Collaboration and the Knowledge Economy: Issues, Applications, Case Studies
Paul Cunningham and Miriam Cunningham (Eds)
IOS Press, 2008 Amsterdam
ISBN 978-1-58603-924-0

BALTICTIME – the Demonstration of Open Source Technologies Application for Legal and Accountable Digital Time Stamping

Rimantas MIŠKINIS[1], Bogdan MALYŠKO[2], Dmitrij SMIRNOV[1], Emilis URBA[1], Andrius BUROKAS[1], Peeter LAUD[3], Monika OIT[3], Francesco ZULIANI[4], Marco MESSINEO[4], Aldo FANELLI[4], Jerzy NAWROCKI[5], Pawel NOGAS[5], Leona BANZAITYTĖ[6], Vitalijus SIMANAVIČIUS[6]
[1]*Semiconductor Physics Institute, A. Goštauto 11, Vilnius, LT-01108, Lithuania*
Tel: +370 5 2620194, Fax: +370 5 2627123, Email: time@pfi.lt
[2]*State Tax Inspectorate, Vasario 16-osios 15, Vilnius, LT-01514, Lithuania*
Tel: +370 5 2687506, Fax: + countrycode localcode number, Email: b.malysko@vmi.lt
[3]*Cybernetica AS, Akadeemia Tee 21, Tallinn, 12618, Estonia*
Tel: +372 6 542422, Fax: +372 6 397992, Email: peeter@tartu.cyber.ee
[4]*Nergal S.r.l., Viale Battista Bardanzellu 8, Roma, 00155, Italy*
Tel: +39 0 640801173, Fax: +39 0 640801283, Email: zuliani@nergal.it
[5]Astrogeodynamical Observatory of Space Research Centre P. A. S.,
Bartycka 18a, Warszawa, 00-716, Poland
Tel: +48 61 8170187, Fax: +48 61 8170219, Email: j.nawrocki@cbk.poznan.pl
[6]*Applied Research Institute for Prospective Technologies,*
J.Galvydžio 5,Vilnius, LT-08236, Lithuania
Tel: +370-5-2745845, Fax: +370 5 2745459, Email: protecho@protechnology.lt

Abstract: The overall objective of the BALTICTIME project is to develop the legal and accountable Digital Time Stamping (DTS) system providing the layer of Trust in eGovernmental transaction environment and to demonstrate system performance for time critical functions. The main technological tasks of BALTICTIME are: to develop the interface between National Time Standard Authorities, keepers of National Coordinated Time Scales UTC(k), and Time Stamping Authorities TSA; to develop the issued time stamps archiving system integral with UTC(k) Time Scale; to develop security system based on interoperable certificate data base; to demonstrate BALTICTIME system performance for eGovernment services with a cross-border time stamping possibilities. All components of the BALTICTIME system are developed using open source LINUX UBUNTU software technologies and fitted for the hardware common for the National Time Standard laboratories. Presented test results demonstrate the BALTICTIME system capabilities and compatibility to the requirements of international standards.

1. Introduction

Electronic signature is an increasingly important data protection mechanism used for security and trust in electronic business, electronic services and communications for generating non-repudiation evidence and receiving legal recognition. In many countries the laws and regulations have been adopted which equalize the use and functions of digital signature to handwritten signature. A typical approach to secure digital signatures as non-repudiation evidence relies on the existence of an on-line trusted Time Stamping Authority (TSA). Each newly generated digital signature has to be time-stamped by a TSA so that the

trusted time of signature generation can be identified. Such an approach may be employed in high value transactions where security is the most important requirement.

However, TSA being integral part of eGovernment services using electronic signatures is to large extent responsible for the confidence and accountability of the service. Technological solutions for time stamping services need to be proposed as integral for Enlarged Europe approach overcoming obstacles hampering eGovernment services acceptance by end users: low trust on existing solutions due to disputed time stamp accountability and fragmentation of eGovernment services on European level.

BALTICTIME project is designed to enhance the confidence on time stamping service through adopting capacities of Time Standard Authorities of EU metrology system as most authoritative and reliable backbone for Time Stamping Authority. In such a way level of the confidence and accountability of the service eligible for employment in eGovernment services will be created and integral for Europe solution having strong potential to overcome existing fragmentation will be introduced.

The overall objective of the BALTICTIME project, which is under implementation from 2006, is to develop the legal and accountable Digital Time Stamping (DTS) system providing the layer of Trust in eGovernmental transaction environment and to demonstrate DTS system performance for time critical functions or validation data for digital signature.

