Multiple Authentications With a Nonce-Based Protocol Using Generalized Timestamps

A. Kehne* J. Schönwälder H. Langendörfer

Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig, Bültenweg 74/75, D-3300 Braunschweig

Abstract

The Kerberos authentication service, a part of MIT's Project Athena, is based on the Needham and Schroeder protocol. Timestamps depending on reliable synchronized clocks are used to guarantee the freshness of messages. As an improvement, we present a nonce-based protocol offering the same features as Kerberos. We generate a ticket in an initial message exchange which includes a generalized timestamp. Checking this generalized timestamp is left to the principal who created it. Consequently we do not need synchronized clocks. Our protocol has the property of using a minimal number of messages to establish an authenticated session key.

Keywords: Security, Authentication, Kerberos

1 Introduction

In computer networks and distributed computing systems a mechanism is needed to provide secure communication. To trust the identity of each other, two principals must run a procedure resulting in mutual authentication. Key distribution protocols establish secret authenticated session keys using conventional cryptography, also called symmetric private key cryptography. A very promising implementation is the Kerberos system [5]. It is a well-known problem that Kerberos depends on reliable synchronized clocks in the network.

As an improvement, we present a nonce-based protocol not relying on synchronized clocks. Generalized timestamps are used to allow repeated authentications. Our protocol has the property of using a minimal number of messages to achieve its authentication goals, assuming conventional cryptography, no initially shared secrets and not relying on timestamps.

* A. Kehne is now at Philips Research Labs.,
P.O. Box 1980, D-5100 Aachen.

In section 2 we shortly review some properties of Kerberos and its dependence on synchronized clocks. In the next section our nonce-based protocol is presented. In section 4 we analyze our protocol using a logic introduced by Burrows, Abadi and Needham. We conclude in section 5 with a discussion of our protocol.

2 Kerberos' dependence on synchronized clocks

The Kerberos authentication service [5] was developed at MIT as part of Project Athena. It is based on the Needham and Schroeder trusted third-party authentication protocol [6] and uses timestamps suggested by Denning and Sacco [3]. A trusted authentication server S generates a ticket including a session key K_{ab} for secure communication between two principals A and B. To reduce communication overhead and to decrease server traffic, this ticket can be used for repeated authentication without contacting the server. The validity of the ticket is limited by a timestamp and a lifetime.

A problem with this protocol is its dependence on reliable synchronized clocks. As discussed in [2], a trusted time server using an authenticated time synchronization protocol is needed. Currently, many unauthenticated time protocols are in use, so the following kind of replay attack is possible:

First an intruder copies a valid ticket. Then he takes as much time as needed to break the session key K_{ab}. When successful, he can impersonate principal A. He sends a forged time server message to B to "correct" B's local clock to the time he copied the ticket. Then, he can send a forged message containing the ticket to make B believe that K_{ab} is an authenticated session key for talking to A. A much more dramatic scenario is given in [4], and some more points are discussed in [2] which can be bypassed by using a nonce-based protocol that does not need any timestamps relying on synchronized clocks.

3 Nonce-based multiple authentication

Our authentication protocol is divided in two parts. First the principals A and B run an initial protocol to establish an authenticated session key K_{ab} and a ticket. In the second part this ticket is used for repeated authentications.

3.1 The initial protocol

In this protocol two principals A and B, not sharing any secret, want to authenticate each other. A trusted authentication server S shares a secret key with each principal. Since no timestamps relying on synchronized clocks are used in this protocol, the freshness of the messages is achieved with nonces, which are a kind of once used keys. The initial protocol is shown below:

$$\begin{array}{lll}
1 & A \to B: & N_a, A, \\
2 & B \to S: & N_a, A, N_b, B, \\
3 & S \to B: & \{N_b, A, K_{ab}\}_{K_{bs}}, \{N_a, B, K_{ab}\}_{K_{as}}, \\
4 & B \to A: & \{N_a, B, K_{ab}\}_{K_{as}}, \{T_b, A, K_{ab}\}_{K_{bb}}, \\
 & & N_c, \{N_a\}_{K_{ab}}, \\
5 & A \to B: & \{N_c\}_{K_{ab}}.
\end{array}$$

Principal A generates a nonce N_a, and sends the cleartext message 1 to B, who safes the nonce, generates a second one N_b, and sends the cleartext message 2 to S. Now, the server chooses a good session key K_{ab}, looks up the secret keys K_{as} and K_{bs}, and sends message 3 to B. K_{as} (K_{bs}) denotes the secret key shared by A and S (B and S).

