Proving Security Protocols Correct

Lawrence C. Paulson

Computer Laboratory
How Detailed Should a Model Be?

too detailed  too simple

concrete    abstract

not usable  not credible

``proves``

``attacks``

everything

everything

publications
Case Study: the Plight of Monica and Bill
An Internet Security Protocol (TLS)
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

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

- 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, \{Na, A, B\}_Ka

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

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

4: Na, \{Na, Kab\}_Ka
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: \(A, Na\)

2: \(B, Nb, \{A, Na\}_{Kb}\)

3: \(\{B, Kab, Na, Nb\}_{Ka}, \{A, Kab\}_{Kb}\)

4: \(\{A, Kab\}_{Kb}, \{Nb\}_{Kab}\)
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

\[ \text{agent } A, B, \ldots = \text{Server } \mid \text{Friend } i \mid \text{Spy} \]

\[ \text{message } X, Y, \ldots = \text{Agent } A \]
\[ \quad \mid \text{Nonce } N \]
\[ \quad \mid \text{Key } K \]
\[ \quad \mid \{X, X'\} \quad \text{compound message} \]
\[ \quad \mid \text{Crypt } KX \]

free algebras: we assume \text{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 $H$

\[
\begin{align*}
\text{Crypt } KX & \in \text{analz } H \\
K^{-1} & \in \text{analz } H
\end{align*}
\]
\[X \in \text{analz } H\]

\[
\begin{align*}
X & \in H \\
\{X, Y\} & \in \text{analz } H
\end{align*}
\]
\[X \in \text{analz } H\]

\[
\begin{align*}
\{X, Y\} & \in \text{analz } H \\
Y & \in \text{analz } H
\end{align*}
\]

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 $A \in \text{synth } H$

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

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

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

\[ \text{parts(parts } H \text{)} = \text{parts } H \]  \hspace{1cm} \text{transitivity}

\[ \text{analz(synth } H \text{)} = \text{analz } H \cup \text{synth } H \]  \hspace{1cm} \text{“cut elimination”}

Symbolic Evaluation:

\[ \text{analz(\{Crypt } K X \text{\}) } \cup \text{ } H \} = \]

\[ \begin{cases} \text{\{Crypt } K X \text{\} } \cup \text{analz(\{X\} } \cup \text{ } H \} & \text{if } K^{-1} \in \text{analz } H \\ \text{\{Crypt } K X \text{\} } \cup \text{analz } H & \text{otherwise} \end{cases} \]
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 \( X \in \text{synth(ana lz(spies e vs))} \)

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

If the server distributes session key \( K \)

may add \( \text{NotesSpy \{Na, Nb, K\}} \)  \( \) (Oops rule)

Nonces show the time of the loss
Overview of Results

- Facts proved by induction & classical reasoning
- Simplifying analysis $H$: 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