The main technological tasks of BALTICTIME are:

1. To develop the interface between National Time Standard Authorities, keepers of National Coordinated Time Scales UTC(k), and Time Stamping Authorities;
2. To develop eDocuments archiving system integral with time scale generator traceable with Universal Coordinated Time Scale (UTC);
3. To develop security system based on interoperable certificate data base;
4. To demonstrate BALTICTIME system performance for eGovernment services with a cross-border time stamping possibilities.

All components of the BALTICTIME system are developed using open source software technologies LINUX UBUNTU distribution and fitted for the hardware common for the National Time Metrology laboratories. Technical specifications will be assessed according to requirements of the standards [1 – 4].

2. Architecture of the BALTICTIME System

To cope with a large number of time stamping requests, the TSA has been deployed in a distributed manner, with one of the sites acting as a main system (TSA main site), keeping its subsystems (Time Stamping Units (TSU), Archive Units (AU) and others) synchronized with the Universal Coordinated Time Scale (directly connected to the National Metrology Institute (NMI)), while the other sites (TSA remote sites) synchronized and linked with the main site. A distributed system can also provide better availability which is important in time stamping applications where some party is always interested in obtaining the timestamp as soon as possible.

3. Timing Subsystem

The timing subsystem generates local TSA timescale, keeps it synchronized to UTC(k) generated by National Metrology Institute and provides it to the all components of main and remote TSA sites. The key parts of time mark synchronization system are:

1. TSA timing Subsystem (located in TSA Main Site);
2. NMI timing Subsystem (located in NMI);
3. Remote Timing Subsystem (located in TSA Remote Site).

Time Stamping Authorities will issue the legal Time Stamps (TS), *id est* TS should be traceable with UTC(k). The main purpose of TSA Timing Subsystem is to generate TSA local time scale synchronized to UTC(k) - time scale generated by National Metrology

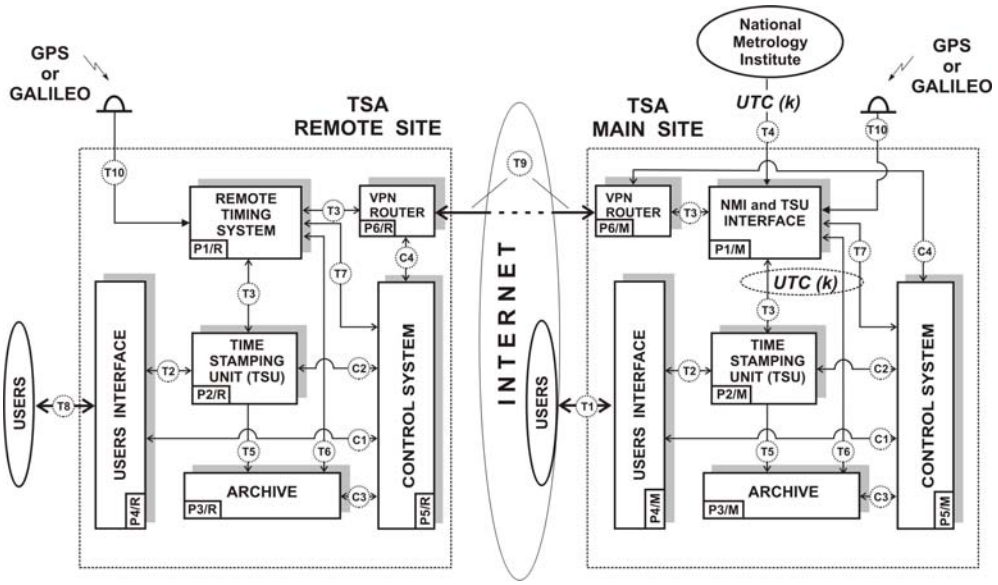

Figure 1. Architecture of the BALTICTIME system. UTC(k) – Universal Coordinated Time Scale of the "k" National Metrology Institute

Institute (NMI) and provide it to Time Stamping Units (TSU) of Time Stamping Authority.

In order to provide system redundancy TSA Timing Subsystem should be equipped with at least 3 time standards. Frequency stability of time standards depends on required time stamping resolution. System after initial synchronization must work autonomously at least for 10 days without external time comparisons to NMI.

