Foundations of Computer Security
Lecture 65: The BAN Logic: Needham-Schroeder

Dr. Bill Young
Department of Computer Sciences
University of Texas at Austin
Recall the Needham-Schroeder protocol:

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

Needham-Schroeder is idealized as follows:

1. omitted since all components are plaintext
2. \( S \rightarrow A : \{N_a, (A \xleftrightarrow{K_{ab}} B), \#(A \xleftrightarrow{K_{ab}} B), \{A \xleftrightarrow{K_{ab}} B\}\}_{K_{bs}} \) \( K_{as} \)
3. \( A \rightarrow B : \{A \xleftrightarrow{K_{ab}} B\}_{K_{bs}} \)
4. \( B \rightarrow A : \{N_b, (A \xleftrightarrow{K_{ab}} B)\} \) \( K_{ab} \) from \( B \)
5. \( A \rightarrow B : \{N_b, (A \xleftrightarrow{K_{ab}} B)\} \) \( K_{ab} \) from \( A \)
The following initial assumptions are given for Needham-Schroeder:

\[ A \equiv A \xleftarrow{Kas} S \quad B \equiv B \xleftarrow{Kbs} S \quad S \equiv A \xleftarrow{Kas} S \]

\[ S \equiv B \xleftarrow{Kbs} S \]

\[ S \equiv A \xleftarrow{Kab} B \]

\[ A \equiv (S \rightarrow A \xleftarrow{K} B) \quad B \equiv (S \rightarrow A \xleftarrow{K} B) \]

\[ A \equiv (S \rightarrow \#(A \xleftarrow{K} B)) \]

\[ A \equiv \#(N_a) \quad B \equiv \#(N_b) \quad S \equiv \#(A \xleftarrow{Kab} B) \]

\[ B \equiv \#(A \xleftarrow{K} B) \]

The very last of these is pretty strong. Needham and Schroeder did not realize they were making it, and were criticized for it.
From step 2 of the (idealized) protocol:

\[
A \triangleleft \{ N_a, (A \xleftarrow{K_{ab}} B), \#(A \xrightarrow{K_{ab}} B), \{A \xleftarrow{K_{ab}} B\}K_{bs}\}K_{as}
\]

The \textit{Nonce Verification Rule} says:

\[
\frac{A \equiv (\#(X)), A \equiv (S \sim X)}{A \equiv (S \equiv X)}
\]

Since A believes \(N_a\) to be fresh, we get:

\[
A \equiv (S \equiv A \xleftarrow{K_{ab}} B)
\]
The *Jurisdiction Rule* says that:

\[ A \equiv (S \implies X), A \equiv (S \equiv X) \]

\[ \implies A \equiv X \]

From this we obtain:

\[ A \equiv A \xleftarrow{K_{ab}} B \]

\[ A \equiv \#(A \xleftrightarrow{K_{ab}} B) \]
Since A has also seen the part of the message encrypted under B’s key, he can send it to B. B decrypts the message and obtains:

\[ B \equiv (S \sim A \xleftarrow{K_{ab}} B) \]

meaning that B believes that S once sent the key.

At this point, we need the final dubious assumption:

\[ B \equiv \#(A \xleftarrow{K} B) \]

With it, we can get:

\[ B \equiv A \xleftarrow{K_{ab}} B \]
From the last two messages, we can infer the following. How?

\[ A \equiv A \leftrightarrow^{K_{ab}} B \]

\[ B \equiv A \leftrightarrow^{K_{ab}} B \]

\[ A \equiv (B \equiv A \leftrightarrow^{K_{ab}} B) \]

\[ B \equiv (A \equiv A \leftrightarrow^{K_{ab}} B) \]

These are the point of the protocol. The proof exhibits some assumptions that were not apparent.
Lessons

Use of a logic like BAN shows what is provable and also what must be assumed.

Using BAN effectively requires a lot of practice and insight into the protocol.

Next lecture: PGP