Verification of security protocols

January 19, 2006

– models of security protocols
– typical examples of protocols and attacks
– modeling in BAN logic
– verification methods

Security Protocols and their Importance

Need for secret communication dates back to antiquity
likewise the discovery of ciphers and beginnings of cryptography

Security involves multiple aspects:
authentification, authorization, integrity, confidentiality, nonrepudiation.

Solutions are complex and reasoning them about them difficult.
Security of a protocol must not depend on the secrecy of the algorithm (no "security through obscurity")
– Subtle errors in existing (and widely used) protocols have been discovered after very long time (17 years, for Lowe's attack on Needham-Schroeder)
$\Rightarrow$ there are great risks in compromising a weak algorithm, unknown and thus unscrutinized by specialists
$\Rightarrow$ importance of formal verification even greater

Symmetric and asymmetric key encryption

A fundamental problem: establishing a two-way secret communication channel between two entities

Symmetric key encryption (secret key, shared key)
– shared key known only by the two participants
– decryption and encryption key related by a simple transform (conventionally considered as being the same)
– examples: Data Encryption Standard (1975, outdated), Triple DES, AES (Rijndael)

Asymmetric key encryption (public key)
– each participant A: has a pair of keys, one is inverse of the other
– publică key, K_a / private key K_a^{-1}
– A sends $K_b(K_a^{-1}(M))$: only A can create, only B can decrypt
– ex. Rivest-Shamir-Adleman (1976), El-Gamal

Protocol Models

[Dolev & Yao '83]: stressed the importance of clear assumptions in modeling, analysis and verification of protocols:

1) In a public key encryption system:
a) encryption functions cannot be broken
b) public key directory has guaranteed identity
c) everybody has access to all public keys E_X, $\forall X$
d) only X has access to its decryption key D_X

2) A protocol between two entities does not require the assistance of a third for encryption or decryption

3) In a uniform protocol, all communicating parties use the same message format

Sample attacks

Protocol 1:

(1) $A \rightarrow B$: $E_B(M)$

(2) $B \rightarrow A$: $E_A(M)$

Attack 1:

(1) $A \rightarrow B$: $E_B(M)$, intercepted by Z

(2) $Z \rightarrow B$: $E_B(M)$

(3) $B \rightarrow Z$: $E_Z(M)$; Z decodes M

Protocol 2:

(1) $A \rightarrow B$: $E_B(E_B(M)A)$

(2) $B \rightarrow A$: $E_A(E_A(M)B)$

Attack 2:

(1) $Z \rightarrow A$: $E_A(E_A(E_A(M)B)Z)$

(2) $A \rightarrow Z$: $E_Z(E_Z(E_A(M)B)A)$

(3) Z decodes $E_A(M)B$, obtaining $E_A(M)$

(4) $Z \rightarrow A$: $E_A(E_A(M)Z)$

(4) $A \rightarrow Z$: $E_Z(E_Z(M)A)$, thus Z has M

The Dolev-Yao intruder model

Intruders are "active": they can eavesdrop on the communication, acquiring messages and doing everything possible to decrypt them.

An intruder:

a) can obtain every message from the network

b) is a legitimate user of the system; in particular, s/he can initiate a conversation with any user

c) will have the opportunity to receive messages from any user (more generally: any user B can become recipient for any user A)

First formal results

Dolev & Yao discuss two types of protocols, defined by their allowed operations:

1. Cascade protocols
– encryption with any public key
– decryption only with own key

2. Name stamp protocols. In addition:
– appending a participant's name to a messaje
– deleting a certain participant's name
– deleting any name

Correctness problem becomes a rewriting problem for strings over an alphabet, decidable in polyomial time
– but undecidable for more complex problems

Authentication protocols

protocols by which participants convince each other of their identity
– and either establish shared secrets (keys) for communication
– or recognize the use of partners' secret keys

– are the most widely studied security protocols in the literature

Notations: A, B: participants. S: authentication server
N_a, N_b: "nonce" (from: number once) = random pattern (number) generated to avoid reuse of old messages by an intruder
$\{X\}_K$: message X cripted with key K

[Needham & Schroeder '78] "Using Encryption for Authentication in Large Networks of Computers": classic article, the first to predict the importance of formal verification methods

Needham-Schroeder shared key protocol

(1) $A \rightarrow S$: A, B, N_a

A announces to server S the intention to communicate with B
(and guarantees freshness of the message with a nonce N_a)

(2) $S \rightarrow A$: $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{bs}}\}_{K_{as}}$

S sends to A the key K_{ab}, together with an encrypted message which A will retransmit to B:

(3) $A \rightarrow B$: $\{K_{ab}, A\}_{K_{bs}}$

B extracts the key K_{ab} and announces A by sending a nonce N_b:

(4) $B \rightarrow A$: $\{N_b\}_{K_{ab}}$

A confirms by retransmitting a message based on N_b
(conventionally, decremented by 1 to avoid replay)

(5) $A \rightarrow B$: $\{N_b - 1\}_{K_{ab}}$

Now, both participants know they can communicate with K_{ab}

Needham-Schroeder: attack # 1

[Denning & Sacco, 1981]
Problem: an intruder who eavesdropped on a previous session can force B to accept an old key, potentially compromised

Intruder I impersonates A (denoted $I(A)$) and sends to B message (3) from the earlier session, with the old key K_c:

(3) $I(A) \rightarrow B$: $\{K_c, A\}_{K_{bs}}$
(4) $B \rightarrow I(A)$: $\{N_b\}_{K_c}$
(5) $I(A) \rightarrow B$: $\{N_b - 1\}_{K_c}$

Danger: I has practically unlimited time to compromise (break) K_c

Correction: timestamps or extra nonce

Needham-Schroeder: attack # 2

[Lowe ’95] finds an error in the public key version (after 17 years!)

