A novel stateless authentication protocol

Chris J. Mitchell

Information Security Group
Royal Holloway, University of London
http://www.chrismitchell.net
me@chrismitchell.net

Abstract. The value of authentication protocols which minimise (or even eliminate) the need for stored state in addressing DoS attacks is well-established — the seminal paper of Aura and Nikander [1] is of particular importance in this context. However, although there is now a substantial literature on this topic, it would seem that many aspects of stateless security protocols remain to be explored. In this paper we consider the design of a novel stateless authentication protocol which has certain implementation advantages. Specifically, neither party needs to maintain significant stored state. The protocol is developed as a series of refinements, at each step eliminating certain undesirable properties arising in previous steps.

1 Introduction

It has long been recognised that a requirement for stored state is an undesirable feature in almost any protocol (although the absence of statefulness can also cause practical problems, as has been the case with HTTP). This is particularly significant for security protocols, in which a requirement for stored state can be used as a means of launching denial of service (DoS) attacks (see, for example, section 1.6.6 of Boyd and Mathuria [2]). During the 1990s considerable efforts were made to devise authentication and key establishment protocols which minimise the requirements for stored state at the server in client-server protocols; one major objective of this work was to minimise the threat of DoS attacks.

Whilst preventing exhaustion of table space was the original motivation for state elimination, there are other good reasons to minimise requirements for stored state. For example, state minimisation can greatly simplify network protocols by reducing the complexity of the associated state machines. The cost is typically slightly longer messages (since messages become the new repository of state). Of course, this observation is not new at all — indeed, HTTP cookies are hardly a revolutionary new idea!

In 1997, Aura and Nikander published a key paper on this topic [1]. They describe how protocols can be made stateless by 'passing the state information between the protocol principals along[side] the messages'. Such state information (forming a cookie — as in HTTP) can be protected using a MAC computed using a secret key known only to the server.

Oakley, a protocol proposed for use in the Internet, and which also can be configured to avoid the need for server state, was proposed at around the same time by Orman [3]. Photuris, another session key management protocol that can be regarded as a development of Oakley, due to Karn and Simpson, is defined in RFC 2522 [4]. A much broader discussion of some of the anti-DoS measures incorporated in these and other similar protocols is provided by Oppliger [5].

The emphasis of past state-minimisation work has primarily focussed on eliminating stored state at the server. However, in the new world of transient relationships and peer/peer communications (which are not restricted to the client/server paradigm), it is necessary to try to protect both parties engaging in a protocol. This provides the motivation for the work described in this paper, i.e. to design an authentication protocol which minimises (ideally eliminates) stored state for *both* parties in a secure interaction.

2 A Simple Scheme

One simple possibility would be to use a time-stamp based protocol. A simple example of such a protocol is the following one-pass protocol, in which entity B is authenticated by entity A:

$$B \to A : \ t_B || f_{K_{AB}}(t_B || i_A)$$

In the above description, $||$ denotes concatenation[1], t_B is a timestamp generated by B, f is a MAC function, K_{AB} is a secret key shared by A and B, and i_A is an identifier for A. Such protocols are widely known and have been carefully analysed. The protocol specified is actually standardised in ISO/IEC 9798-4 [8] and similar protocols have been discussed in the literature (see, for example, Protocol 3.4 of Boyd and Mathuria [2]). This protocol can be used twice to achieve mutual authentication.

This approach requires securely synchronised clocks. This may cause serious problems in practice; in particular, this does not seem like a good solution for a transient relationship scenario. In the absence of an existing security management infrastructure, it is unclear who would define how clocks should be synchronised. In any case, this protocol, like any timestamp-based protocol, does not prevent replays within a short time window of the original transmission (see, for example, section 10.4.1 of [9]).

One way of avoiding the use of timestamps (and the associated problems) is to use a key idea from the Aura-Nikander paper [1]. Whilst the emphasis in this paper was on eliminating server state, the ideas presented there work just as well in eliminating client state. The key idea is to pass ‘the state information between the protocol principals along[side] the messages’. This motivates the work we describe next.

