Proving Security Protocols Correct

Lawrence C. Paulson

Computer Laboratory

UNIVERSITY OF CAMBRIDGE
How Detailed Should a Model Be?

- too detailed
- not usable
- concrete
- ``proves'' everything
- publications

- too simple
- not credible
- abstract
- ``attacks'' everything
Case Study: the Plight of Monica and Bill
An Internet Security Protocol (TLS)

A, Na, Sid, Pa

Client hello

Nb, Sid, Pb

Server hello

cert(B, Kb)

Server certificate

cert(A, Ka)

Client certificate

\{PMS\}_Kb

Client key exchange

\{(Hash(Nb, B, PMS))_K_a^{-1}\}

Certificate verify

M = PRF(PMS, Na, Nb)
Finished = Hash(M, messages)

\{(Finished)_{client}K(Na, Nb, M)\}

Client finished

\{(Finished)_{server}K(Na, Nb, M)\}

Server finished
Why Are Security Protocols Often Wrong?

- they are trivial programs built from simple primitives, but they are complicated by
- concurrency
- a hostile environment
  - a bad user controls the network
- obscure concepts
- vague specifications
  - we have to guess what is wanted
Typical Protocol Goals

- **Authenticity**: who sent it?
- **Integrity**: has it been altered?
- **Secrecy**: who can receive it?
- **Anonymity**
- **Non-repudiation** …

*all SAFETY properties*
What Are Session Keys?

- used for a single session
- not safeguarded forever
- distributed using long-term keys
- could eventually become compromised
- can only be trusted if FRESH
Freshness, or Would You Eat This Fish?

wine: six years old

fish: ? weeks old
Packaging a Session Key for Bill

\[ \{K, A, Nb\}_{K_b} \]

- session key
- sealed using Bill's key
- person it's shared with
- nonce specified by Bill:
  - proof of freshness
A Bad Variant of the Otway-Rees Protocol

1: $Na; A; B; fj; Na; A; B; jg; Ka$

2: $Na; A; B; fj; Na; A; B; jg; Kb; Nb; fj; Na; A; B; jg; Kb$

3: $Na; fj; Na; Kab; jg; Ka$

4: $Na; fj; Na; Kab; jg; Ka$

L. C. Paulson
A Splicing Attack with Interleaved Runs

1. \( A \rightarrow C_B : Na, A, B, \{Na, A, B\}_{Ka} \)
1'. \( C \rightarrow A : Nc, C, A, \{Nc, C, A\}_{Kc} \)

2'. \( A \rightarrow C_S : Nc, C, A, \{Nc, C, A\}_{Kc}, Na', \{Nc, C, A\}_{Ka} \)
2''. \( C_A \rightarrow S : Nc, C, A, \{Nc, C, A\}_{Kc}, Na, \{Nc, C, A\}_{Ka} \)

3'. \( S \rightarrow C_A : Nc, \{Nc, Kca\}_{Kc}, \{Na, Kca\}_{Ka} \)

4. \( C_B \rightarrow A : Na, \{Na, Kca\}_{Ka} \)

Alice thinks the key \( Kca \) is shared with Bill, but it's shared with Carol!
A Bad Variant of the Yahalom Protocol

1: A, Na

S

2: B, Nb, {A, Na}\_Kb

3: \{B, Kab, Na, Nb\}_Ka, \{A, Kab\}_Kb

A

4: \{A, Kab\}_Kb, \{Nb\}_Kab

B
A Replay Attack

1. \( C_A \rightarrow B : A, Nc \)

2. \( B \rightarrow C_S : B, Nb, \{ A, Nc \}_{Kb} \)

4. \( C_A \rightarrow B : \{ A, K \}_{Kb}, \{ Nb \}_K \)

Carol has broken the old key, \( K \). She makes Bill think it is shared with Alice.
Verification Method I: Authentication Logics


Short proofs using high-level primitives:

- Nonce $N$ is fresh
- Key $K_{ab}$ is good
- Agent $S$ can be trusted

- good for freshness
- not-so-good for secrecy or splicing attacks
Verification Method II: State Enumeration

Specialized tools (Meadows)
General model-checkers (Lowe)

Model protocol as a finite-state system

- automatically finds splicing attacks
- freshness is hard to model

Try using formal proof!
Why An Operational Model?

