Some Timestamping Protocol Failures

Mike Just*

Abstract

Protocol failures are presented for two timestamping schemes. These failures emphasize the importance and difficulty of implementing a secure protocol even though there exist secure underlying algorithms, as well as the importance of clearly defining the goals for a protocol. For the scheme of Benaloh and de Mare (Eurocrypt '93), it is shown that although an indication of time can be included during the computation of the timestamp, the verification of the timestamp does not allow for the recovery of this temporal measure. For the scheme of Haber and Stornetta (Journal of Cryptology '91), we demonstrate how a collusion attack between a single user and a timestamping service allows for the backdating of timestamps. This attack is successful despite the claim that the timestamping service need not be trusted. For each of these schemes, we discuss methods for improvement.

Keywords: protocol failure, timestamp, notary, temporal authentication, absolute, relative, non-repudiation, adjudication.

Carleton University, School of Computer Science, Technical Report, TR-97-16, August, 1997

*School of Computer Science, Carleton University, Ottawa, ON, Canada, K1S 5B6, e-mail: just@scs.carleton.ca. Partially supported by an NSERC grant.

1 Introduction

There currently exist many "believed to be strong" cryptographic algorithms. Privacy is obtained through the application of an encryption function (whether it be of the public key or secret key variety). Authenticity can be obtained through the application of keyed hash functions (for the secret key case) or digital signatures (for the public key case) (see [MOV97] for examples).

Failures do not usually occur from the complete break of an algorithm but are more often the result of a poor implementation, i.e., a poor protocol design. The literature is filled with examples of protocols designed using strong cryptographic primitives yet susceptible to the simplest of attacks [Moor88, AnNe95].

The use of a notary is currently receiving a great amount of attention for its application to digital communications. Timestamping is the simplest form of a notary, essentially involving only an authentic appendage of time to a document.[1] Just as with other protocols, a timestamping protocol uses (believed to be) secure primitives such as hash functions and digital signatures and must be designed with clear and secure objectives in mind, e.g., [AbNe94].

1.1 Definitions

The purpose of a timestamping protocol is to assign a temporal ordering to a document y. Just as we can provide for the data integrity and message authentication of y, a timestamping protocol provides for the *temporal authentication* of y. Such a concept is useful, for example, in aiding the provision of non-repudiation of digital signatures. The timestamping of a message signed with a particular private signature key allows for the determination of whether the signed message existed before or after the expiry or reported compromise of the corresponding public verification key.

A protocol provides *data integrity* of a message y if it ensures that the message has not been altered in an unauthorized manner since its latest authorized alteration. *Message authentication (data-origin authentication)* is provided when a party is assured of the source (i.e., entity authorized to alter y) of a given message. Temporal authentication (transaction authentication in [MOV97]) combines message authentication with the notion of *uniqueness* and *timeliness* of messages. Note here that the authentication of the message y will result in the production of a timestamp s which will be some function of both the message y (which itself may be the function of some document) along with a piece of data from which the temporal position of y can be inferred. Thus, although the same y may be stamped many times, each timestamp for y is unique since each will have a unique temporal interpretation (though see special case below). The combination of y with its timestamp, plus any other relevant information comprises the *timestamp capsule* (or simply *time capsule*).

We can view the application of a timestamp as an instance of a larger, ongoing timestamping protocol. Indeed, the uniqueness of a particular stamp is measured in relation to other stamps produced via alternate instances of the protocol. Each

[1] A stronger form of notary might perform such operations as verifying the signature on a submitted document or even verifying the form or content of the signed data.

such instance is referred to as a *round*. More than one document can be stamped during a given round though only a single timestamp is produced. Within a given round, the temporal ordering of the messages is not necessarily important. Thus, the same message submitted twice during the same round may receive an identical stamp associated with each submission. In practice, the length of the round may be determined either by fixing an upper bound of the number of messages that will be jointly stamped in the round or on the amount of time that is allowed to elapse before a stamp is output.