Principal B decrypts the first part of message 3 with his secret key K_{bs} and verifies the nonce N_b and the name A. If they match, B can be sure of this message being fresh, and he picks up K_{ab} as a trusted session key with A. Afterwards he generates a new nonce N_c and encrypts N_a received in message 1 with K_{ab}. To allow repeated authentications with fewer messages, B generates a ticket $\{T_b, A, K_{ab}\}_{K_{bb}}$. This ticket contains a generalized timestamp T_b, the name of principal A and the session key K_{ab}, and is encrypted with a secret key K_{bb} known only to B. After receiving message 4, A decrypts the first part with his secret key K_{as} and verifies the nonce N_a and the name B. If they are correct, he trusts the session key K_{ab} and decrypts the last part of the message. Verifying N_a, he is convinced of sharing this key with the correct principal B. Then he safes the ticket together with K_{ab} and the name B in a secure memory for later repeated authentications. At last, he encrypts the nonce N_c with the session key and sends message 5 to B to convince him of possessing K_{ab}.

3.2 The ticket

Before presenting the protocol for repeated authentications, let us consider the ticket. Principal A can use it to convince B again to use and trust K_{ab} as a secret session key. Since there is needed something that limits the validity of the ticket, we suggest a generalized timestamp T_b which consists of three parts:

1. A current timestamp from the local clock of principal B,
2. a lifetime which limits the validity of the ticket, and
3. a time identifier to protect B from replay attacks with forged time server messages and old tickets.

A time identifier, e.g. a long sequence number or a long random number, is managed by principal B together with his local clock. Every time a server contacts B for correcting his local clock, B changes his time identifier. Verifying a ticket means testing timestamp and lifetime, if still valid, and comparing the local time identifier with the one in the ticket. If they do not match, B will reject the ticket.

We emphasize that this timestamp is not based on synchronized clocks, because B checks only self-generated timestamps. To protect the ticket from any tampering, B encrypts it using a secret key K_{bb} shared with no one else.

3.3 The protocol for repeated authentications

Principal A, knowing the session key K_{ab} and possessing a valid ticket, can repeat the authentication in the following way:

$$\begin{array}{lll}
1' & A \to B: & N'_a, \{T_b, A, K_{ab}\}_{K_{bb}}, \\
2' & B \to A: & N'_b, \{N'_a\}_{K_{ab}}, \\
3' & A \to B: & \{N'_b\}_{K_{ab}}.
\end{array}$$

A sends a new nonce N'_a together with the ticket to B. After verifying the ticket in the way described above, B trusts the session key K_{ab}. The following messages are exchanges of nonces, partly encrypted with K_{ab}, to convince each other of presently existing and possessing the session key analogous to the last part of message 4 and message 5 in the initial protocol.

4 Analysis

Before starting the analysis of our protocol, we give a short introduction in a formal method useful for analyzing authentication protocols.

4.1 A logic of authentication

The logic of authentication was presented by Burrows, Abadi and Needham in [1] to serve the need for a formalism to exactly understand all initial assumptions a protocol is based on and to clearly set out all final states. Here we just give a short overview over this logic. Thus readers not familiar with it should be able to understand the analysis of our protocol. We do not discuss the semantics of this logic and refer the reader to [1] for more details.

A, B, S	Principals.
K_{ab}, K_{as}, K_{bs}	Shared secret keys.
N_a, N_b, N_c	Statements.
P, Q, R	Range over principals.
X, Y	Range over statements.
K	Ranges over keys.

Table 1: Symbols of the logic.

The tables 1 and 2 summarize all symbols of the logic needed for our analysis. A very important construct of this logic is $P \models X$, which means that P may act as though X is true. Another important construct is $\#(X)$, which expresses that X was never used in any message before the current run of the protocol. Nonces meet this property by definition, i.e. they are expressions generated for the purpose of being fresh.

comma	Conjunction of statements.
$P \models X$	P believes X.
$P \triangleleft X$	P sees X.
$P \mid\!\sim X$	P once said X.
$P \Rightarrow X$	P has jurisdiction over X.
$\#(X)$	X is fresh.
$P \stackrel{K}{\leftrightarrow} Q$	P and Q share a secret key K.
$\{X\}_K$	X encrypted under K.
(X,Y)	concatenation of X and Y

Table 2: Relational symbols.

