

# Proving Properties of Security Protocols by Induction

Week 10

Jason, Josh, Steve

# Presentation Overview

## ● Introduction

- Previous Methods to Prove Security Protocols
  - BAN,
  - Finite State Systems
- Inductive method

## ● Background

- Proof by Induction?
- Finite State Machine Models
- Isabelle/HOL

## ● Paper Overview

- Introduction and Setup
  - Agents and Messages
  - The Attacker
  - Message Analysis
  - Modeling Protocol
- Proving the Protocol

## ● Conclusion

# Introduction-Previous Methods

## ● Belief Logic (Burrows, Abadi, Needham)

- Formalized theory to describe and analyze authentication protocols
- Defines what each agent can infer from a given set of interactions.
- Shows what beliefs for a given protocol are provable.

# Introduction-Previous Methods

## ● Belief Logic (Burrows, Abadi, Needham)

### ■ Pros:

- Concise “Idealized” proofs
- Can identify problems in protocols that do not involve breaking encryption (message manipulation for advantage to attacker)

### ■ Cons:

- Excessive simplification can take logic away from real world solutions.
- Original proposed BAN logic incomplete and unable to describe some aspects of protocols such as Secrecy/Confidentiality.
- Logics may be incorrectly applied or incorrect (proof is false due to rule problems).

# Background Info

## ● Finite State Systems

- All states in a given system are defined.
- Transitions between states defined.
- All sequence of states (start to end) can be examined to see if security protocol definitions are upheld.

# Background Info

## ● Finite State Systems

- Pros:

- Capability to exhaustively examine a protocol for adherence to principles

- Cons:

- Due to the space explosion, protocols and agents usually need to be significantly simplified.

- May be limited by finiteness...

# Background Info

## ● Proof by Induction

- General form

- Prove some initial base case  $P(0)$  is true.

- Then show that  $P(n+1)$  is true, by assuming  $P(n)$  is true

# Combination of Methods - Inductive Solution

- Using State based methods, we can concretely model events, such as A sending X to B.
- Using belief logics, we can derive guarantees from each protocol message.
- Protocols are formalized simply as the set of all possible traces of events.
- Properties about the protocol are then proved by induction on these traces

# Background: Isabelle/HOL

- Generic theorem proving analysis tool.
  - Allows a variety of formulas to be represented in a formal language for verification.
  - Available for free download at <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/>
- Proves correctness and does not just identify flaws in the protocol

# Agents and Messages

## ● Agents can be any of the following:

- Server – S
- Client – A, B, etc...
- Spy – M

## ● Messages can be any of the following:

- Agent names – A, B, S...
- Nonces – Na, Nb...
- Keys – K, Kab, Shrk (K) ...
- Compound Message - {X, Y}
- Encrypted Message - Crypt K X

# Message Analysis

## Inductively Defined Operators:

- **$H$** : Includes an agent's initial knowledge and the history of all the message sent in a trace.
- '**parts  $H$** ': Includes all components from all messages in set  $H$ , both encrypted and unencrypted.
- '**analz  $H$** ': Includes all decrypted and unencrypted components of any encrypted message in set  $H$  as long as the specific encryption key is known.
- '**synth  $H$** ': Includes any possible message that a spy can create from set  $H$ .

# Defining parts H

$$\frac{X \in H}{X \in \text{parts } H}$$

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

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

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

# Defining analz H

$$\frac{X \in H}{X \in \text{analz } H}$$

$$\frac{\text{Crypt } K \ X \in \text{analz } H \quad K \in \text{analz } H}{X \in \text{analz } H}$$

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

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

# Defining synth H

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

Agent  $A \in \text{synth } H$

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

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

# Message Analysis

## What we can prove about the operators:

- $\text{parts } G \cup \text{parts } H = \text{parts } (G \cup H)$
- $\text{analz } H \subset \text{parts } H$
- $\text{analz } G \cup \text{analz } H \subseteq \text{analz}(G \cup H)$
- $\text{synth } G \cup \text{synth } H \subseteq \text{synth}(G \cup H)$

# Profile of an Attacker



- Observes all traffic on the network –  $H$
- Sends fraudulent messages drawn from the set  $\text{synth}(\text{analz } H)$ .
- Other agents view the attacker as an honest agent
- Has access to an unspecified number of “lost” long-term keys.
- Has access to “oops” messages containing session keys.

# Modelling Protocols

Consider a variant of the Otway-Rees protocol:

1.  $A \rightarrow B : Na, A, B, \{Na, A, B\}_{K_{as}}$
2.  $B \rightarrow S : Na, A, B, \{Na, A, B\}_{K_{as}}, Nb, \{Na, A, B\}_{K_{bs}}$
3.  $S \rightarrow B : Na, \{Na, Kab\}_{K_{as}}, \{Nb, Kab\}_{K_{bs}}$
4.  $B \rightarrow A : Na, \{Na, Kab\}_{K_{as}}$

# Modeling Protocols

When modeling sent messages:

$A \rightarrow B : X \quad \Rightarrow \quad \text{Says } A \ B \ \{X\}$

# Modeling Protocols

Protocol steps are modeled as possible extensions of a trace with new events.

