Proving Properties of Security Protocols by Induction

Lawrence C. Paulson

Computer Laboratory

University of Cambridge

Cryptographic Protocol Analysis

- Finite-state checking — Lowe, Millen, . . .
  - \+ find attacks quickly
  - − drastic simplifying assumptions
- Belief logics — Burrows, Abadi, Needham, . . .
  - \+ short, abstract proofs
  - − some variants are complicated & ill-motivated

An Inductive Approach

- Traces of events: A sends X to B
- Any number of interleaved runs
- Algebraic theory of messages
- A general attacker
- Modelling of accidents
- Mechanized proofs

Agents and Messages

$$agent \quad A, B, \ldots \quad = \quad \text{Server} \mid \text{Friend}\ i \mid \text{Spy}$$

$$\begin{aligned} msg \quad X, Y, \ldots \quad = \quad & \text{Agent}\ A \\ \mid \quad & \text{Nonce}\ N \\ \mid \quad & \text{Key}\ K \\ \mid \quad & \{\!|X, X'|\!\} \\ \mid \quad & \text{Hash}\ X \\ \mid \quad & \text{Crypt}\ K\, X \end{aligned}$$

Processing Sets of Messages

parts: message components $\quad$ Crypt $K\,X \leadsto X$

analz: message decryption $\quad$ Crypt $K\,X,\ K^{-1} \leadsto X$

synth: message faking $\quad$ $X,\ K \leadsto$ Crypt $K\,X$

Regularity lemmas stated using parts H

Secrecy theorems stated using analz H

Spoof messages drawn from synth(analz H)

Inductive Definition: parts H

$$\frac{X \in H}{X \in \text{parts}\ H} \qquad \frac{\text{Crypt}\ K\, X \in \text{parts}\ H}{X \in \text{parts}\ H}$$

$$\frac{\{\!|X, Y|\!\} \in \text{parts}\ H}{X \in \text{parts}\ H} \qquad \frac{\{\!|X, Y|\!\} \in \text{parts}\ H}{Y \in \text{parts}\ H}$$

$$\text{parts}\ G \cup \text{parts}\ H = \text{parts}(G \cup H)$$

Inductive Definition: analz H

$$\frac{X \in H}{X \in \text{analz } H} \qquad \frac{\text{Crypt } K\, X \in \text{analz } H \qquad K^{-1} \in \text{analz } H}{X \in \text{analz } H}$$

$$\frac{\{\!|X, Y|\!\} \in \text{analz } H}{X \in \text{analz } H} \qquad \frac{\{\!|X, Y|\!\} \in \text{analz } H}{Y \in \text{analz } H}$$

$$\text{analz } G \cup \text{analz } H \subseteq \text{analz}(G \cup H)$$

Inductive Definition: synth H

$$\frac{X \in H}{X \in \text{synth } H} \qquad \text{Agent } A \in \text{synth } H$$

$$\frac{X \in H}{\text{Hash } X \in \text{synth } H}$$

$$\frac{X \in \text{synth } H \qquad Y \in \text{synth } H}{\{\!|X, Y|\!\} \in \text{synth } H} \qquad \frac{X \in \text{synth } H \qquad K \in H}{\text{Crypt } K\, X \in \text{synth } H}$$

$$G \subseteq H \Longrightarrow \text{synth } G \subseteq \text{synth } H$$

Simplification Laws

$$\left.\begin{aligned} \text{parts}(\text{parts}\ H) &= \text{parts}\ H \\ \text{analz}(\text{analz}\ H) &= \text{analz}\ H \\ \text{synth}(\text{synth}\ H) &= \text{synth}\ H \end{aligned}\right\} \text{idempotence}$$

$$\text{parts}(\text{analz}\ H) = \text{analz}(\text{parts}\ H) = \text{parts}\ H$$

