

CESNET Technical Report 15/2008

Time-Stamp Authority

VLADIMÍR SMOTLACHA, MILAN SOVA

Received 4.12.2008

Abstract

This document describes the time-stamp authority (TSA) that we have designed and built with focus on providing of accurate and trustful time information and on immunity from service compromising. Our system is based on open software and utilizes a hardware security module (HSM) which keeps the private key and speeds up cryptographic operations. We also use special hardware supporting clock synchronization and calibration.

Keywords: time-stamp authority, PKI, time synchronization

1 Time-stamp Authority principles

Time-stamp authority (TSA) is a service providing electronic message containing the trustworthy time of issuance. The time-stamp protocol (TSP) specified in the RFC 3161 [1] defines both the communication protocol and the time-stamp format. The time information must be provided with a resolution of 1 second or finer. The TSP is based on a request message sent by a client (Time Stamp Query – TSQ) and a response message (Time Stamp Reply – TSR) sent by the server. The TSQ can contain arbitrary information (e.g., one-way hash of a file), which is returned back after being signed in the TSR. This way, the information is bound with the time stamp. The TSA clock must be synchronized by a trustworthy method to any time system that ensures traceability to UTC. According to the RFC3161, the TSP can be implemented either on the 3rd network level (TCP) or on the 4th network level (e.g., HTTP or SMTP).

Although there exists an open source implementation OpenTSA [2], the majority of operating TSAs are commercial proprietary solutions and therefore it is very difficult even for the operator to give an evidence of TSA time service accuracy. In some countries, the law forces every licensed TSA operator to calibrate its TSA system. We will further discuss the issue in the section Section 5.

2 TSA hardware

The TSA runs at standard server with 64-bit Intel processor. The hardware includes specialized cards that support data encryption and system clock synchronization. CPU performance is not critical, however the data encryption card requires 64-bit operating system.

2.1 Hardware security module

In our application, the hardware security module serves mainly for safe keeping of the private key. We decided to use SCA6000 card¹ from Sun Microsystems – we already successfully utilized it in the CESNET Certification Authority. This card also accelerates cryptographic operations (up to 13000 RSA operations per second) therefore time stamp signing is not a bottleneck of our TSA. However, more important is the system immunity from service compromising as the private key is neither stored in the system memory nor used by the CPU. If the intruder hacks the operating system or even steals the computer, he has no way how to read the private key.

2.2 Clock

The system is also equipped with the PCT-7424 card² from company Tedia s.r.o., which processes the incoming PPS (pulse per second) signal without interrupt latency. The PPS signal represents external time source which is used by the NTP daemon for system clock synchronization. The clock stability is further improved by the ovenized oscillator which replaces the motherboard crystal 14.318 MHz.

3 TSA software

The card SCA6000 is the Sun Microsystems proprietary hardware and the manufacturer provides driver only for Solaris and Linux. We decided to use Linux but only limited number of distributions and kernels is supported – the newest one is RedHat 5.0 with x86_64 kernel version 2.6.9-22.

The whole TSA functionality is implemented in the OpenTSA package [2] which includes:

- patch enhancing the OpenSSL by the time-stamp protocol according to RFC-3161
- Apache2 module that implements the service on the HTTP protocol

The OpenTSA package unfortunately lacks the SCA6000 card support and we were forced to program this interface ourselves.

Another important system program is the NTP daemon (we have installed recent version 4.2.4) which is responsible for TSA clock synchronization.

4 Clock synchronization

Basic assumption of every time stamp is that the issuing TSA guarantees declared level of internal clock accuracy. The TSA operator must arrange the clock synchronization by a trustworthy method to any time system that ensures traceability to UTC. The most common method is to utilize a GPS receiver generating the PPS signal. Alternatively, another PPS source can be used as well, e.g., the Cesium clock. PPS signal from our GPS Trimble Acutime 2000 has maximal time error 50

¹ <http://www.sun.com/products/networking/ssllaccel/suncryptoaccel6000/index.xml>.

² <http://www.tedia.cz/produkty/pct7424.html>.

nanoseconds and the PPS capture card PCT-7424 adds uncertainty 50 nanoseconds, too. Passing the PCI bus and further software processing increase the total uncertainty to about 300 nanoseconds. The running NTP daemon disciplines system clock which represents the time scale T(TSA).

TSA clock stability is about 10^{-9} – it depends on the ovenized oscillator (OCXO). Even when no PPS signal is available for a day, the TSA time uncertainty is still better than 100 microseconds.

We developed a simple method that allows to calibrate the system clock: the utility *gen_pps* transmits T(TSA) to a serial port in the form of generated PPS signal and this signal is compared with reference clock. Long time measurement proved that we synchronize time scale of our TSA prototype with uncertainty less than 2 microseconds. Figure 1 shows example of such one-day measurement.

