A Note on the Use of Timestamps as Nonces

B. Clifford Neuman   Stuart G. Stubblebine
Information Sciences Institute
University of Southern California

The use of timestamps in key distribution protocols was suggested by Denning and Sacco [DS81]. Timestamps are now used in most production authentication services including Kerberos [SNS88]. Concerns have been raised about the security implications of this practice [Gon92]. Timestamps are necessary in authentication protocols that support multiple authentication without multiple requests to an authentication server. Kehne, Schönwälder, and Langendörfer [KSL92] have proposed a nonce-based protocol for multiple authentications that they claim improves upon the Kerberos protocol because it does not depend on the presence of synchronized clocks.

This note discusses the use of timestamps as nonces[1] and demonstrates a nonce-based mutual-authentication protocol requiring only four messages, one less than described in [KSL92], and the same number of messages required for mutual-authentication in Kerberos. The note concludes by suggesting extensions to our protocol that allow the use of verifier issued timestamps as nonces while recovering some (though not all) of the benefits of traditional timestamps.

1 Timestamp pros and cons

The design of an authentication system, as with any system, requires that tradeoffs be made. When these tradeoffs involve security, it is important that they be fully understood. The debate about reliance on timestamps is beneficial since it clarifies these tradeoffs.

One consideration when designing an authentication service is reducing the number of messages needed for authentication. The use of timestamps in Kerberos allowed the elimination of one message from the protocol, and two if mutual authentication was not required. This is important when round trip latency is an issue. The use of a timestamp as a nonce also allows one-way authentication when communication with the server is in one direction only.

Authors' address: University of Southern California, Information Sciences Institute, 4676 Admiralty Way, Marina del Rey, California 90292. U.S.A. (bcn@isi.edu, stubblebine@isi.edu)

[1] A nonce is an identifier that is used only once [NS78].

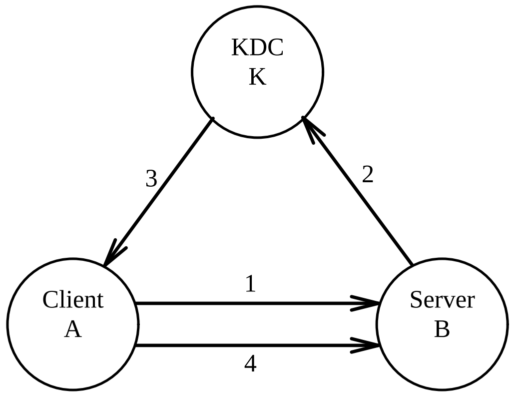
A second consideration in the design of Kerberos was minimizing state at the end-server. By using timestamps the need for per-connection state was avoided. This was important because many of the targeted applications used simple, stateless, request response protocols. While it is still necessary for the end-server to keep track of the timestamps seen within the allowable clock-skew window, such state can be maintained on a system wide basis and does not require the server to remain active while waiting for subsequent messages. Additionally, the amount of potential state can be reduced by narrowing the window.

A drawback of using a timestamp in place of a nonce is the need for loosely synchronized clocks. Some claim that synchronizing clocks in a secure manner is difficult, yet Kerberos itself can be used to authenticate the response from a timeserver since synchronized clocks are only required when authenticating a client to a server and not the reverse. In essence, since a timeserver has little need to authenticate its clients, the timestamp in the request can be treated as a client generated nonce. Further, since the clock on the Kerberos server is returned at the time of the initial request, it can be used to determine that the time needs to be reset, or to record an offset to be used for subsequent messages.

Gong [Gon92] discusses the difficulty of recovering from a post-dated clock. If an authenticator is sent before the clock is reset, that authenticator can be replayed when the time incorrectly recorded in the authenticator is reached. While of concern, we feel that the likelihood of generating a post-dated authenticator can be reduced if the time on the authentication server (AS) is correct and the client carefully checks its current time against that returned by the AS, taking appropriate action if a discrepancy is detected. Further, in the unlikely event that an otherwise valid post-dated authenticator is received, the end-server can take an exception and record the timestamp in non-volatile storage (e.g. disk) so that it would be available across reboots[2].

[2] Note that the window for post-dated authenticators is limited by the maximum lifetime of tickets for the server.

B. Clifford Neuman and Stuart G. Stubblebine. A Note on the Use of
Timsestamps as Nonces. *Operating Systems Review*, 27(2):10-14, April, 1993.



