

A Survey of the Timestamping Problem

Michael de Mare
Computer Science Department
SUNY
Institute of Technology

Lincoln DeCoursey
Computer Science Department
SUNY
Institute of Technology

Abstract

The timestamping problem is one of showing that one possessed a document before a certain time. In this paper several protocols are discussed for solving this problem. These protocols were developed by Haber and Stornetta [11] and by Benaloh and de Mare [5, 6]. In addition, this paper discusses the Byzantine Generals' Problem, which is necessary for the protocols to operate and discusses the relative strength of the protocols and ways to avoid certain types of attacks.

1 Introduction

The problem of timestamping can be broken down into two problems. The first problem, showing that one had a document after a given date, is trivial. Consequently, research efforts have focused on the second problem. This is to show that one had a document before a given date.

To illustrate why the first problem is trivial and the second is not, we use the example of a hostage. [5] We can show a picture of the hostage holding

up a newspaper to show that the hostage was alive after the newspaper was printed, but it says nothing about how early the hostage was taken.

The timestamping problem was first studied by Haber and Stornetta in 1990. [11] This was followed by Benaloh and de Mare in 1992. [5] Benaloh and de Mare did additional research with applications to the timestamping problem in 1993, and presented their results at Eurocrypt '93. [6]

Due to the importance of secure authenticated broadcast networks for the operation of the protocols presented, we will include a section on the Byzantine Generals' Problem. This is to show the reader that the time stamping systems can be implemented without any unreasonable assumptions about the underlying communications channel.

2 Document Chaining

The first methods proposed for timestamping were protocols involving document chaining. These methods were proposed by Haber and Stornetta in 1990. In these methods, linking information is included in the time stamp certificate. The linking information is bits from the previous timestamp certificate. This assures that the document was produced after the previous document, hence the previous document came before this document. [11]

Let $\sigma(x)$ be the signature function of the timestamping authority. A certificate would be of the form

$$\sigma(n, t_n, ID_n, y_n, L_n)[11]$$

where n is the sequence number, t_n is the time, ID_n is the client number, y_n is the hash of the document to be time stamped and L_n consists of bits from previous documents. [11]

In order to verify a timestamp certificate, a challenger first verifies that the certificate is of the correct format and contains the correct information. Then the challenger calls ID_{n+1} to see if his linking information includes this certificate's bits. If he is still suspicious he checks out the $n + 1$ certificate in a similar way.

It can be argued that this is expecting an unreasonable amount of cooperation from other participants.

3 Distributed Trust

Another method, also proposed by Haber and Stornetta, for digital timestamping, is to get a large number of other participants to sign your document along with the time. Such a system would have k other participants sign the document and provide security based on the trustworthiness of those k participants. [11]

This approach has multiple problems. One is that if k is chosen large enough to provide sufficient trust in the system, it will become subject to denial of service if enough participants refuse to sign. Another is that many participants must participate in the generation of each individual certificate. [11]

4 Timestamping in Rounds

In 1992 Benaloh and de Mare proposed timestamping in rounds. [5] The observation behind this proposal is that documents that occur within a certain length of time of one another can be said to have occurred simultaneously. Benaloh and de Mare cite the example of two patent applications arriving at the patent office on the same day. This way documents can be grouped into time intervals and one timestamp can be issued for each time interval. Certificates show that the document can be hashed into the round's timestamp. [5, 6]

The benefits of such a system are clear. Documents can be related to one another. In fact, for a sufficient number of participants, it is similar to Haber and Stornetta's k -signature scheme [11] because all of the round's participants and any honest observer will vouch for the validity of the round's timestamp. [5, 6] The challenge is to produce certificates which do not take too much space store or computation (or other assumptions) to verify.

5 Tree Based Timestamping

In 1992 de Mare and Benaloh proposed a timestamping system using rounds that require secure broadcast networks. Such a system can be simulated with an agreement protocol (see Section 7). This timestamping system requires logarithmic storage on the number of participants to hold a certificate.

The only assumptions this protocol makes are one way functions and secure broadcast networks.

The participants are organized as the leaves of a tree. Leaves for which there is no participant are assigned a default value. Each participant hashes together all its documents along with the previous round's timestamp to produce its value. The value of each node of the tree is the hash of its descendants. Each participant stores as its timestamp certificate its value, its ancestor and its ancestor's siblings. [5]

6 One Way Accumulators

In the previous section we looked at a system which could be built using secure broadcast networks and one way functions. If we add in the RSA assumption ([21]) we can build a system that requires only $\mathcal{O}(1)$ storage to hold a certificate. [6]

This system uses one way accumulators ([6]). The idea is to produce a value that, when hashed with the document, produces the day's timestamp. Note that by document we mean a value representing the document which is, most likely a hash value itself. [6]

One way accumulators are a type of hash function that have a quasi-commutative property. This property can be expressed as,

$$f(f(x, y), z) = f(f(x, z), y)$$

. The function

$$f(x, y) = x^y \text{ mod } n$$

is an example of a one way accumulator if n is hard to factor. [6]