We elaborate on what is meant by the timelineness of messages by distinguishing two types of temporal authentication. *Absolute temporal authentication* is provided if an *absolute timestamp* is associated with a message. An absolute timestamp positions the message at a particular point in time, based upon the time given by a trusted, mutually agreed upon source. *Relative temporal authentication* is provided if a *relative timestamp* is associated with a message. A relative timestamp positions a message at some general point in time relative to messages stamped before and after it. The ordering of these messages is a partial ordering. If every two elements are comparable, the set of stamps are a total order or chain. We refer to this set as a *temporal chain (or temporal order)* since elements of the set are comparable based on their temporal interpretation. *Hybrid temporal authentication* is provided if both an absolute and relative timestamp are associated with a message.

There are two measures by which the temporal ordering (authentication) of y can be verified. The *absolute (temporal) measure* determines the placement of y with respect to a particular time, e.g. determining the time at which a digital signature may have been applied to a given document. The *relative (temporal) measure* determines the placement of y relative to other y', e.g. determining the precedence relationship between two patent claims. We say that a document y' has been *backdated* (backstamped) if a temporal measurement infers that y' was stamped before y when in fact, the timestamp for y' was constructed after the timestamp for y. A similar definition follows for *forward dating* (forward stamping).

1.2 Timestamping Model

Let u represent a user in a distributed system with unique identification ID_u who would like to obtain a timestamp s for a message y. Two components in any timestamping scheme are the *Stamping Protocol (SP)* and the *Challenge/Verification Protocol (CVP)* (similar to those presented by Benaloh and de Mare [BeMa91] and mentioned briefly by Haber and Stornetta [HaSt91]).

Stamping Protocol (SP) Initiated by the user requesting a timestamp, the protocol takes y as input and produces an absolute and/or relative timestamp s.

Challenge/Verification Protocol (CVP) Initiated by a user c challenging the temporal position of a particular y (whose position is defined by the stamp s). In the *Challenge Stage*, user u submits the timestamp capsule consisting of y along with the

stamp s as well as any other relevant information (e.g., x where $y = h(x)$) to c. In the *Verification Stage*, c verifies the authenticity of the temporal stamp s with respect to the message y.

1.3 Outline

In Section 2 we discuss how the inclusion of a time in the production of a timestamp in the scheme of Benaloh and de Mare [BeMa93] is not recoverable during stamp verification. Several methods for rectifying this shortcoming are discussed. Section 3 presents the collusion attack to the scheme of Haber and Stornetta [HaSt91]. We demonstrate how the use of only an absolute temporal measure allows one to backdate documents. Several methods for preventing this attack are discussed.

2 A Decentralized Timestamping Protocol

Benaloh and de Mare [BeMa93] present a protocol which allows the resultant timestamp for a round to be computed in such a manner that an on-line, centralized entity is not required. We demonstrate here that the constructed timestamp does not allow for an absolute temporal measure during its verification, even though such information may be included during stamp construction.

Let $n = pq$ be the product of two primes. The construction of n may be undertaken via a trusted outside source, a special purpose physical device, or a secure multiparty computation (selection of p and q is described in [BeMa93]). During round r, a value x is first agreed upon (e.g. the current date), from which the starting seed $x_0 = x^2 \bmod n$ is obtained. The scheme simply proceeds by having users use modular exponentiation as an accumulator where the y_i's are hashed documents that are submitted within a given round via some sort of multi-party broadcasting. We have the resultant timestamp $a_r = x_0^{y_1 y_2 \cdots y_m} \bmod n$ whereby u_i maintains $\{z_i = x_0^{y_1 \cdots y_{i-1} y_{i+1} \cdots y_m} \bmod n, y_i\}$ as his timestamp capsule. Notice that the size of the partial accumulation, z_i, is independent of the number of submissions made during the round.

2.1 Verification - Where's the Time?

Responding to a challenge by c, u_i would demonstrate that ${z_i}^{y_i} \bmod n$ was indeed equivalent to a_r modulo n (the task of cheating this procedure is discussed in [BeMa93]). However, notice that although an absolute time may have been included in the stamp computation (i.e., by setting x to be the current date), no such temporal measure is available to the challenger for timestamp verification. Even if u_i had x, he could not demonstrate that it contributed to the computation of a_r (unless of course he also stored $y_1, \ldots, y_m$ – but the whole point of the scheme from [BeMa93] is to not have to store all of the submissions).

