Time Stamp Protocol

© Copyright 2013 by Peter C. Chapin
Vermont Technical College

Last Revised: December 18, 2013

1 Introduction

This document describes the time stamp protocol. See RFC-3161 for specifics on how this protocol can be encoded for actual use.

The principals are $A$, $B$, and $T$.

- $A$ (Alice): Owns document $d$ and wishes to obtain a cryptographically verifiable time stamp for that document.
- $B$ (Bob): Wishes to verify that document $d$ existed on or before a specific time.
- $T$ (Trent): A trusted time stamp authority that creates time stamps and maintains an accurate notion of time. This authority has keys $\kappa_{T}^{(p)}$ and $\kappa_{T}^{(r)}$. As usual the public key is made known to all principals by a means unspecified by this protocol. The private key is assumed to be known only to $T$.

The protocol is intended to have the following properties. The primary property is emphasized.

- $A$ is unable to generate a time stamp for any document $d$ containing a time earlier than when $d$ was last modified.
- $A$ is able to store time stamps with the documents to which they are associated.
- $A$ does not need to reveal $d$ to $T$.
- $A$ can obtain time stamps from $T$ anonymously.
- $B$ can verify a time stamp at any time as long as $T$’s private key remains valid.
2 Protocol

The protocol proceeds as follows:

1. \( A \rightarrow H(d) \rightarrow T \). Alice sends a hash of her document to Trent. The hash algorithm used must be acceptable to Bob.

2. \( T \rightarrow s = S(\kappa_\tau^T, H(d) \parallel t) \rightarrow A \). Trent concatenates the current time \( t \) to the hash and signs the result with his private key. He returns the resulting time stamp \( s \) to Alice.

3. Alice verifies \( s \). This entails a) verifying Trent’s signature, and b) checking to be sure the hash contained in \( s \) agrees with that originally sent by Alice. Alice should also check the reasonableness of \( t \).

4. At some later time: \( A \rightarrow d \parallel s \rightarrow B \). Alice sends her document, together with the corresponding time stamp, to Bob.

5. Bob verifies \( s \). This entails a) verifying Trent’s signature, and b) computing \( H(d) \) and checking that it agrees with the hash inside the time stamp. Bob can then be assured that the document \( d \) was prepared on or before time \( t \) in the time stamp.

Notice that this protocol does not give Bob any assurance that Alice is the author of \( d \). If that service is needed additional steps are required. It is interesting to note that Alice could make a digital signature over \( d \) without invalidating \( s \). Thus she could sign the document just before sending it to Bob and send \( d \parallel s \parallel S_d(\kappa_\tau^A, d) \) instead.

3 Informal Analysis

I will begin with an analysis of the trust requirements of each principal.

Alice does not need to Trent. She does not need to show her document to Trent and she fully verifies all aspects of the time stamp returned by Trent. She does need to reveal her document \( d \) to Bob but presumably that is acceptable to her as a user of this protocol (the point of the protocol is to support the submission of her document to Bob).

Bob needs to trust that Trent maintains the correct time and, of course, that his private key is not compromised. Notice that Alice is not concerned about the compromise of Trent’s key except to the extent that such a compromise would undermine the credibility of her time stamp.

Now I will do an analysis of network traffic. As usual I assume a Dolev-Yao attacker. I acknowledge that a D-Y attacker can create a denial of service situation. However, I assert that all such situations are detectable by one of the principals. I consider each message in turn.
• In step 1 Alice sends her hash to Trent. If that message is blocked or delayed she will detect it because no time stamp will be delivered. If the hash is modified she will detect it in step 3.

• In step 2 Trent sends $s$ to Alice. If that message is blocked or delayed Alice will detect it because no time stamp will be delivered. If $s$ is modified Alice will detect it in step 3 because Trent’s signature will be invalid.

• In step 4 Alice sends her document and time stamp to Bob. If that message is blocked or delayed, the message might be lost (there is no expected acknowledgment from Bob in this protocol). However, Alice can later prove that $d$ existed at or before the specified time in $s$ so any confusion could be resolved when (if!) the problem is discovered.

   If Alice’s message is modified, Bob will detect it because either the hash in $s$ won’t match or Trent’s signature on $s$ will be invalid.

   It is true that a D-Y attacker can read the contents of $d$ at this point. If that’s a problem the protocol will need to be extended to include support for confidentiality of $d$ in step 4.

The protocol seems to meet its intended goals. It could potentially be improved by adding an acknowledgment step after Alice sends her document to Bob so that Alice is assured that Bob received the document in a timely manner. Support for authenticity and confidentiality of Alice’s document might also be useful additions.