A simple authorization protocol

Burton Rosenberg
University of Miami

Abstract

We present an authorization protol designed around the assumptions of a thin-client. The client does not need to generate random values, keep time or hold state. The protocol uses symmetric key encryption and a central key repository and permissions manager. Services are assured the identity and authorization of the user for the requested service. Users have some assurances concerning the action performed on their behalf. The protocol is presented and formally analyzed. The limitations of the assurances made to the user are discussed.

Introduction

We present an authorization protocol designed around the assumptions of a thin-client. In particular, we assume that the client does not have access to high quality random values, that it does not keep reliable time, and that it is inconvenient for it to hold state or to be personalized with user specific data. A typical scenario would be a small computer equiped with a SmartCard reader and a Web browser. Using the browser, the user converses with the service in an informal way and agrees upon an action. The protocol then provides for a central permissions manager to authorize the action and assures the service and the permissions manager of the identity of the user. The protocol requires four messages, only two of which involve the client.

While there is much in common with the classical authentication using symmetric key encryption, such as Needham-Schroeder [1, 2], Otway-Rees [3], and Kerberos [4], this protocol is designed to be light weight, but does not achieve all that these protocols achieve. In particular, the user does not authenticate the service, and is therefore useful only in situations of asymmetric trust relations. Such is the case, for instance, in the permission to access a printer. A user hardly ever complains about being given gratuitous permission to a printer, whereas the printer's owner generally wants controls.

The protocol uses symmetric key encryption. The authentication system could be incoporated in freeware products, and therefore needs to avoid expensive or legally encombered cryptographic products. Since in any case the system must refer to a central permission mananger to grant actions and accept liability, the inconveniences of symmetric key are offset by a transparent legal structure.

The protocol is presented in the next section. Following we transform and analyze the protocol using BAN logic [5]. We conclude with a discussion of the limitations of this protocol.

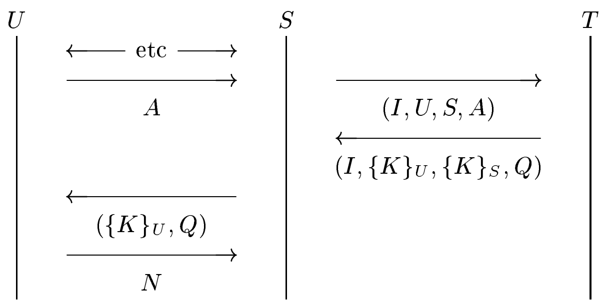

Figure 1: A simple authentication protocol

Description

Our protocol is diagramed in Figure 1. Let U, S and T represent the user, service and a trusted third party, respectively. The trusted third party functions as the permissions manager and shares keys with U and S. After a conversation between U and S, U requests action A. The service S tags the request with an identifier I and forwards it to T for authorization. If T authorizes the request, it generates a session key K, a timestamp t, and an authorization nonce N. It sends the reply $Q = \{U, S, A, N, t\}_K$, where $\{R\}_K$ denotes message R encrypted and integrity protected with the symmetric key K. The service will forward a copy of Q to U, along with $\{K\}_U$, and await the return of N in clear. If U does not agree that it asked A of S, it will not echo N. If t has expired, S will not honor the returned N.

Analysis of protocol

The protocol is analyzed using the method of Burrows, et. al. [5]. The notation is reviewed in the Appendix. Formalized, the protocol is reduced to three messages:

1. $T \longrightarrow U : \{M, N\}_U$.
2. $T \longrightarrow S : \{t, N, (U \mid\!\sim N \Rightarrow U \mid\!\equiv M)\}_S$.
3. $U \longrightarrow S : N$.

The request M reads: U wants S to do A. The permission N reads: T allows that S does A for U. In the formalization, U and S receive these messages unsolicited. The goal of the protocol is $U, S \mid\!\equiv N, M$.

Our axioms are:

1. $U \mid\!\equiv M$, else abort protocol.
2. $S \mid\!\equiv U \mid\!\Rightarrow M$.
3. $U \mid\!\equiv \sharp M$.
4. $S \mid\!\equiv \sharp t$.

