

Providing Digital Time Stamping Services to Mobile Devices

D. Cotroneo, C. di Flora, A. Mazzeo, L. Romano, S. Russo, G. P. Saggese -
Universita' degli Studi di Napoli Federico II
{cotroneo,diflora,mazzeo,lrom,sterusso,saggese}@unina.it

Keywords: Digital Time-stamping, Ubiquitous Computing, Mobile Systems.

Abstract

Technology evolution in wireless communication is enabling pervasive connectivity to Internet scale systems. In this scenario, security critical applications are being deployed over platforms which include mobile devices. It is thus key that security services be provided to mobile devices as well. Since security functions are typically based on computationally intensive cryptographic algorithms, achieving this goal is none of a simple task, for the following characteristics of mobile devices: 1) limited computing power, and 2) constraints imposed by peculiarities of the software platforms. This work presents an architecture which allows the provision of digital Time-Stamping services to mobile devices with limited resources. The architecture is described with respect to a case study system.

1 Introduction

Recent advances in wireless technologies have enabled pervasive connectivity to Internet scale systems which include heterogeneous mobile devices (such as mobile phones and Personal Digital Assistants), and a variety of embedded computer systems (such as smart cards and vehicle on-board computers). This trend is generally referred to as ubiquitous computing [9].

This leads to the need of providing security functions to applications which are partially deployed over wireless devices, and poses a number of new challenges, since security-related functionalities – such as digital signature, time stamping, and secure connection – are mostly based on cryptographic algorithms which are typically computational-intensive. At present, implementing these algorithms on mobile devices is not straightforward (and sometimes not feasible at all) because mobile devices are typically characterized by a low computational power and

are equipped with software platforms with limited functionalities. As an example, PDAs provide a limited supported to security-enabled Java applications. As an example, most modern PDAs are based on the Windows Pocket PC operating system. Unfortunately, to the best of our knowledge, all existing Java virtual machines for this operating system are compliant with the standard J2ME/MIDP (Java 2 Platform, Micro Edition/Mobile Information Device Profile) specification. J2ME/MIDP platforms lack standard XML APIs and support for the SSL protocol, which aren't included in the upcoming MIDP 2.0 specification either.

This work presents an architecture for providing security services to mobile devices, with regard to a case study application, namely a Time Stamping service. By Time Stamping it is meant a service supporting assertion of proof that a datum existed before a particular time [1]. The real importance of time-stamping becomes clear when there is a need for a legal use of electronic documents with a long lifetime. Without time-stamping it is not possible to trust signed documents when the cryptographic primitives used for signing have become unreliable, nor to solve the cases when the signer himself repudiates the signing, claiming that he has accidentally lost his signature key. Providing Time Stamping services to mobile devices is thus crucial, if applications (such as e-business, e-commerce, and e-transactions) must be deployed on such devices. While there have been some attempts to provide security services in the context of ad-hoc mobile systems [10], to the best of our knowledge the field of information security for mobile devices in Internet-scale distributed applications is still widely unexplored. This experience has demonstrated that multi-tier architectures, which exploit a clear separation of concerns, represent an effective means of providing security services based on computationally-intensive cryptographic algorithms to mobile devices with low computing power.

The rest of the paper is organized as follows. Section 2 provides an overview of the main existing solutions for Time Stamping, in terms of algorithms and standards. Section 3 details the specific algorithm implemented in our system. Section 4 describes the conceptual architecture of the multi-tier system we have developed. Section 5 describes

the case study system, including implementation choices and adopted technologies. Section 6 concludes the paper with final remarks and directions of future research.

2 Existing Solutions for Digital Time Stamping

In this section, we provide an overview of the main existing solutions for Time Stamping. The first subsection deals with the description of the main algorithms. The second subsection describes the most relevant standards and recommendation.

2.1 Overview of schemes and algorithms

Time-stamping is a set of techniques enabling one to ascertain whether an electronic document was created or signed at a certain time (i.e., temporal authentication). Without time-stamping we neither can trust signed documents when the cryptographic primitives used for signing have become unreliable nor solve the cases when the signer himself repudiates the signing, claiming that he has accidentally lost his signature key.