Described method of system clock synchronization was adopted from our design of CESNET primary NTP servers [5].

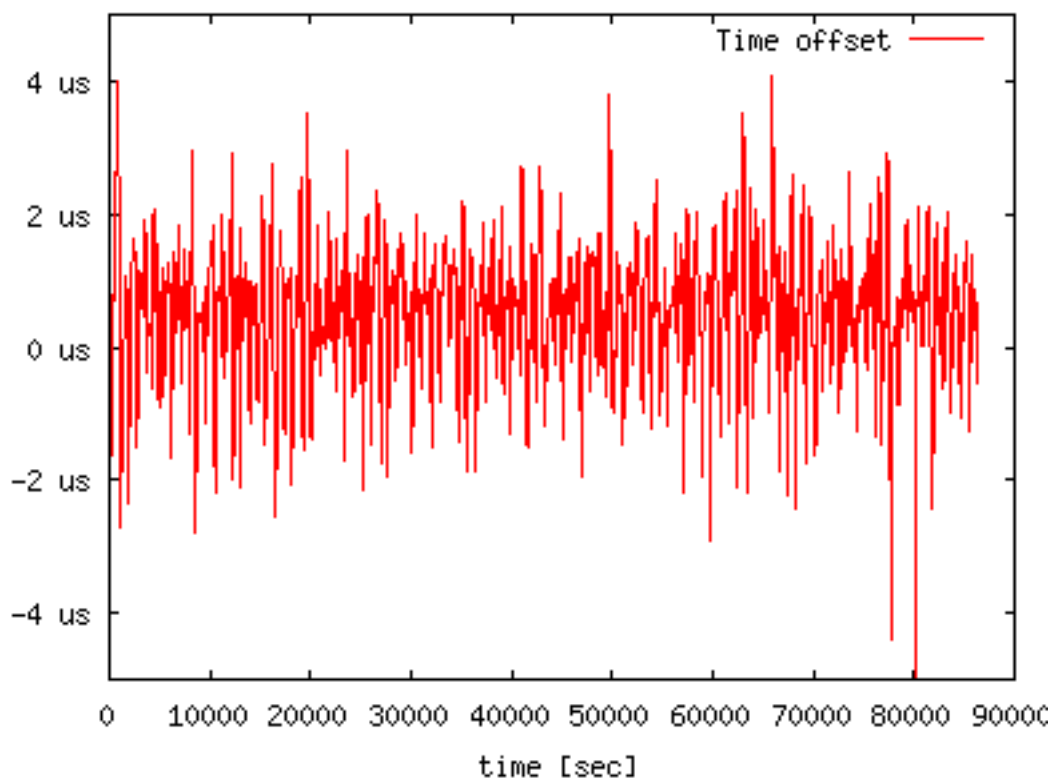


Figure 1. TSA clock accuracy

5 TSA calibration

The TSA provides time information with specified accuracy. As any other time service, it should be subject of calibration that allows to trace local time scale to UTC as requested in time metrology. However, TSA calibration is a new topic – we designed one of such methods in [3] and further elaborated it in [4].

TSA issues time stamp that binds an event (the client's request TSQ) with an epoch with some uncertainty which includes:

- Time offset of the TSA clock: It is the essential source of inaccuracy and it has to be considered in TSA service specification. Accuracy of TSA clock has been discussed in the section Section 4.
- Delay in TSQ and TSR processing: The delay summarizes particular delays inside the TSA system, including client authorization and authentication, response generation and queuing of network interfaces. This delay is mainly done by the TSA software design and also may depend on the instantaneous TSA load.
- Network transport delay of the TSQ from client to service provider: The network delay (and the delay variation) significantly influences the overall TSA accuracy. While the mean network delay can be easily measured, it is difficult to estimate the maximum network delay.

It is evident that the service provider can not guarantee any parameters that depend on network between service provider and service client. While it is important and meaningful to evaluate the service accuracy at the particular client site, real calibration is possible only at the provider site.

The TSA calibration is based on evaluation of the difference between issued time stamp and the epoch at which the TSQ occurred at the reference point (e.g., TSA network interface, network border router) specified by the service provider.

Time stamps are generated from the server time scale $T(S)$ synchronized to UTC with an uncertainty $\pm u(S)$. The uncertainty $u(TS)$ of the time stamps is in principle larger and practically much larger than $u(S)$ due to limited resolution of the time stamps (RFC 3161 [1] states that $u(TS)$ can be up to 1 s). Uncertainties $u(S)$ and $u(TS)$ are sometimes confused with the uncertainty $u(UTC)$ of the local source of UTC (typically $u(TS) \gg u(S) \gg u(UTC)$). The basic calibration objective is to verify the uncertainty $u(TS)$ which is associated with the access point defined by the TSA provider (typically at the server interface).

