RENEWING CRYPTOGRAPHIC TIMESTAMPS

Sattam S. Al-Riyami and Chris J. Mitchell
Information Security Group, Royal Holloway, University of London
Egham, Surrey TW20 0EX, UK
S.Al-Riyami@rhul.ac.uk, C.Mitchell@rhul.ac.uk

Abstract This paper shows that the scheme described in Haber and Stornetta [Haber and Stornetta Jr., 1994] for extending the validity of a cryptographic timestamp for a Time Stamping Service contains security shortcomings. A modification is proposed to rectify the identified shortcomings.

Keywords: timestamping, TSA, TSS, digital signature, PKI, security, protocol failure

1. Introduction

A time-stamping service (TSS) has been identified by both the IETF and ISO/IEC as a potentially important part of a Public Key Infrastructure (PKI), and draft standards have been produced by both bodies [Adams et al., 2001, ISO/IEC, 2001]. Conventionally, and as originally proposed by Haber and Stornetta in 1991 [Haber and Stornetta, 1991], a TSS will take as input a hash-code of a data string supplied by a client, and will return a digital signature computed on a concatenation of this hash-code and a time-stamp (the *cryptographic timestamp*). This cryptographic timestamp can then be used as evidence that the original data string existed at the time indicated, without revealing the data string to the TSS. The hash-code should be computed using a collision-resistant one-way hash-function (see for example [Menezes et al., 1997]).

One particularly important application of a TSS is to prolong the lifetime of a digital signature. Without use of a TSS, when a public key certificate expires or is revoked, all signatures computed using the corresponding private key potentially lose their validity. This is because, if the private key becomes known, it is possible to forge signed documents that are indistinguishable from documents produced prior to the point

at which the private key was compromised. On the other hand, if a TSS adds a signed timestamp to a signed document, then this proves that the signature on the document was created prior to the time at which the timestamp was created. Even if the signing key is subsequently revoked, the fact that the original signature can be shown to have been created prior to the time of revocation means that the signature remains valid. Of course, in some environments, and depending on the policy in force, signatures may not lose their validity if the public key revocation occurs for reasons other than key compromise. However, even in such cases, problems may eventually arise because key lengths deemed secure at the time the signature was created may no longer be deemed secure at some later date.

This is particularly relevant for circumstances where signatures are needed to have long-term validity. One obvious example where this will be the case is for the signing of high-value financial transactions and/or contracts, where, as with handwritten signatures, digital signatures will be expected to last indefinitely. It is therefore very likely that timestamping services will be of particular importance for PKIs used to support security for financial applications. This issue is also discussed in RFC 3161, [Adams et al., 2001, Appendix B].

If the signature key of the TSS itself is about to expire, then it is typically necessary to re-timestamp the original cryptographic timestamp with a new TSS signature key, if the original timestamp is to remain valid. This issue has been widely discussed in the literature (see for example [Bayer et al., 1993]). However, independently of the security of the digital signature created by the TSS, the strength of a cryptographic timestamp also relies on the security of the hash function used to compute the hash-code submitted to the TSS [Preneel et al., 1998]. This paper focuses on ways of dealing with the situation where this hash-function is broken, since such an event has the potential to invalidate all cryptographic timestamps computed with this hash-function.

Other authors discuss witnessing and linking techniques in order to improve the security and accountability of the TSS scheme [Buldas et al., 2000]. However, if a pre-image of the hash-code is discovered at any stage after the cryptographic timestamp is produced, the additional techniques will not provide proof of prior existence for the document. Increases in computing power and new algorithmic techniques mean that current "trusted" hash-functions are likely to eventually need replacement. Hence, as soon as any doubts about future use of a particular hash-function arise, and before it is known to be broken, the cryptographic timestamp should be renewed.

Note that loss of confidence in the hash-function is not the only reason for renewal of a cryptographic timestamp. Indeed, timestamp renewal is more commonly discussed in the context of update of the TSS key pair. However, for the purposes of this paper, renewal of cryptographic timestamps refers only to updating the hash-function.