[1] As discussed in [6, 7], this concatenation must be done in such a way that the resulting data string can be uniquely resolved into its constituent data strings, i.e. so that there is no possibility of ambiguity in interpretation.

3 Working Towards a Solution

We use shared secret-based unilateral authentication protocols throughout as simple examples. It seems reasonable to suppose that these protocols can be extended and modified to use asymmetric cryptography and/or to provide mutual authentication.

3.1 Putting a Nonce into a Cookie

We first consider the following two-pass (stateless) nonce-based unilateral authentication protocol:

$$\begin{aligned} A \to B &: n_A || f_{K_A}(i_B || n_A) \\ B \to A &: n_A || f_{K_A}(i_B || n_A) || f_{K_{AB}}(n_A || i_A) \end{aligned}$$

where n_A is a nonce chosen by A, K_A is a key known only by A (and used only for generating and verifying cookies), and other notation is as before. The string '$n_A || f_{K_A}(i_B || n_A)$' functions as a cookie for the nonce n_A, which is sent with the nonce to avoid A having to remember it (cf. Photuris [4] and/or page 27 of Boyd and Mathuria [2]). This protocol is based on a 2-pass protocol standardised in ISO/IEC 9798-4 [8], where the modification essentially transfers A's stored state into the message. When A receives the response from B, A can first process the cookie to recover its stored state prior to verifying the response from B.

One advantage of this protocol is that A now only has to remember a single secret K_A, which can be the same for all interactions with other parties. The main remaining problem is that A cannot verify that the cookie $[n_A || f_{K_A}(i_B || n_A)]$ is fresh. That is, B can use the cookie to send a series of responses, all of which will be accepted. Even worse, a third party could intercept and replay B's original response, which will also be accepted.

Of course, as is stated on page 27 of [2], the cookie could be bound to the connection. However, if A allocates a new number for each authentication protocol instance, makes the cookie a function of this number, and keeps a log of current protocol instances, then this simply makes the protocol stateful again!

3.2 Using a Timestamp

One solution to this problem would be to use a timestamp instead of a nonce in a two-pass protocol. One exchange of this type is as follows:

$$\begin{aligned} A \to B &: t_A \\ B \to A &: t_A || f_{K_{AB}}(t_A || i_A) \end{aligned}$$

where t_A is a timestamp chosen by A, and other notation is as before. It is important to note that, just because the scheme uses a timestamp, does not mean that synchronised clocks are needed. Only A checks the timestamp that it generates itself.

Unfortunately, this scheme allows Gong-style preplay attacks (cf. pages 22/23 of [2]), as follows. Suppose C wishes to impersonate B to A at some future time. C (pretending to be A) engages in the protocol with B, using a future value of A's clock. C can now replay this message to A at the future specified time, and successfully impersonate B.

4 The Main Proposal

The scheme we now propose combines the ideas introduced in the previous subsection, i.e. to use cookies and a timestamp-based nonce. That is, we use the same scheme as just described except that A also computes and sends a cookie with the nonce/timestamp (which must also be sent back by B). B responds with a MAC on the concatenation of the nonce/timestamp, the cookie, and the name of A, and also sends copies of the nonce/timestamp and the cookie. A can verify the cookie, verify the MAC, and finally verify the freshness of the timestamp.

An example protocol of this type is as follows:

$$\begin{aligned} &A \to B : t_A || f_{K_A}(i_B || t_A) \\ &B \to A : t_A || f_{K_A}(i_B || t_A) || f_{K_{AB}}(t_A || i_A || f_{K_A}(i_B || t_A)) \end{aligned}$$

where the notation is as before. As previously, this scheme does not require A's clock to be synchronised with anyone. In fact, the timestamp could simply be generated using a continuously clocked counter. Use of the cookie prevents preplay attacks involving a third party.

The inclusion of a session identifier in the cookie would enable A to match the response from B to a higher-layer protocol communications request (e.g. from an application).

However, replays within a time window are still possible. There are two obvious ways of fixing this. The first (and the widely discussed solution to this problem) would be to keep a log of recently accepted messages. In our context this solution is not so elegant, since it re-introduces state, albeit of a bounded size. The second solution would be to keep track of the timestamp/counter for the most recently received (accepted) message and to only accept 'newer' messages. This also involves the maintenance of state, but in this case of fixed size, regardless of the number of concurrent communications partners.