Unfortunately, although the creation of a digital data item is an observable event in the physical world, the moment of its creation cannot be ascertained by observing the data itself. Thus, the association of an electronic document uniquely with a certain moment of time is very complicated, if not impossible. The best one can do is to check the relative temporal order of the created data items (i.e., prove the *relative temporal authentication*) using one-way dependencies defining the arrow of time.

In the following we give a definition of Time Stamping systems applicable in legal situations [6]. A *Time Stamping system (TS-System)* consists of a set of principals with the Time Stamping server together with a triple $(S; V; A)$ of protocols. The Time Stamping server (*TS-Server*) is an entity providing digital time-stamping services. The stamping protocol S allows each participant to post a message. The verification protocol V is used by a principal having two Time Stamps to verify the temporal order between those Time Stamps. The audit protocol A is used by a principal to verify whether the TS-System carries out his duties. Additionally, no principal (in particular, TS-Server) should be able to produce fake Time Stamps without being caught. A Time Stamping system has to be able to handle Time Stamps which are anonymous and do not reveal any information about the content of the stamped data. The TS-System is not required to identify the initiators of Time Stamping requests.

Time stamping schemes have been generally classified into three types [2]: simple, linking, and distributed schemes. In the *simple* scheme, a time stamp is generated

by a completely trusted third party (the Time Stamping Authority, TSA) in such a way that it does not involve data included in other time stamps. The main weakness of this scheme is that the TSA has to be unconditionally trusted. It offers evidence that specific data existed at a certain point in time and guarantees the correctness of the time parameter. However, there is no control over the correctness of the value t attached to document x . If the TSA fraudulently alters the time parameter of a certain time stamp, nobody can detect the alteration. Also, if a leakage of the signature key of the TSA has occurred, fake time stamps can be forged at will. In these cases, verifying whether s was issued actually at time t stated in the time stamp is impossible.

Some ten years ago the only known time stamping methods were based on the use of TSAs representing unconditionally trusted third parties. Thus, applications which needed digital time stamping had no choice but resorting to a TSA. In 1991 the seminal publication of Haber and Stornetta [3] showed that the trust to the TSA can be greatly reduced by using the linking schemes or alternatively the distributed schemes. Several papers were published during the last years, which further improved the original schemes [4, 5, 6].

In the *linking scheme*, the TS-System generates a time stamp which involves data included in other time stamps. In practical implementations, the TS-System combines requests from individual clients which arrive within a given time window. As a result, a chain of time stamps is constructed, typically by using a one-way hash function. If an issuer is willing to fraudulently alter a certain time stamp, it has to alter all the time stamps relating to that time stamp. This is why it is considered to be more difficult for an issuer to manipulate a time stamp in the linking scheme than in the simple scheme. Furthermore the TS-System cannot deliberately forge time stamps, since the output of the process depends on all client requests falling in a given time window. Among the linking schemes, two algorithms particularly interesting, are the one proposed in [3] and the so-called Tree-like scheme [5]. Section 3 details these two algorithms. The specific algorithm implemented in our system is the Tree-like scheme.

Finally, the *distributed* scheme is one in which multiple issuers cooperatively generate a time stamp. The main goal of this scheme is to strengthen security against the issuers manipulation of a time stamp by sharing the secret data used to generate a time stamp among the issuers. If the number of collusive issuers is less than a specific predetermined number, they cannot recover the secret data completely and therefore can not manipulate a time stamp.

2.2 Standards and Recommendations

The main source of information about the data format of a time-stamp token, the sequence of the actions that a requestor client and a TS-System have to realize, and the transport mechanism for the exchanged messages, is the Internet RFC 3161 [1].

The RFC 3161 specifies some requirements that a TS-System has to verify in order to be considered as a valid TS-System. Among these recommendations the most important are reported in the following. Clearly a valid TS-System has to use a trustworthy source of time, and it has to produce a time-stamp token upon receiving a valid request from the requestor. When it is possible, a TS-System should not to examine the imprint being time-stamped in any way (other than to check its length). A TS-System has to sign each time-stamp message with a key reserved specifically for that purpose.

