Mechanized Proofs for a Recursive Authentication Protocol

Lawrence C. Paulson

Computer Laboratory

University of Cambridge
Overview of the Protocol

• Based on Otway-Rees

• Distributes session keys for any number of agents

• Can be implemented as remote procedure calls

• “application components are in control of security policy and its enforcement” — John Bull

• Some modifications to assist proofs
The Protocol: Accumulation of Requests

Hashing to make Message Authentication Codes:

\[ \text{Hash}_X Y \equiv \{ \text{Hash}\{ X, Y \}, Y \} \]

1. \( A \to B \) : \( \text{Hash}_{K_a}\{ A, B, N_a, - \} \)

2. \( B \to C \) : \( \text{Hash}_{K_b}\{ B, C, N_b, \text{Hash}_{K_a}\{ A, B, N_a, - \} \} \)

2'. \( C \to S \) : \( \text{Hash}_{K_c}\{ C, S, N_c, \text{Hash}_{K_b}\{ B, C, N_b, \cdots \} \} \)

No limit on the nesting of requests
The Protocol: Distribution of Certificates

3. \( S \rightarrow C : \{Kcs, S, Nc\}_{Kc}, \{Kbc, B, Nc\}_{Kc}, \{Kbc, C, Nb\}_{Kb}, \{Kab, A, Nb\}_{Kb}, \{Kab, B, Na\}_{Ka} \)

4. \( C \rightarrow B : \{Kbc, C, Nb\}_{Kb}, \{Kab, A, Nb\}_{Kb}, \{Kab, B, Na\}_{Ka} \)

4'. \( B \rightarrow A : \{Kab, B, Na\}_{Ka} \)
The Verification Method

- Formal proof, not finite state checking
- Trace semantics, no beliefs or other modalities
- Inductive definitions: a simple, general model of action
- Any number of interleaved runs
- A general & uniform attacker
- Mechanized using Isabelle/HOL
Processing Message Histories

- parts: message components
  \[
  \text{Crypt } KX \rightsquigarrow X
  \]
  parts \(H\) contains everything potentially recoverable from \(H\)

- analz: message decryption
  \[
  \text{Crypt } KX, K^{-1} \rightsquigarrow X
  \]
  analz \(H\) contains everything currently recoverable from \(H\)

- synth: message faking
  \[
  X, K \rightsquigarrow \text{Crypt } KX
  \]
  synth \(H\) contains everything expressible using \(H\)
The Introduction of Hashing

Allow the message Hash $X$. How to extend the operators?

- Don’t add $\text{Hash } X \in \text{parts } H \implies X \in \text{parts } H$
- Don’t add $\text{Hash } X \in \text{analz } H \implies X \in \text{analz } H$
- Do add $X \in \text{synth } H \implies \text{Hash } X \in \text{synth } H$

Hashing is one-way, so hash values are atomic

Components vs Ingredients
1. If $e_{vs}$ is a trace and $Na$ is fresh, may add

$$\text{Says } A \, B \left( \text{Hash}_{shrK \, A} \{ A, B, Na, - \} \right)$$

2. If $e_{vs}$ has Says $A' \, B \, Pa$ and $Pa = \{ Xa, A, B, Na, P \}$ and $Nb$ is fresh, may add

$$\text{Says } B \, C \left( \text{Hash}_{shrK \, B} \{ B, C, Nb, Pa \} \right)$$

*B doesn’t know* the true sender & *can’t verify* hash $Xa$
3. If $evs$ contains the event $\text{Says } B' \ S \ Pb$, may add a suitable response $\text{Says } S \ B \ Rb$

4. If $evs$ contains the events

$$\text{Says } B \ C \ (\text{Hash}_{\text{shrK}_B} \{ B, C, Nb, Pa \})$$

$$\text{Says } C' \ B \ \{ \text{Crypt}(\text{shrK}_B) \{ Kbc, C, Nb \}, \text{Crypt}(\text{shrK}_B) \{ Kab, A, Nb \}, R \}$$

may add $\text{Says } B \ A \ R$
1. If $Kab$ is a fresh key (not used in $evs$) then

\[
\begin{align*}
\text{Hash}_{shrK_A}\{A, B, Na, -\}, & \quad \text{(request)} \\
\text{Crypt}(shrK_A)\{Kab, B, Na\}, & \quad \text{(response)} \\
Kab \in \text{respond evs} & \quad \text{(last key)}
\end{align*}
\]

Only if the hash can be verified
2. If \((Pa, Ra, Kab) \in \text{respond evs and } Kbc\) is fresh and
\[ Pa = \text{Hash}_{\text{shrK} A}\{A, B, Na, P\} \text{ then} \]
\[
( \text{Hash}_{\text{shrK} B}\{B, C, Nb, Pa\}, \text{ (request)} \\
\{\text{Crypt(shrK} B\}\{Kbc, C, Nb\}, \text{ (response)} \\
\text{Crypt(shrK} B\}\{Kab, A, Nb\}, \\
Ra\}, \]
\[ Kbc \) \in \text{respond evs} \text{ (last key) } \]
An Easy Proof: Long-Term Keys Aren’t Lost

By induction over \((P, R, K') \in \text{respond evs}\):

\[ K \in \text{parts}\{R\} \implies K \text{ is fresh} \]

By induction over \(evs \in \text{recur lost}\):

\[ K \in \text{parts } H \iff K \in \text{lost} \]

(any long-term key \(K\) found in traffic was lost initially)

Typically need \textbf{two} nested inductions
Unicity of Nonces

At most one hash in the history $H$ contains

- the key of an uncompromised agent ($A \not\in \text{lost}$)
- any specified nonce value, $Na$

$\exists B' \ P'. \forall B \ P.$

$\text{Hash}\{\text{Key}(\text{shrK} \ A), A, B, Na, P\} \in \text{parts} \ H$

$\rightarrow B = B' \land P = P'$
Unicity of Session Keys

At most two certificates in the response \((R)\) contain

- any particular session key, \(Kab\) ...

- made for two uncompromised agents \((A, B \notin \text{lost})\)

\[
\exists A' \ B'. \ \forall A \ B \ N. \\
\text{Crypt}(\text{Key}(\text{shrK} \ A)) \{ Kab, B, N \} \in \text{parts}\{ R \} \\
\rightarrow (A' = A \land B' = B) \lor (A' = B \land B' = A)
\]
Essential lemma, for any session key \(Kab\):

\[ K \in \text{analz}(\{Kab\} \cup H) \iff K = Kab \lor K \in \text{analz} H \]

Guarantee between uncompromised agents \(A\) and \(B\):

\[ \text{Crypt}(\text{shr}K A)\{Kab, B, N\} \in \text{parts} H \implies Kab \not\in \text{analz} H \]

Nonces not involved in proofs
Difficulties involving Certificates

- Danger of re-ordering
- Need for explicitness: name of other agent
- Special treatment of first & last agents
- Complexity of respond’s definition
  Simpler version: arbitrary lists of certificates
Limitations of the Proofs

- Authentication of $B$ to $A$ not proved
- Authentication of $A$ to $B$ not provable!
- No dynamic loss of long-term keys
- Encryption assumed secure
- Type confusion not considered (not relevant?)
Statistics

- Two weeks human effort for proofs
- 30 lemmas and theorems
- 135 tactic commands
- Under five minutes CPU time
- Savings from protocol’s symmetries
Conclusions

- Inductive definitions can model non-trivial processes
- Nested inductions cause no problems
- Multiple session keys are no obstacle
- Many types of protocols can be analyzed
- Must distinguish abstract level from implementation