Crypto Protocols, part 2

Today's talk includes slides from:

Jonathan Millen and Dan Wallach

Example - Needham-Schroeder

- The Needham- Schroeder symmetric-key protocol [NS78]

  A -> S: A, B, Na

  S -> A: {Na, B, Kc, {Kc, A}Kb }Ka

  A -> B: {Kc, A}Kb

  B -> A: {Nb}Kc

  A -> B: {Nb-1}Kc
- A, B are "principals;" S is a trusted key server
- Ka, Kb are secret keys shared with S
- {X, Y}K means: X concatenated with Y, encrypted with K
- Na, Nb are "nonces;" fresh (not used before)
- Kc is a fresh connection key

Denning-Sacco Attack

- Assumes that the attacker has recorded a previous session, and compromised the connection key Kx used in that one.

  A -> B: {Kx, A}Kb *attacker replayed old message*

  B -> A: {Nb}Kx

  A -> B: {Nb-1}Kx *forged by attacker*

- B now believes he shares a fresh secret key Kx with A.
- Denning-Sacco moral: use a timestamp (calendar clock value) to detect replay of old messages.

Belief Logic

- Burrows, Abadi, and Needham (BAN) Logic [BAN90a]
  - Modal logic of belief ("belief" as local knowledge)
  - Special constructs and inference rules

    e.g., P sees X (P has received X in a message)
  - Protocol messages are "idealized" into logical statements
  - Objective is to prove that both parties share common beliefs

Constructs

P bel X	P believes X
P sees X	P received X in a message
P said X	P once said X
P controls X	P has jurisdiction over X
fresh(X)	X has not been used before
P <-K-> Q	P and Q may use key K for private communication
K-> P	P has K as public key
P <-X-> Q	X is a secret shared by P and Q
{X}K	X encrypted under K
<X>Y	X combined with Y
K^{-1}	inverse key to K

(This symbolism is not quite standard)

BAN Inference Rules

- These inferences are supposed to be valid despite attacker interference.

(1) Message-meaning rules

P bel Q <-K-> P, P sees {X}K |- P bel Q said X

P bel K-> Q, P sees {X}K^{-1} |- P bel Q said X

P bel Q <-Y-> P, P sees <X>Y |- P bel Q said X

(2) Nonce-verification

P bel fresh(X), P bel Q said X |- P bel Q bel X

(3) Jurisdiction

P bel Q controls X, P bel Q bel X |- P bel X

More BAN Rules

(4) Sees rules

P sees (X, Y) |-   P sees X, P sees Y

P sees <X>Y |-   P sees X

P bel Q <-K-> P, P sees {X}K |-   P sees X

P bel K-> P, P sees {X}K |-   P sees X

P bel K-> Q, P sees {X}K^{-1} |-   P sees X

(5) Freshness

P bel fresh(X) |-   P bel fresh(X, Y)   (inside encryption)

- Symmetry of <-K-> and <-X-> is implicitly used
- Conjunction is handled implicitly