Initially a trace  $evs$  is an empty list –  $[]$

# Modelling Protocols

1.  $A \rightarrow B : Na, A, B, \{Na, A, B\}_{Kas}$

If  $evs$  is a trace,

$Na$  is fresh

$B$  is an agent distinct from  $A$  and  $S$

Then extend  $evs$  with:

$Says\ A\ B\{Na, A, B, \{Na, A, B\}_{Kas}\}$

# Modelling Protocols

2.  $B \rightarrow S : Na, A, B, \{Na, A, B\}_{Kas}, Nb, \{Na, A, B\}_{Kbs}$

If  $evs$  has an event of the form:

*Says*  $A' B\{Na, A, B, X\}$

*Nb is fresh*

$B \neq S$

Then extend  $evs$  with:

*Says*  $B S\{Na, A, B, X, Nb, \{Na, A, B\}_{Kbs}\}$

# Modelling Protocols

3.  $S \rightarrow B : Na, \{Na, Kab\}_{Kas}, \{Nb, Kab\}_{Kbs}$

If  $evs$  has an event of the form:

$Says\ B'\ S\ \{Na, A, B, \{Na, A, B\}_{Kas}, Nb, \{Na, A, B\}_{Kbs}$

$Kab$  is fresh

$B \neq S$

Then extend  $evs$  with:

$Says\ S\ B\ \{Na, \{Na, Kab\}_{Kas}, \{Nb, Kab\}_{Kbs}\}$

# Modelling Protocols

4.  $B \rightarrow A : Na, \{Na, Kab\}_{Kas}$

If  $evs$  has an event of the form:

$Says\ B\ S\ \{Na, A, B, X', Nb, \{Na, A, B\}_{Kbs}\}$

$Says\ S'\ B\ \{Na, X, \{Nb, K\}_{Kbs}\}$

$A \neq B$

Then extend  $evs$  with:

$Says\ B\ A\ \{Na, X\}$

# Modelling Protocols

A protocol description usually requires some additional rules:

- If  $evs$  is a trace,  $X \in \text{synth}(\text{analz } H)$  is a fraudulent message, then extend  $evs$  with:

*Says Spy B X*

- If  $evs$  is a trace and  $S$  distributed key  $K$  to  $A$ , and  $Na$  and  $Nb$  were used, then extend  $evs$  with:

*Says A Spy{Na, Nb, K}*

# Ottway-Rees Attack

1.  $A \rightarrow B : Na, A, B, \{Na, A, B\}_{K_{As}}$  ( $M$  intercepts)
- 1'.  $M \rightarrow A : Nm, M, A, \{Nm, M, A\}_{K_{Ms}}$
- 2'.  $A \rightarrow S : Nm, M, A, \{Nm, M, A\}_{K_{Ms}}, Na', \{Nm, M, A\}_{K_{As}}$  ( $M$  intercepts)
- 2''.  $M_A \rightarrow S : Nm, M, A, \{Nm, M, A\}_{K_{Ms}}, Na, \{Nm, M, A\}_{K_{As}}$
- 3'.  $S \rightarrow A : Nm, \{Nm, K_{Ma}\}_{K_{Ms}}, \{Na, K_{Ma}\}_{K_{As}}$  ( $M$  intercepts)
4.  $M_B \rightarrow A : Na, \{Na, K_{Ma}\}_{K_{As}}$

# Protocol Proofs: What to Prove?

## ● Ideal:

### ■ Secrecy

- Transactions between authenticated parties are protected from observation.

### ■ Mutual Authentication (Knowledge of each other)

- Not just between the server and the agent, but between agents

### ■ Freshness

- Are keys used “fresh”, not subject to replay?

# Protocol Proofs: Inductive Method

- Inductive Method
  - Combines aspects of BAN, SPI, and Finite state Systems.
- (Ban): Provides a Higher ordered logic to describe security protocol
- (Spi): Has mathematical/logical operations that can extend to security and confidentiality.
- (Finite State Systems): All states in a given conversation between agents can be defined and verified.

# Protocol Proofs: Quick Review—Sets

Parts H: Consist of all parts from H, plus all messages decrypted ('God' Decryption)

Analz H: Spy Seen Messages + Decrypt w/Key  
(Max Spy Knowledge)

H: All messages seen by  
an Agent (Finite)

Synth H: Spy created from Analz

Lost:  
Lost Long Term  
Keys/Agents

# Protocol Proofs: Review—Terms

- **EVS:** Event Traces
  - Set of all messages from a given start state to ending state
- **Set\_of\_list:** This is the complete set of event traces being examined.
- **Lost:** Set of all lost long term keys
- **Shrk (A):** Shared key A and the Server Kas
- **'analz (sees lost Spy evs)'** means that the spy has access to all lost long term keys.