We now shortly summarize the most important logical postulates.

(1) The message-meaning rule:

$$\frac{P \models Q \stackrel{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \models Q \mid\!\sim X}.$$

If P believes that he shares the key K with Q and sees a message X encrypted under K, then P believes that Q once said X.

(2) The nonce-verification rule:

$$\frac{P \models \#(X), \quad P \models Q \mid\!\sim X}{P \models Q \models X}.$$

If P believes that X was never used before and that Q once said X, then P believes that Q believes X.

(3) The jurisdiction rule:

$$\frac{P \models Q \Rightarrow X, \quad P \models Q \models X}{P \models X}.$$

If P believes that Q has jurisdiction over X and that Q trusts X as though it is true, then P believes X.

(4) A rule for extending the freshness:

$$\frac{P \models \#(X)}{P \models \#(X,Y)}.$$

It is important to note that if one part of a message is known to be fresh, then the whole message must also be fresh.

Additional rules allow the decomposition and decryption of messages. Another rule reflects the commutativity of shared keys. The protocol analysis is performed in the following four steps:

1. The original protocol must be transformed into an idealized one, because the messages of the concrete protocol are not a useful basis for a formal analysis. As a general rule cleartext is omitted in the idealized protocol, because it can simply be forged. More details about transformation rules can be found in [1].

2. The assumptions about the initial state are denoted.

3. Logical formulars are inserted after each protocol step as assertions about the state of the system.
4. The logical postulates are applied to the assumptions and assertions to discover the final believes held by the parties.

4.2 Analysis of the initial protocol

We start the analysis by first transforming the initial protocol into the idealized form:

$$\begin{array}{lll} 3 & S \to B: & \{N_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}, \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \\ 4 & B \to A: & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \{T_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bb}}, \\ & & \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}, \\ 5 & A \to B: & \{N_c, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}. \end{array}$$

As stated above, the cleartext messages are omitted in the idealized protocol. So it seems that the authentication server S acts spontaneously. Comparing message 3 with the concrete protocol, it should be intelligible that the two parts contain the nonce and the formula $A \overset{K_{ab}}{\leftrightarrow} B$ to express that K_{ab} is a secret key to be shared by A and B. More interesting is the third part of message 4. In the concrete protocol B sends $\{N_a\}_{K_{ab}}$ to A. Principal B, trusting the session key K_{ab}, decrypts and verifies the second part of message 3. Then he picks up the nonce N_a received in message 1, and encrypts it under K_{ab} to tell A that he trusts K_{ab} as a secret session key shared with principal A. So we have $\{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}$ as the last part of message 4 in the idealized protocol. Message 5 is transformed analogously.

$$\begin{array}{ll} \text{A1} & A \models A \overset{K_{as}}{\leftrightarrow} S, \quad B \models B \overset{K_{bs}}{\leftrightarrow} S, \\ & S \models A \overset{K_{as}}{\leftrightarrow} S, \quad S \models B \overset{K_{bs}}{\leftrightarrow} S; \\ \text{A2} & S \models A \overset{K_{ab}}{\leftrightarrow} B; \\ \text{A3} & A \models (S \Rightarrow A \overset{K}{\leftrightarrow} B), \;\; B \models (S \Rightarrow A \overset{K}{\leftrightarrow} B); \\ \text{A4} & A \models \#(N_a), \;\; B \models \#(N_b), \;\; B \models \#(N_c). \end{array}$$

Let us consider the assumptions of our protocol stated above. The first group is about shared secret keys between the principals and the server. Group A2 denotes that the server generates the session key K_{ab}. The third group indicates the trust of A and B in the server S for generating good encryption keys. The last group deals with the freshness of the nonces. Note that each principal trusts only the freshness of self-generated nonces. The ticket for repeated authentications is not needed in the initial protocol run.