Benaloh and de Mare present their scheme as advantageous over the "save everything you see" approach in terms of storage. They compare it to the variation whereby

a central authority (with a public key) signs each y_i and returns it to the submitting user. The solution they present here removes the need for a central authority, yet maintains similar storage. However, the scheme is more suited to a closed system. Each user stores every a_r that he contributes to the production of and measures the authenticity of a user's timestamp capsule by comparing ${z_i}^{y_i} \bmod n$ to the value they store.

However, the scheme does not allow for (what we define to be) an appropriate temporal measure of the position of y_i. The most it does is to allow a user to demonstrate (to other users that participated in the same stamp computation) the round that his document was stamped in. We note here that this fact was recognized by Benaloh and de Mare [BeMa93] though the failure to provide a more robust temporal measure was not. Indeed, by suggesting that x may be chosen to represent the current time, it is implied that an absolute measure is recoverable during verification.

2.2 Providing a Temporal Measure

In this section, we consider some ideas for allowing a temporal measure to be obtained through the verification of a timestamp in the Benaloh/de Mare scheme. In an earlier work of Benaloh and de Mare [BeMa91], the authors introduced the concept of rounds. Using rounds is claimed to provide for increased efficiency at the cost of a loss of granularity. The latter is caused by the fact that the same "time" is associated with each document submitted during a given round.

Providing Absolute Temporal Authentication A simple way to allow for an absolute temporal measure during the verificaton protocol is to authenticly store a_r along with t. Use of a hash by computing $a'_r = h(a_r, t)$ and storing a'_r and t allows for increased storage efficiency. However, there may be additional overhead for such a decentralized protocol for the users agree upon a time t.

Providing Relative Temporal Authentication The provision of a relative temporal measure would allow users to demonstrate a precedence ordering for documents submitted in distinct rounds. This will necessitate that users from one round are aware of the timestamp from other rounds (since relative authentication requires a relation between stamps in order to show precedence). This could be accomplished by a widespread publication (e.g., newspaper) of the stamps for each round as suggested by Merkle [Merk80] (though not directly in the context of timestamping) and as well by Bayer, Haber and Stornetta [BHS93], provided that it can be done authenticly.

For example, if a_r is the originally obtained timestamp, we let $a'_r = h(a_r, a_{r-1})$ be the published value where h is a collision resistant hash function. A similar scheme was suggested by Benaloh and de Mare [BeMa91], though there, the linking was done for timestamps produced by an individual user, not the result of a round computation. This is essentially the scheme suggested used by Bayer, Haber and Stornetta [BHS93] for linking their rounds.

The concern with this scheme is that besides having to rely on just number theoretic assumptions (as in the Benaloh and de Mare scheme [BeMa93]), we now also have to rely on the strength of the hash h. Observe that the order of the documents submitted to the original stamp computation is irrelevant, motivating the use of the commutative multiplication operation. Exponentiation is not commutative. This leads us to present the following alternative to any hash-based linking scheme. Once again, a_r is the originally obtained timestamp for round r. We let the published value be a'_r. Definined recursively we have (similar to the multiuser key agreement protocol by Steer et al. [SSDW89])

$$\begin{aligned} a'_1 &= a_1 \\ a'_r &= (a'_{r-1})^{a_r} \bmod n, \text{ for } r \geq 2. \end{aligned}$$

Verification would now involve demonstrating that ${z_i}^{y_i} \bmod n$ was indeed equivalent to a_r modulo n (i.e., providing evidence that the document y_i belongs to a given round) as well as demonstrating that for that particular a_r, $a'_r = (a'_{r-1})^{a_r} \bmod n$ (i.e., providing evidence that a_r was indeed produced after a'_{r-1} and before a'_{r+1} – the latter follows since a'_r is used in the computation of a'_{r+1}). Such a scheme fits quite naturally into the scheme from Benaloh and de Mare [BeMa93], though its security requires further verification.