In this protocol all the documents are hashed together using one way accumulators to produce the round's timestamp. Each participant stores as his certificate his document and the hash, again using one way accumulators, of all the other documents. To verify a certificate, one merely needs to hash the document with the hash of the other documents together to form the day's timestamp. [6]

In order to avoid attacks based on the knowledge of a large number of roots of the timestamp, n should be of the form pq where $p = 2p' + 1$ and $q = 2q' + 1$. p, q, p', q' are all prime. This gives the number rigidity and it can be shown that unless you can compute a root anyway, you gain no additional advantage in knowing a number of the roots. [6]

The way such attacks work is through a combination of knowing the roots from other documents being timestamped and choosing document(s) that are the product of many small primes during the formation of the timestamp, the forger hopes that some combination of the roots, when multiplied, will produce the document that forger hopes to forge. As noted above, this method of finding collisions is not feasible for suitably chosen n , ie: n is a product of two large primes and $\phi(n)$ has large prime factors. [6, 20]

7 Agreement Protocols

The Byzantine Generals problem is one of reaching agreement in the presence of faults. Examples of such faults may include the failure of dishonest participants to adhere to protocol, the spurious operation of a failed component, or the breakdown of a communication link.

Abstractly illustrated, this classic problem is most applicable to the areas of distributed trust and decision making. The original solutions to this problem allow a super-majority of well-behaved participants to reach definite agreement as to a reasonable solution to a problem notwithstanding the behavior of dishonest participants. [4]

We examine a contemporary solution to the problem which was developed by Lamport, Shostak, and Pease. [13]

Consider several divisions of the Byzantine army, each commanded by an individual general. These generals, camped around a single enemy city, must collectively decide upon a common plan of action attack or retreat.

However, some number of these generals may be traitors. Traitorous generals will use any means at their disposal to prevent such collective agreement, or to attempt to cause a bad decision to be made.

Each general formulates his own opinion as to which decision to make, and this opinion is distributed to each other general. We let v_i represent the opinion distributed by the i 'th general. [13]

Communicating only by message, each honest general will execute the same decision-making algorithm to reach an ultimate conclusion. [4]

We must ensure that all generals reach the same conclusion. Also, we must prevent a small number of traitors from exercising undue influence on the decision-making process.

To satisfy these conditions, constraints are formulated which dictate algorithm requirements: 1. any two loyal generals must use the same value of

v_i , and 2. if the i 'th general is loyal, then the value that he sends must be used by every loyal general for v_i . [13]

These constraints are not trivial to satisfy, as a traitorous general will resort to sending different information to different generals, masquerading as other generals, and dropping or altering in-transit messages, all in an attempt to cause different generals to reach different conclusions, reach a bad conclusion, or fail to come to agreement. [4]

Given only oral message passing capabilities, solutions to this problem exist which withstand up to m traitors given $3m + 1$ total generals. We assume that every message that is sent is delivered correctly, that the receiver of a message knows who sent it, and that the absence of a message can be detected. [13, 18]

Subsequent work by Rompel has demonstrated that one way functions are equivalent to secure digital signature schemes [19]. One-way functions are an assumption of both the tree-based timestamping system presented in section 5 and the system of section 6 based on one-way accumulators.

Since we enjoy cryptographic benefits, and since we treat this topic mainly to demonstrate the existence of secure broadcast communication channels, we assume that our communications protocol meets both authentication and integrity constraints with respect to our messages.

The introduction of message authentication and integrity constraints allow any number of well-behaved participants to reach agreement, and alleviate the need for a super-majority to guarantee a reasonable solution. [13, 18]

We examine the means by which a single general, general i , certifies and distributes his opinion of value x to all other generals. We refer to this general as the authoritative general with respect to his personal opinion.

Authoritative general i signs his message x , which we denote as $x : i$, and distributes it to each other general. Each other general instantiates a list, V_i , of all opinions originating from authoritative general i . Then, each general j adds his signature to this message, now denoted as $v : i : j$, and redistributes it to all other generals. [13, 2]

In this manner, each general will receive the opinion of authoritative general i first from general i himself, and subsequently from each other general. Clearly, these opinions should all have the same value x . In the case where the authoritative general is a traitor, however, we may find that he has sent different opinions to different generals.

Whenever any general j receives a message in the form $x : i : g_1 : \dots : g_k$ containing a value x not a member of his list V_i , he adds the value x to his

list V_i , adds his signature to the message, and redistributes to each general not $g_1 \dots g_k$. [13, 2]

This process should proceed within a specified timeframe, and a timeout will be employed so as to disallow a traitorous general from preventing agreement by withholding an opinion. [13]

Each loyal general, in this manner, is guaranteed to ultimately receive the exact same set of information regarding the opinion of any particular general. This holds even if that authoritative general sent opinion A to some, opinion B to others, and no message at all to still others. [13, 18]

Each loyal general having the same list, V_i , of opinions issued by authoritative general i , will execute the same algorithm of choice. In the event that authoritative general i is a loyal general, the list V_i will necessarily contain only one value, and the algorithm will yield that single value. [13]

In the event that the list V_i contains multiple values, clearly general i is a traitor. In this case, the particular details of the algorithm are arbitrary, excepting that it must be deterministic. An example algorithm may be to sort V_i , taking the median value. This algorithm should also specify a default value when presented with an empty list V_i . Another alternative is to simply discard the opinion of a traitorous general identified in this manner. [13, 4]

At this point, we have specified how each loyal general will have reached the same conclusion as to the opinion of another particular general. We extend this specification to apply to all generals in our system. We consider each general to be the authoritative general with respect to his own opinion.