As far as the steps composing a valid transactions are concerned, as the first message of this mechanism, the requesting entity requests a time-stamp token by sending a request (which is or includes a *TimeStampReq*, as defined in the request format given in [1]) to the TS-System. As the second message, the TS-System responds by sending a response (which is or includes a *TimeStampResp*) to the requesting entity. Upon receiving the response (which normally contains a *TimeStampToken*), the requesting entity shall verify some information in order to ascertain that the response of the TSS can be considered as a valid time-stamping. In fact the requesting entity shall verify the status error returned in the response and it shall verify the various fields contained in the *TimeStampToken* and the validity of the digital signature of the *TimeStampToken*. In particular, it shall verify that what was time-stamped corresponds to what was requested to be time-stamped. It shall then verify the timeliness of the response by verifying either the time included in the response against a local trusted time reference, if one is available. If any of the verifications above fails, the *TimeStampToken* shall be rejected. Other checks that should be carried out by the requestor are the validity of the TS-System's certificate (e.g., checking the appropriate Certificate Revocation List) and the policy field to determine whether or not the policy under which the token was issued is acceptable for the application.

For the RFC 3161, there is no mandatory transport mechanism for TS-System messages. Different optional mechanisms (such as e-mail, File Based Protocol, Socket Based Protocol, or HTTP) are described, but additional optional mechanisms may be defined in the future.

3 Adopted Algorithm

In this Section we summarize two linking schemes by Haber and Stornetta: the one proposed in [3] and the so-called Tree-like scheme [5]. The algorithm we implemented in our Time-Stamping System is the Tree-like scheme.

The hashing functions (such as SHA-1 [7] and MD5 [8]) are the fundamental components for these algorithms, so we give some details about them.

The *collision free one-way hash functions* is a family of function $H : \{0, 1\}^* \rightarrow \{0, 1\}^l$ compressing bit-strings of arbitrary length to bit-strings of a fixed length l , with the following properties:

- The functions H are easy to compute, and it is easy to pick a member of the family at random.
- It is computationally infeasible, given one of these functions H , to find a pair of distinct strings X, X' satisfying $H(X) = H(X')$. Such a pair is called a *collision* for H .

The practical importance of such functions has been known for some time, and researchers have used them in a number of schemes. In the linking schemes described in this Section, the goals to be reached using hashing functions are: 1) to use one-way dependencies to define the arrow of time, and 2) to compute a digest to be time-stamped instead of the whole document.

The former goal is related to the observation that given a one-way hash function H , if $H(X)$ and X are known to a principal P at a moment t , then someone (possibly P himself) used X to compute $H(X)$ at a moment prior to t .

Furthermore hash functions are used to obtain a hash value $H(X) = Y$ of the document X to be Time Stamped. For the purpose of authentication, Time Stamping Y is equivalent to Time Stamping X . This greatly reduces the bandwidth problem and the storage requirements which would arise if the whole document X were processed. Resorting to hash functions solves a privacy issue as well, since the content of the document to be Time Stamped need not to be revealed. The originator of the document computes the hash values himself, and sends them to the Time Stamping service. The plain document is only needed for verifying the Time Stamp. This is very useful for many reasons (like protecting something that one might want to patent). Depending on the design goal for an implementation of Time Stamping, there may be a single hash function used by everybody, or different hash functions for different users.

A further precondition needed to implement a Time Stamping service is the availability of digital signature schemes. With a secure signature scheme available, when the TS-System receives the document or its hash value to be

time-stamped, it produces the time-stamp, appends it to the document (or to its hash value), then signs this compound document and sends it to the client. By checking the signature, the client is assured that the TS-System actually did process the request, that the hash was correctly received, and that the correct time is included. This takes care of the problem of present and future incompetence on the part of the TS-System.

The first scheme proposed by Haber and Stornetta is referred to as *Linking scheme*. In order to diminish the need for trust, the TS-System links all Time Stamps together into a chain using the collision-resistant hash function H . In this case the Time Stamp for the n th submitted document X_n is $s = sig_{TSS}(n, t_n, ID_n, X_n, L_n)$, where t_n is the current time, ID_n is the identifier of the submitter and L_n is the linking information defined by the recursive equation:

$$L_n := (t_{n-1}, ID_{n-1}, X_{n-1}, H(L_{n-1})).$$

There are several complications with the practical implementation of this scheme. First, the number of steps needed to verify the one-way relationship between two Time Stamps is linear with respect to the number of Time Stamps between them. Hence, a single verification may be as costly as it was to create the whole chain. It has been shown that this solution has impractical trust and broadcast requirements. Haber and Stornetta proposed a modification where every Time Stamp is linked with $k > 1$ Time Stamps directly preceding it. This variation decreases the requirements for broadcast by increasing the space needed to store individual Time Stamps.