(1) A → B: A, N_a
(2) B → K: $B, \{A, N_a, T_b\}K_{kb}, N_b$
(3) K → A: $\{B, N_a, K_{ab}, T_b\}K_{ka}, \{A, K_{ab}, T_b\}K_{kb}, N_b$
(4) A → B: $\{A, K_{ab}, T_b\}K_{kb}, \{N_b\}K_{ab}$

Figure 1: The initial exchange

The above steps will be sufficient for most applications. Where stronger guarantees are needed, protocols such as Version 5 of Kerberos provide mechanisms whereby a server can require the use of a traditional nonce (and the additional protocol messages needed to use it). Note also that even without this option the Kerberos protocol only uses timestamps to assure the freshness of the initial message from the client to the end-server. The protocol uses a more traditional nonce to assure the freshness of the response from the authentication server and for mutual authentication.

To summarize, use of timestamps in authentication protocols is beneficial in many situations, and although it is important to understand their limitations, their use as nonces does not necessarily reduce the security of an authentication protocol.

2 A nonce-based protocol

Having discussed the pros and cons of timestamps, let us consider a new protocol and some variations as an exercise to better understand the benefits of the alternatives. This protocol is a nonce-based protocol for repeated mutual authentication requiring only four messages initially and three messages for subsequent authentication to a single server.

2.1 Initial authentication

The initial exchange of our protocol is similar to that of Yahalom [Yah, BAN89], but extended to allow subsequent reuse of the credentials issued to A. We also do not require the secrecy of the nonce (N_b) in messages (2) and (3).

Our protocol begins with principal A initiating authentication by sending a cleartext message (1) to B containing A's name, together with a nonce generated by A. B receives the message then sends message (2) to the authentication server (K) containing B's name, a nonce generated by B, and instructions to the KDC to issue credentials to A. These instructions specify A as the intended recipient of the credentials, a suggested expiration time for the credentials, and the nonce received from A. The instructions are encrypted in K_{bk}, the key registered for principal B with the KDC.

Upon receiving instructions from B, K decrypts the instructions with K_{bk}, thus verifying that they were issued by B at some point in the past (K does not care about replays). K then issues a ticket allowing A to share a session key (K_{ab}) with B until the expiration time specified by B. In message (3), K sends the ticket and B's nonce (N_b) to A together with the session key K_{ab}, B's identity, the expiration time specified by B, and the nonce originally generated by A (the last four items encrypted in K_{ak}, the key shared by the KDC and A).

Upon receiving message (3), A decrypts that part encrypted in K_{ak} and verifies that the nonce N_a is the same as that included in message (1). If it matches A is assured that K received the nonce from B encrypted in K_{bk}. Further, A knows B's nonce (N_b) and can use it to prove its identity to B.

Finally, in message (4), A forwards the ticket to B together with B's nonce (N_b) encrypted under the session key from the ticket, thus proving to B the identity of A.

2.2 The ticket

As was the case in the protocol described in [KSL92], execution of the protocol leaves A in possession of a ticket that may be used for subsequent authentication to B, avoiding the need to repeatedly contact the authentication server. The ticket specifies a session key to be used between A and B (K_{ab}) and an indication of the time after which that key should no longer be considered valid. Like the ticket in [KSL92], the time in the ticket (in this case an expiration date) is relative to B's clock, and the time has been generalized to include an epoch, an identifier for the current strand of time. If B's clock is set back, the epoch is changed, and any tickets not issued in the current epoch are rejected.

Unlike the ticket in [KSL92], the current timestamp from B is not included. The expiration time is sufficient since the ticket can not be used before it is issued. A second, and perhaps more fundamental dif-

(1') A $\rightarrow$ B: $N_a', \{A, K_{ab}, T_b\}K_{kb}$
(2') B $\rightarrow$ A: $N_b', \{N_a'\}K_{ab}$
(3') A $\rightarrow$ B: $\{N_b'\}K_{ab}$

Figure 2: Subsequent authentication

ference between the ticket in [KSL92] and that in our protocol is that in our protocol, the ticket is issued by K and encrypted in the key shared by K and B, rather than issued by B in a key known only to itself [3].

