Efficient Broadcast Time-Stamping (Extended Abstract)

Josh Benaloh
Clarkson University

Michael de Mare
Clarkson University

April 21, 1992

Efficient Broadcast Time-Stamping (Extended Abstract)

Abstract

Even using an authenticated synchronous broadcast model, the task of unforgeably time-stamping digital documents still presents some problems. It is simply not practical to assume that all participants will record and store everyone else's documents so that creation times can be verified. This paper presents a time *and* space efficient method for accumulation of time-stamped documents. Whenever a question arises, a claimant can convince any "honest" participant of the time that its document was posted.

1 Introduction

The problem of unforgeably time-stamping documents has long been believed to be "open and shut". Given any unpredictable stream of data (e.g. sun-spot activity), one can digitally sign a document together with a sufficiently long portion of the stream at the time of signing. If the document is later revealed, it is plain that it must have been signed at a time after the relevant portion of the stream was generated. This method is commonly used in hostage photographs where a hostage is shown holding a newspaper in order to demonstrate that the hostage was alive on the date the newspaper was printed.

It has, until recently, been equally plain to researchers that the problem of showing that an event occurred *before* a certain time was impossible since it would involve requiring the prover to forget information. (A photograph of a hostage holding the Magna Carta does not provide compelling evidence that the hostage was being held in the year 1215.) Recently, however, Haber and Stornetta ([HaSt90]) show that in a reasonable model, forward *and* backward time-stamping are both possible! They offer several solutions, but all require at least one of the following: a central authority, substantial cooperation among participating agents to validate a time-stamp, substantial cooperation to post a document, or a substantial amount of storage.

This paper presents a solution to the time-stamping problem in which none of the above are required. For simplicity, the communication model used is that of authenticated broadcast channels. Since such channels can be simulated by a variety of more easily realizable methods (see, for example, [CGMA85], [Fisc83], [BenO83], or [Rabi83]), authenticated broadcast channels serve as a useful communication primitive on which to develop interactive protocols.

The model used in this work is somewhat different from that used by Haber and Stornetta. We assume that our time-stamping protocols run in fixed time periods (rounds) and allow a claimant to efficiently demonstrate to any challenger the round in which a document was stamped. Thus, unlike Haber and Stornetta, we cannot guarantee that our protocols will, for every pair of documents, determine which was stamped first. However, in exchange for this loss of fine discrimination, we are able to eliminate other requirements and substantially improve efficiency.

The actual best choice for the length of a round would be dependent on the level of usage of the system. It could be an hour, a day, a week, or any other suitable time period. Shortening the period increases the complexity. However, it does not seem unreasonable that, for example, two claims submitted to a patent office on the same day be regarded as having arrived together. Note that the document which is stamped could be digitally signed, encrypted, or hashed with the authors' names (perhaps to be revealed at some later time), so that it would not be possible for a dishonest agent to duplicate the contents of a document and send a rival claim the same day.

2 Definitions

We begin by giving a formal definition of time-stamping.

Definition A *time-stamping system* with security parameter N consists of a set of participants $P = \{p_1, p_2, \ldots, p_m\}$ which are synchronous communicating processes together with a triple of protocols $(\mathcal{S}, \mathcal{C}, \mathcal{V})$. The *stamping* protocol $\mathcal{S}$ proceeds in rounds and allows each participant to *post* one or more messages during that round. The *claimant* protocol $\mathcal{C}$ can at any subsequent time be run by a participant who posted a message to convince any "honest" participant running the *verification* protocol $\mathcal{V}$ of the round in which the message was actually posted. Finally, there is no polynomial P such that one or more "dishonest" participants running substitute protocols $\mathcal{S}'$ and $\mathcal{C}'$ can, with probability $1/P(N)$, falsely convince an honest verifier running protocol $\mathcal{V}$ that a message was submitted during a given round.

This definition is very broad since it allows any model of communication and does not require symmetry among the protocols (since a protocol may require a participant to use its own index to determine what action to take). Some of the protocols may not require any interaction at all. The model attempts to capture the notion that time-stamping can continually be performed using protocol $\mathcal{S}$. If it becomes necessary to demonstrate a time-stamp at some later time, protocols $\mathcal{C}$ and $\mathcal{V}$ are provided to allow a claimant to demonstrate a time-stamp to a verifier.