2. The Haber-Stornetta renewal protocol

Haber and Stornetta considered the hash-function renewal problem very briefly in their original 1991 paper [Haber and Stornetta, 1991], and Bayer et al. proposed some modifications in [Bayer et al., 1993]. A much more concrete proposal appeared in the Haber and Stronetta patent [Haber and Stornetta Jr., 1994], and it is this latter scheme we consider here. We first describe their basic time-stamping protocol, and then go on to outline their proposed solution to dealing with hash-functions that require renewal. Note that the form of the timestamping protocol included in the latest draft standards [Adams et al., 2001, ISO/IEC, 2001] is very similar to the Haber and Stornetta proposal.

The basic timestamping protocol, as described in [Haber and Stornetta Jr., 1994], has two steps. First the client requests a timestamp from the TSS, as follows:

A→TSS: $h(M\|X) = h(R)$,
where A denotes the client of the TSS, M is the 'message' to be timestamped, $\|$ denotes concatenation of data items, X is other data of unspecified form as chosen by the client, and h is a hash-function. The concatenation of M and X is also written as R (for *receipt*). Note that the Haber and Stornetta patent [Haber and Stornetta Jr., 1994] is not completely clear as to the purpose of the data string X — they simply suggest that it is used to identify M by, for example, including the 'author data'.

Second, the TSS responds with the cryptographic timestamp C:

TSS→A: $C = S_{TSS}(h(R)\|T)$,
where S_{TSS} denotes the digital signature function of the TSS, and T is the time/date stamp.

On receipt of C, the client A stores it as evidence that M existed at time T.

Note that in their patent specification, [Haber and Stornetta Jr., 1994] Haber and Stornetta actually propose the use of a cryptographic timestamp function F, which is used to compute C, i.e. $C = F(R)$. We have chosen the simplest interpretation of F, namely that F involves concatenation with a timestamp and applying the TSS's digital signature

function. In fact, F could also involve the concatenation of data in addition to the timestamp, to support more complex variants of the scheme. However, this is outside the scope of this paper.

The protocol for renewing the cryptographic timestamp to extend its lifetime is as follows:

A→TSS: $h'(C\|M)$,
where h' denotes the replacement hash-function.

The TSS responds by sending the extended cryptographic timestamp back to the client;

TSS→A: $C' = S'_{TSS}(h'(C\|M)\|T')$,
where C' is the extended cryptographic timestamp, T' is the time of the renewal request and S'_{TSS} is the new signature function of the TSS.

It is also stated in [Haber and Stornetta Jr., 1994] that an alternative more secure way of obtaining a cryptographic timestamp involves use of a compound cryptographic timestamp. Compound cryptographic timestamps are essentially the same as the previous cryptographic timestamps but use two different trusted hash functions, which are applied in parallel to the same receipt. To initiate the basic protocol the client sends $h(R)\|h'(R)$ and consequently two signatures are produced by the TSS. This potentially increases the lifetime of the cryptographic timestamp before a renewal is required.

3. Attack and Observation

We now describe a possible attack against the Haber and Stornetta hash-function renewal protocol. In the attack we suppose that the hash-function used in the original timestamping protocol has been compromised at some point after the renewal process, i.e. the very situation which renewal is supposed to deal with.

The attack is initiated after the first basic cryptographic timestamp is issued and is completed at a later time. In this attack there is no need to break h' and we only assume that the initial hash h is compromised. The attack is performed as follows:

1 Get C: Using the basic protocol, the attacker obtains the cryptographic timestamp $C = S_{TSS}(h(R)\|T)$. Note that we assume $R = (M\|X)$ for some X.

2 Obtain C': The attacker at time T' wants to backdate a forged message M' to T. He does this by requesting a renewal for the

original cryptographic timestamp C by sending $h'(C\|M')$ to the TSS, thus obtaining $C' = S'_{TSS}(h'(C\|M')\|T')$. Now at time T', both hash-functions h and h' remain secure, and therefore at this stage C' cannot be used as the basis of an attack on the scheme.

