \longleftarrow Br$: $\{cr\}$
10) From now on, Cl's end user may use cr as proof of identity, until cr's expiration date.

4.3. Secure messaging

Messenger primitives define how to directly exchange simple text messages between client peers, such as a chat service, without requiring broker intervention. The two main primitives are:

- *sendMsgPeer*: Sends a simple message to some other client peer.
- *sendMsgPeerGroup*: Sends a simple message to all members of a group. It is actually resolved by iteratively calling *sendMsgPeer*.

The secure versions for both primitives provide lightweight privacy, data integrity and message source authentication in a stateless, best effort manner. This is in contrast, for example, with JXTA's secure pipes, which rely on TLS and require some previous negotiation between endpoints, as explained in section 3.

4.3.1. secureMsgPeer and secureMsgPeerGroup. The necessary steps for some user connected to client peer Cl_1 to send a simple text message to another one connected to Cl_2 are:

1) Cl_1 retrieves Cl_2's pipe advertisement.
2) Cl_1 validates the advertisement signature in order to ensure that it has not been compromised, using the method described in [15].
   - If the signature does not validate, the advertisement has been tampered, and is deemed invalid. If the message is sent, no guarantees exist on regards to its security.
3) Cl_1 retrieves PK_{Cl_2} from the signed advertisement's enclosed credential, $Cred^{Br}_{Cl_2}$.
4) $Cl_1 \longrightarrow Cl_2$: $\{E_{PK_{Cl_2}}(m, S_{SK_{Cl_1}}(m))\}$
5) Cl_2 decrypts the message using SK_{Cl_2}.
6) Cl_2 retrieves Cl_1's pipe advertisement and repeats steps 2, 3.
7) Cl_2 validates the message signature using PK_{Cl_1}, obtained via Cl_1's signed advertisement.

The need to locate a pipe advertisements in steps 1 and 6 puts no burden or constraint on the system, since such advertisements are always necessary in order to exchange messages between peers. In fact, both steps always occur, regardless of which primitive is used (the original or the secure one). Therefore, if the advertisement is not available for some reason, it would be impossible to exchange any kind of message nevertheless. This is an approach to secure messaging which seamlessly integrates with JXTA-Overlay's messenger primitives by making use of the exactly the same core mechanisms, instead of relying on additional protocols for key distribution.

Just as it is the case for the standard primitives, the *secureMsgPeerGroup* primitive just iteratively uses the *secureMsgPeer* to send the same message to a group of peers.

5. Security Cost

Security always comes at a cost in protocol efficiency by adding some overhead. In order to asses the impact on performance, two different sets of scenarios have been taken into account in order to run a set of experiments: time overhead until a client peer joins the JXTA-Overlay network and delay in simple message transmission. Such tests have been run using a PC with a 1.20 GHz Intel Pentium M processor and 1 Gb of RAM under Ubuntu 8.10 and SUN's Java Runtime Environment version 1.6.0.10 (which includes the Java Cryptographic Extension, JCE) . We decided to use a computer which is below today's average standards to assess the impact of using JXTA's security mechanisms on lower end machines.

The resulting overhead for joining the network via the *secureConnection* and *secureLogin* primitives amounts to about 81.76%. The overhead is high from an absolute standpoint, however, we must take into account that it is in comparison to a scenario where no security exists at all.

Overhead in secure messaging has also been tested for different data lengths, as shown in Figure 2. Even though overhead is high for the same reasons as the network join process, it quickly falls as network latency becomes more relevant.

6. Conclusions and Further work

In this paper, a secure extension to JXTA-Overlay primitives has been presented. Borker issued credentials are used to identify users and encapsulate cryptographic data. We also take advantage of an approach based on XMLdsig signed

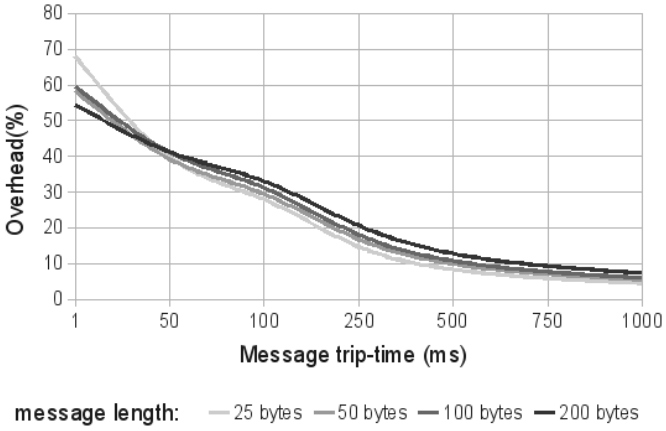

Figure 2. *secureMsgPeer* primitive overhead.

advertisements to distribute credentials to group members in order to ensure secure data exchange. The main contribution of this work is providing effective security to an existing framework which has none at all using an approach which takes into account which kind of entities interact and how the network is set up.