Step 4 of the analysis starts with the server S, who generates the key K_{ab} and sends message 3 to B, i.e.

$$B \lhd \{N_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}, \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}.$$

Decrypting the first part of this message with K_{bs}, B can check the nonce N_b. Applying the message-meaning rule to the first part, we can deduce

$$B \models S \mid\sim (N_b, A \overset{K_{ab}}{\leftrightarrow} B).$$

The assumption $B \models \#(N_b)$ can be extended to $B \models \#(N_b, A \overset{K_{ab}}{\leftrightarrow} B)$ using rule (4). Applying the nonce-verification rule, we get

$$B \models S \models (N_b, A \overset{K_{ab}}{\leftrightarrow} B).$$

Splitting this message and using the jurisdiction rule, we can deduce

$$B \models A \overset{K_{ab}}{\leftrightarrow} B.$$

B, trusting the session key K_{ab}, generates message 4 and sends it to A, i.e.

$$A \lhd \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}, \{T_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bb}}, \\ \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}.$$

Considering the first part, we can deduce the formula

$$A \models A \overset{K_{ab}}{\leftrightarrow} B$$

in a way similar to the one described above. Applying the message-meaning rule to the third part of message 4, we get

$$A \models B \mid\sim (N_a, A \overset{K_{ab}}{\leftrightarrow} B).$$

Extending the trust in the freshness of the nonce N_a using rule (4), we can apply the nonce-verification rule and obtain

$$A \models B \models (N_a, A \overset{K_{ab}}{\leftrightarrow} B)$$

and decomposing the message, we get the formula

$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B.$$

Then principal A generates message 5 and sends it to B, i.e.

$$B \lhd \{N_c, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}.$$

Analyzing this formula analogously to the third part of message 4, we finally get

$$B \models A \models A \overset{K_{ab}}{\leftrightarrow} B.$$

The final beliefs held by the principals A and B are summarized below:

$$A \models A \overset{K_{ab}}{\leftrightarrow} B, \quad B \models A \overset{K_{ab}}{\leftrightarrow} B,$$

$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B, \quad B \models A \models A \overset{K_{ab}}{\leftrightarrow} B.$$

Burrows, Abadi and Needham name these statements the formalized goals of authentication. The first two formulas show that each principal trusts K_{ab} as a secret session key shared with the other one. The last two state that each principal believes that the other one currently trusts the session key.

4.3 Minimality of the initial protocol

The initial protocol has the property of using a minimal number of messages to achieve the goals of authentication under the following assumptions:

- Conventional cryptography is used.
- The principals A and B do not share any secret before running the protocol.
- No timestamps relying on synchronized clocks are used, and the freshness of messages is guaranteed by using nonces.
- Both principals share a secret key with a trusted authentication server S (A1 in subsection 4.2).
- The authentication server generates the session key (A2).
- The authentication server is trusted by A and B, e.g. for generating good session keys (A3).
- Each party trusts only the freshness of self-generated nonces (A4).
- The formalized goals of authentication stated above have to be deducible.

Our initial protocol fulfills these assumptions. To illustrate, why we need 5 messages at minimum, take a look at the analysis. First, we must be able to deduce the final beliefs

$$A \models A \overset{K_{ab}}{\leftrightarrow} B \quad \text{and} \quad B \models A \overset{K_{ab}}{\leftrightarrow} B.$$

As shown in the analysis each principal must receive a message and then apply the message-meaning rule, the nonce-verification rule and the jurisdiction rule (perhaps with some other useful rules in between) to deduce these formulas. The nonce-verification rule can only be used, if the message contains a nonce that is believed to be fresh. Because each principal trusts only the freshness of self-generated nonces and because the server must know these nonces to put them together with the session key in a message, there must have been at least two messages sent, before the server sends any message containing the session key. Thus, the third message is the first one that could include the session key.

Now consider the other two final beliefs

$$A \models B \models A \overset{K_{ab}}{\leftrightarrow} B \quad \text{and} \quad B \models A \models A \overset{K_{ab}}{\leftrightarrow} B.$$

Only one of the principals A and B can receive the session key in a third message, let us say A. Then, B cannot receive K_{ab} before the fourth message is sent. Now have a look at the formula $A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$. To deduce this formula it is necessary that A receives a message that contains a nonce and is encrypted under K_{ab}, using the message-meaning rule and the nonce-verification rule. Because B receives the session key earliest in the fourth message, the message considered must at least be a fifth one. So at least 5 messages are needed to reach the demanded goals of authentication. The initial protocol examplifies that no more than 5 messages are needed.