3 A Centralized Timestamping Protocol

Haber and Stornetta [HaSt91] present the following protocol for timestamping a digital document x. It uses a central timestamping service T that requires no record-keeping. Here, r denotes the rth round, where 1 document is stamped per round.

1. u sends $y_r = h(x_r)$ and $ID_r = ID_u$ to T, where x_r is the original message belonging to u_r that is hashed with the collision resistant hash h for privacy as well as decreased submission size.

2. T computes the certificate z_r for this rth submission, namely $z_r = sig_T(C_r)$, where

$$\begin{aligned} C_r &= (r, t_r, ID_r, y_r; L_r) \\ L_r &= (t_{r-1}, ID_{r-1}, y_{r-1}, H(L_{r-1})) \end{aligned}$$

   and H is a second collision resistant hash function and t_r is the (absolute) time of the submission. L_r is referred to as the *linking information* and contains the respective information pertaining to the submission from the previous round.

3. Upon receiving the next request for a stamp from user v, T sends the timestamp capsule $(z_r, ID_{r+1} = ID_v)$ to u who verifies that the timestamp has been computed properly (note that the ability to verify implicitly requires that C_r and L_r are included in the capsule as well).

To challenge a timestamp, u would give the timestamp capsule (z_r, ID_{r+1}) to the challenger who first verifies that z_r is a valid signature. To verify that there hasn't been a collusion with T, the challenger contacts ID_{r+1} and obtains (z_{r+1}, ID_{r+2}) where

$$z_{r+1} = sig_T(r+1, t_{r+1}, ID_{r+1}, y_{r+1}; L_{r+1})$$

and checks that L_{r+1} contains both y_r and $H(L_r)$. As well, the challenger can also check ID_{r+2}'s stamp or go backwards with ID_{r-1} (as it is included in L_r). It is assumed that the collision resistance of H prevents T from either backdating or forward dating documents (see [HaSt91]). Thus, verification of a challenge mainly consists of verifying that a challenged document belongs to some apparantly legitimate temporal chain.

We note here that the HS scheme is actually a hybrid scheme, and not just a relative stamping scheme since explicit times t_i are included in each timestamp. Indeed, the verification process determines the position of only a single document rather than the relative positioning of a number of challenged documents (thereby not taking full advantage of its ability to provide relative temporal authentication). We therefore assume that if z_{i-1}, z_i and z_{i+1} are stamps that are consecutively linked in a temporal chain and the respective absolute times associated with each are t_{i-1}, t_i and t_{i+1}, then $t_{i-1} < t_i < t_{i+1}$. We make this assumption for the successful running of the protocol, i.e., we assume that any challenger that moves along the chain will check that the times will follow the same temporal order as the stamps to which they are associated.

3.1 Attack

A *fake chain attack* is recognized by Haber and Stornetta [HaSt91], where they claim that

> the only possible spoof is to prepare a fake chain of time-stamps, long enough to exhaust the most suspicious challenger that one anticipates.

Since each timestamp requires a signature by T, this attack would presumably require a collaboration with T. This attack might appear not that difficult to implement except for the fact that for assigning fake stamps, a number of additional collaborators would be required. After all, a suspicious challenger might only be convinced of the legitimacy of a chain if a large number of distinct participants are contacted for verification.

In the following, we present a new attack whereby an attacker can collude with T and *partially insert* a single false stamp into a valid chain of stamps. In this way, only a small fake chain need be produced, that can be "fused" into the valid chain (though only one end of the fake chain is connected to the valid chain). This fake chain is the lower chain in Figure 1. The attack demonstrates that an untrusted, centralized T with no record-keeping is not sufficient for providing the claimed level of security.

