Relations Between Secrets:
The Yahalom Protocol

Extended Abstract

Lawrence C. Paulson

Computer Laboratory, University of Cambridge
Pembroke Street, Cambridge CB2 3QG, England
lcp@cl.cam.ac.uk

1 Introduction

There is something subtle about the Yahalom protocol. Protocols such as Otway-Rees [3] distribute certificates, signed by a trusted authority. Each principal typically receives a session key packaged with a nonce to ensure freshness. But in Yahalom, principal B receives two certificates. One contains a key but no evidence of freshness, while the other is signed using the same doubtful key. To accept the latter certificate as evidence of freshness for the key requires a convoluted argument. It relies on the secrecy of the nonce Nb, which is encrypted using the very key in question; that it still works is surprising.

The Yahalom protocol is largely of academic interest, but equally awkward protocols have been deployed. Kerberos version IV [1] uses session keys to encrypt other session keys. If one session key is compromised, many others could be lost. Despite this vulnerability, the protocol can be analyzed using essentially the same technique that proves the secrecy of Nb in Yahalom. The idea is to reason formally about the relation that holds between the secrets.

The technique is based upon the inductive method [5], which models a protocol as the set of traces that could arise over time. Agents drawn from an infinite population may engage, playing various roles, in any number of possibly interleaved protocol runs. The formal definition resembles informal protocol notation, but contains additional rules to allow the empty trace, enemy action and accidental security breaches. Properties are proved by induction over this definition. If the inductive argument appears not to hold, one can easily identify the offending rule and the circumstances under which the desired property fails. Then one must generalize the induction formula, prove further lemmas to bridge the gap in the reasoning, or look for a weakness in the protocol.

2 The Yahalom Protocol

This protocol, described by Burrows et al. [2, page 257], distributes a session key Kab to parties A and B with the help of a trusted authentication server. At

* The full version of this paper, including details of the Isabelle proofs, is available [4].
the end of a run, each party can be sure that the other was recently present.

1. \( A \rightarrow B : A, Na \)
2. \( B \rightarrow \text{Server} : B, \{ A, Na, Nb \}_{Kb} \)
3. \( \text{Server} \rightarrow A : \{ B, Kab, Na, Nb \}_{Ka}, \{ A, Kab \}_{Kb} \)
4. \( A \rightarrow B : \{ A, Kab \}_{Kb}, \{ Nb \}_{Kab} \)

Now we can see in detail why Yahalom is problematical. When \( B \) receives the fourth message, he obtains a session key from the certificate \( \{ A, Kab \}_{Kb} \), but it does not mention \( Nb \) and could therefore be a replay of an old message. Freshness evidence comes from \( \{ Nb \}_{Kab} \), but why should \( B \) trust a certificate that is signed with an old, possibly compromised key?

The protocol is correct because \( Nb \) is kept secret: only \( A \) could have formed \( \{ Nb \}_{Kab} \), so \( A \) associates \( Kab \) with the fresh nonce. Proving that \( Nb \) remains secret is harder than it looks. In an ideal model, one could prove that \( Kab \) always remains secret, and the secrecy of \( Nb \) would follow immediately for all runs between uncompromised agents. Such reasoning is faulty. It could ‘verify’ a version of Yahalom that sent \( Nb \) in clear in message 2:

2. \( B \rightarrow \text{Server} : B, Nb, \{ A, Na \}_{Kb} \)

But this version can be attacked. Suppose an intruder \( I \) has managed to crack one of \( B \)’s old certificates \( \{ A, K \}_{Kb} \), extracting the session key \( K \). He can then masquerade as \( A \), using \( Nb \) to forge message 4:

1. \( I_A \rightarrow B : A, Nc \)
2. \( B \rightarrow I_{\text{Server}} : B, Nb, \{ A, Nc \}_{Kb} \)
4. \( I_A \rightarrow B : \{ A, K \}_{Kb}, \{ Nb \}_{K} \)

We must be realistic. Old session keys or nonces do sometimes leak out. The inductive model admits such accidents. In order to analyze the protocol, we must then examine the associations between session keys and nonces in the runs of a Yahalom trace. If a particular key \( Kab \) is secret then \( Nb \) is secret too, even if other keys or nonces are compromised. The proof is long and detailed—as it must be in a non-trivial model.

3 Formalization of the Protocol

The Isabelle formalization (omitted here) follows the usual conventions of the inductive method. The Oops rule is critical to the discussion below. This rule can hand the triple \( \{ Na, Nb, Kab \} \) to the spy if the server has used these nonces and key together in a run. Oops is intended to model compromise of the session key by any means. Including the nonces in the message identifies the run associated with the key. With Yahalom, however, the loss of \( Nb \) is itself a breach of security. Our guarantee to \( B \) will say that—provided the run involving \( Nb \) has not been compromised in this way—the session key is both fresh and secure.