2.3 Subsequent authentication

At the end of the initial exchange, A is in possession of a ticket and session key that may be used to communicate with B. A sends B the ticket, together with a newly generated nonce (N_a') with which B will authenticate itself to A. Upon receipt of the message, B decrypts the ticket, verifies that it has not expired, extracts the session key (K_{ab}), then responds to A with A's nonce (N_a') encrypted in the session key (K_{ab}) together with a new nonce (N_b') with which A will prove its identity to B. A decrypts the message from B, verifies N_a' and is assured of B's identity. A then encrypts N_b' in the session key and returns it to B, thus proving its identity to B.

3 Analysis

In this section, we discuss the importance of deriving formalized goals from functional objectives, and hence, explain why our protocol achieves the same functional objectives with fewer messages. We also note an inconsistency between the description of the earlier protocol for subsequent authentication [KSL92] and the analysis using the logic of [BAN89].

3.1 Initial authentication

Designers who use logic in analyzing protocols frequently fail to explicitly define the formalized goals (i.e., final beliefs and sometimes possessions [GNY90]) of a protocol. Perhaps this failure to define formalized goals is partly due to the subtle variability of functional objectives in different protocols. For example, one protocol may merely need to prove the presence of one party to the other, while another must prove the presence of each party to each other. As authentication often precedes future communication, protocols may require unconfirmed or confirmed distribution of a fresh session key to one or more communicants.

[3]Note that we could allow B to generate the ticket and session key (K_{ab}) itself and send it to K for forwarding to A, in which case the ticket would be similar to that in [KSL92].

The failure to define formalized goals may also be in part due to the lack of guidance on how formalized goals are derived from functional objectives. Though it is beyond the scope of this note to adequately address this issue, we point out that it is imperative for designers to define the minimal set of formalized goals from functional objectives when making statements about the minimality of the number of messages required to satisfy a functional objective.

An important distinction between our protocol and that of [KSL92] is that we need only satisfy a reduced set of formalized goals to achieve the functional objectives under the same assumptions. We reason that the functional objectives of the initial protocol proposed by Kehne et al. and that of Kerberos, is to (1) prove the presence of A and B to each other, and (2) for A to obtain a session key and ticket that can be used for subsequent authentication.

We claim that the following set of formalized goals supports these functional objectives [4]:

$$A \ \textbf{believes}\ A \overset{\mathrm{K_{ab}}}{\longleftrightarrow} B$$
$$A \ \textbf{believes}\ \mathrm{B}\ \textbf{believes}\ N_a$$
$$B \ \textbf{believes}\ A \ \textbf{believes}\ A \overset{\mathrm{K_{ab}}}{\longleftrightarrow} B$$

In constrast with the formalized goals of the initial protocol of [KSL92], we need not obtain the belief

$$A \ \textbf{believes}\ B \ \textbf{believes}\ A \overset{\mathrm{K_{ab}}}{\longleftrightarrow} B$$

to satisfy functional objectives (1) and (2) above.

3.2 Subsequent authentication

Our protocol for repeated authentication is similar to that in the original paper [KSL92], however the beliefs we obtain are weaker than those incorrectly obtained therein.

The pitfalls of mapping an actual protocol to the logic (i.e. idealization) [GKSG91] account for the incorrect beliefs obtained in the original paper [KSL92]. The flaw in the original analysis stems from an inconsistency between the protocol description and the idealization of the protocol to obtain the following assumption about the freshness of the timestamp:

$$B \ \textbf{believes fresh}\ T_b$$

Recall, a formula X is fresh when X has not been sent in a message at any time before the current run of the protocol [BAN89]. According to the protocol

[4]When the secrecy of nonce N_b is maintained in messages (2) and (3), the analysis of the initial protocol follows closely with that of the strengthened version of the Yahalom Protocol [BAN89] and for brevity is not included herein. Extensions to the logic of [BAN89] are necessary to obtain these beliefs for the initial protocol shown in Figure 1.

description, any T_b that has the current time identifier and lifetime that has not expired is considered valid[5]. Thus the description of the state information maintained by B is insufficient for B to determine the freshness of Message $1'$. This precludes the application of the nonce-verification rule of [BAN89] to obtain:

$$B \ \textbf{believes}\ A \stackrel{K_{ab}}{\longleftrightarrow} B.$$

4 Verifier issued timestamps

We have shown a protocol for nonce-based repeated authentication requiring four messages for initial authentication. Like the protocol in [KSL92] the server (B) must communicate with a third party (K) as part of the initial exchange. This requires per-connection state. Unfortunately, the need for per-connection state can make it difficult to integrate an authentication protocol with applications that support a simple, stateless, request response protocol. With several changes, however, we can reduce the required state to that required by Kerberos without the need for synchronized clocks.