The attack proceeds with user u_i colluding with T to backstamp a document y_i (with corresponding stamp z_i). Referring to Figure 1, we see how z_i should appear immediately after z_{i-1}, i.e., normal running of the protocol assumes that additions

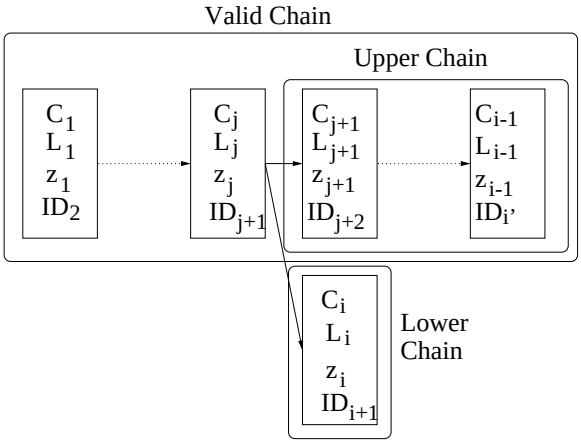

Figure 1: Fork in the temporal chain of documents. Each of the smaller rectangles represents the timestamp capsules for a user. The valid chain is an example of what might be produced from a normal running of the Haber/Stornetta protocol.

take place at the end of the valid chain. However, as in the figure, z_i is placed immediately after z_j. What advantage does this give us? Suppose that z_j, z_{j+1} and z_{i-1} contained the respective times t_j, t_{j+1} and t_{i-1} where $t_j < t_{j+1} < t_{i-1}$. If we were to place the stamp z_i (corresponding to document y_i) in its correct place (i.e. after z_{i-1}), we would have to associate a time $t_i > t_{i-1}$ with it. By placing it immediately after z_j, we can assign any time t_i to y_i (in stamp z_i) such that $t_i > t_j$. Since $t_j < t_{i-1}$, we have backstamped y_i by assigning it a time earlier than the time associated with the most recently stamped document.

Subsequent to the linking of y_i in this new chain, all future stamp requests can either be added in the lower chain (i.e., after y_i whereby $ID_{i'}$ in the upper chain could simply be assigned ID_i; see Figure 1), or alternately added to first the lower then the upper chain (whereby $ID_{i'}$ in the upper chain would be assigned ID_{i+2}; see Figure 1). The latter technique ensures that challenges can proceed in the forward direction for documents contained in the upper chain (though how a challenger would even know when a chain is supposed to end when moving forward must be considered). However, it does require that the stamps for two documents will have the same round number associated with them (i.e., the same r). This is under the assumption that consecutive stamps must have consecutive round numbers associated with them. This is discussed further in Section 3.3.

Now that we have produced z_i (from submission y_i), suppose that someone were to challenge z_i. If they proceed forward with their challenge, they will not discover any faults since documents are subsequently, legitimately stamped after z_i. However, if the challenger proceeds backwards, they will see that the owner of z_j, namely u_j was previously given ID_{j+1} by T, whereas the challenger expects u_j to have ID_i. But there are still some options for our attacker:

1. T and u_i can also collude with u_j to get him to keep ID_i instead of ID_{j+1}. This requires a single additional collusion which is still much less work than the fake chain attack.

2. Have $ID_{j+1} = ID_i$. This can be accomplished by having u_i periodicly stamp (possibly meaningless) documents. Subsequent to this, u_i can backstamp (with T's help) at any point where his dummy documents are.

The second option is clearly more favourable since no additional colluding partners are required. Additional considerations for these options are discussed in Section 3.3.

3.2 Analysis

Although we can cheat this hybrid scheme with respect to its underlying application of an absolute time, this attack is not successful when the scheme is treated as a truly relative stamping scheme. In other words, we cannot backstamp to show the precedence of one stamp over another unless the verification protocol performs this comparison based only on the absolute times associated with the timestamps for the respective documents. There is no relationship between stamps that are solely on the upper or lower chains here. For example there does not exist a relative temporal relationship between the stamps z_{j+1} and z_i as they are not linked together (i.e., via a series of directed edges).[2]

Let us further examine why this attack is successful. Let $f : P \rightarrow P$ be a function where P is the set of all possible timestamp capsules. For example, from Figure 1, we have $p_1 \in P$ belonging to u_1, where $p_1 = \{C_1, L_1, z_1, ID_2\}$. Then $p_2 = f(p_1) = \{C_2, L_2, z_2, ID_3\}$. As long as H is collision resistant, the assumption is that f is a one-to-one function.