As will be seen in the next section, the definition admits many simplistic solutions to the time-stamping problem. However, the simplistic solutions all prove to be inefficient, and a slightly more sophisticated protocol will be given which both satisfies the definition and gives reasonable efficiency.

One of the fundamental tools that will be used for our time-stamping systems is that of *one-way hash functions*.

Definition A family of *one-way hash functions* is an infinite set of functions $\{h_\ell\}$ such that the functions $h_\ell : \{0,1\}^* \to \{0,1\}^\ell$ have the following properties:

1. There exists a polynomial P such that for each integer ℓ, $h_\ell(x)$ is computable in time $P(\ell, |x|)$ for all $x \in \{0,1\}^*$.

2. There is no polynomial P such that there exists a probabilistic polynomial time algorithm which, for all sufficiently large ℓ, will when given ℓ and some $x \in \{0,1\}^*$, find an $x' \in \{0,1\}^*$ such that $h_\ell(x) = h_\ell(x')$ with probability greater than $1/P(\ell)$ when x is chosen uniformly among all elements of $\{0,1\}^{|x|}$.

By applying a one-way hash function to a document, it becomes computationally infeasible to find another document which hashes to the same value. For this reason, Haber and Stornetta utilize one-way hash functions in their schemes as do we here.

It follows from the above definition that a family of one-way hash functions is itself a family of one-way functions. Work by Naor and Yung ([NaYu89]) and by Rompel ([Romp90]) has shown that one-way hash functions exists if and only if one-way functions exist which, in turn, exist if and only if secure signature schemes exist. It has also been shown ([ILL89]) that the existence of one-way functions is equivalent to the existence of secure pseudo-random number-generators.

3 Simple Broadcast-Based Protocols

There are several simple protocols which will satisfy the definition but are not desirable because they require excessive storage or cooperation. We will briefly describe them for completeness.

3.1 Remember Everything

This protocol simply requires each participant to broadcast (perhaps in encrypted form) any document it wants time-stamped. All the other participants simply remember all posted documents together with the time of posting. To verify a time-stamp, any honest participant can simply go back through its archives, find the document, and (if it's present) check its posting time to verify that the document was indeed posted at the claimed time.

In this case, the stamping protocol $\mathcal{S}$ simply requires broadcast of any message to be stamped and a recording of all broadcasts, the claimant protocol $\mathcal{C}$ is null, and the verification protocol $\mathcal{V}$ simply requires a look-up in local memory.

While this is simple, and any machine can post any document at any time, the space required is unacceptable, as every machine must permanently store every document.

3.2 Chaining

The above system can be made quite storage efficient by chaining everything using one-way hash functions in a manner similar to a method proposed by Haber and Stornetta ([HaSt90]). When a participant wants to time-stamp a message, it computes a hash of its message together with the most recently posted hash value. This new hash value is then posted for possible use by others. A participant need only remember, for each message it posts, the message, and the hash of the previous document. The current (most recent) hash value is always saved by all participants. This describes the $\mathcal{S}$ protocol.

To verify a time-stamp, one participant may have to enlist the help of many other participants. The entire chain of hash values from the claimed message through to some hash value known by the verifier (if necessary, the current hash) must be re-hashed and verified. Thus, protocol $\mathcal{C}$ requires a claimant to obtain (from other participants) all intervening messages, and protocol $\mathcal{V}$ requires the verifier to re-hash every message in this sequence to make sure that a match is obtained.

This approach is very storage efficient, requiring each participant to remember only its own documents and one hash value per document it posts. The problem is that it requires a very high degree of cooperation; even one participant refusing to produce its documents could destroy the entire verification process. This is clearly unacceptable, since the system must still function when participants within it are hostile to one another.