The TSA service calibration is carried out by the Calibration Computer (CC) which is described in [3]. We connected the CC to the same segment of LAN as the TSA and made the active calibration which employs the CC as a substitute for the TSA client as shown in Figure 2.

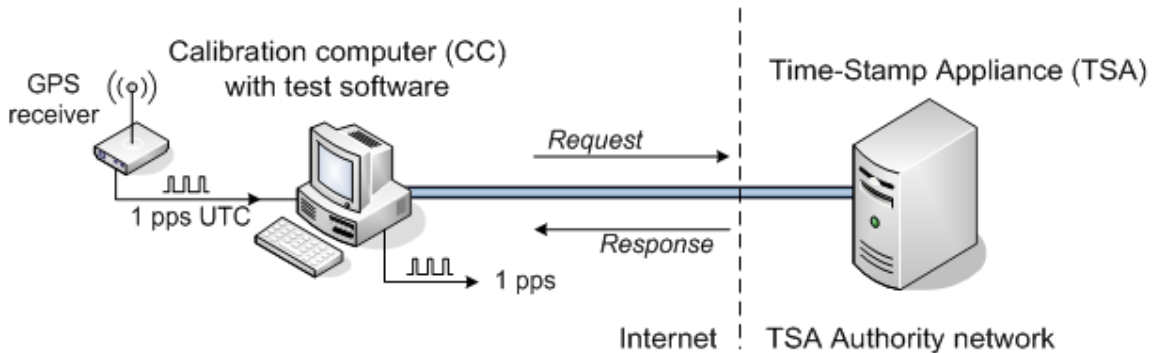


Figure 2. Method of TSA calibration

Let the CC clock represents the time scale $T(C)$ and TSA clock represents the time scale $T(S)$. The TSQ is sent by the CC at time $T_q(C)$ (with uncertainty $u(Q)$)

and the TSR that contains the time stamp $T_s(S)$ is received by the CC at time $T_r(C)$. The uncertainty of time stamp is $u(TS)$. The time scales difference is

$$x_0 = T_s(S) - T_q(C)$$

where the uncertainty of x_0 is given by $u(TS)$ since $u(Q) \ll u(TS)$.

The *tsa_mon* utility evaluates times $T_q(C)$ and $T_r(C)$, matches corresponding TSQ and TSR and decodes the $TS(S)$.

Values $T_s - T_q$ in Figure 3 show calibration result of our TSA. For comparison, we also evaluated the total response time $T_r - T_q$.

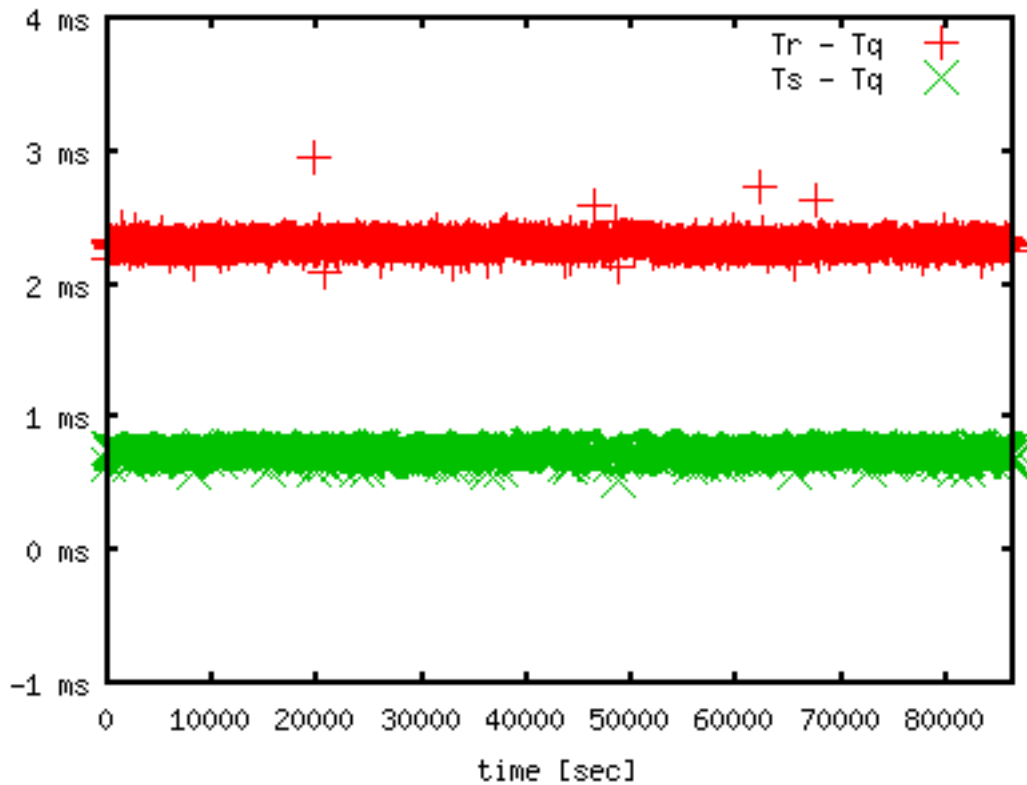


Figure 3. Calibration results

We see that mean value of $T_s - T_q$ is 0.8 millisecond. Standard deviation of shown data is 0.16 milliseconds.