The claim of Haber and Stornetta [HaSt91] was that T need not be trusted since an attack would *require* finding a collision for H. From our attack, we can in fact state the following.

Lemma 1 *The collision-resistance of H does not imply that trust in T is not required for the provision of valid timestamps in the Haber/Stornetta timestamping scheme.*

In fact, if T is dishonest, then f is not necessarily a function at all – it is a relation. For example, from Figure 1 we have that $f(p_j) = p_{j+1}$ as well as $f(p_j) = p_i$. Therefore, rather than forming a total order, the set P of timestamp capsules forms a partial order. Let each stamp be a vertex in a graph with an edge from stamp z_i to stamp z_j if z_i was stamped before z_j and one can follow a directed chain from z_i to z_j (or vice-versa). Rather than exclusively forming a single chain, we form a tree, directed from the root. We have a tree (and hence no cycles) since each vertex has no more than a single incoming edge (dictated by the collision resitance of H), but can have more than one outgoing edge (allowing for the creation of multiple paths). Each path from root to leaf is a potentially valid chain which represents a total ordering on its own.

[2] A relative comparison of stamps z_{j+1} and z_i would involve starting at z_i (w.l.o.g.) and successively challenging documents along the chain from z_i until z_{j+1} is reached. The direction taken to arrive at the other stamp would determine the order of precedence. This is the type of verification technique specified by Pinto and Freitas [PiFr96].

3.3 Attack Detection?

We define a *valid state* as one in which only a single temporal chain has been produced by T (i.e., the valid chain in Figure 1). Attack detection is the discovery of a state that is not valid. A successful detection allows one to return to a valid state from a non-valid one. For each of the cases discussed here, detection of the attack is not successful unless certain preventative measures are taken and explicitly required during stamp creation and stamp verification. Suggestions for attack prevention are given in Section 3.4.

In item 1 given at the end of Section 3.1, for the next stamp in the "chain", u_j holds ID_{j+1} and ID_i where $ID_{j+1} \neq ID_i$. ID_{j+1} points to the upper chain while ID_i points to the lower chain. In item 2, this "fork" in the temporal chain is advanced ahead one link. In other words, u_j holds $ID_{j+1} = ID_i$ pointing to each of the next stamps. On the other hand, u_i, possessing a stamp in both the upper and lower chains, holds $ID_{j+2} \neq ID_{i+1}$.

Proceeding backward or forward starting from u_j (in the first case) and u_i (in the second case) causes no suspicion on the part of the challenger since moving backward continues along the valid chain and for moving forward, it does not matter which chain the challenger is lead on to by u_j or u_i. Likewise for approaching either of the stamps held by u_j and u_i from earlier stamps. However, consider item 1 (item 2 is analogous) and suppose that stamp z_{j+k} is currently being challenged where $(j+k) < (i-1)$ and thus the stamp appears in the upper chain (see Figure 1). Working backwards to z_{j+1}, the challenger will eventually obtain ID_j from L_{j+1} (which is stored by u_{j+1}) and hence asks u_j for his timestamp capsule, namely $\{z_j, ID'\}$. Notice that if the challenger were proceeding on the upper chain then he would expect $ID' = ID_{j+1}$ whereas on the lower chain he would expect $ID' = ID_i$. Notice also that u_j has no way of knowing which chain the challenger is proceeding on.

However, consider that u_j may possess many stamps (all presumably along the same chain from the challengers point of view). The challenger will have to inform u_j about which stamp he wishes to challenge. This may include information which identifies which chain he may be proceeding with his challenge on.[3] As well, since the entire capsule (i.e. $\{z_j, ID'\}$) was not signed by T, there is no reason that any integrity should be expected to be associated with it by any challenger.

A second possible suggestion for detection is that subsequent to the partial insertion of the false (lower) chain (see Figure 1), maintenance of both the upper and lower chains requires that some stamps will share the same corresponding identification (round) number. However, the same identification number will only be shared by stamps that appear on different chains. Stamps will have unique identification numbers relative to the chain that they are on. Therefore, unless two documents are compared for their relative positioning, such number repetition is not detected during a stamp verification. In Section 3.2, we acknowledged that our attack is susceptible to detection should a relative measure be performed. Relative ordering is discussed further in Section 3.4.