$$\text{parts}(\text{synth}\ H) = \text{parts}\ H \cup \text{synth}\ H$$

$$\text{analz}(\text{synth}\ H) = \text{analz}\ H \cup \text{synth}\ H$$

$$\text{synth}(\text{analz}\ H) = ??$$

Symbolic Evaluation of $\mathsf{parts}(\mathsf{ins}\,X\,H)$

$$\mathsf{ins}\,X\,H = \{X\} \cup H$$

$$\begin{aligned}
\mathsf{parts}(\mathsf{ins}(\mathsf{Key}\,K)\,H) &= \mathsf{ins}(\mathsf{Key}\,K)(\mathsf{parts}\,H) \\
\mathsf{parts}(\mathsf{ins}(\mathsf{Hash}\,X)\,H) &= \mathsf{ins}(\mathsf{Hash}\,X)(\mathsf{parts}\,H) \\
\mathsf{parts}(\mathsf{ins}\{\!|X, Y|\!\}\,H) &= \mathsf{ins}\{\!|X, Y|\!\}(\mathsf{parts}(\mathsf{ins}\,X\,(\mathsf{ins}\,Y\,H))) \\
\mathsf{parts}(\mathsf{ins}(\mathsf{Crypt}\,K\,X)\,H) &= \mathsf{ins}(\mathsf{Crypt}\,K\,X)(\mathsf{parts}(\mathsf{ins}\,X\,H))
\end{aligned}$$

Symbolic Evaluation of $\mathsf{analz}(\mathsf{ins}\ X\ H)$

$$\mathsf{analz}(\mathsf{ins}(\mathsf{Key}\ K)H)$$

$$= \mathsf{ins}(\mathsf{Key}\ K)(\mathsf{analz}\ H) \qquad K \notin \mathsf{keysFor}(\mathsf{analz}\ H)$$

$$\mathsf{analz}(\mathsf{ins}(\mathsf{Crypt}\ K\ X)H)$$

$$= \begin{cases} \mathsf{ins}(\mathsf{Crypt}\ K\ X)(\mathsf{analz}(\mathsf{ins}\ X\ H)) & K^{-1} \in \mathsf{analz}\ H \\ \mathsf{ins}(\mathsf{Crypt}\ K\ X)(\mathsf{analz}\ H) & \mathsf{otherwise} \end{cases}$$

Deductions from synth H

$$\text{Nonce } N \in \text{synth } H \Longrightarrow \text{Nonce } N \in H$$

$$\text{Key } K \in \text{synth } H \Longrightarrow \text{Key } K \in H$$

$$\text{Crypt } K\, X \in \text{synth } H \Longrightarrow \text{Crypt } K\, X \in H$$

or $\quad X \in \text{synth } H \wedge K \in H$

A similar law for $\{\!|X, Y|\!\} \in \text{synth } H$

Spoof Messages: Limiting the Damage

Breaking down the spoof message:

$$\{\!|X, Y|\!\} \in \mathsf{synth}(\mathsf{analz}\, H) \iff X \in \mathsf{synth}(\mathsf{analz}\, H) \land Y \in \mathsf{synth}(\mathsf{analz}\, H)$$

Eliminating the spoof message:

$$X \in \mathsf{synth}(\mathsf{analz}\, G) \Longrightarrow \mathsf{parts}(\mathsf{ins}\, X\ H) \subseteq \mathsf{synth}(\mathsf{analz}\, G) \cup \mathsf{parts}\, G \cup \mathsf{parts}\, H$$

The Shared-Key Model

Traces as lists of events:	Says $A\ B\ X$
Alice's shared key:	shrK A
Items already used in this trace:	used evs

Reading the traffic (with the help of lost keys):

$$\text{spies}\left(\text{Says}\ A\ B\ X \# evs\right) = \{X\} \cup \text{spies}\ evs$$

$$\text{spies}\ [] = \{\text{shrK}\ A \mid A \in lost\}$$

The Simplified Otway-Rees Protocol

1. $A \to B : Na, A, B, \{\!|Na, A, B|\!\}_{Kas}$
2. $B \to S : Na, A, B, \{\!|Na, A, B|\!\}_{Kas}, Nb, \{\!|Na, A, B|\!\}_{Kbs}$
3. $S \to B : Na, \{\!|Na, Kab|\!\}_{Kas}, \{\!|Nb, Kab|\!\}_{Kbs}$
4. $B \to A : Na, \{\!|Na, Kab|\!\}_{Kas}$

