A Logic for Authentication

M. Burrows, M. Abadi, R.Needham

15-712: Software Systems Core

25 February 1999

Motivation

1. Even the simplest protocols (as anything else in the world of distributed systems) are very difficult to design correctly.

2. We need a framework for formal reasoning about certain properties of protocols (in this case authentication protocols).

3. BAN logic concentrates on the following questions:

   – What does this protocol achieve?

   – Does this protocol need more assumptions than another one?

   – Does this protocol do anything unnecessary that could be left out without weakening it?

   – Does this protocol encrypt something that could be sent in clear without weakening it?

Assumptions

1. If you've sent Joe a number that you have never used before and if you subsequently receive from Joe something that depends on knowing that number, then you ought to believe that Joe's message originated after yours.

2. If you believe that only you and Joe know K, then you ought to believe that anything you receive encrypted with K as key comes originally from Joe.

3. If you believe that K is Joe's public key, then you should believe that any message that you can decrypt with K comes originally from Joe.

4. If you believe that only you and Joe know X, then you ought to believe that any encrypted message you receive containing X comes originally from Joe.

Notation

1. Three sorts of objects:
   - Principals - range over A, B, S, P, Q
   - Encryption keys - K, K_{ab}, K_a, K_b, K_a^{-1}, K_b^{-1}
   - Formulas (or statements) - N_a, X, Y

2. Predicates:
   - $P \mid\equiv X$: *P believes X*. At a given moment in the run of the algorithm, the principal P believes that the sentence X holds.
   - $P \triangleleft X$: *P sees X*. Someone has sent a message containing X to P who can read it and repeat it.
   - $P \mid\sim X$: *P once said X*. The principal P once sent a message containing X. It is not known whether it happened in this run of the protocol or before.
   - $P \mid\Rightarrow X$: P has *jurisdiction* over X. If P believes X, we should trust him.
   - $\sharp(X)$: The formula X is *fresh*. It originates from the current run of the algorithm.

Notation (cont'd)

- $P \overset{K}{\leftrightarrow} Q$: P and Q use *shared key* K to communicate. The designer of the protocol guarantees that K will not be misused.
- $\overset{K}{\mapsto} P$: P has K as a *public key.* The matching *secret key* is K^{-1}.
- $P \overset{Y}{\rightleftharpoons} Q$: Y is a *secret* known only to P and Q.
- $\{X\}_K$ (from P): The formula X is encrypted with K and received from principal P.
- $\langle X \rangle_Y$: The secret Y is concatenated to X.

Derivation Rules

1. Message-meaning rules:

   – for shared key protocols

$$\frac{P \mid\equiv Q \overset{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \mid\equiv Q \mid\sim X}$$

   – for public key protocols

$$\frac{P \mid\equiv \overset{K}{\mapsto} Q, \quad P \triangleleft \{X\}_{K^{-1}}}{P \mid\equiv Q \mid\sim X}$$

   – for shared secrets

$$\frac{P \mid\equiv Q \overset{Y}{\rightleftharpoons} P, \quad P \triangleleft \langle X \rangle_Y}{P \mid\equiv Q \mid\sim X}$$

2. Nonce-verification rule:

$$\frac{P \mid\equiv \sharp(X), \quad P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

3. Jurisdiction rule:

$$\frac{P \mid\equiv Q \mid\Rightarrow X, \quad P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

Derivation Rules (cont'd)

4. Pairing rules for $\mid\equiv$:

$$\frac{P \mid\equiv X, \quad P \mid\equiv Y}{P \mid\equiv (X, Y)} \quad \frac{P \mid\equiv (X, Y)}{P \mid\equiv X} \quad \frac{P \mid\equiv Q \mid\equiv (X, Y)}{P \mid\equiv Q \mid\equiv X}$$

5. Pairing rule for $\mid\sim$:

$$\frac{P \mid\equiv Q \mid\sim (X, Y)}{P \mid\equiv Q \mid\sim X}$$

6. Pairing rules for $\triangleleft$:

$$\frac{P \triangleleft (X, Y)}{P \triangleleft X} \quad \frac{P \triangleleft \langle X \rangle_Y}{P \triangleleft X} \quad \frac{P \mid\equiv Q \stackrel{K}{\leftrightarrow} P, \quad P \triangleleft \{X\}_K}{P \triangleleft X}$$

$$\frac{P \mid\equiv \stackrel{K}{\mapsto} P, \quad P \triangleleft \{X\}_K}{P \triangleleft X} \quad \frac{P \mid\equiv \stackrel{K}{\mapsto} Q, \quad P \triangleleft \{X\}_{K^{-1}}}{P \triangleleft X}$$

7. Pairing rule for $\sharp$:

$$\frac{P \mid\equiv \sharp(X)}{P \mid\equiv \sharp(X, Y)}$$

8. Symmetry rules for $\stackrel{K}{\leftrightarrow}$ (similarly for $\stackrel{Y}{\rightleftharpoons}$):

$$\frac{P \mid\equiv R \stackrel{K}{\leftrightarrow} R'}{P \mid\equiv R' \stackrel{K}{\leftrightarrow} R} \quad \frac{P \mid\equiv Q \mid\equiv R \stackrel{K}{\leftrightarrow} R'}{P \mid\equiv Q \mid\equiv R' \stackrel{K}{\leftrightarrow} R}$$

Protocol Analysis

1. Represent the protocol in its *idealized* form. E.g. transform:

   $$A \to B\colon \{A, K_{ab}\}_{K_{bs}}$$

   into

   $$A \to B\colon \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$$

2. Give the set of formulas which should be assumed true before the protocol starts.

3. Attach logical formulas to the statements of the protocol, as assertions about the state of the system after each statement.

4. At each step, apply the derivation rules in order to discover the beliefs held by the parties of the protocol.

5. Goal of the authentication is to derive

   $$A \models A \overset{K}{\leftrightarrow} B \quad \text{and} \quad B \models A \overset{K}{\leftrightarrow} B$$

   in the case of shared-key schemes, or

   $$A \models \overset{K}{\mapsto} B$$

   in public-key schemes.

Examples

1. Otway-Rees protocol
2. Needham-Schroeder protocol (with shared keys)
3. Kerberos
4. Wide-mouthed-frog protocol
5. Andrew Secure RPC Handshake
6. Yahalom protocol
7. Needham-Schroeder Public-Key protocol
8. CCITT X.509 protocol

Needham-Schroeder protocol (with shared keys)

1. Description of the protocol
2. Idealized protocol
3. Assumptions
4. Analysis
   - The principals reach similar conclusions.
   - Assumes that B believes the key is fresh. This makes it vulnerable since its often not the case: an intruder has unlimited time to find an old session key and reuse it.
   - Can be repaired by letting B initiate the interaction with the server.
   - Can be implemented with less encryption then is actually used.

Kerberos

1. Description
2. Idealized protocol
3. Assumptions
4. Analysis
   - Depends on totally synchronized clocks.
   - As Needham-Schroeder, can also be implemented with less encryption.

Andrew Secure RPC Handshake

1. Description
2. Idealized protocol
3. Assumptions
4. Analysis
   - As Needham-Schroeder, it also assumes the freshness of the key.
   - Can easily be repaired by proving the key fresh.

Conclusions

1. BAN logic cannot handle:
   - errors in implementations
   - untrustworthy principals
   - weaknesses of encryption schemes
2. But it still:
   - found subtle bugs in well-known protocols
   - indicated that formal methods are useful for design and testing of security protocols