Improving the Efficiency and Reliability of Digital Time-Stamping

Dave Bayer*
Barnard College
Columbia University
New York, N.Y. 10027 U.S.A.
dab@math.columbia.edu

Stuart Haber
Bellcore
445 South Street
Morristown, N.J. 07960 U.S.A.
stuart@bellcore.com

W. Scott Stornetta
Bellcore
445 South Street
Morristown, N.J. 07960 U.S.A.
stornetta@bellcore.com

March 1992

Abstract

To establish that a document was created after a given moment in time, it is necessary to report events that could not have been predicted before they happened. To establish that a document was created before a given moment in time, it is necessary to cause an event based on the document, which can be observed by others. Cryptographic hash functions can be used both to report events succinctly, and to cause events based on documents without revealing their contents. Haber and Stornetta have proposed two schemes for digital time-stamping which rely on these principles [HaSt 91].

We reexamine one of those protocols, addressing the resource constraint required for storage and verification of time-stamp certificates. By using trees, we show how to achieve an exponential increase in the publicity obtained for each time-stamping event, while reducing the storage and the computation required in order to validate a given certificate.

We show how time-stamping can be used in certain circumstances to extend the useful lifetime of different kinds of cryptographic certifications of authenticity, in the event that the certifying protocol is compromised. This can be applied to digital signatures, or to time-stamping itself, making the digital time-stamping process renewable.

*Partially supported by NSF grant DMS-90-06116.

1 Introduction

Causality fixes events in time. If an event was determined by certain earlier events, and determines certain subsequent events, then the event is sandwiched securely into its place in history. Fundamentally this is why paper documents have forensic qualities allowing them to be accurately dated and examined for signs of after-the-fact tampering. However, documents kept in digital form need not be closely tied to any physical medium, and tampering may not leave any tell-tale signs in the medium.

Could an analogous notion of causality be applied to digital documents to correctly date them, and to make undetected tampering infeasible? Any solution would have to time-stamp the data itself, without any reliance on the properties of a physical medium, and would be especially useful and trustworthy if the date and time of the time-stamp could not be forged.

In [HaSt 91], Haber and Stornetta posed this problem, and proposed two solutions. Both involve the use of cryptographic hash functions (discussed in §2 below), whose outputs are processed in lieu of the actual documents. In the *linking* solution, the hash values of documents submitted to a time-stamping service are chained together in a linear list into which nothing can feasibly be inserted or substituted and from which nothing can feasibly be deleted. This latter property is insured by a further use of cryptographic hashing. In the *random witness* solution, several members of the client pool must date and sign the hash value; their signatures form a composite certification that the time-stamp request was witnessed. These members are chosen by means of a pseudorandom generator that uses the hash of the document itself as a seed. This makes it infeasible to deliberately choose which clients should and should not act as witnesses.

In both of these solutions, the record-keeping requirements per time-stamping request are proportional to the number of (implicit) observers of the event. In §3 below we address the following problem: What if an immense flood of banal transactions want their time-stamps to become part of the historical record, but history just isn't interested? We propose to merge many unnoteworthy time-stamping events into one noteworthy event, using a tournament run by its participants. The winner can be easily and widely publicized. Each player, by remembering a short list of opponents, can establish participation in the tournament. We do this by building trees in place of the linked list of the linking solution, thus achieving an exponential increase in the number of observers. Such hash trees were previously used by Merkle [Merk 80] for a different purpose, to produce authentication certificates for a directory of public enciphering keys.

There are several ways in which a cryptographic system can be compromised. For example, users' private keys may be revealed; imprudent choice of key-lengths may be overtaken by an increase in computing power and improved algorithmic techniques may render feasible the heretofore intractable computational problem on which the system is based. In §4 below we show how time-stamping can be used in certain circumstances to extend the useful lifetime of digital signatures. Applying the same technique to time-stamping itself, we demonstrate that digital time-stamps can be

renewed.

Finally, in §5 we discuss the relationships between the different methods of digital time-stamping that have been proposed.

2 Hash functions

The principal tool we use in specifying digital time-stamping schemes, here as in [HaSt 91], is the idea of a cryptographic hash function. This is a function compressing digital documents of arbitrary length to bit-strings of a fixed length, for which it is computationally infeasible to find two different documents that are mapped by the function to the same *hash value*. (Such a pair is called a *collision* for the hash function.) Hence it is infeasible to fabricate a document with a given hash value. In particular, a fragment of a document cannot be extended to a complete document with a given hash value, unless the fragment was known before the hash value was created. In brief, a hash value must follow its associated document in time.