(1) $A \rightarrow B$: $A, B, \{N_a, A\}_{K_b}$ A asks to communicate, sends nonce N_a

(2) $B \rightarrow A$: $B, A, \{N_a, N_b\}_{K_a}$ B replies with nonce N_b

(3) $A \rightarrow B$: $A, B, \{N_b\}_{K_b}$ A confirms reception

Attack with two concurrent sessions: A initiates session α with intruder I; the latter impersonates A in session β with B

(α.1) $A \rightarrow I$: $A, I, \{N_a, A\}_{K_i}$

(β.1) $I(A) \rightarrow B$: $A, B, \{N_a, A\}_{K_b}$

(β.2) $B \rightarrow I(A)$: $B, A, \{N_a, N_b\}_{K_a}$

(α.2) $I \rightarrow A$: $I, A, \{N_a, N_b\}_{K_a}$

(α.3) $A \rightarrow I$: $A, I, \{N_b\}_{K_i}$

(β.3) $I(A) \rightarrow B$: $A, B, \{N_b\}_{K_b}$

Discovered: with FDR model checker for CSP language

Correction: including the encrypted name of the sender in message (2)

Types of attacks

"Cryptography is not broken, it is circumvented" – A. Shamir

Clark & Jacob, "A Survey of Authentication Protocol Literature, '97:

• freshness attacks (replay attacks)
– a message (or fragment) from an earlier communication session is stored an inserted by the intruder in a new session

• type flaw attacks
– A message is composed of fields, each with a given interpretation (data, nonce, participant name, key value)
– Attack based on accepting a message with another interpretation of bit pattern than the one initially sent

Types of attacks (cont'd.)

• patallel session attacks
– two or more concurrent sessions of the same protocol
– messages from a session used to attack another

• implementation-dependent attacks
– type flaw attacks can be eliminated if the representation of message components contains redundancy to distinguish the type
– interaction between protocol and encryption method (e.g., changing of a bit in bitwise encryption)

• binding attacks (key integrity attacks)
– tampering with partner's public key (replacing it with intruder's key)

• ... and many others

Modeling in BAN logic

[Burrows, Abadi, Needham '89: "A logic of authentication"]

– most important method for modeling using logic
– a *logic of belief*, as opposed to a *logic of knowledge*
– deals with what every participant *believes* is true

Goal: to express precisely
– initial assumptions about the workings of a protocol
– the final conclusions reached by the participants

Examples:
– what does the protocol achieve ?
– does it need more assumptions than another protocol ?
– does it send/encrypt something which is not necessary ?

BAN logic: basic notions

$P \mid\equiv X$	P believes X
$P \triangleleft X$	P sees (receives, reads) message X
$P \mid\sim X$	P said (sent) X sometime in the past
$P \mid\Rightarrow X$	P has jurisdiction over X. P is an authority on X (e.g., a key) and must be believed
$\sharp(X)$	X is fresh (has not been sent so far)
$P \overset{K}{\leftrightarrow} Q$	P and Q can use shared key K to communicate
$\overset{K}{\mapsto} P$	P has public key K
$P \overset{X}{\rightleftharpoons} Q$	X is a secret known only by P and Q
$\{X\}_K$	message X encrypted with key K
$\langle X\rangle_Y$	X combined with secret Y (for identification)

BAN logic: inference rules

Rules on meaning of messages:

– for shared keys:

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

– for public keys:

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

– for shared secrets:

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

Rules on freshness of messages

$$\frac{P \models \sharp(X),\ P \models Q \mid\sim X}{P \models Q \models X} \qquad \frac{P \models \sharp(X)}{P \models \sharp(X, Y)}$$

BAN logic: inference rules (cont.)