[3]The protocol description given in [HaSt91] is not specific enough to determine the exact steps taken during such a challenge.

A third method for possible detection may involve the following. Depending on how far back z_i is partially inserted into the valid chain (to produce the lower chain – see Figure 1) the amount of time between the time recorded in the stamp for z_i and the stamp following z_i in the lower chain may be uncomfortably large.[4] Once again though, it is difficult to determine how this might be interpreted by a challenger. Should a scheme such as this require that an upper bound be placed on the amount of time that might elapse between the construction of two consecutive stamps? Prior to knowledge of this attack, such an additional constraint would be unmotivated. Given knowledge of this attack, it may still be difficult to enforce.

If it happened that multiple chains were detected and this evidence is given to an adjudicator, then this essentially brings some suspicion on T. At this point, T may claim that his private signature key must have been compromised. Either he can refute having created one temporal chain, or the other or even both. Note that this loss of key scenario is not the same as if we were to have a single chain for which T refutes some or all of the stamps that he produced. In such a case, the adjudicator may have a choice to believe or not believe T and to not accept or to accept the temporal chain. However, in the case given above, the adjudicator does not have this luxury. Even if he choses not to believe T, how can he tell which chain is correct?

We note again here that such malicious action by T may be unlikely, though not impossible. In Section 3.4, we discuss some additional measures that might be taken to limit the extent of such attacks. Though we must also keep in mind that the relevance of our attack is mainly to point out how the assumption that T need not be trusted does not necessarily allow for a secure protocol.

3.4 Attack Prevention

The purpose of *attack prevention* is quite simply to prevent the attack discussed in Section 3.1.

Two ways to prevent the aforementioned attack are

1. authenticly publish (possibly a subset of) the stamps,

2. or treat the protocol as only a relative scheme.

It is important to realize that the use of newspaper publishing does not simply provide for an extra level of security (should you claim that T need not be trusted) since we have shown the scheme to be insecure without it. With this provision, producing an alternative chain is made more difficult, since the one true chain is widely published, e.g., in the local newspaper. This relies on the ability to ensure that the authenticity of the published stamps can be verified. As well, notice that providing for such publication of the timestamps is not simply an extra feature that can be added to the protocol since its addition produces an entirely new protocol – i.e., the interactions

[4] Consider that stamps following z_i will likely be stamped with a time that is at least as late as the (actual) current time. Whereas z_i (since it is being backstamped) will be stamped with a time that is earlier (possibly much earlier) than the (actual) current time.

required for the verification protocol (the main feature of the scheme) appear to be unnecessary in this case.

For treating the protocol as only a relative scheme (as done by Pinto and Freitas [PiFr96]), there are some drawbacks. First is a loss of fine granularity. It can no longer be determined exactly when a document was timestamped, but rather only when it was timestamped relative to when other documents were timestamped. The second drawback is that more things must be stamped. For the non-repudiation example given at the beginning of the paper, an absolute measure of a signed document can be made by comparing the time in its timestamp with the time of revocation as reported (possibly) by a Certification Authority. For a relative scheme, all such occurrences must be timestamped as well. This also necessitates a single timestamping authority responsible for all such requests. After all, how would one compare the times produced by different timestamping authorities that are producing only relative timestamps?

We note here that an attack such as this is presented relative to the model in which the original scheme was given. For the Haber/Stornetta scheme, the model assumes that the Timestamping Service (T) need not be trusted. We have shown here that a dishonest T can subvert the scheme unless certain precautions are taken. One should not presume however that such an attack is only relevant to the specific Haber/Stornetta scheme. Indeed, one should be careful when designing similar linking schemes. In environments where it is not too unreasonable to trust T, such an assumption, and therefore the attack and preventions, may be unnecessary.