To do this we apply the technique from [KSL92] that allows timestamps without synchronized clocks, but in this case we use a timestamp (with epoch) generated by B as a nonce instead of the life of ticket. The nonce will not be recorded for future comparison. Instead, upon receipt of the nonce, B will check to see that the epoch is current and that the timestamp falls within the allowable window. Like Kerberos, the server must keep track of timestamps seen within the window but because the timestamp is based on the local clock, that window can be very small (the maximum reasonable round trip time through K and A).

A similar modification can be made to the protocol for subsequent authentication, but both messages 1' and 3' will have to be accompanied by the ticket.

4.1 Epochs

Within an epoch a clock must increase monotonically. If a clock is set back its epoch must change. Epochs may be useful even when a timestamp is verified by a separate party. For example an epoch associated with the time on an authentication server can provide a mechanism to quickly invalidate outstanding tickets if the time on the authentication server has to be set back.

[5] This follows from the protocol description which states that principal B manages the time identifier and local clock and verifies a ticket by testing the timestamp and local lifetime and comparing the local time identifier with the one in the ticket.

5 Conclusion

In this note we discussed advantages and disadvantages of using timestamps to prevent replay in authentication protocols. We presented a four message protocol for initial authentication that supports subsequent authentication in three messages without contacting the authentication server and without the use of synchronized clocks. Where it is important to eliminate per-connection state on the server we suggested a modification to our protocol that uses verifier issued timestamps as nonces.

When designing an authentication protocol certain tradeoffs must be made. It is important to understand the issues in order to choose the best solution. Of the protocols discussed, issues to be considered are per-connection state on the server, the number of messages required, the presence of loosely-synchronized clocks, and the ability to recover from a post-dated clock.

Where the number of messages is the dominant factor, perhaps due to round trip latency or the need to fit the communication pattern of an existing application, the use of timestamps from loosely-synchronized clocks is appropriate. Where recovery from post-dated protocol messages is a dominant issue, a timestamp or nonce issued by the verifier is required. Where reduction of per-connection server state is of concern the use of a timestamp is the appropriate choice. Depending on other factors the timestamp can be issued either by the verifier or based on loosely-synchronized clocks.

	Random	Timestamp	
	Nonce	Verifier	Synch
Number of Messages	-	-	+
Connection State	-	+	+
No Synch Clocks	+	+	-
Postdate Recovery	+	+	-

Acknowledgments

We would like to thank Martin Abadi, Celeste Anderson, Li Gong, Axel Kehne, Steve Lunt, and Juergen Schoenwaelder for comments on a draft of this note.

References

[BAN89] M. Burrows, M. Abadi, and R. Needham. A logic for authentication. Technical Report 39, Digital Equipment Corporation Systems Research Center, March 1989.

[DS81] Dorothy E. Denning and Giovanni Maria Sacco. Timestamps in key distribution protocols. *Communication of the ACM*, 24(8):533–536, August 1981.

[GKSG91] V. Gligor, R. Kailar, S. Stubblebine, and L. Gong. Logics for cryptographic protocols - virtues and limitations. In *Proceedings of the Computer Security Foundations Workshop IV*, pages 219–226, June 1991.

[GNY90] L. Gong, R. Needham, and R. Yahalom. Reasoning about in cryptographic protocols. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1990.

[Gon92] Li Gong. A security risk of depending on synchronized clocks. *Operating Systems Review*, 26(1):49–53, January 1992.

[KSL92] A. Kehne, J. Schonwalder, and H. Langendorfer. A nonce-based protocol for multiple authentication. *Operating Systems Review*, 26(4):84–89, October 1992.

[NS78] Roger M. Needham and Michael D. Schroeder. Using encryption for authentication in large networks of computers. *Communication of the ACM*, 21(12):993–999, December 1978.

[SNS88] J. G. Steiner, B. C. Neuman, and J. I. Schiller. Kerberos: An authentication service for open network systems. In *Proceedings of the Winter 1988 Usenix Conference*, pages 191–201, February 1988.

[Yah] Raphael Yahalom. The shortest asynchronous secure data exchange protocol. Submitted for Publication.