- good fit to informal protocol proofs: inductive
- simple foundations
- readable protocol specifications
- easily explained to security experts
- easily mechanized using Isabelle
An Overview of Isabelle

- uses higher-order logic as a logical framework
- generic treatment of inference rules
- logics supported include ZF set theory & HOL
- powerful simplifier & classical reasoner
- strong support for inductive definitions
Overview of the Model

- **Traces of events**
  - A sends B message X
  - A receives X
  - A stores X

- **A powerful attacker**
  - is an accepted user
  - attempts all possible splicing attacks
  - has the same specification in all protocols
Agents and Messages

agent \( A, B, \ldots = \text{Server} \mid \text{Friend } i \mid \text{Spy} \)

message \( X, Y, \ldots = \text{Agent } A \)

| \(\text{Nonce } N\)  
| \(\text{Key } K\)  
| \(\{X, X'\} \quad \text{compound message}\)  
| \(\text{Crypt } KX\)

free algebras: we assume PERFECT ENCRYPTION
Functions over Sets of Messages

- **parts \( H \):** message components
  \[ \text{Crypt } KX \mapsto X \]

- **analz \( H \):** accessible components
  \[ \text{Crypt } KX, K^{-1} \mapsto X \]

- **synth \( H \):** expressible messages
  \[ X, K \mapsto \text{Crypt } KX \]

**RELATIONS** are traditional, but **FUNCTIONS** give us an equational theory.
**Operational Definition: analz \( H \)**

\[
\text{Crypt } KX \in \text{analz } H \quad K^{-1} \in \text{analz } H
\]

\[
X \in \text{analz } H
\]

\[
X \in H \quad \{X, Y\} \in \text{analz } H \quad \{X, Y\} \in \text{analz } H
\]

\[
X \in \text{analz } H \quad X \in \text{analz } H \quad Y \in \text{analz } H
\]

**Typical derived law:**

\[
\text{analz } G \cup \text{analz } H \subseteq \text{analz } (G \cup H)
\]
**Operational Definition: synth H**

\[
\frac{X \in H}{X \in \text{synth } H}
\]

Agent \( A \in \text{synth } H \)

\[
\frac{X \in \text{synth } H \quad Y \in \text{synth } H}{\{X, Y\} \in \text{synth } H}
\]

\[
\frac{X \in \text{synth } H \quad K \in H}{\text{Crypt } KX \in \text{synth } H}
\]

- agent names can be guessed
- nonces & keys cannot be!
A Few Equations

\[
\text{parts(parts } H) = \text{ parts } H \quad \text{transitivity}
\]
\[
\text{analz(synth } H) = \text{ analz } H \cup \text{ synth } H \quad \text{"cut elimination"}
\]

Symbolic Evaluation:

\[
\text{analz}\{\text{Crypt } KX\} \cup H =
\]
\[
\begin{cases}
\{\text{Crypt } KX\} \cup \text{analz} \{\{X\} \cup H\} & \text{if } K^{-1} \in \text{ analz } H \\
\{\text{Crypt } KX\} \cup \text{analz } H & \text{otherwise}
\end{cases}
\]
What About Freshness?

The only thing you can predict from examining that fish is that anyone who eats it will be ill!
Modelling Attacks and Key Losses

If \( X \in \text{synth(analz(spies evs))} \)

\[
\text{may add} \quad \text{SaysSpy} \ B \ X \quad \text{(Fake rule)}
\]

If the server distributes session key \( K \)

\[
\text{may add} \quad \text{NotesSpy} \ \{N_a, N_b, K\} \quad \text{(Oops rule)}
\]

Nonces show the **time** of the loss
Overview of Results

- facts proved by induction & classical reasoning
- simplifying analz $H$: case analysis, big formulas
- handles REAL protocols: TLS, Kerberos, ...
- lemmas reveal surprising protocol features
- failed proofs can suggest attacks

Proofs require days or weeks of effort
Generalizing induction formulas is hard!
The Recursive Authentication Protocol

- designed in industry (APM Ltd)
- novel recursive structure: variable length
- **VERIFIED** by Paulson
  - assuming perfect encryption
- **ATTACKED** by Ryan and Schneider
  - using the specified encryption (XOR)

*Doesn’t proof give certainty?* Not in the real world!
So Then, How Detailed Should a Model Be?

- detailed enough to answer the relevant questions
- abstract enough to fit our budget
- model-checking is almost free (thanks to Lowe, Roscoe, Schneider)
- formal proofs give more, but cost more
Don’t let theory displace reality