6 System utilization

From the user viewpoint, obtaining a time stamp consists of several steps:

- TSQ generation
- time stamp request, i.e., sending TSQ to the TSA
- TSR receiving and optional parsing

6.1 TS command

The OpenTSA package contains a tool *ts*, which is accessed as a new *openssl* command. The *ts* has three subcommands:

query

The *query* subcommand generates TSQ. It is used by the TSA client.

reply

The *reply* subcommand generates TSR as a response to TSQ. It is used by TSA server (in standard implementation with HTTP protocol, it is executed by the Apache module).

verify

The *verify* subcommand checks the TSR integrity. It is used by the client or any other third party to check if received TSR matches the TSQ.

Example:

```
openssl ts -query -data design1.txt -no_nonce -out design1.tsq
```

OpenTSA web pages contain a detailed description of *ts*³.

6.2 Time stamp requesting

URL of our TSA is *http://tsa.cesnet.cz:3161/tsa*, which means that it is accessible by the HTTP protocol. The TSA request can be made by the following command:

```
cat design1.tsq | curl -s -S -H \
'Content-Type: application/timestamp-query' \
--data-binary @- http://tsa.cesnet.cz:3161/tsa -o dat.tsr
```

assuming that *design1.tsq* is the TSQ file generated in previous example and TSR is stored as *dat.tsr*.

We also compiled a static executable *ts* binary (32-bit Linux), that can be simply installed without the need of patched OpenSSL build. The package contains scripts for arbitrary file timestamping and time stamp verification. Until we finish the official CESNET TSA web pages, the packaged can be downloaded from a temporary repository⁴.

7 System performance and security consideration

The system is supposed to provide accurate, reliable, and trustworthy signatures. To achieve this goal, the security of the signing key is essential. The signing key is generated within an HSM as an unexportable object, i.e., it can never leave the HSM. The key is activated by an operator by entering an activation passphrase on every start of the system. Thus the only process able to use the key is the TSA server, even the root account cannot use the key without the passphrase.

³ <http://www.opentsa.org/ts/ts-20060923.html>.

⁴ <https://perfmon.cesnet.cz/software/tsa/TSA.tar.gz>

To mitigate a possible attack involving modification of the system time, the time is being constantly monitored from NTPMON system [6]. Any significant time discrepancy would invoke an inquiry into the system with the possibility of revoking the TSA certificate and effectively revoking the time-stamps issued within the respective time interval.

The system is naturally vulnerable to denial-of-service attack. However, our stress tests proved the throughput of almost 3000 time stamps per second. In case of need, the system and the firewall in front of it may be configured to limit the service to authenticated clients from authorized network only.

References

- [1] ADAMS, C.; CAIN, P.; PINKAS, D.; ZUCCHERATO, R. *Internet X.509 Public Key Infrastructure, Time-Stamp Protocol (TSP)*, RFC 3161⁵, IETF, August 2001.
- [2] GLOZIK, Z. *OpenTSA*, 2006. Available online⁶.
- [3] SMOTLACHA, V.; TYML, P.; ČERMÁK, J. "Calibration of Time Stamp Authorities", Proc. *15th IMEKO TC4 International Symposium on Novelty in Electrical Measurements and Instrumentations*, Volume I, Iasi, September 2007.
- [4] SMOTLACHA, V.; ČERMÁK, J.; PALACIO, J. "On Calibration of Network Time Services". *Metrologia*, Volume 45, p. S51–S58, December 2008, [accepted].
- [5] SMOTLACHA, V. *NTP Servers with a Stable Oscillator*. Technical report 7/2007⁷, Praha: CESNET, 2007.
- [6] SMOTLACHA, V. *NTP Monitoring System*. Technical report 23/2007⁸, Praha: CESNET, 2007.

⁵ <http://www.ietf.org/rfc/rfc3161.txt>

⁶ <http://www.opentsa.org/site/>

⁷ <http://www.cesnet.cz/doc/techzpravy/2007/ntp-server>

⁸ <http://www.cesnet.cz/doc/techzpravy/2007/ntpmon>