The latter approach could be generalised to the maintenance of a list of information regarding accepted messages for recent timestamps ('recent' is a well-defined concept as long as the counter increases at a roughly steady rate). Again, although this is stored state, this should not be such a big problem since it will only contain information about valid messages received, which can be removed as soon as the timestamp 'expires'.

5 Concluding Remarks

We have proposed an authentication protocol which minimises state for both parties in an authentication exchange. In particular, the client only needs to maintain a small list of values which does not grow as the number of parallel communications sessions grows. This is at the cost of slightly longer messages. The scheme builds on ideas of Aura and Nikander, [1].

Although we have proposed a stateless authentication protocol, there are many unresolved issues, including the following.

– The scheme we have proposed provides only unilateral authentication (of B to A). It would be interesting to devise a mutual authentication scheme. Of course, this could be achieved by simply using two instances of the scheme we have proposed, although it would be interesting to see if efficiencies could be achieved by devising a more integrated protocol.
– In this paper we have almost exclusively considered schemes based on the use of MACs. Whilst MACs are simple to compute and are widely used, a wide range of other cryptographic primitives can be used to achieve entity authentication. It would therefore be interesting to see if similar protocols could be devised using, for example, digital signatures.
– The presentation given here has been completely informal, and no attempt has been made to establish the security of the proposed schemes. Whilst this is reasonable when looking at what might be achieved, if schemes of this type are to be used in practice then it would be highly desirable to prove them secure in an appropriate model (first addressing any identified issues, where necessary).
– Finally, it would also be interesting to see how protocols of this type could be integrated into practical protocols.

More generally, it would be interesting to consider the applicability of stateless (cookie-based) protocols to a variety of communications models. For example, if all the interactions in the application are request-response, then stored state may be completely unnecessary from the application perspective. Even where a connection is set up, only a party wishing to initiate message transmissions, rather than responding to a request, needs to maintain state. A security protocol that avoids state would be particularly apposite in such applications.

6 Acknowledgements

The ideas described here evolved out of helpful discussions with Colin Boyd and Andreas Pashalidis, although any errors and omissions are entirely my fault.

References

1. Aura, T., Nikander, P.: Stateless connections. In Han, Y., Okamoto, T., Qing, S., eds.: Information and Communication Security, First International Conference,

ICICS '97, Beijing, China, November 11–14, 1997, Proceedings. Volume 1334 of Lecture Notes in Computer Science., Springer-Verlag, Berlin (1997) 87–97

2. Boyd, C.A., Mathuria, A.: Protocols for key establishment and authentication. Springer-Verlag (2003)
3. Orman, H.: RFC 2412, The OAKLEY key determination protocol. Internet Engineering Task Force. (1998)
4. Karn, P., Simpson, W.: RFC 2522, Photuris: Session-key management protocol. Internet Engineering Task Force. (1999)
5. Oppliger, R.: Protecting key exchange and management protocols against resource clogging attacks. In Preneel, B., ed.: Secure Information Networks: Communications and Multimedia Security, IFIP TC6/TC11 Joint Working Conference on Communications and Multimedia Security (CMS '99), September 20-21, 1999, Leuven, Belgium. Volume 152 of IFIP Conference Proceedings., Kluwer (1999) 163–175
6. Chen, L., Mitchell, C.J.: Parsing ambiguities in authentication and key establishment protocols. Journal of Electronic Security and Digital Forensics **3** (2010) 82–94
7. International Organization for Standardization Genève, Switzerland: ISO/IEC 9798–4: 1999/Cor 1:2009, Technical Corrigendum 1. (2009)
8. International Organization for Standardization Genève, Switzerland: ISO/IEC 9798–4: 1999, Information technology — Security techniques — Entity authentication — Part 4: Mechanisms using a cryptographic check function. 2nd edn. (1999)
9. Dent, A.W., Mitchell, C.J.: User's Guide to Cryptography and Standards. Artech House, Boston, MA (2005)