Some Critical Aspects of the PKIX TSP

Cristian Marinescu and Nicolae Tapus

University "Politehnica" Bucharest, Romania
cristian.marinescu@omicron.at and ntapus@cs.pub.ro

Abstract. Authentication, non-repudiation, and digital signatures, require the ability to determine if a data token existed at a certain moment in time when the creator's credentials were valid. Time-stamps are tokens which contain a verifiable cryptographic link between a time value and a data representation. The paper presents some critical aspects of the X.509 Public Key Infrastructure Time Stamp Protocol, trying to suggest some possible improvements to the protocol.
Keywords: PKI, security, time-stamp, TSA, PKIX TSP.

1 The PKIX Time Stamp Protocol

In an effort to solve some of the current security problems, many security solutions and services require the ability to establish the existence of data at a certain moment in time. Time-stamps (TSs) are a digital solution to this problem, providing the proof that the signed data existed prior to the indicated time. The Time Stamping Authority (TSA) is the trusted third party (TTP) that generates the digital TS and guarantees that the time parameter is correct. TSs can indicate weather or not an electronic signature was generated before the private key expired or was compromised, non-repudiation and authenticity beeing guaranteed if this is the case [6]. The moment when the document was time-stamped is also an important part of the requirement to present undeniable information about who, what and *when* e-documents were issued, in order to be used in a court of law [5].
RFC3161 specifies a simple time-stamp scheme based on digital signatures and a typical client-server architecture. The PKIX TSP specifies the format of the packets, along with some possible transport protocols and some verifications to be done by the server and by the client. The communication mechanism consists of a one-step transaction: the TSP client sends a request to the TSA; the server has to check, upon receiving a packet, that it contains a valid TS request, and to send a valid TS token back [1]. The requester has the responsibility to verify that the received TS token is what it has requested. The verifier does not have to be the same as the requester, any third party may check the TS. In case of a dispute, the claimer has to provide the TS, to prove that the data existed at the specified moment. It should also be noted that this does not prove sole possession or origination of the data, other mechanisms should be used in conjunction with a TSA to accomplish this task.

2 Some Critical Aspects of the PKIX TSP

Like many PKI standards, the PKIX TSP does not consider real-life conditions, such as incompatibility problems, software bugs, or the interconnection of software modules. This rather *optimistic* approach can cause security problems, even though a certain abstraction is quite unavoidable in a standardization process. The resulting protocol has been designed to be a part of the PKI, and therefore should not be regarded as a stand-alone solution [3].
RFC3161 suggests several transport protocols that can be used: e-mail, FTP, HTTP, and raw sockets. Unfortunately, the standard specifies more options for the raw sockets solution and disregards the basic rule of network protocols to completely ignore the underlying transport layer. The raw socket polling support is unlikely to simplify any implementation, and just adds unnecessary complexity to the protocol. Under these circumstances, interfacing an HTTP solution to a raw socket implementation is difficult to achieve because the protocol behaves different depending on the transport layer. Since interoperability is in our opinion an important issue, we argue that the next version of the standard should dispense with the polling operations.
The standard contains some questionable *features*, like the *ordering* field or the *policy* information, which can cause problems if implemented like the standard suggests. The main benefit of the *policy* field, like defined by RFC3161, should be the possibility to provide more information about the conditions under which a TS may be used, the availability of a log, etc. Unfortunately, it is neither specified what policies must be provided, nor what the TSA should do under these policies. Another unsolved problem is the procedure to be used for advertising and parsing the supported policies. In order to be able to request a certain policy, the client has to find out the available policies, but at the moment, this has to be solved outside the standard . We suggest to define a frame inside the standard, so that the client could optionally start by parsing first the available policies and other parameters of the TSA.
Another issue is generated by the usage of the *ordering* field in the TS token. If the field is set to true, all TSs generated by the same TSA can be ordered based on the time parameter. Otherwise, ordering the TSs is just possible if the difference of the time parameters is greater than the sum of the accuracies. This is rather a mistake, ordering TSs generated by the same TSA should always be possible, any other approach is not acceptable. Establishing a timeline is an important feature of TS schemes in general. We strongly suggest to avoid the usage of the *ordering* field, any practical implementation should serialize the TS generation process, in order to guarantee the timeline.
The several security considerations specified by RFC3161, are rather thin and insufficient. In case that the key expires or gets compromised, the certificate has to be revoked with a specified reason, but auditing, notarizing or even applying a new signature to all existing TSs is difficult to achieve and a tremendous task to accomplish [4], even if assuming that all the generated TSs have been stored locally (which is normally not required). This still does not solve the problem in case that the private key of the TSA gets compromised, because in most cases

it is difficult to find out the exact moment when this happened. In our opinion, a much more simple approach, borrowed from the linking schemes, would be to embed information from the previous generated token in the TS; another solution, borrowed from the distributed schemes, would be to time-stamp the same message digest at two or more different TSAs. A protocol improvement should include this possibility, since this would not just increase the security of the scheme, but also solve one of the biggest PKI problems [2].
In an effort to prevent the *man-in-the-middle* attack, RFC3161 makes an *interesting* recommendation: to consider any response as suspect if it takes too much time between request and reply. We argue that this approach is futile, since the time necessary to process a request is not an indisputable argument for an attack; it can be just the sign of a simple network congestion. The question that also rises is how to define *an acceptable period of time*, since this parameter would be different depending on the transport protocol [4]?
The TS is done on a message digest, having no constraints on the data format, but this apparent simplicity hides another problem when time-stamping CMS digital signatures. As defined, the TS token is placed inside a client's CMS digital signature as an unsigned/unauthenticated attribute within the signer info, with a special OID. Since two different CMS data structures are needed (the one to be signed and the one to place the TS inside), the implementation can be rather complex. RFC3126 tried to solve this problem by extending CMS to include TSs, but we believe that it would have been better if the TSA would have time-stamped not just a hash but also the signature of the requester, if desired [4].
The virtual world of the PKIX TSP does not consider security threats, and interoperability issues. Unfortunately, it is impossible for practical implementations to avoid all the problems presented, if an RFC3161 compliant version is the goal. This is in our opinion the main reason why a second version should improve and correct at least some of these mistakes. Failing to solve them will have negative effects on the acceptance of RFC3161 as the *de facto* time-stamp standard of the Internet.

References

1. Adams, C., et al: RFC3161 Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP), ftp://ftp.rfc-editor.org/in-notes/rfc3161.txt (2001)
2. Adams, C. and Lloyd S.: Understanding PKI, Addison-Wesley, NY, USA (2002)
3. Housley, R. and Polk, T.: Planning for PKI - Best Practices Guide for Deploying Public Key Infrastructure, John Wiley & Sons, NY, USA (2001)
4. Marinescu, C., et al: A Case Study of the PKI Time Stamp Protocol Based On A Practical Implementation, in: Proceedings of the CSCS15, Bucharest, (2005)
5. Merill, C.R.: Time is of the Essence, CIO Magazine, http://www.cio.com/archive/031500_fine.html (2000)
6. Pinto, F. and Freitas, V.: Digital Time-stamping to Support Non Repudiation in Electronic Communications, in: Proceedings of the SECURICOM'96, CNIT, Paris, (1996) 397-406