There are practical implementations of hash functions, for example those of Rivest [Riv 90] and of Brachtl, *et al.* [BC$^+$ 88], which seem to be reasonably secure.

In a more theoretical vein, Damgard defined a family of *collision-free hash functions* to be a family of functions $h : \{0,1\}^* \to \{0,1\}^l$ compressing bit-strings of arbitrary length to bit-strings of a fixed length l, with the following properties:

1. The functions h are easy to compute, and it is easy to pick a member of the family at random.

2. It is computationally infeasible, given a random choice of one of these functions h, to find a pair of distinct strings x, x' satisfying $h(x) = h(x')$.

He gave a constructive proof of their existence, on the assumption that there exist one-way "claw-free" permutations [Dam 87]. For further discussion of theoretical questions relating to the existence of families of cryptographic hash functions (variously defined) see [HaSt 91] and the references contained therein.

In the rest of this paper, we will assume that a cryptographic hash function h is given: either a particular practical implementation, or one that has been chosen at random from a collision-free family.

3 Trees

In the linking scheme, the challenger of a time-stamp is satisfied by following the linked chain from the document in question to a time-stamp certificate that the challenger considers trustworthy. If a trustworthy certificate occurs about every N documents, say, then the verification process may require as many as N steps. We may reduce this cost from N to $\log N$, as follows.

Suppose we combine the hash values of two users' documents into one new hash value, and publicize only the combined hash value. (We will consider a "publicized"

value to be trustworthy.) Either participant, by saving his or her own document as well as the other contributing hash value, can later establish that the document existed before the time when the combined hash value was publicized.

More generally, suppose that N hash values are combined into one via a binary tree, and the resulting single hash value is widely publicized. To later establish priority, a participant need only record his own document, as well as the $\lceil \log_2 N \rceil$ hash values that were directly combined with the document's hash value along the path to the root of the tree. In addition, along with each combining hash value, the user needs to record its "handedness," indicating whether the newly computed value was placed before or after the combining hash value. Verification consists simply of recomputing the root of the tree from this data.

Once hashing functions are chosen, such a scheme could be carried out like a world championship tournament: Heterogeneous local networks could govern local subtrees under the scrutiny of local participants, and regional "winners" could be combined into global winners under the scrutiny of all interested parties. Global communication facilities are required, and a broadcast protocol must be agreed upon, but no centralized service bureau need administer or profit from this system. For example, given any protocol acceptable separately to the western and eastern hemispheres for establishing winners for a given one-hour time period, the winners can be broadcast by various members of the respective hemispheres, and anyone who wants to can carry out the computations to determine the unique global winner for that time period. Winners for shorter time periods can similarly be combined into unique winners for longer time periods, by any interested party.

At a minimum, daily global winners could be recorded in newspaper advertisements, to end up indefinitely on library microfilm. The newspaper functions as a widely available public record whose long-term preservation at many locations makes tampering very difficult. An individual who retains the set of values tracing the path between his document and the hash value appearing in the newspaper could establish the time of his document, without any reliance on other records. Anyone who wishes to be able to resolve time-stamps to greater accuracy needs only to record time-stamp broadcasts to greater accuracy.

4 Using time-stamping to extend the lifetime of a threatened cryptographic operation

The valid lifetime of a digitally signed document can be extended with digital time-stamping, in the following way. Imagine an implementation of a particular digital signature scheme, with a particular choice of key lengths, and consider a plaintext document D and its digital signature σ by a particular user. Now let the pair (D, σ) be time-stamped. Some time later the signature may become invalid, for any of a variety of reasons, including the compromise of the user's private key, an increase in available computing power making signatures with keys of that length unsafe, or the discovery of a basic flaw in the signature scheme. At that point, the document-

signature pair becomes questionable, because it may be possible for someone other than the original signer to create valid signatures.

However, if the pair (D, σ) was time-stamped at a time before the signature was compromised, then the pair still constitutes a valid signature. This is because it is known to have been created at a time when only legitimate users could have produced it. Its validity is not in question even though new signatures generated by the compromised method might no longer be trustworthy.