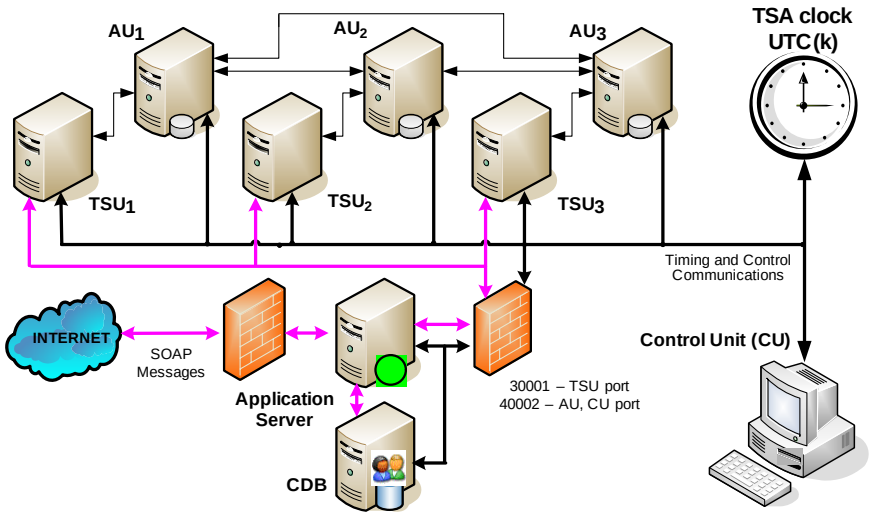

Figure 2. Structure of the BALTICTIME system Main Site.

1PPS signal from time standards should be connected through multiplexer and time interval counter to Timing Subsystem's Control Unit and compared continuously. Control Unit should be able to detect improper results from one of time standards, and eliminate the bad-working time standard. In case when less than 2 time standards work correctly control unit should stop the time stamping process.

To provide required accuracy, and for security reasons Trusted TSA must be synchronized to NMI using at least two independent methods. Use of GNSS based time transfer as a primary method , and NTPv4 as secondary method is recommended. Required accuracy of time transfer methods depends from required time stamping resolution.

4. Time Stamping Units

BALTICTIME's timestamping unit (TSU) implements an RFC3161-compliant timestamping server, also including the security-enhancing extensions specific to BALTICTIME. The implementation is based on the open-source cryptographic library OpenSSL, and in particular its extension OpenTSA implementing the functionality described in RFC3161. OpenTSA implements a typical hash-and-sign timestamping server where the document digest contained in a client's query is signed by the timestamping server, using a dedicated signing key. Key management of the TSU is done according to the standard [ETSI TS 102 023 v1.2.1]. The certificates for the public key are distributed in the same way as certificates for signing keys of any other party. In such a timestamping system, the leakage of the server's signing key would mean the invalidation of all timestamps issued with this key, unless the signing time of those timestamps can be determined using other means. To protect against such leakage, and to also prevent a malicious timestamping server from reordering the timestamps, all timestamps issued by the same TSU are submitted to the archive unit (AU) that will link them together using a one-way hash function (refer to the description of the AU for a more thorough description of linking).

The current state of the linking chain (called the last linking item, and having the size comparable to the size of the output of the hash function) has to be periodically included in the timestamps to complete the chain of one-way dependencies between different timestamps. Hence the TSU will periodically (preferably before each time it issues a timestamp) query the AU for the last linking item. This item is included in the timestamps as a signed attribute (RFC3161 contains provisions for this). Tha details of such inclusion were worked out in the scope of Baltictime project. The implementation of the TSU consists of the necessary modifications to OpenTSA to construct timestamps also containing the last linking item (implemented in C), as well as a small program communicating with the client and the AU, implementing the necessary protocol logic, and using the modified OpenTSA-library (implemented in OCaml).

5. Archival Units

The archival unit keeps a log of timestamping server's activity, thereby allowing to audit it and to compare the timestamps after the keys used to sign them have expired. The archival unit achieves this objective by linking the timestamps together using a one-way cryptographic hash function.

Since a TSA may have multiple TSUs, and hence multiple sites, the audit mechanism must take into account that two timestamps might have been issued by different TSUs. In order to create a single arrow of time between two timestamps issued by two different TSUs the archival units periodically send their last log item to all of the other archival units in order for them to include it in their linking items chain. In this way it is possible to create a linking chain between two timestamps stored in two different archival units.

A cryptographically secure hash function h has been fixed. It is used to link the log items together. The hash function is not hardcoded, but the archival unit stores its identity. If necessary, the hash function may be changed (although this is supposed to be an infrequent and always an extraordinary event). The chosen hash-function must be considered collision-resistant for the foreseeable future. The log is a set of items of the form (n, X_n, L_n), where X_n is the n-th bitstring (timestamp) saved in the log and $L_n = h(X_n, L_{n-1})$ is

the linking information. The log items may also need to store the source of the bit-string X_n (which may be the TSU, or the time mark synchronization unit, or the control system, or some remote archival unit) and the recipient(s) of L_n, if there are any (they may be the TSU or some remote archival units).