3 Break h: At a later time T'', suppose h has been broken, so that it is no longer a one-way function. That is, we assume that pre-images can be found for h. Suppose that, as a result, the attacker is able to find a pre-image for $h(R)$ under h, i.e. the attacker can find a string $R^\star$ such that $h(R^\star) = h(R)$. Suppose, moreover, that the attacker can choose the first part of the pre-image $R^\star$ — in particular we suppose that the attacker chooses $R^\star$ so that $R^\star = M'\|X^\star$ for some string $X^\star$. We are thus supposing that h has been subject to a particularly severe form of failure, i.e. so that we can find pre-images for which the first part can be freely chosen. Note however that, given that h involves the iterative use of a round-function (as is the case for all commonly used hash-functions), such a catastrophic failure of the one-way property is not unlikely if the round-function is found to be weak.

The attacker can now claim that the message M' was signed at time T, with C as proof and renewed cryptographic timestamp C' as supporting evidence. Before proceeding, observe that, during the attack described above, in order to find the string $R^\star$ such that $h(R^\star) = h(R)$, where $R^\star = M'\|X^\star$ for some string $X^\star$, some part of $X^\star$ will need to be effectively 'random'. It might consequently be argued that this will reveal the attack, since the third party adjudicating in a dispute will observe that $X^\star$ is a meaningless sequence.

However, whilst this may be true in some circumstances, there is a danger that the situation will not always be so clear cut. First, it might be possible to choose the first part of $X^\star$ to conform to what is expected of such strings, and then to arrange for the random part to be disguised (e.g. as random padding of some kind). Second, the patent specification merely states that the string X can be chosen by the client, and hence it is not reasonable to expect a third party adjudicator to decide what was in the client's mind when he created the string X. Third, it is not good practice to design protocols which rely on third parties making judgements about whether a string is meaningful or not. The protocol should be designed to avoid such issues, and we show below how this can be achieved in a very simple way.

It should be clear that the main reason that this attack is possible is that the original timestamp involves signing $h(R)$, where $R = M\|X$, and the timestamp renewal involves signing $h'(C\|M)$. That is the two

timestamps actually involve different data strings, namely $M\|X$ in the first case, and M in the second case. It is this difference that allows the attack to take place. This observation motivates the proposed protocol modifications discussed in the next section.

4. Revised versions of the protocol

Based on the above observation on the cause of the attack, we propose that the protocol should be modified in the following minimal way.

To obtain a basic cryptographic timestamp, the client initiates the request and the TSS responds, as follows:

A→TSS: $h(Y)$,
where Y denotes either M or $R = M\|X$. Whichever version of Y is adopted, it must be preserved exactly in the renewal protocol.

TSS→A: $C = S_{TSS}(h(Y)\|T)$.

The protocol for renewing the cryptographic timestamp in order to extend its lifetime is as follows:

A→TSS: $h'(C\|Y)$,

TSS→A: $C' = S'_{TSS}(h'(C\|Y)\|T')$.

The only change is to fix the fundamental problem which led to the previous attack, namely that the two timestamps involve different bit-string inputs into the hash-function. Note that, in the case $Y = M$, this modification has previously been described in various places, including in [Haber and Stornetta, 1997]. However, it is important to note that the reason to adopt this variant (or the variant with $Y = M\|X$) in favour of the protocol described in [Haber and Stornetta Jr., 1994], i.e. the existence of the attack described above, has never previously been discussed.

We now consider certain other possible modifications to both the basic and the renewal protocols. First note that we are essentially using $h(Y)$ in the basic protocol and $h'(h(Y)\|Y)$ for the renewal request. An equally secure alternative would be to use $h(Y)$ and $h(Y)\|h'(Y)$ in the respective steps. However, there is no added value in this alternative. In fact the scheme will require the TSS to differentiate between new timestamps and renewal ones. This will unnecessarily complicate the practical implementation of the TSS, in addition to using more bandwidth.