In [5], Haber and Stornetta proposed a tree-like scheme in which the Time Stamping procedure is divided into rounds. The Time Stamp R_r for round r is a cumulative hash of the Time Stamp R_{r-1} for round $r - 1$ and of all the documents submitted to the TS-System during the round r . After the end of the r th round a binary tree T_r is built. Every participant P_i who wants to Time Stamp at least one document in this round, submits to the TS-System a hash $y_{r,i}$ which is a hash of R_{r-1} and of all the documents he wants to Time Stamp in this round. The leaves of T_r are labelled by different $y_{r,i}$. Each inner node k of T_r is recursively labelled by $H_k := H(H_{k'}, H_{k''})$, where k' and k'' are correspondingly the left and the right child nodes of k , and H is a collision-resistant hash function. The TS-System has to store only the Time Stamps R_r for rounds (see Figure 1). All the remaining information, required to verify whether a certain document was Time Stamped during a fixed round, is included into the individual Time Stamp of the document.

These schemes are feasible but provide the relative temporal authentication for the documents issued during the same round only if we unconditionally trust the TS-System to maintain the order of Time Stamps in T_r . Therefore, this method either increases the need for trust or otherwise limits

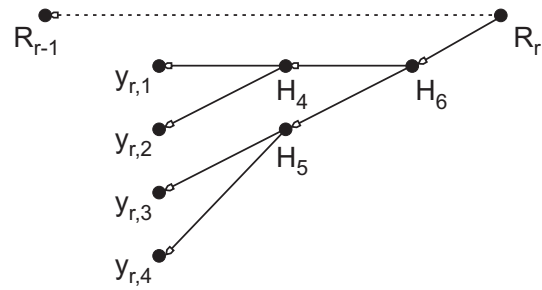


Figure 1. An example of Time Stamp for round r by the Haber and Stornettas scheme.

the maximum temporal duration of rounds to the insignificant units of time.

In the linking scheme, the challenger of a Time Stamp is satisfied by following the linked chain from the document in question to a Time Stamp certificate that the challenger considers trustworthy. If a trustworthy certificate occurs about every N documents then the verification process may require as many as N steps. By using the tree-like scheme, an exponential increase in the publicity obtained for each Time Stamping event is achieved, reducing the storage and the computation required in order to validate a given certificate, as the cost from this operation is reduced from N to $\log N$.

4 System Architecture

This section describes the proposed architecture for providing time-stamping services with emphasis on ubiquity and interoperability. In order to achieve these goals, we suggest the adoption of a multi-tier architecture, whose characteristics are described in the following. The overall organization of the system is depicted in Figure 2.

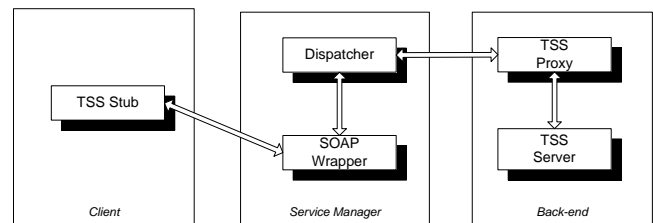


Figure 2. Overall architecture

As shown in the figure, the clients, the service manager (SM) component, and the time stamping servers are located in the first, second, and third tier of a three-tier architecture, respectively. This favors a development approach which exploits clean separation of responsibilities. We use the

emerging Web Services technology to facilitate service access from / to any platform. Unlike traditional client/server models, such as a Web server/Web page system, Web services do not provide the user with a GUI. Web services instead share business logic, data, and processes through a programmatic interface across a network. The main objective is to allow applications to interoperate without human intervention.

As far as the *Client* is concerned, since the client devices have limited resources, the architecture is based on a thin-client model. In fact, the client tier consists of the *TSS Stub*, composed of a simple interface to the middle tier. Clients access the services by using the simple object access protocol (SOAP), and are not aware of the implementation details of the service provided by the middle-tier. All they know is which services are available and what their interface is. SSL and XML processing are needed in order to enable secure SOAP-based communication between the client and the *service manager*. The implementation of such a communication scheme on modern PDAs poses some technical challenges which are described in the next Section, which also presents an effective solution to them.