# Protocol Proofs: Greatly Simplified...

Paper provides a lot of detail, and many simple ideas are obscured...

First, build all of the possible traces relating to the protocol and goals

## Set of Traces

### Step 1: Run Protocol

- OR Protocol 1:  $A \rightarrow B$  ...
- Or Protocol 2:  $B \rightarrow S$  ...

- + Say A B {...}
- + Say B S {...}

### Step 2: Oops Statements

- Loss of Session Key(s)
- Loss of Nonce(s)

- + Say A Spy {... Key}
- + Say A Spy {... Nonce}

### Step 3: Analz (past agent/ Long term Keys, lost session keys)

- Messages from 'Lost'
- Messages using leaked session Keys

- + Say S A {X}Kas (decrypte)
- + Say B A {X}Kab (decrypte)

• End up with a Complete Set of all possible messages...

---

Set\_of\_List EVS

# Protocol Proofs: Greatly Simplified...

- Then you go through and prove that in every case (all messages), that key principles are not violated.
- Does the Protocol Work (Possibility Properties)?
- Does the protocol violate any protocol properties (regularity properties)?
  - For the set 'set\_of\_list evs' do any messages violate these two properties?

# Proving OR Protocol: Fundamental Properties to Prove

## ● “Possibility Properties”

- Mainly agree that message formats agree from one message to the next.
- Essentially, there are no screw-ups, and the protocol can complete successfully.

## ● “Regularity Properties”

- Defines where we expect to see messages and keys.
- Used to show where messages are exposed or where keys may appear.
- Converse: if messages and keys can only appear in certain messages, then they **cannot** appear elsewhere.

# Proving OR Protocol: Fundamental Properties to Prove

- Paper breaks the Possibility properties into 3 separate components.
  - #1: Message Agreement
  - #2: No  $A \rightarrow A$  Messages
  - #3: Forwarding
- The more specific goals of security are each separately addressed in the “regularity” properties.
- #4: Regularity
- #5: Unicity ( $\sim$ Freshness)
- #6: Secrecy ( $\sim$ Secrecy)

# OR “Possibility” Properties #1

## Message Agreement

### ● Message Agreement:

- Show message formats agree from one step to the next
- Protocol can complete successfully

### ● Otway-Rees protocol

- Examine a complete Trace (State Start)→(State End)
- For given set of Agents A,B, Key K, and Nonce there Exists final message,  $B \rightarrow A: Na, \{Na, Kab\}Ka$

(Start State)  $A \rightarrow B : Na, A, B, \{Na, A, B\}Ka$

(State N+)  $B \rightarrow S : Na, A, B, \{Na, A, B\}Ka, Nb, \{Na, A, B\}Kb$

(State N++)  $S \rightarrow B : Na, \{Na, Kab\}ka, \{Nb, Kab\}Kb$

(Ending State)  $B \rightarrow A : Na, \{Na, Kab\}Ka$

# OR “Possibility” Properties #2

## No $A \rightarrow A$ Messages

- No  $A \rightarrow A$  Messages: Messages not sent to self
  - The communication protocols are designed to work between distinct and separate agents.
- For messages  $X$ , and Agents  $A$ ,
  - Essentially: there should be no event traces within (set\_of\_list evs) where Agent  $A$  sends message  $X$  to Agent  $A$

Isabelle:  $\forall A X, \text{ Says } A A X \notin \text{ set\_of\_list evs}$

# OR “Possibility” Properties #3

## Forwarding

- Protocol Requires Agents to forward ‘unknown’ items
  - Case #1: The forwarded items/messages are not encrypted (example  $N_a, A, B$ )
  - Case #2: The forwarded items/messages are encrypted, (example  $\{N_a, K_{ab}\}K_a$ )

### Otway-Rees Protocol (Original Version)

- (1)  $A \rightarrow B : N_a, A, B, \{N_a, A, B\}K_a$
- (2)  $B \rightarrow S : N_a, A, B, \{N_a, A, B\}K_a, N_b, \{N_a, A, B\}K_b$
- (3)  $S \rightarrow B : N_a, \{N_a, K_{ab}\}k_a, \{N_b, K_{ab}\}K_b$
- (4)  $B \rightarrow A : N_a, \{N_a, K_{ab}\}K_a$

# OR Proof Forwarding

## Case #1—Unencrypted Data

If the message  $X$  is in the clear, nothing new is gained if  $X$  is seen

- $\text{analz } H$  includes compound messages,  $X$  if its in the clear.
- A Spy can learn nothing additional by seeing  $X$  again.

Says A' B:  $\{N, \text{Agent A}, \text{Agent B}, X\} \in \text{set\_of\_list evs}$   
 $X \in \text{analz}$  (sees lost Spy evs)

- If a message is sent that includes  $X$  and its not encrypted, then that message is known to the Spy.

- