6.915 Computer and Network Security	Fall Term, 1995
Lecture 8 : October 3, 1995	
Lecturer: Ron Rivest	*Scribe: Yoav Yerushalmi*

1 Outline

A great concern with most security-oriented systems is the need to 'prove' those systems secure. For example, in a distributed system, we have much complicated communications occuring, and many possible adversaries. It would be nice to have a technique that can help us test and assure the security of protocols, preferably even proving them to be secure. This lecture focuses on a specific method proposed to do just that: the *BAN Logic* system. Furthermore, we will analyze one particular protocol, Kerberos, and see if we can show it to be secure.

2 Proving Security

In order to "prove" the security of a cryptographic protocol, we need to be able to model it mathematically. We define a language to deal with everything: Principals, Messages (statements and beliefs), time (and how things change over time), encryption (public or secret), secrets and who might have access to them, and of course, adversaries.

After we know what we're dealing with, we have two ways to prove our system secure: We can either prove that the only way the adversary can break our security is by solving a hard problem (proof by reduction), or we can use logic rules to end up with some stated conclusion. Although the proof by reduction is usually sufficient when talking about a cryptosystem (for example, finding d in RSA), it is sometimes more convenient to use *BAN* or some other system to prove the security of the protocol instead.

What do we get from using special logic (i.e., *BAN*)? Well, we end up with a formal system for explaining what it is we want to ensure. It lets us derive new statements from others via deduction rules. It forces us to have expressive notation and be able to describe exactly what is going on.

There are concerns that one must address should one choose to use the second method. Are we sure we are proving what we need to prove? Do we know that the proof is sound (is stuff that is provable true?). Are our axioms and rules true? All these concerns need to be addressed if one wishes to create a special logic system for formal proofs of security.

Finally, when we are sure we have such a system (for example *BAN*) we need to use it properly. If we CAN'T prove what we want, we probably have a problem with the protocol. If we CAN, there might still be problems (implementation for example). One thing to always do is **be explicit**.

3 The BAN system

The *BAN System* was proposed by Michael Burrows, Martin Abadi, and Roger Needham as one such logic system for authentication protocols. It attempts to help answer the following questions about protocols:

- Does this protocol work? Can it be made to work?
- Exactly what does this protocol achieve?
- Does this protocol need more assumptions than another protocol?
- Does this protocol do anything unnecessary?

the following sections are straight from their paper.

3.1 The *BAN* terms

In order to be able to describe what is going on, we need to be able to refer to things formally. The following terms are used in *BAN*:

Principals are those agents involved in the protocol (usually people or programs). P, Q, R are all possible symbols for the principals.

Keys are the things used to encrypt messages symmetrically. K_{ab} and K_{as} are possible keys.

Public Keys are similar except that they are used in pairs, one of which is known to all. (K_b, K_b^{-1}) is a possible pair.

Nonces are message parts that are not meant to be repeated. N_a, N_b are nonces.

Timestamps are similar to nonces in that they are unlikely to be repeated. However, they reflect the current time of the person creating them. T_a, T_b are timestamps.

These terms are then used to compose the assumptions and deductions necessary for the proof.

3.2 The *BAN* Statements

Once we have a formal definition of terms, we need a formal definition of the statements we may make (the constructs)

P **believes** X : P *believes* X, or P would be entitled to believe X. In particular, P may act as though X is true.

P **sees** X : P *sees* X. Someone has sent a message containing X to P, who can read and repeat X.

P **said** X : P *once said* X. The principal P at some time sent a message including the statement X. It is not known whether this is a replay, though it is known that P believed X when he sent it.

P **controls** X : P has *jurisdiction over* X. The principal P is an authority on X and should be trusted on this matter. For example, a server is often trusted to generate encryption keys properly. One can represent that as the server controlling the keys.

fresh(X) : The formula X is *fresh*; that is, X has not been sent in a message at any time before the current run of the protocol. This is usually true for *nonces*.

$P \overset{K}{\leftrightarrow} Q$: P and Q may use the *shared key* K to communicate. The key K is good in that it will never be discovered by any principal except P and Q.

$\overset{K}{\mapsto} P$: P has K as a *public key*. The matching *secret key* (denoted K^{-1}) will never be discovered by any principal excpet P or a principal trusted by P.

$P \stackrel{X}{\rightleftharpoons} Q$: The formula X is a *secret* known only to P and to Q, and possibly to principals trusted by them. Only P and Q may use X to prove their identities to one another. An example is a secret password.

$\{X\}_K$ **or** $[X]_K$: This represents the formula X encrypted under the key K. This is short for $[X]_K from P$.

$\langle X \rangle_Y$: This represents X *combined with* the formula Y. Y is intended to be secret and that its presence proves the identity of whoever utters $\langle X \rangle_Y$. In implementations, X can simply be concatenated with the password Y. We assume that this is encrypted so that replay cannot be used.

With those basic constructs, we can now create a few basic rules of deduction which we can then use in our analysis.

3.3 The *BAN* Rules of Deduction

We have five basic rules which we use to reach conclusions. Those are:

1. *Message meaning rules* concern the interpretation of messages. They all derive beliefs about the origin of messages.

   For shared keys, we postulate

   $$\frac{P \text{ believes } Q \stackrel{K}{\leftrightarrow} P, P \text{ sees } [X]_K}{P \text{ believes } Q \text{ said } X}$$

   That is, if P believes that the key K is shared with Q and sees X encrypted under K, then P believes that Q once said X.

   For public keys, we postulate

   $$\frac{P \text{ believes } \stackrel{K}{\mapsto} Q,\ P \text{ sees } [X]_{K^{-1}}}{P \text{ believes } Q \text{ said } X}$$

   That is, if P believes that K is Q's public key, and P receives a message encoded with Q's secret key, then P believes Q once said X.

   For shared secrets, we postulate

   $$\frac{P \text{ believes } Q \stackrel{Y}{\rightleftharpoons} P,\ P \text{ sees } \langle X \rangle_Y}{P \text{ believes } Q \text{ said } X}$$

That is, if P believes that the secret Y is shared with Q and sees $\langle X\rangle_Y$, then P believes that Q once said X.

2. The *nonce-verification* rule expresses the check that a message is recent, and hence, that the sender still believes in it:

$$\frac{P \text{ believes fresh } (X),\ P \text{ believes } Q \text{ said } X}{P \text{ believes } Q \text{ believes } X}$$

That is, if P believes that X could have been uttered only recently and that Q once said X, then P believes that Q believes X. For simplicity, X should be "cleartext".

3. The *jurisdiction* rule states that if P believes that Q has jurisdiction over X, then P trusts Q on the trueth of X:

$$\frac{P \text{ believes } Q \text{ controls } X,\ P \text{ believes } Q \text{ believes } X}{P \text{ believes } X}$$

4. If a principal sees a formula, the he also sees its components, provided he knows the necessary keys:

$$\frac{P \text{ sees } (X,Y)}{P \text{ sees } X}, \quad \frac{P \text{ sees } \langle X\rangle_Y}{P \text{ sees } X}, \quad \frac{P \text{ believes } Q\overset{K}{\leftrightarrow}P(,)P \text{ sees } [X]_K}{P \text{ sees } X},$$
$$\frac{P \text{ believes } \overset{K}{\mapsto}P,\ P \text{ sees } [X]_K}{P \text{ sees } X}, \quad \frac{P \text{ believes } \overset{K}{\mapsto}P,\ P \text{ sees } [X]_{K^{-1}}}{P \text{ sees } X}.$$

Note that if P sees X and P sees Y it does NOT follow that P sees (X,Y) since that means that X and Y were uttered at the same time.

5. If one part of the formula is fresh, then the entire formula must be fresh:

$$\frac{P \text{ believes fresh}(X)}{P \text{ believes fresh}(X,Y)}.$$

With those postulates, one can proceed to using this system for proving the security of a protocol, for example, Kerberos.

4 Kerberos

Now that we have a formal proof system, let's see how well it works on a real-world protocol.

4.1 The protocol

The kerberos protocol is described in Figure 1

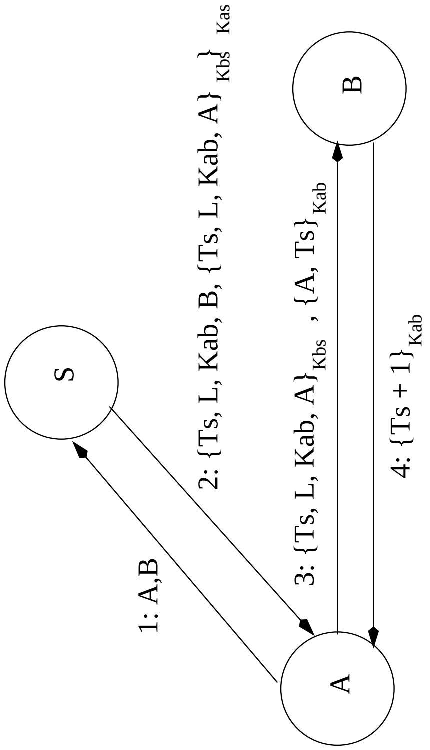

Figure 1: The Kerberos Protocol.

In this system, there is a server S, which has shared secrets with every participant. The object of the protocol is to allow every participant to mutually authenticate to each other.

In order to achieve this, the following idealization is studied:

1. $A \rightarrow S : (A, B)$ – A sends a request for an authenticator for B to S.

2. $S \rightarrow A : [T_s, A \overset{K_{ab}}{\leftrightarrow} B, [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{bs}}]_{K_{as}}$ – S replies with a message only decodable by A which contains a timestamp and an authenticator A can send to B to demonstate he is real.

3. $A \rightarrow B : [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{bs}}, [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{ab}}$ – sends an initial message to B telling him he is there, has a valid authenticator, and wants to know if he knows how to decode it and reply.

4. $B \rightarrow A : [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{ab}}$ – Finally, B demonstrates that he can parse the authenticator and sends back a reply to A's challenge.

4.2 The proof : assumptions and goals

Now that we have defined the protocol, we list out basic assumptions and what we are aiming to prove. Our assumptions are:

$$\begin{array}{rl}
A \text{ believes } A \overset{K_{as}}{\leftrightarrow} S, & B \text{ believes } B \overset{K_{bs}}{\leftrightarrow} S, \\
S \text{ believes } A \overset{K_{as}}{\leftrightarrow} S, & S \text{ believes } B \overset{K_{bs}}{\leftrightarrow} S, \\
S \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B, & B \text{ believes } (S \text{ controls } A \overset{K}{\leftrightarrow} B), \\
A \text{ believes } (S \text{ controls } A \overset{K}{\leftrightarrow} B), & B \text{ believes fresh}(T_s), \\
A \text{ believes fresh}(T_s), & B \text{ believes fresh}(T_a),
\end{array}$$

We also want to end up with the following conclusions:

$$\begin{array}{rl}
A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B & B \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B \\
A \text{ believes } B \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B & B \text{ believes } A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B
\end{array}$$

This functionally means that both A and B believe they each have a valid secure key, AND believe that each other has the same key. It is debatable whether this is what you want out of kerberos. Wouldn't you prefer to know that the other person and you are the only two people (except for S) who know the key? This is the goal the *BAN* authors chose, however, and they then went on to prove that this is achieved in this system.

4.3 Proof that the Kerberos protocol is secure

We analyze the ideal version of the Kerberos protocol, starting with our assumptions, using the postulates, and hopefully reaching all of the stated goals. after receiving message 2, we have

$$A \text{ sees } [T_s, (A \overset{K_{ab}}{\leftrightarrow} B), [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{bs}}]_{K_{as}}$$

and using the hypothesis

$$A \text{ believes } A \overset{K_{as}}{\leftrightarrow} S$$

we use the *message-meaning rule* for shared keys to get

$$A \text{ believes } S \text{ said } (T_s, (A \overset{K_{ab}}{\leftrightarrow} B), [T_s, A \overset{K_{ab}}{\leftrightarrow} B]_{K_{bs}})$$

which of course means that

$$A \text{ believes } S \text{ said } (T_s, (A \overset{K_{ab}}{\leftrightarrow} B))$$

Moreover, we have the hypothesis

$$A \text{ believes fresh}(T_s)$$

so the *nonce-verification rule* applies, and yields

$$A \text{ believes } S \text{ believes } (T_s, A \overset{K_{ab}}{\leftrightarrow} B)$$

Once again, we break the conjunction to get

$$A \text{ believes } S \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B$$

or by instantiating K to K_{ab} in the hypothesis

$$A \text{ believes } S \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B$$

deriving the more concrete

$$A \text{ believes } S \text{ controls } A \overset{K_{ab}}{\leftrightarrow} B$$

The *jurisdiction rule* applies now, to yield

$$A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B$$

concluding the analysis of message 2.

A passes the ticket on to B, together with a new message containing a timestamp created by A. Initially, B can decrypt only the authenticator:

$$B \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B$$

(which can be achieved much in the same way as that conclusion for message 2). Knowledge of the new key allows B to proceed by decrypting the rest of Message 3. Through the *message meaning* and *nonce-verification* rules we deduce that

$$B \text{ believes } A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B$$

The fourth and final message simply assures A that B believes in the key and has received A's last message. After once again applying the *message-meaning* and the *nonce-verification* postulated, the final result is:

$$\begin{array}{ll} A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B & B \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B \\ A \text{ believes } B \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B & B \text{ believes } A \text{ believes } A \overset{K_{ab}}{\leftrightarrow} B \end{array}$$

Which is exactly what we were aiming for. This indicates that in theory, our protocol is probably not weak, but we still have to worry about things such as implementations, and whether the conclusion is actually what we are expecting of this protocol. One side effect of this analysis was noticing that the double-encryption of the authenticator A is meant to send to B is unnecessary.

5 Handouts and Readings

The answers to problem sets 1 and 2 were handed out. A small reading packet containing *A Logic of Authentication* and *EC plans encryption rules* were given as reading material. In this reading material, apart from Kerberos, the *Andrew Secure RPC Handshake* is discussed and also proven, though there wasn't enough time in class to discuss it. Finally, Problem set 3 is due Thursday.