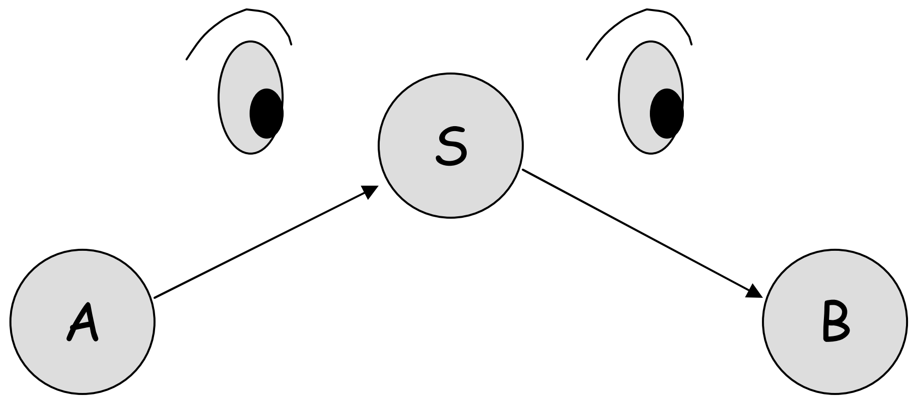
P bel (X, Y) |-   P bel X and P bel Y

P bel Q said (X, Y) |-   P bel Q said X, P bel Q said Y

Protocol Idealization

- Convert a protocol into a collection of statements
  - Assumptions
  - Message idealizations
  - Security goals
- Message idealization conveys intent of message
  - Example: A -> B: {A, Kab}Kbs
  - Idealized: B sees {A <-Kab-> B}Kbs
- ***Note:*** *only encrypted fields are retained in the idealization.*

Example - Wide-Mouthed Frog



A -> S: A, {T, B, Kab}Kas --> (M1) S sees {T, A <-Kab-> B}Kas
S -> B: {T, A, Kab}Kbs --> (M2) B sees {T, A bel A <-Kab-> B}Kbs

(A1) P bel fresh(T), for P = A, B, S	T is a timestamp
(A2) B bel A controls A <-Kab-> B	A generates Kab
(A3) S bel A <-Kas-> S, B bel B <-Kbs-> S	Kas, Kbs are shared with S
(A4) B bel S controls A bel A <-Kab-> B	S should check this
(A5) A bel A <-Kab-> B	Justifies A said A <-Kab-> B

Analysis

- Goal: prove that B bel A <-Kab-> B.
- Proof:

B sees {T, A bel A <-Kab-> B}Kbs	M2
B bel S said (T, A bel A <-Kab-> B)	A3, rule 1
B bel fresh(T, A bel A <-Kab-> B)	A1, rule 5
B bel S bel (T, A bel A <-Kab-> B)	rule 2
B bel S bel A bel A <-Kab-> B	conjunction
B bel A bel A <-Kab-> B	A4, rule 3
B bel A <-Kab-> B	A2, Rule 3

- Exercises:
  - Prove that S bel A bel A <-Kab-> B
  - Add the message B -> A: {T}Kab (M3) and show that

    A bel B bel A <-Kab-> B

Nessett's Critique

- Awkward example in [Nes90]

  A -> B: {T, Kab}Ka^{-1} --> B sees {T, A <-Kab-> B}Ka^{-1}

- Assumptions

  (A1) B bel Ka-> A

  (A2) A bel A <-Kab-> B

  (A3) B bel fresh(T)

  (A4) B bel A controls A <-Kab-> B

- Goal: B bel A <-Kab-> B
- Proof:

B bel A said (T, A <-Kab-> B)	A1, rule 1
B bel fresh(T, A <-Kab-> B)	A3, rule 5
B bel A bel (T, A <-Kab-> B)	rule 2
B bel A <-Kab-> B	A4, rule 3

- ***Problem**: Ka is a public key, so Kab is exposed.*

Observations

- According to "Rejoinder" [BAN90b], "There is no attempt to deal with … unauthorized release of secrets"
- The logic is monotonic: if a key is believed to be good, the belief cannot be retracted
- The protocol may be inconsistent with beliefs about confidentiality of keys and other secrets
- More generally - one should analyze the protocol for consistency with its idealization
- Alternatively - devise restrictions on protocols and idealization rules that guarantee consistency

Subsequent Developments

- Discussions and semantics, e.g., [Syv91]
- More extensive logics, e.g., GNY (Gong-Needham-Yahalom) [GNY90] and SVO [SvO94]
- GNY extensions:
  - Unencrypted fields retained
  - "P possesses X" construct and possession rules
  - "not originated here" operator
  - Rationality rule: if X |– Y then P bel X |– P bel Y
  - "message extension" links fields to assertions
- Mechanization of inference, e.g, [KW96, Bra96]
  - User still does idealization
- Protocol vs. idealization problem still unsolved