5. $S \models T \Rrightarrow (U \mid\!\sim N \Rightarrow U \models M)$.
6. $U, S \models T \Rrightarrow N$.
7. $U \models U \overset{U}{\longleftrightarrow} T$.
8. $S \models S \overset{S}{\longleftrightarrow} T$.
9. $S \models U \overset{X}{\longleftrightarrow} S$ for secrets X shared only by U, S and T.

The calculation for U is not surprising, but it does use the freshness of M,

$$U \triangleleft \{M, N\}_U,\ U \models U \overset{U}{\longleftrightarrow} T \quad \Rightarrow \quad U \models T \mid\!\sim (M, N)$$

$$U \models T \mid\!\sim (M, N),\ U \models \sharp M \quad \Rightarrow \quad U \models T \models (M, N).$$

$$U \models T \models (M, N),\ U \models T \Rrightarrow N, \quad \Rightarrow \quad U \models N.$$

The calculation for S is more involved. Using the freshness of t, we find:

$$S \models T \models (t, (U \mid\!\sim N \Rightarrow U \models M), N).$$

Using the the belief of S of the jurisdiction of T, we conclude:

$$S \models (U \mid\!\sim N \Rightarrow U \models M), N.$$

With the arrival of N to S:

$$S \triangleleft N \text{ from } U,\ S \models U \overset{X}{\longleftrightarrow} S \text{ for } N \quad \Rightarrow \quad S \models U \mid\!\sim N.$$

$$S \models (U \mid\!\sim N \Rightarrow U \models M),\ S \models U \mid\!\sim N \quad \Rightarrow \quad S \models U \models M$$

$$S \models U \models M,\ S \models U \Rrightarrow M \quad \Rightarrow \quad S \models M.$$

Hence we conclude $U, S \models M, N$.

Conclusions

This protocol has the unique advantage of requiring little of the client. However, an important limitation must be noted.

Since a client machine stands between the user and the client, the client machine can hold two conversations, one with the service the user desires, and a second with a service the client desires, showing the user only one of the conversations, and the client the other. In formal terms, how can the user and client know that they share S and A before releasing N?

Even with no further improvements, this weakness is mitigated by the fact that the permissions manager holds a complete paper trail of any such deceit and can attempt a roll-back of damages once the deceit is detected. A more complete solution is left for future research.

Appendix

We collect the notation and inference rules for BAN logic.

$A \mid\equiv N$	A believes N
$A \mid\sim N$	A has said N
$A \triangleleft N \mathit{from}\ B$	A sees N from B
$\sharp N$	N is fresh
$A \overset{K}{\longleftrightarrow} B$	A and B share secret key K
$\{N\}_K$	N is encrypted and integrity protected with K
$A \mid\Rightarrow N$	A has jurisdiction over N

The message-meaning rule:

$$\text{If } A \triangleleft \{N\}_K \text{ from } B, \text{ and } A \mid\equiv A \overset{K}{\longleftrightarrow} B, \text{ then } A \mid\equiv B \mid\sim N.$$

The nonce-verification rule:

$$\text{If } A \mid\equiv B \mid\sim N, \text{ and } A \mid\equiv \sharp N, \text{ then } A \mid\equiv B \mid\equiv N.$$

The jurisdiction rule:

$$\text{If } A \mid\equiv B \mid\equiv N, \text{ and } A \mid\equiv B \mid\Rightarrow N, \text{ then } A \mid\equiv N.$$

References

[1] Roger Needham, Schroeder, M. D., *Using encryption for authentication in large networks of computers,* Comm. of the ACM, Vol. 21. No 12., pp. 993–999, 1978.

[2] Roger Needham, Schroeder, M. D., *Authentication revisited,* Op. Sys. Review, Vol. 21, No. 1 p, 7. 1987.

[3] Otway, D., Rees, O. *Efficient and timely mutual authentication,* Op. Sys. Review, Vol 21, No. 1 pp 8–10. 1987.

[4] Miller, S. P., Neuman, C., Schiller, J. I., Saltzer, J. H., *Kerberos authentication and authorization systems,* Proj. Athena Tech. Plan, Section E.2.1., MIT, 1987.

[5] Michael Burrows, Martin Abadi, Roger Needham, *A logic of authentication,* SRC Research Report 39, DEC. 1989.