The Modelling and Analysis of Security Protocols
Notes for Lecture 1

Tom Chothia
T.Chothia@cwi.nl

I. MODELLING PROTOCOLS: THE BASICS
Protocols describe how communication between computers takes place. While most protocols are simple enough to be written down in a few lines, designing correct protocols is notoriously hard.

A. Course Information
You will get handouts that contain all the important information for each lecture. All the handouts, lecture slides, exercises and related papers and links for this course can be found at:
http://homepages.cwi.nl/~chothia/Teaching
I work in Amsterdam, so I will not have office hours in Leiden, but you may e-mail me with course related questions at the above email address.

B. Modelling the Protocol
We model protocols at a very high level; we do not look at how the communications are actually made. This is because we want to find bugs in the protocol design, not bugs in a single implementation of that protocol.

Protocols are made up of a number of steps. In each step a principal, sends a message to another principal. We write the principal “A” sending the message “M” to “B” as:

A → B : M

1) Message Types: We only model the really important parts of the protocol, which normally are:

- Principal names: A, B, S, E, ...
- Keys: K, K_a, K_ab, pri, pub, ...
- Nonces: N, N_a, N_b, ...
- Encryptions: symmetric key: {M}_K, {A, K_a}_K_ab, ...
- public key: E_p(M), ...
- Signed messages: S_K(M), MAC_K(M), {M}_{prv(A)}, ...

and sometimes timestamps (T, T_a) and numbers (L, N, ...).

Definition 1 (nonce): A nonce is a “fresh” message, e.g. a random number. We consider each nonce to be unique.

Nonces are used to make sure that messages are “fresh”, i.e., not a message from an old run of the protocol that is being replayed. If you generate and send a new nonce and later you see a message that contains your nonce then the message must have been put together after you sent your nonce.

2) Protocol Aims: The aim of a protocol is usually either:
- transfer of security data
- key establishment
- principal authentication

We will see more about the different security goals for these types of protocol next time.

C. Example protocol:
The Needham-Schroeder authentication protocol uses public key encryption:

1) A → B : E_B(N_a, A)
2) B → A : E_A(N_a, N_b)
3) A → B : E_B(N_b)

This is a well know protocol for authentication; at the end of the run N_a and N_b should be known only to A and B. These values can then be used to generate a session key.

D. The Attacker
Protocols must defend against an attacker that...
- may read all the messages sent across the network.
- may delete messages passing over the network.
- may create and send new messages.
- may act as one on more of the principals of the protocols.
- may run multiple sessions of the same protocols.
- may not perform a brute force attack on the encryption.

The Needham-Schroeder authentication protocol is not safe: the attacker (C) acts as a man-in-the-middle and starts a second run of the protocol. This lets C pretend to be A.

1. A → C : E_C(N_a, A)
2. C → A : E_A(N_a, N_b)
3. C → C : E_C(N_b)

Changing line 2 of the protocol can stop this attack:

2. B → A : E_A(N_a, N_b, B)
E. Kerberos

Kerberos is a widely used authentication protocol, it starts with A and B both sharing symmetric keys \( (K_{ab}, K_{as}) \) with a trusted third part \( S \) but not with each other. Both \( A \) and \( B \) trust \( S \) to generate a new key. After the protocol run \( A \) and \( B \) share the key \( K_{ab} \):

1. \( A \to S : \{A, B, N_a\} \)
2. \( S \to A : \{K_{ab}, B, L, N_a\}_{K_{as}}, \{K_{ab}, A, L\}_{K_{as}} \)
3. \( A \to B : \{A, T_a\}_{K_{as}}, \{K_{ab}, A, L\}_{K_{as}} \)
4. \( B \to A : \{T_a + 1\}_{K_{as}} \)

1) First \( A \) sends a request for a new key to the server. The request includes \( A \)'s identity, the identity of the principal that \( A \) wishes to talk to \( (B) \) and a random nonce.

2) The server generates a new key for \( A \) and \( B \) called \( K_{ab} \). It returns this key to \( A \) along with the nonce \( N_a \) (which proves that the message really is a reply to \( A \)'s request), the identity that \( A \) requested communication with (so that no one can impersonate \( B \)) and an expiry time for the session key \( L \). The server also sends a certificate \( \{K_{ab}, A, L\}_{K_{as}} \) that \( A \) can send to \( B \) in order to prove to \( B \) that the key \( K_{ab} \) is good to use to communicate with \( A \).

3) In the third message \( A \) sends this certificate to \( B \) along with a challenge consisting of \( A \)'s identity and a timestamp \( T_a \) encrypted with their new key \( K_{ab} \). When \( B \) receives this message it decrypts the part encrypted by \( S \) and checks the expiry time \( L \), this lets \( B \) know that the key \( K_{ab} \) really did come from \( S \) and is fresh, it then decrypts the part of the message from \( A \) and checks that the identities match.

4) Finally, \( B \) sends a message that back to \( A \) using the session key, to prove to \( A \) that it does have a correct copy of the key.

II. Questions

These questions are for you to test your understanding. I will look at answers but they are not compulsory.

A. Find a Protocol Attack

The following protocol lets \( B \) authenticate \( A \) using a trusted server \( S \):

1. \( A \to B \) : \( A \)
2. \( B \to A \) : \( N_b \)
3. \( A \to B \) : \( \{N_b\}_{K_{as}} \)
4. \( B \to S \) : \( \{A, \{N_b\}_{K_{as}}\}_{K_{bs}} \)
5. \( S \to B \) : \( \{N_b\}_{K_{bs}} \)

1) How does this protocol work? What should \( B \) believe at the end of the protocol run?
2) Assume that there is an attacker \( E \) that shares the key \( K_{es} \) with the server. Show how, by starting two runs of the protocol at the same time an attacker can trick \( B \) into believing that it is \( A \).

B. Design a Protocol

Design a key establishment protocol, similar to the Kerberos protocol, in which \( A \) and \( B \) use a trusted server \( S \) to agree a new session key \( K_{ab} \). You may assume that \( A \) and \( S \) share the key \( K_{as} \) and that \( B \) and \( S \) share the key \( K_{bs} \). Design your protocol such that \( A \) initiates communication to the server \( S \) and the server then makes first contact with \( B \).

The attacker \( E \) might also be allowed to use the key server and has the shared key \( K_{es} \). Be sure that an attacker cannot trick \( B \) into believing that they are \( A \).

1) Write down your protocol using the \( A \to S : M \) notation and state any important assumptions you are making.
2) Why do people prefer not to implement protocols in which the server initiates contact with other principals?