The *middle-tier* wraps the services provided by the back-end server, and addresses interoperability issues, which are due to the heterogeneity of the clients. It consists of the following components:

SOAP wrapper: it is in charge of *i*) processing requestor's messages, and *ii*) coordinating the flow of SOAP messages through the subsequent components. It is also responsible for ensuring that the SOAP semantics are followed;

Dispatcher: it is responsible for acting as a bridge between the SOAP wrapper and the functional components; it *i*) converts the SOAP messages from XML into the specific protocol for communicating with the functional back-end, and *ii*) converts any possible response data back into XML and places it in the response SOAP message.

This tier also decouples service implementation (i.e., the back-end) from its interface.

The *back-end* is in charge of actually providing the specific security services. In other words, the back-end satisfies the application functional requirements. Since these services are based on cryptographic algorithms which are usually very computationally intensive, this tier has to fully exploit the potential of server platform to boost performance. To this end, a promising approach is to use hardware accelerators. The conceptual model of the back-end is composed of the following components:

TSS proxy: it is in charge of receiving messages delivered by the middle-tier in a common format which does not

depend on the particular implementation of the time-stamping servers. The proxy builds the appropriate message to be exchanged with the relevant back-end server;

TS Service: it represents the system that provides the time-stamping services.

5 Case Study

This section describes the implementation of the proposed architecture with respect to a case-study application. The architecture allows the provisioning of a digital Time Stamping web-Service (TSS) to Java-based PDAs. We implemented the time-stamping service relying on the algorithm described in Section 3. The UML deployment diagram of the case-study application is depicted in Figure 3. The multi-tier architecture has been effectively deployed on an experimental testbed, consisting of *i*) a PDA, *ii*) an application server, and *iii*) a back-end server.

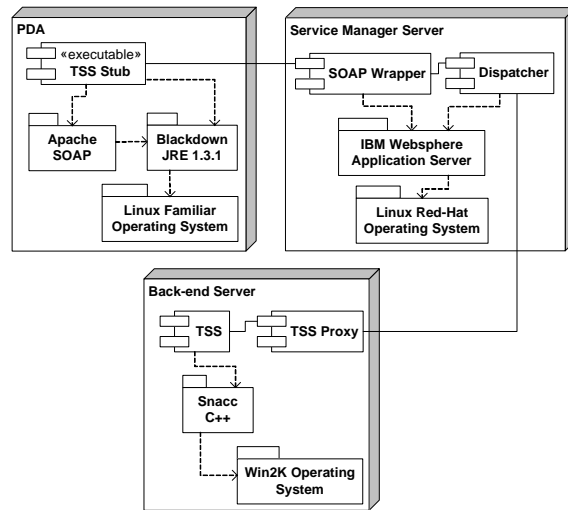


Figure 3. Deployment of the digital Time Stamping service

As far as the PDA is concerned, we used a Compaq IPAQ 3870 handheld device. System requirements of security services may be not suitable for modern personal digital assistants (PDAs), such as the one we used, even though the proposed model is based on a web services-based thin-client approach. In fact, since all the Java virtual machines (JVMs) available for PocketPc-based devices are compliant with J2ME/MIDP specifications, these JVMs lack standard XML APIs and support for the SSL protocol, which are not included in the upcoming MIDP 2.0 specification either. Moreover, these devices are equipped with web browsers

that are not capable of managing Java applets. Hence third party J2ME/CLDC (Connected Limited Device Configuration) libraries are needed that can handle XML, especially Web services-specific XML protocols. KXML and KSOAP libraries contain XML parsing and SOAP-based communication primitives for J2ME applications; however, performance of these libraries is not suitable for most applications, as demonstrated in [11]. For the foregoing reasons, we implemented the client-tier using a Java 2 Standard Edition (J2SE) platform. In particular, we adopted the Blackdown JRE 1.3.1 [12] virtual machine. To enable the setup of such an environment, the PDA runs the Linux Familiar Operating System. This solution allows us to use standard libraries for XML and SOAP processing (i.e., Apache SOAP libraries), and to make the *TSS Stub* executable either from a web-browser or as a stand-alone application. Moreover, the resulting executable does not depend on a specific device, and can be used on several client devices, i.e., laptop and personal computers too.