The same technique applies to other instances of cryptographic protocols. In particular, the technique can be used to renew the time-stamping process itself. Once again, imagine an implementation of a particular time-stamping scheme, and consider the pair (D, C), where C is a valid time-stamp certificate (in this implementation) for the document D. If (D, C) is time-stamped by an improved time-stamping method before the original method is compromised, then one has evidence not only that the document existed prior to the time of the new time-stamp, but that it existed at the time stated in the original certificate. Prior to the compromise of the old implementation, the only way to create a certificate was by legitimate means. (The ability to renew time-stamps was mentioned in [HaSt 91] but an incorrect method was given. The mistake of the previous work was in assuming that it is sufficient to renew the certificate alone, and not the document-certificate pair. This fails, of course, if the compromise in question is a method of computing hash collisions for the hash function used in submitting time-stamp requests.)

5 Different methods of time-stamping

To date, three different digital time-stamping techniques have been proposed: linear linking, random witness and linking into trees. What is the relationship between them? Does one supersede the others? Initially, one might think that trees satisfy time-stamping requirements better than the two previously proposed methods, because the tree protocol seems to reduce storage requirements while increasing the number of interested parties who serve as witnesses. But there are other tradeoffs to consider.

First we consider the linking protocol. In certain applications, such as a laboratory notebook, it is crucial not only to have a trustworthy date for each entry but also to establish in a trustworthy manner the exact sequence in which all entries were made. Linear linking of one entry to the next provides the most straightforward means of achieving this.

Next we consider the difference between the random-witness method and the tree method. While trees increase the number of witnesses to a given time-stamping event in proportion to the number of documents time-stamped, they do not guarantee a minimum number of witnesses. Neither do they guarantee that witnesses will retain their records. In contrast, in random witness the effective number of witnesses is the entire population, though only a small fraction are actually involved in any given time-stamping event. Furthermore, the set of signatures computed by the random-witness protocol explicitly creates a certificate which is evidence that a time-stamping

event was widely witnessed. Thus the protocol does not depend for its final validity on witnesses keeping records. Publishing a hash is somewhat analogous to placing an advertisement in the newspaper, as discussed earlier, but with an additional refinement. Like the newspaper ad, it is effectively a widely witnessed event, but in addition it creates a record of the witnessing.

Given these tradeoffs, we imagine that the three methods may be used in a complementary fashion, as the following example illustrates. An individual or company might use linear linking to timestamp its own accounting records, sending the final summary value for a given time period to a service maintained by a group of individuals or parties. This service constructs linked trees at regular intervals. The root of each tree is then certified as a widely viewed event by using the random-witness protocol among the participants. In this way, individual and group storage needs can be minimized, and the number of events which require an official record of witnessing can be greatly reduced.

References

[BC+ 88] B. O. Brachtl, D. Coppersmith, M. M. Hyden, S. M. Matyas, Jr., C. H. W. Meyer, J. Oseas, S. B. Pilpel, and M. Shilling. Data authentication using modification detection codes based on a public one way encryption function. U.S. Patent No. 4,908,861, issued March 13, 1990. (*Cf.* C. H. Meyer and M. Shilling, Secure program load with modification detection code. In *Securicom 88: 6ème Congrès mondial de la protection et de la sécurité informatique et des communications*, pp. 111–130 (Paris, 1988).)

[Dam 87] I. Damgård. Collision free hash functions and public-key signature schemes. In *Advances in Cryptology—Eurocrypt '87*, Lecture Notes in Computer Science, Vol. 304, pp. 203–217, Springer-Verlag (Berlin, 1988).

[HaSt 91] S. Haber, W. S. Stornetta, How to time-stamp a digital document, *Journal of Cryptography*, Vol. 3, No. 2, pp. 99–111 (1991). (Presented at Crypto '90.)

[Merk 80] R. C. Merkle, Protocols for public key cryptosystems. In *Proc. 1980 Symp. on Security and Privacy*, IEEE Computer Society, pp. 122–133 (Apr. 1980).

[Riv 90] R. L. Rivest. The MD4 message digest algorithm. In *Advances in Cryptology—Crypto '90*, Lecture Notes in Computer Science, Vol. 537 (ed. A. J. Menezes, S. A. Vanstone), pp. 303–311, Springer-Verlag (Berlin, 1991).