Inductively Defining the Protocol, 1–2

1. If evs is a trace and Na is unused, may add

$$\text{Says } A\; B\; \{\!|Na, A, B, \text{Crypt}(\text{shrK } A)\{\!|Na, A, B|\!\}|\!\}$$

2. If evs has Says $A'\; B\; \{\!|Na, A, B, X|\!\}$ and Nb is unused, may add

$$\text{Says } B\; \text{Server}\; \{\!|Na, A, B, X, Nb, \text{Crypt}(\text{shrK } B)\{\!|Na, A, B|\!\}|\!\}$$

B doesn't know the true sender & can't read X

Inductively Defining the Protocol, 4

4. If evs contains the events

$$\text{Says } B \text{ Server } \{\!|Na, A, B, X', Nb, \text{Crypt}(\text{shrK } B)\{\!|Na, A, B|\!\}|\!\}$$

$$\text{Says } S' \; B \; \{\!|Na, X, \text{Crypt}(\text{shrK } B)\{\!|Nb, K|\!\}|\!\}$$

may add

$$\text{Says } B \; A \; \{\!|Na, X|\!\}$$

Rule applies only if nonces agree, etc.

Modelling Attacks and Accidents

Fake. If $X \in \text{synth}(\text{analz}(\text{spies}\, evs))$, may add

$$\text{Says Spy}\ B\ X$$

Oops. If server distributes key K, may add

$$\text{Says}\ A\ \text{Spy}\ \{\!|Na, Nb, K|\!\}$$

Nonces show the time of the loss

Regularity & Unicity

- Agents don’t talk to themselves
- Secret keys are never lost (except initially)
- Nonces & keys uniquely identify creating message

Easily proved by induction & simplification of parts

Secrecy

- Keys, if secure, are never encrypted using any session keys
- Distributed keys remain confidential — to recipients!
- Yahalom: nonce Nb remains secure

Simplification of analz: case analysis, big formulas

An Attack

$$
\begin{array}{ll}
1. & A \rightarrow B\times : Na, A, B, \{\!|Na, A, B|\!\}_{Kas} \\
1'. & C \rightarrow A \quad : Nc, C, A, \{\!|Nc, C, A|\!\}_{Kcs} \\
2'. & A \rightarrow S\times : Nc, C, A, \{\!|Nc, C, A|\!\}_{Kcs}, Na', \{\!|Nc, C, A|\!\}_{Kas} \\
2''. & C_A \rightarrow S : Nc, C, A, \{\!|Nc, C, A|\!\}_{Kcs}, Na, \{\!|Nc, C, A|\!\}_{Kas} \\
3'. & S \rightarrow A\times : Nc, \{\!|Nc, Kca|\!\}_{Kcs}, \{\!|Na, Kca|\!\}_{Kas} \\
4. & C_B \rightarrow A : Na, \{\!|Na, Kca|\!\}_{Kas}
\end{array}
$$

New Guarantees of Fixed Protocol

B can trust the message if he sees

$$\text{Says } S' \; B \; \{\!|Na, X, \text{Crypt}(\text{shrK } B)\{\!|Nb, K|\!\}|\!\}$$

$$\text{Says } B \; \text{Server} \; \{\!|Na, A, B, X', \text{Crypt}(\text{shrK } B)\{\!|Na, Nb, A, B|\!\}|\!\}$$

A can trust the message if she sees

$$\text{Says } B' \; A \; \{\!|Na, \text{Crypt}(\text{shrK } A)\{\!|Na, K|\!\}|\!\}$$

$$\text{Says } A \; B \; \{\!|Na, A, B, \text{Crypt}(\text{shrK } A)\{\!|Na, A, B|\!\}|\!\}$$

Statistics

- 200 theorems about 10 protocol variants

  ($3 \times$ Otway-Rees, $2 \times$ Yahalom, Needham-Schroeder, . . .)

- 110 laws proved concerning messages

- 2–9 minutes CPU time per protocol

- few hours or days human time per protocol

- over 1200 proof commands in all

Conclusions

- A feasible method of analyzing protocols
- Guarantees proved in a clear framework
- Complementary to other methods:
  - Finite-state: finding simple attacks automatically
  - Belief logics: freshness analysis
- Related work by Dominique Bolignano