Model-Checking

- Application of software tools designed for hardware CAD
  - Verification by state space exploration - exhaustive on model
- Like earlier Prolog tool approach, but
  - Forward search rather than reverse search
  - Special algorithms (BDDs, etc.)
  - A priori finite model (no unbounded recursion)
  - Fully automatic once protocol is encoded
- Practicioners:
  - Roscoe [Ros95], using FDR (the first)
  - Mitchell, et al, using Murphi [MMS97]
  - Marrero, et al, using SMV [MCJ97]
  - Denker, et al, using Maude [DMT98]
  - … and more

Model-Checking Observations

- ***Very effective*** at finding flaws, but
- No guarantee of correctness, due to artificial finite bounds
- Setup and analysis is quick when done by experts
- Automatic translation from simple message-list format to model-checker input is possible [Low98a, Mil97]
- "Killer" example: Lowe attack on Needham-Schroeder public-key protocol, using FDR [Low96]

NSPK Protocol

- Na, Nb are nonces; PKA, PKB are public keys
- The protocol - final handshake

  A -> B: {Na, A}PKB

  B -> A: {Na, Nb}PKA

  A -> B: {Nb}PKB

- Exercise: use BAN Logic to prove

  B bel A bel A <-Nb-> B [BAN90a]

Lowe Attack on NSPK

- X is the attacker acting as a principal
- X masquerades as A for B

Session 1: A to X	Session 2: X (as A) to B
A -> X: {Na, A}PKX	
	A(X) -> B: {Na, A}PKB
	B -> A(X): {Na, Nb}PKA
X -> A: {Na, Nb}PKA	
A -> X: {Nb}PKX	
	A(X) -> B: {Nb}PKB

(Lowe's modification to fix it: B -> A: {Na, Nb, B}PKA)

Finiteness Limitation

- How many sessions must be simulated to ensure coverage?
  - Lowe attack needed two sessions
  - Example 1.3 in Dolev-Yao [DY83] needed three sessions

    A -> B: {{M}PKb, A}PKb

    B -> A: {{M}Pka, B}Pka
- No algorithmically determined bound is possible for all cases

    Because of undecidability for the model
- Possible bounds for limited classes of protocols
  - Lowe "small system" result [Low98b]: one honest agent per role, one time, if certain restrictions are satisfied:

    Encrypted fields are distinguishable

    Principal identities in every encrypted field

    No temporary secrets

    No forwarding of encrypted fields

Inductive Proofs

- Approach: like proofs of program correctness
  - Induction to prove "loop invariant"
- State-transition model, objective is security invariant
- General-purpose specification/verification system support
  - Kemmerer, using Ina Jo and ITP [Kem89] (the first)
  - Paulson, using Isabelle [Paul98] (the new wave)
  - Dutertre and Schneider, using PVS [DS97]
  - Bolignano, using Coq [Bol97]
- Can also be done manually [Sch98, THG98]
  - Contributed to better understanding of invariants
  - Much more complex than belief logic proofs
- Full guarantee of correctness (with respect to model)
  - Proofs include confidentiality

Summary

- Cryptographic protocol verification is based on models where
  - Encryption is perfect (strong encryption)
  - The attacker intercepts all messages (strong attacker)
  - Security is undecidable in general, primarily because the number of sessions is unbounded.
- Belief logic analysis:
  - Requires "idealization" of the protocol
  - Does not address confidentiality
  - Can be peformed easily, manually or with automated support
- State-exploration approaches
  - Use model-checking tools
  - Are effective for finding flaws automatically
  - Are limited by finiteness

Summary, cont'd

- Inductive proofs
  - Can prove correctness
  - Require substantial effort
  - Can be done manually, but preferably with verification tools
- Protocol security verification is still a research area
  - But experts can do it fairly routinely
- "Real" protocols are difficult to analyze for practical reasons
  - Specifications are not precise
  - They use operators with more complex properties than simple abstract encryption
  - Flow of control is more complex - protocols negotiate alternative encryption algorithms and other parameters
  - Messages have many fields not relevant to provable security