As far as the *service manager server* is concerned, we adopted the Java 2 Enterprise Edition technology. In particular, the adopted application server is IBM WebSphere 4, which supports web-services, running on Dell PowerEdge 1400SC with two 1400MHz Pentium III processors, running a Linux Red Hat kernel 2.4.18-3 with dual processor support. The *SOAP wrapper* and the *dispatcher* have been implemented as Java applications. The TSS web-service is presented as a web-service by means of WebSphere's web services engine.

As far as the back-end is concerned, we adopted Microsoft technology, and we used Sample Neufeld Asn.1 to C Compiler (SNACC) efficient C++ routines and data structures to support BER encoding and decoding of ASN.1 data structures.

6 Conclusions and Future Work

This work presented an architecture which allows the provision of digital Time-Stamping services to mobile devices with limited resources. The architecture was described with respect to a case study system. We implemented the time-stamping service relying on the tree-like algorithm by Haber and Stornetta. The architecture is structured as a multi-tier system, which has been effectively deployed on an experimental testbed, consisting of *i*) a PDA, *ii*) an application server, and *iii*) a back-end server. The objectives of future research are:

- to improve the performance of the Back-end, by means of Field-Programmable Gate Array technology.
- to develop mechanisms to allow dynamic download of the stub code at client-side.

As far as the former objective is concerned, we are currently conducting experiments on a Commercial Off The Shelf (COTS) programmable PCI hardware board, namely a Celoxica RC1000 board, mounting a Xilinx Virtex-E 2000 FPGA part. We expect to reach a speed-up of 2 orders of magnitude, as compared to a high-end workstation. As far as the latter is concerned, we are investigating Jini-based techniques for wrapping the implemented service on a Jini system.

Acknowledgements

This work has been carried out partially under the financial support of the Ministero dell'Istruzione, dell'Università e della Ricerca (MIUR) in the framework of the FIRB Project "Middleware for advanced services over large-scale, wired-wireless distributed systems (WEB-MINDS)", by the University of Naples Federico II, and by the Consorzio Interuniversitario Nazionale per l'Informatica (CINI), and by the National Research Council (CNR).

References

- [1] RFC 3161, Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP) RFC Time Stamping, <http://www.ietf.org/rfc/rfc3161.txt?number=3161>, August 2001.
- [2] International Organization for Standardization and International Electrotechnical Commission, *ISO/IEC Standard 18014: Information technology - Security techniques - Time stamping services* 2002.
- [3] S. Haber and W.S. Stornetta, *How to timestamp a digital document*, J. of Cryptology, Vol. 3, pp. 99-111, 1991.
- [4] D. Bayer, S. Haber, and W.S. Stornetta, *Improving the efficiency and reliability of digital timestamping*, Proceedings Sequences II: Methods in Communication, Security, and Computer Science, Springer-Verlag, pp. 329-334, 1993.
- [5] S. Haber and W.S. Stornetta, *Secure Names for Bit-Strings*, Proceedings of the 4th ACM Conference on Computer and Communications Security, pp. 28-35, 1997.
- [6] A. Buldas, P. Laud, H. Lipmaa, J. Villemson, *Time Stamping with Binary Linking Schemes*, Advances in Cryptology CRYPTO '98, 1998.
- [7] National Institute of Standards and Technology, *Secure Hash Standard, FIPS PUB 180-1*, Federal Information Processing Standards Publication, 1995 (available on-line at <http://www.itl.nist.gov/fipspubs/fips180-1.htm>.)

- [8] R. Rivest, *The MD5 Message-Digest Algorithm*, RFC 1321, MIT LCS & RSA Data Security Inc, 1992.
- [9] Weiser M. Some computer science issues in ubiquitous computing. *Comm ACM* 1993; **36**(7):72–84.
- [10] Jiejun Kong, Petros Zerfos, Haiyun Luo, Songwu Lu and Lixia Zhang, Providing Robust and Ubiquitous Security Support for Mobile Ad-Hoc Networks, in *Proceedings of the ninth International Conference on Network Protocols (ICNP)* 2001, Riverside, CA, Nov. 2001:41–46.
- [11] Vinay Bansal, Angela Dalton, A Performance Analysis of Web Services on Wireless PDAs,
<http://www.cs.duke.edu/vkb/advnw/project/>
- [12] <http://www.blackdown.org/>
- [13] <http://www.jini.org/>