3.3 A Flat System Using Rounds

Using rounds one can construct a scheme which does not require extensive cooperation and where the storage for participants is not substantial during times when they don't have documents to post. During each round, all the participants broadcast their documents to be stamped. All of these documents are hashed together with the previous round's time-stamp to produce a single value which serves as the current round's time-stamp. This $\mathcal{S}$ protocol requires every participant that produced a document that round to remember *all* the documents for that round. All participants (whether or not they are submitting documents that round) must remember the round's time-stamp.

To demonstrate that a document is properly stamped, the claimant protocol $\mathcal{C}$ requires a participant to produce all documents submitted the same round. The verification protocol $\mathcal{V}$ simply requires a participant to compute the hash of these messages and check that it matches the round's time-stamp.

Since the cost of storage is levied on only those participants actually producing documents to be tagged, this is better then the "Remember Everything" model, and less cooperation is required then in the "Chaining" model.

3.4 A Scheme which Assumes Associativity

If one is willing to assume the existence of one-way hash functions which are associative, then it is easy to make the "flat" scheme above extremely space-efficient.

Let x_i denote the document set submitted by participant p_i in a given round. The process of hashing together *all* of the document sets of all of the participants could be completed by starting with y_0 as the previous round's time-stamp and forming and $y_i = h(y_{i-1}, x_i)$ for $1 \leq i \leq m$. The resulting hash value y_m would then be used as the current round's time-stamp.

It is clear that to reproduce the round's time-stamp as a hash which includes x_i, participant p_i, need only save y_{i-1} and x_j for $i \leq j \leq m$ (rather than saving all x_j). However, for most p_i, this will not result in more than a small constant factor improvement in the storage requirements.

If, however, the one-way hash function h were associative, then p_i could simply store y_{i-1}, x_i, and a value z_{i+1} which represents the cumulative hash of all of the document sets x_j for $i < j \leq m$. Since the order of application of the hashes would not matter, the round's time-stamp would be equal to $h(h(y_{i-1}, x_i), z_{i+1})$ regardless of i.

In fact, if h were also commutative, then it would only be necessary for p_i to store one value along with x_i: the cumulative hash of all document sets x_j for $j \neq i$. In either case, the number of fixed sized hash values which must be stored by a participant in order to demonstrate the validity of its documents in a given round would be constant.

Unfortunately, although this scheme is very efficient and provides for simple verification, the assumption of an associative one-way hash function may be unrealistic. The next section describes a scheme which, although not quite as efficient, depends only on the existence of one-way hash functions which, in turn, is known to be equivalent to the existence of arbitrary one-way functions.

4 A Hierarchical System that Proceeds in Rounds

We now describe a time-stamping system which is both time and space efficient and depends only on the assumption that one-way functions (and hence one-way hash functions) exist. The method is similar to the "Flat" system of the previous section, but by structuring things hierarchically, it is possible to eliminate the bulk of the storage requirements. The auxiliary storage required of each machine in this scheme is only $\mathcal{O}(\log m)$ per round where m is the number of participants.

The idea is to structure things such that, as before, the time-stamp for a round depends on the time-stamp for the previous round and on *all* documents submitted in the current round. The difference is that it will no longer be necessary to store *all* of a round's documents to prove any one document's time-stamp. The m participants in the time-stamping system are organized into a hierarchy with each participant placed at a leaf of a complete K-ary tree.

4.1 The Stamping Protocol $\mathcal{S}$

1. In round k, each participant p_i gathers *all* documents it wishes to stamp that round and hashes them together with the previous round's hash value R_{k-1} to produce its hash value $h_{k,i}$. Each participant then broadcasts its $h_{k,i}$. (Defaults can be substituted for R_0 and for the hash values of any participants which have no messages to stamp in a given round.)

2. The hash values $h_{k,i}$ are associated with the leaves of the K-ary tree into which the participants are organized. (Default values are used for leaves with which there is no participant associated.) The value associated with an internal node of the tree is the hash of the values of its K children. The value associated with the root is denoted by R_k, and this value is called the time-stamp of the round.

3. Each p_i independently computes (from the broadcast information and the fixed defaults), *all* of the tree values for round k. However, only the values associated with ancestors of a participant's leaf *and its ancestors' siblings* are saved.