Jurisdiction rule:

$$\frac{P \models Q \Rightarrow X,\ P \models Q \models X}{P \models X}$$

Composition: P believes a compound $\Leftrightarrow$ believes the parts

Projection: P said a compound $\Rightarrow$ said the parts

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

Decryption rules:

$$\frac{P \models \overset{K}{\mapsto} Q,\ P \triangleleft \{X\}_{K^{-1}}}{P \triangleleft X}$$

Bidirectionality of keys and secrets among participants

$$\frac{P \models R \overset{K}{\leftrightarrow} R'}{P \models R' \overset{K}{\leftrightarrow} R}$$

Example: Needham-Schroeder with shared keys

(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) $A \rightarrow B$: $\{N_b - 1\}_{K_{ab}}$

We idealize the protocol: instead of bit messages, we send *logical formulas*, corresponding to message meaning:
(1) Message 1 is only a request, has no logic value
(2) $S \rightarrow A$: $\{N_a, B, (A \overset{K_{ab}}{\leftrightarrow} B), \sharp(A \overset{K_{ab}}{\leftrightarrow} B), \{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}\}_{K_{as}}$
(3) $A \rightarrow B$: $\{A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{bs}}$
(4) $B \rightarrow A$: $\{N_b, (A \overset{K_{ab}}{\leftrightarrow} B)\}_{K_{ab}}$ from B
(5) $A \rightarrow B$: $\{N_b, (A \overset{K_{ab}}{\leftrightarrow} B)\}_{K_{ab}}$ from A

Reasoning: Needham-Schroeder with shared keys

We start with the assumptions (denote $P \models X$ and $Q \models X$ by $P, Q \models X$):

$A, S \models A \overset{K_{as}}{\leftrightarrow} S \qquad B, S \models B \overset{K_{bs}}{\leftrightarrow} S \qquad S \models A \overset{K_{ab}}{\leftrightarrow} B$

$A, B \models (S \Rightarrow A \overset{K}{\leftrightarrow} B) \qquad A \models (S \Rightarrow \sharp(A \overset{K}{\leftrightarrow} B)) \qquad$ (a good key is fresh)

$A \models \sharp(N_a) \qquad B \models \sharp(N_b) \qquad S \models \sharp(A \overset{K_{ab}}{\leftrightarrow} B)$

From $A \models \sharp(N_a)$, and (2) we deduce:

$A \models S \models A \overset{K_{ab}}{\leftrightarrow} B$, $A \models S \models \sharp(A \overset{K_{ab}}{\leftrightarrow} B)$

and from the jurisdiction rule: $A \models A \overset{K_{ab}}{\leftrightarrow} B \qquad A \models \sharp(A \overset{K_{ab}}{\leftrightarrow} B)$

After receiving message (3) from A, we deduce: $B \models S \mid\!\sim A \overset{K_{ab}}{\leftrightarrow} B$

We cannot obtain $B \models A \overset{K_{ab}}{\leftrightarrow} B$ without the premise $B \models \sharp(A \overset{K}{\leftrightarrow} B)$ (!!)
From the freshness of messages (4) and (5) we deduce: $A \models B \models A \overset{K_{ab}}{\leftrightarrow} B$ and $B \models A \models A \overset{K_{ab}}{\leftrightarrow} B$, thus each participant is believes both that the key is valid, and that the other participant knows this

Reasoning explicitates the missing premise, which allows an intruder to substitute a compromised key.

BAN logic: applicability and limitations

– allows to prove properties about a protocol
– if property can't be proved, there are serious reasons for doubt
– can identify dubious/missing/non-explicit premises

But:
– monotone logic: an existing fact cannot be retracted
– cannot handle the notion of key confidentiality or their being compromised (e.g. sending a key in plaintext)

Verification: model checking

Protocol is asynchronous composition b/w participants and intruder. Intruder can listen to anything, and delete, change or insert messages according to its current knowledge set.

State space is given by execution point for each participant and knowledge set of intruder (set of terms constructed by certain rules)

Intruder is modeled by a relation $\vdash$ by which the intruder can derive messages m from an initial set of information I:
– if $m \in I$ then $I \vdash m$
– concatenation: if $I \vdash m_1$ and $I \vdash m_2$ then $I \vdash m_1 \cdot m_2$
– projection: if $I \vdash m_1 \cdot m_2$ then $I \vdash m_1$ and $I \vdash m_2$
– encryption: if $I \vdash m$ and $I \vdash k$ then $I \vdash \{m\}_k$
– decription: if $I \vdash \{m\}_k$ and $I \vdash k^{-1}$ then $I \vdash m$

Model checkers: FDR (for CSP), OFMC (on-the-fly), SATMC (SAT-based)

Verification: theory generation

Problem in model checking: representing potentially infinite state space
Theory generation (RVChecker, REVERE [Kindred&Wing] for extended BAN logic)
– a syntactic method of saturation-based theorem proving
– produces a finite representation of a potentially infinite theory (all theorems generated from some premises and rules)
– termination based on limiting the application of those inference rules which can generate conclusions of larger size than premises.

Combination with model checking:
– a dubious premise found by theory generation: used for modeling an attack
– conversely, a counterexample, modeled as logic deduction, can identify a dubious premise

Verification: theorem proving

Model checking requires finite model $\Rightarrow$ finite participants and sessions
Theorem provers do not have this limitation

Rewriting of reasoning in Prolog: Interrogator [Millen'87]
NRL Protocol Analyzer [Meadows et al.]
– combination of theorem-proving + model checking
– starts from an error state (should be inaccessible)
– searches backwards using inductive techniques

Athena [Song et al., CMU/Berkeley]
– representation using *strand spaces* based on causality and not individual executions $\Rightarrow$ reduces state space significantly