3.5 Extended Linking

Modifying the original scheme, Haber and Stornetta [HaSt91] present the following. Let

$$L_r = [(t_{r-k}, ID_{r-k}, y_{r-k}, H(L_{r-k})), \ldots, (t_{r-1}, ID_{r-1}, y_{r-1}, H(L_{r-1}))].$$

After k requests, ID_r is given the list $(ID_{r+1}, \ldots, ID_{r+k})$, i.e. the identifications of all the users that include L_r in their linking piece. A *challenger* can now check any of the previous or next k clients, ID_{r+i}, $i = 1, \ldots, k$. Just as before, we can apply our attack by having u_i periodicly stamp k consecutive documents. Though there may be some practical difficulties here (depending on the size of k), similar techniques do follow.

4 Conclusion

The purpose of this paper was two-fold. The first is to emphasize that more thorough and specific verification protocols are required and that one ensures that they satisfy the intended goals, e.g., if you include a provision for an (absolute) time during stamp creation, you should be able to identify such a temporal measure during stamp verification. The second (closely related to the first) is to present a model as well as specifically define some of the goals or requirements of a timestamping protocol, e.g., the provision of absolute vs. relative temporal authentication. Notably, one should be careful when setting goals that may be difficult to meet, e.g., assuming that you do

not have to trust the centralized entity that is responsible for providing the users with temporal authentication. We hope that the definitions and model given in Section 1 demonstrate the need for such specifications as well as independently provide some much needed foundation for the role of a timestamping authority.

More generally, our goal was to point out again that protocols that use (believed to be) secure underlying algorithms do not necessarily provide an equivalent level of security should they be carelessly implemented. In this way, we make an attempt at warning potential users or implementors that protocol design is very difficult and that precautions (such as the widespread publication of timestamps) that may be thought of as an additional feature to the security of a system, may in fact be necessary in some situations.

Acknowledgements

Thanks to Paul Van Oorschot for some helpful comments regarding the presentation of the results.

References

[AbNe94] M. Abadi, R. Needham, "Prudent Engineering Practice for Cryptographic Protocols", *DEC/SRC Research Report*, No. 125, June 1994.

[AnNe95] R. Anderson, R. Needham, "Robustness Principles for Public Key Protocols", *Advances in Cryptology – Proceedings of Crypto'95*, Springer-Verlag, pp. 236-247, 1995.

[BHS93] D. Bayer, S. Haber, W.S. Stornetta, "Improving the Efficiency and Reliability of Digital Time-Stamping", *Sequences II: Methods in Communication, Security and Computer Science*, Springer-Verlag, 1993.

[BeMa91] J. Benaloh, M. de Mare, "Efficient Broadcast Time-Stamping", *Clarkson University, Department of Math and Computer Science*, TR 91-1, August 1991.

[BeMa93] J. Benaloh, M. de Mare, "One-Way Accumulators: A Decentralized Alternative to Digital Signatures", *Advances in Cryptology - Proceedings of Eurocrypt '93*, Springer-Verlag, 1993. April 1993).

[HaSt91] S. Haber, W.S. Stornetta, "How to Time-Stamp a Digital Document", *Journal of Cryptology*, Vol. 3, No. 2, pp. 99-111, 1991.

[MOV97] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1997.

[Merk80] R. Merkle, "Protocols for Public-Key Cryptosystems", *Proceedings of the 1980 IEEE Symposium on Security and Privacy*, April 1980.

[Moor88] J. Moore, "Protocol Failures in Cryptosystems", *Proceedings of the IEEE*, Vol. 76, No. 5, pp. 594-601, May 1988.

[PiFr96] F. Pinto, V. Freitas, "Digital Time-stamping to Support Non Repudation in Electronic Communications", *Proceedings SECURICOM '96 - 14th Worldwide Congress on Computer and Communications Security and Protection*, ed. MCI (Manifestations and Communications Internationales), CNIT, Paris, France, pp. 397-406, June 5-6, 1996. (Available from `http://marco.uminho.pt/CCG/ccom-pub.html`.)

[SSDW89] D. Steer, L. Strawczynski, W. Diffie, M. Wiener, "A Secure Audio Teleconference System", *Advances in Cryptology - Proceedings of Crypto'88*, Springer-Verlag, pp. 520-528, 1988.