4.2 The Claimant Protocol $\mathcal{C}$

To demonstrate that a document was submitted in round k,

1. The claimant p_i produces *all* documents which it submitted in round k (note that a document may be encrypted or pre-hashed so that revealing a document does not necessarily reveal its contents).

2. The claimant then produces all of the hash values (of ancestors and their siblings) which it stored from round k.

4.3 The Verification Protocol $\mathcal{V}$

To verify a claim that a document was stamped in round k,

1. The verifier checks that the documents released by the claimant p_i together with the time-stamp R_{k-1} for round $k-1$ hash to the value $h_{k,i}$ supplied by the claimant.

2. The verifier then checks that the hash values produced by the claimant (starting from $h_{k,i}$) do indeed (following the tree towards the root) ultimately hash to the time stamp R_k for round k.

If all of the above hashes match, the verifier accepts the time-stamped document. Otherwise, the time-stamp is rejected.

Note that it is not essential that the verifier store the time-stamp for every round. If the verifier does not know the time-stamp for a given round, a claimant can provide information

to chain together the time-stamps for two consecutive rounds. This can be continued until a round is found for which the verifier knows the time-stamp. The current round's time-stamp should always be stored by all participants.

It turns out, that for the protocols described above, a branching factor of $K = 3$ in the hierarchy is optimal. However, a slightly refined set of protocols in which participants *do not* save the hash values of their ancestor nodes but instead save *only* their ancestors' siblings' values (which — together with the relevant document hash $h_{k,i}$ — is sufficient to reconstruct the missing values) would yield an optimal branching factor of $K = 2$.

5 Proof of the Hierarchical Scheme

The fact that the hierarchical scheme satisfies the definition of a time-stamping system stems from the assumption that given a one-way hashed document, it is infeasible to find another document which has the same hash. Hashes are chained so as to make it infeasible to match outputs by matching inputs.

Most of the conditions of the definition of a time-stamping system are trivially satisfied by the hierarchical scheme. For example, if the inputs used by a claimant to compute a hash value are released for verification, there is no question that an honest verifier will be able to compute the same hash value. The only condition which requires effort is the assertion that no alternate claimant protocols $\mathcal{S}'$ and $\mathcal{C}'$ will be able to falsely convince an honest verifier running protocol $\mathcal{V}$ of an allegedly stamped document.

Theorem 1 *The hierarchical protocols $(\mathcal{S}, \mathcal{C}, \mathcal{V})$ when used with a one-way hash function h_N chosen (according to security parameter N) from a family of one-way hash functions $\{h_\ell\}$ constitutes a time-stamping system.*

Proof:(sketch)

Assume that $(\mathcal{S}, \mathcal{C}, \mathcal{V})$ does not constitute a time-stamping system. Then, by the arguments which precede this theorem, it must be the case that there exist alternate protocols $\mathcal{S}'$ and $\mathcal{C}'$ which can fool protocol $\mathcal{V}$ with unacceptably high probability. This would imply the existence of a polynomial P such that $\mathcal{V}$ will be fooled with probability $1/P(N)$.

In order for $\mathcal{S}'$ and $\mathcal{C}'$ to fool $\mathcal{V}$, they must produce a document which is alleged to have been stamped in a given round and a sequence of values which includes this document which will hash to the known time-stamp for the round in question.

Consider the sequence of values which originally led up to the round's time-stamp and compare it to the "false" sequence released by $\mathcal{C}'$. Let y denote the first value where the two sequences coincide. (Such a y must exist since the final values of the sequences must coincide.) Since the two sequences must differ (else the document stamp would be valid), it must be the case that y is computed both as $y = h(x)$ and as $y = h(x')$ where x is taken from the original sequence and x' is taken from the false sequence. The value x' is either the document which the dishonest participants want to falsely stamp or it is some derived hash value dependent on this value. In either case, it is a fixed value for which x must

have been chosen to yield a collision. (Note that a one-way hash function does *not* preclude the possibility of finding colliding pairs of inputs. It only requires that a collision cannot be found for a given input.) Thus, even though x may have been derived from a document chosen so as to facilitate collisions, the distribution of the value x' which is derived through a constant number of iterations of a one-way hash function cannot be "too far" from uniform.