In this proposal, only some of the most basic, but not least important, primitives have been secured. The extension of the JXTA-Overlay framework to secure every single primitive is beyond the space limitations of this work (about 122 primitives and 84 events related to different functions). However, once the building blocks for a secure system have been established, an integral key distribution and authenticity scheme, it is feasible to extend security to every single primitive. Any message exchange can be secured using an approach similar to that defined for messenger primitives.

Currently, the extended primitives have been implemented and integrated into the latest version of JXTA-Overlay. They correctly interact with the key distribution method, providing secure messaging. Further work includes extending security to other JXTA-Overlay primitives which are deemed sensitive to attacks. Of special note are those of the executable set of primitives, related to remote code execution.

References

[1] "Gnutella", 2000, http://rfc-gnutella.sourceforge.net.

[2] K. Matsuo, L. Barolli, F. Xhafa, A. Koyama, and A. Durresi, "Implementation of a jxta-based p2p e-learning system and its performance evaluation", *International Journal of Web Information Systems*, vol. 4, no. 3, pp. 352–371, 2008.

[3] Sun Microsystems Inc., "Jxta v2.0 protocols specification", 2007, https://jxta-spec.dev.java.net/nonav/JXTAProtocols.html.

[4] F. Xhafa, R. Fernandez, T. Daradoumis, L. Barolli, and S. Caballe, "Improvement of jxta protocols for supporting reliable distributed applications in p2p systems", in *International Conference on Network-Based Information Systems (NBiS)*, 2007, pp. 345–354.

[5] Sax D., "Dns spoofing (malicious cache poisoning)", 2003, http://www.sans.org/rr/firewall/DNS_spoof.php.

[6] B. Yeager, "Enterprise strength security on a jxta p2p network", *P2P'03: Proceedings of the 3rd International Conference on Peer-to-Peer Computing*, 2003.

[7] Kille S. Wahl M., Howes T., "Lightweight directory access protocol (v3)", 1997, http://www.ietf.org/rfc/rfc2251.txt.

[8] L. Kawulok, K. Zielinski, and M. Jaeschke, "Trusted group membership service for jxta", in *Computational Science - ICCS 2004*, 2004, Lecture Notes in Computer Science Volume 3038.

[9] Yunhao L. and Jinpeng H. et al, "Access control in peer-to-peer collaborative systems", *First International Workshop on Mobility in Peer-to-Peer Systems (MPPS)*, pp. 835–840, 2005.

[10] Amoretti M., Bisi M., Zanichelli F., and G. Conte, "Introducing secure peer groups in sp2a", 2005, pp. 62–69.

[11] Allen C. Dierks T., "Ietf rfc 2246: The tls protocol version 1.0", 1999, http://www.ietf.org/rfc/rfc2246.txt.

[12] D. Bailly, "Cbjx: Crypto-based jxta (an internship report)", July 2002.

[13] R. Canetti H. Krawczyk, M. Bellare, "Hmac: Keyed-hashing for message authentication", 1997, http://www.ietf.org/rfc/rfc2104.txt.

[14] Ed. S. Josefsson, "Ietf rfc 3548 - the base16, base32, and base64 data encodings", 2003, http://www.ietf.org/rfc/rfc3548.txt.

[15] J. Arnedo-Moreno and J. Herrera-Joancomartí, "Persistent interoperable security for jxta", in *Proceedings of the Second International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC) 2008*. 2008, pp. 354–359, IEEEPress.

[16] W3C, "Xml-signature syntax and processing", 2002.

[17] SUN Microsystems Inc., "Java cryptography architecture (jca)", 2008, http://java.sun.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html.

[18] CCITT, "The directory authentication framework. recommendation", 1988.

[19] B. Kaliski and J. Staddon, "Pkcs1: Rsa cryptography specifications. version 2.0", 1998.

[20] G. Montenegro and C. Castelluccia, "Crypto-based identifiers (cbids): Concepts and applications", *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 97–127, 2004.

[21] W. Simpson, "Ppp challenge handshake authentication protocol (chap)", 1996, http://tools.ietf.org/html/rfc1994.