We say that each loyal general in our system will create a new list, v , containing the accepted value of the opinion of all generals in the system.

We now describe a manner by which each loyal general will reconcile the list of individual opinions into a final determination. Again, each loyal general acts independently, but executes the same algorithm given the same input. The choice of a particular algorithm is arbitrary, but clearly it should tend to favor the most common opinion. [13, 4] This is so as to prevent a small number of traitors from unduly influencing the final determination. An example algorithm would be the majority vote function, augmented with well-defined behavior when presented with multi-modal data or the empty list. [13]

8 Conclusions

We have examined a number of timestamping schemes and investigated the underlying assumptions. The underlying assumption for two of the protocols, secure broadcast networks, is a reasonable one (see section 7). The other assumptions, one way functions for most of the schemes and the RSA problem for one way accumulators seem to be reasonable. More research needs to be done to determine the validity of the RSA assumption. [8, 9, 14, 15] This should be considered not only with the referenced factoring algorithms, some of which are subexponential, but with discrete logarithm algorithms [1] because factoring reduces to the composite discrete logarithm problem [10, 22].

If the assumptions are correct, then the most efficient system for a large number of participants who timestamp many documents seems to be the protocol based on one way accumulators. This is based on a combination of the asymptotic analysis of the storage requirements and the effort and cooperation required to verify a certificate.

References

- [1] Leonard M. Adleman and Jonathon DeMarrais. A subexponential algorithm for discrete logarithms over all discrete fields. In Douglas R. Stinson, editor, *Advances in Cryptology - Crypto '93*, pages 147–158. Springer-Verlag, 1993.
- [2] S. Amitanand, I. Sanketh, K. Srinathant, V. Vinod, and C. Pandu Rangan. Distributed consensus in the presence of sectional faults. In Borowsky and Rajsbaum [7], pages 202–210.
- [3] Robert L. Ashenurst, editor. *Communications Of the ACM Volume 21, Issue 2*, New York, NY, Feb. 1978. ACM Press.
- [4] Michael Barborak, Anton Dahbura, and Minoslaw Malek. The consensus problem in fault-tolerant computing. *ACM Comput. Surv.*, 25(2):171–220, 1993.
- [5] Benaloh and de Mare. Efficient broadcast timestamping. Technical report, Clarkson University, 1992.

- [6] Benaloh and de Mare. One-way accumulators: A decentralized approach to digital signatures. In Helleseeth [12], pages 274–285.
- [7] Elizabeth Borowsky and Sergio Rajsbaum, editors. *Proceedings of the twenty-second annual Symposium on Principles of Distributed Computing*. ACM Press, 2003.
- [8] David Coppersmith. Modifications to the number field sieve. In *Journal of Cryptology Vol 6.*, pages 169–180, 1993.
- [9] David Coppersmith. Small solutions to polynomial equations, and low exponent RSA vulnerabilities. In *Journal of Cryptology Vol 10*, pages 233–260, 1997.
- [10] Michael deMare and Josh Benaloh. Cryptographic tools based on composite discrete log. Technical report, Clarkson University Department of Math and Computer Science, 1992.
- [11] Haber and Stornetta. How to timestamp a digital document. In Menezes and Vanstone [16], pages 437–455.
- [12] Tor Helleseeth, editor. *Advances In Crptology – Eurocrypt ’93*. Springer-Verlag, 1993.
- [13] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.
- [14] A. Lenstra and H.W. Lenstra. *Lecture Notes in Mathematics, Vol 1554*. Springer-Verlag, 1994.
- [15] H.W. Lenstra. Factoring integers with elliptic curves. In *Annals of Mathematics, 126*, pages 126:649–673, 1987.
- [16] Alfred Menezes and Scott Vanstone, editors. *Advances In Cryptology – Crypto ’90*. Springer-Verlag, 1990.
- [17] Harriet Ortiz, editor. *Proceedings of the twenty-second annual ACM Symposium on Theory of Computing*. ACM Press, 1990.
- [18] M. Pease, R. Shostak, and L. Lamport. Reaching agreement in the presence of faults. *J. ACM*, 27(2):228–234, 1980.

- [19] J. Rompel. One-way functions are necessary and sufficient for secure signatures. In Ortiz [17], pages 387–394.
- [20] Adi Shamir. On the generation of cryptographically strong pseudo-random sequences. In *Proceedings ICALP*, 1981.
- [21] R. Rivest A. Shamir and L. Adleman. A method for obtaining digital signatures and public key cryptosystems. In Ashenurst [3], pages 120–126.
- [22] Shmuely. Composite Diffie-Hellman public-key generating systems are hard to break, TR 356. Technical report, Computer Science Dept, Technion Institute, 1985.