This implies that an efficient algorithm has been found to produce a collision with a (perhaps repeated) hash of a given input. In other words, an algorithm has been found which will take one-way hash functions $h_1, h_2, \ldots, h_j$ and a value x and find a value x' such that $h_1(h_2(\ldots h_j(x)\ldots)) = h_1(h_2(\ldots h_j(x')\ldots))$. However, this would imply that at least one of the h_i is in fact *not* one-way. ∎

The fact that the hierarchical scheme satisfies the claimed resource requirements follows immediately from the protocols themselves.

6 Conclusions and Extensions

This paper has described several methods of unforgeably time-stamping documents in a synchronous broadcast environment. While trivial solutions to this problem exist, they either have unacceptably large storage requirements or unacceptably large communication and cooperation requirements. By structuring time into rounds of appropriate length, efficient protocols can be found based only on the assumption of the existence of one-way functions.

The optimal choice for the length of a round cannot be described quantitatively. There is a trade-off. If the length of a round is increased, the storage required to maintain the round time-stamps and auxiliary data will decrease. However, long rounds reduce the fine descrimination of document times since documents submitted during the same round are considered to be simultaneous. It seems that choosing the length of the round to be such that the average number of documents submitted per round is about one per participant may be a good trade-off. In this way, the descrimination is still quite fine while the overhead is not overwhelming. In many practical applications, it may be desirable to choose the length of a round to be a day, since this is already a common unit for time-stamping. Post offices, banks and other places of business typically use the date as the time-stamp for common transactions.

Several problems remain to be addressed. Is it possible to further improve the efficiency of these schemes? Is the existence of associative one-way hash functions equivalent to the existence of arbitrary one-way hash functions? Perhaps, the most widely applicable open question raised by this work is that of whether or not the existence of one-way (hash) functions implies the existence of one-way hash functions for which finding *any* collision is hard. This latter problem seems quite difficult and would be of significant interest in the area of structural cryptography.

References

[BenO83] **Ben-Or, M.** "Another Advantage of Free Choice: Completely Asynchronous Agreement Protocols." *Proc.* 2nd *ACM Symp. on Principles of Distributed Computing*, Montreal, PQ (Aug. 1983), 27–30.

[CGMA85] **Chor, B.**, **Goldwasser, S.**, **Micali, S.**, and **Awerbuch, B.** "Verifiable Secret Sharing and Achieving Simultaneity in the Presence of Faults." *Proc.* 26th *IEEE Symp. on Foundations of Computer Science*, Portland, OR (Oct. 1985), 383–395.

[Fisc83] **Fischer, M.** "The Consensus Problem in Unreliable Distributed Systems", *Proc. 1983 International FCT-Conference*, Borgholm, Sweeden (Aug. 1983), 127–140. Published as *Foundations of Computation Theory*, ed. by M. Karpinski in *Lecture Notes in Computer Science*, vol. 158, ed. by G. Goos and J. Hartmanis. Springer-Verlag, New York (1983).

[HaSt90] **Haber, S.** and **Stornetta, W.** "How to Time-Stamp a Digital Document." *Advances in Cryptology — Crypto '90.* to appear in Springer-Verlag *Lecture Notes in Computer Science.*

[ILL89] **Impagliazzo, R.**, **Levin, L.**, and **Luby, M.** "Pseudorandom Generation from One-Way Functions." *Proc.* 21st *ACM Symp. on Theory of Computation*, Seattle, WA (May 1989), 12–24.

[NaYu89] **Naor, M.** and **Yung, M.** "Universal One-Way Hash Functions and their Cryptographic Applications." *Proc.* 21st *ACM Symp. on Theory of Computation*, Seattle, WA (May 1989), 33–43.

[Rabi83] **Rabin, M.** "Randomized Byzantine Generals." *Proc.* 24th *IEEE Symp. on Foundations of Computer Science*, Tucson, AZ (Nov. 1983), 403–409.

[Romp90] **Rompel, J.** "One-Way Functions are Necessary and Sufficient for Secure Signatures." *Proc.* 22nd *ACM Symp. on Theory of Computation*, Baltimore, MD (May 1990).