The configuration of the archival unit must include the addresses of other archival units, with whom the linking information is exchanged. For each archival unit we have to store its address (probably an IP address and a port number) and the frequency of synchronization.

The TSA Archive System main functionalities are:

- To store issued timestamp and timestamps linkage information;
- To send periodically the last link item information to other archives to link them.

The Archival Unit consists of a database and software performing the required accesses to the database and the network communications with other modules such as the TSU and the other remote archive units. The Archival Unit relies on a relational database system to store its data. Since the usage of open source technologies was strongly suggested, the chosen database system is MySQL. The application interaction to the MySQL database is carried out using the MySQL++ library. This library is a C++ wrapper of the standard MySQL development libraries to interact with MySQL database[i]. This is referred to as the official C++ opensource wrapper of the MySQL connector ANSI C driver.

The communication between Archive Unit and other connected modules is carried out through network protocols in a TCP socket connection. The archival unit works as a server responding to requests that other modules perform. Any TCP/IP network connections and communications is carried out using the library "C++ Sockets "[ii]. In order to encode and decode all the messages that the archival unit exchanges with the other modules, an open-source external package that handles ASN1 has been used[iii]. The generated code has been included in the archival unit software as a static library. Data hashing and base64 encoding is achieved using the open-source OpenSSL library[iv]. Implemented scheme of Archival Units linking has not attempted before [6].

6. User Interface

The TSA Main Site which configuration is presented in Figure 2 serve the needs of small users. For the connection between the small user and TSA, WebServices are used. The user will have a possibility to write their own client application, using any programming environment and language. The necessary information to do that is be available in the WSDL (Web Service Description Language) file and additional documentation is also available at the TSA internet site. As an alternative there are premade user Java applets published on TSA web site. Those applets provide all needed functionality for time stamp generation and time stamp linking chain verification. The only stipulation for client applet usage is that the user must have installed on his/her PC the JAVA run-time environment.

To be able to use the TS service, a small user will has to register. The registration procedure is simplified for project purposes; the user must only provide username, password and user security certificate. To ensure the security, transparency and reliability of the service, there is no application for the generation of the key pair (private and public keys). The user has to generate the key pair manually with JAVA KeyTool (the detailed instructions is available in the site). JAVA KeyStore issued for the storage of user certificate and private key. For the project purposes, self-signed certificates are allowed. In the final version of the TSA only trusted CA's will be accepted and CRL (Certificate Revocation List) checked. The user will have to enter his username, select the private key from the password-protected KeyStore, and then browse for the file to be time-stamped. The user will be able to choose between several hash algorithms to be used for file hash calculation. The following data will be sent to the server: the username, hash, hash algorithm and a signature calculated from hash string with the user private key. After the

submission of the data, the username and certificate is used by the TSA to identify the user. TSA will use the user's public key (from the Certificate submitted at registration) to verify the sender's identity and request integrity as well as to authorize the provision of the TS service. Applet sends the above mentioned data set to the TSA's WebService using HTTPS connection. All data are sent using the SOAP protocol message (look in Appendix). Application server will create ASN1 encoded request to the TSU and will then send it to TSU server through socket connection (TSU server selection is load balanced).

Response is also in SOAP format and consist the Base64 encoded string representing the encoded bytes of the TimeStamp.

7. Test Results

BALTICTIME system was assembled using PCs with: CPU - Intel Core 2 Duo E4400 2 GHz; RAM – 1 GB; HDD – 160 GB. All PCs were running on the LINUX UBUNTU 7.10 Server edition. The first test round was performed. The aim of tests was to determine the time stamping service processing time at the system different loadings. First round tests was performed sending time stamp requests from the user located at the same network node as the BALTICTIME system.

Time stamps requests were sent in two modes – serial and parallel. In serial mode the next time stamp request was sent after receiving the response for previously request.