4.4 Analysis of the protocol for repeated authentications

The transformation of the protocol for repeated authentications results in the following idealized form:

$$\begin{array}{lll} 1' & A \to B: & \{T_b, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bb}}, \\ 2' & B \to A: & \{N_a', A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}, \\ 3' & A \to B: & \{N_b', A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{ab}}. \end{array}$$

The cleartext parts of the concrete messages are omitted. The rest of the transformation is done similar to the last messages of the initial protocol. The following assumptions are made:

$$\begin{array}{ll} \text{B1} & A \models A \overset{K_{ab}}{\leftrightarrow} B, \quad B \models B \overset{K_{bb}}{\leftrightarrow} B; \\ \text{B2} & B \models (B \Rightarrow A \overset{K_{ab}}{\leftrightarrow} B); \\ \text{B3} & A \models \#(N_a'), \quad B \models \#(N_b'), \quad B \models \#(T_b). \end{array}$$

Principal A still knows the session key K_{ab}, and B trusts his secret key K_{bb}, expressed in B1. Assumption B2 expresses that B trusts himself to have put the correct session key into the ticket. B3 describes the trust in the freshness of the nonces and the timestamp T_b.

The analysis starts with message $1'$, i.e.

$$B \lhd \{T_b, A \stackrel{K_{ab}}{\leftrightarrow} B\}_{K_{bb}}.$$

B can decrypt this message to check the generalized timestamp. Applying the message-meaning rule to this formula we get

$$B \models B \mid\sim (T_b, A \stackrel{K_{ab}}{\leftrightarrow} B).$$

Extending the timestamp using rule (4), we can apply the nonce-verification rule to deduce

$$B \models B \models (T_b, A \stackrel{K_{ab}}{\leftrightarrow} B).$$

Splitting the message and using the jurisdiction rule, we obtain

$$B \models A \stackrel{K_{ab}}{\leftrightarrow} B,$$

so B trusts the session key K_{ab}.

The analysis of the messages $2'$ and $3'$ is the same as the analysis of the third part of message 4 and message 5 of the initial protocol. Again, we get the goals of authentication.

5 Conclusion

We have presented a nonce-based protocol for multiple authentications. The initial protocol needs 5 messages and its minimality is explained. We propose a ticket containing a generalized timestamp avoiding dependence on reliable synchronized clocks to allow repeated authentications with only 3 messages.

In the following discussion we adopt Kerberos terminology, where A is a client and B is a network server. In our protocol the network server has complete control over the ticket. Because he generates the timestamp, he can determine the lifetime of the ticket with respect to the trust in the client, the sensitivity of the offered service, and the security of the network. These parameters may change.

After a period of time or if there is suspicion in the secret key K_{bb} being broken, the server can change it without any problems, because this key is shared with no one else. As an implication all tickets become invalid even though they are not expired.

As an extension to the protocol, it is possible for the server to renew the ticket. For this purpose a client presents a valid ticket to the server asking him to renew it. It is in the responsibility of the server to do or not to do so. If a new ticket is granted, it could contain a new carefully chosen session key, e.g. the result of a special function applied to the previous session key. Since the server has complete control over the ticket, many extensions are possible, where the server puts additional information in the ticket. As an example Kerberos-like postdated tickets may be managed this way. And since the ticket contains all information necessary to check the validity of the session key, the server does not need to store old keys.

Depending on the update frequency of the time synchronization protocol in use, another strategy for changing the time identifier is needed. In this strategy the server manages a variable, in which all time synchronization differences are summed up since the last change of the time identifier. If this variable exceeds a limit, the time identifier is changed and this variable is set to 0. With this strategy the time identifier is changed less often.

Comparing with Kerberos, the cost of avoiding dependence on synchronized clocks is one additional message in every authentication. In future work we plan to implement this protocol in a Kerberos-like environment to achieve a higher degree of security and to compare it with Kerberos from a practical point of view.

References

[1] M. Burrows, M. Abadi, R.M. Needham: *A logic of authentication*, Proceedings of the Royal Society of London, A 426, 1989.

[2] S.M. Bellovin, M. Merritt: *Limitations of the Kerberos Authentication System*, ACM Computer Communications Review, 20(5), October 1990.

[3] D.E.R. Denning, G.M. Sacco: *Timestamps in Key Distribution Protocols*, CACM, 24(8), August 1981.

[4] L. Gong: *A Security Risk of Depending on Synchronized Clocks*, ACM Operating Systems Review, 26(1), 1992.

[5] J.T. Kohl, B.C. Neuman: *The Kerberos Network Authentication Service*, Internet-Draft, Project Athena, MIT, June 1991.

[6] R.M. Needham, M.D. Schroeder: *Using Encryption for Authentication in Large Networks of Computers*, CACM, 21(12), December 1978.