A second modification is to include a hash-function identifier with the hash-code [ISO/IEC, 2001]. This is of fundamental importance in ensuring that the hash-code can not be deliberately mis-represented as having been generated using a different (weaker) hash-function.

A third modification is the inclusion of the message length with the hash-code [PKITS, 1998]. This limits the freedom of an attacker attempting to find a second pre-image for the hash-code.

With these latter two modifications, the basic timestamping protocol is now as follows:

A→TSS: $h(Y)\|N\|hID$,
where hID is a one byte hash-identifier and N is the length of Y.

TSS→A: $C = S_{TSS}(h(Y)\|N\|hID\|T)$,

The corresponding modified protocol for renewing a cryptographic timestamp to extend its lifetime is as follows:

A→TSS: $h'(C\|Y)\|N\|hID'$,
where hID' denotes the hash identifier of h'.

TSS→A: $C' = S'_{TSS}(h'(C\|Y)\|N\|hID'\|T')$.

5. Conclusion

It has been shown that, in the case where the original hash-function admits pre-image attacks, the Haber and Stornetta cryptographic timestamp renewal scheme [Haber and Stornetta Jr., 1994] does not prevent retrospective forgeries. However, this renewal scheme was designed to prevent forgeries in precisely these circumstances. A modification to the protocol that prevents these attacks has been proposed. The modified protocol has a computational and communications overhead that is very similar to the original scheme.

References

[Adams et al., 2001] Adams, C., Cain, P., Pinkas, D., and Zuccherato, R. (2001). *Internet X.509 Public Key Infrastructure, Time Stamp Protocol (TSP)*. Internet Engineering Task Force (IETF). Request For Comments: 3161.

[Bayer et al., 1993] Bayer, D., Haber, S., and Stornetta, W. S. (1993). Improving the efficiency and reliability of digital time stamping. In Capocelli, R., De Santis, A., and Vaccaro, U., editors, *Sequences II: Methods in Communication, Security and Computer Science*, pages 329–334. Springer-Verlag.

[Buldas et al., 2000] Buldas, A., Lipmaa, H., and Schoenmakers, B. (2000). Optimally efficient accountable time-stamping. In Zheng, Y. and Imai, H., editors, *Public Key Cryptography*, volume 1751 of *Lecture Notes in Computer Science*, pages 293–305, Melbourne, Australia. Springer Verlag.

[Haber and Stornetta, 1991] Haber, S. and Stornetta, W. S. (1991). How to timestamp a digital document. *Journal of Cryptology*, 3:99–111.

[Haber and Stornetta, 1997] Haber, S. and Stornetta, W. S. (1997). Secure names for bit strings. In *Proceedings of the 4th ACM Conference on Computer and Communication Security*, pages 28–35. ACM.

[Haber and Stornetta Jr., 1994] Haber, S. A. and Stornetta Jr., W. S. (1994). *Method of extending the validity of a cryptographic certificate*. Bell Communications Research Inc., Livingston, N.J. United States Patent number 5,373,561.

[ISO/IEC, 2001] ISO/IEC (2001). *ISO/IEC FCD 18014-1, Information technology — Security techniques — Time-stamping services — Part 1: Framework*. International Organization for Standardization, Geneva, Switzerland.

[Menezes et al., 1997] Menezes, A., van Oorshot, P., and Vanstone, S. (1997). *Handbook of Applied Cryptography*. CRC Press, Boca Raton.

[PKITS, 1998] PKITS (1998). *PKITS — Public Key Infrastracture with Time-Stamping Authority*. Fábrica Nacional de Moneda y Timbre (FNMT), Madrid, Spain. ETS Project: 23.192, Architecture of Time-Stamping Service and Scenarios of Use: Services and Features.

[Preneel et al., 1998] Preneel, B., Rompay, B. V., Quisquater, J.-J., Massias, H., and Avila, J. S. (1998). Design of a timestamping system. Technical report, TIMESEC, Katholieke Universiteit Leuven and Université Catholique de Louvain.