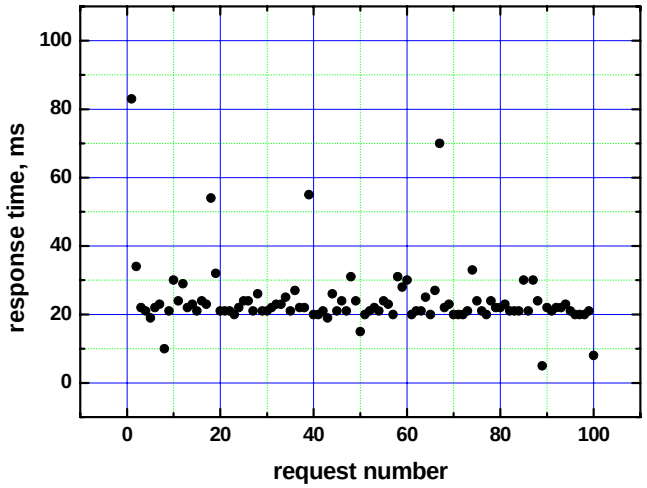

Figure 3. Dependency of Time Stamp Issuing Time (Response Time) on the Time Stamp Request Number at Serial Mode Tests

Typical results are presented in Figure 3. Mean value of the time stamp response is in the range of 24 ÷ 25 ms. Test results demonstrate that BALTICTIME system able to issue up to 25 time stamps per second. The largest obtained response time not exceed 100 ms.

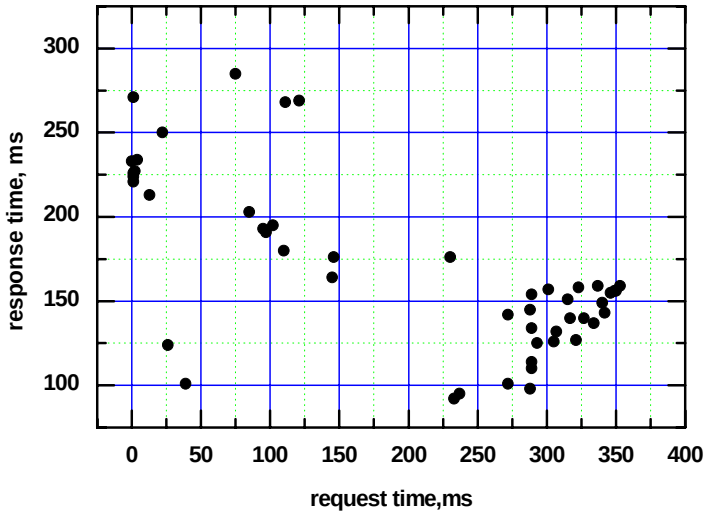

Figure 4. Dependency of time stamp issuing time (response time) on the time stamp request time at parallel mode tests. 50 time requests stream.

Parallel mode tests was done by sending the concurrent time stamps requests in the one stream. The streams containing up to 300 requests was used. Test results obtained in the parallel mode are presented in Figure 4. Request time is a time mark fixed by Application Server when the time stamp request is delivered to the Time Stamping Unit. Test results show that 50 time request stream were processed during within 0,35 second and the mean response time (time stamp issuing time) is 170 ms. Tests with the streams consisting of 300 time stamp requests demonstrate a response time in the range from 0,8 to 1,6 second.

8. Conclusions

All components of the BALTICTIME system are developed using open source software technologies LINUX, OpenTSA, OpenSSL, MySQL and fitted for the hardware common for the National Time Metrology laboratories. Technical specifications will be assessed according to requirements of the standards. First test results demonstrate the BALTICTIME system ability to integrate into the eGovernment, eCommercial and other eServices. Obtained test results are a backbone for further BALTICTIME system tests and developments.

References

[1] ETSI TS 101 733 v1.5.1 (2003-12) “Electronic Signatures and Infrastructure (ESI); Electronic Signature Formats”.

[2] ETSI TS 101 861 v1.2.1 (2002-03) “Time stamping profile”.

[3] ETSI TS 102 023 v1.2.1 (2003-01) “Electronic Signatures and Infrastructure (ESI); Policy requirements for time-stamping authorities”.

[4] RFC 3161 (2001) “Internet X.509 Public Key Infrastructure Time – Stamp Protocol (TSP)”.

[5] J. Cryptology 3(2): 99-111 (1991) Stuart Haber, W. Scott Stornetta: How to Time-Stamp a Digital Document.

[6] Arne Ansper, Ahto Buldas, Märt Saarepera, Jan Willemson: Improving the Availability of Time-Stamping Services. Information Security and Privacy, 6th Australasian Conference, ACISP 2001: 360-375.

[i] http://tangentsoft.net/mysql++/

[ii] http://www.alhem.net/Sockets/

[iii] http://lionet.info/asn1c

[iv] http://www.openssl.org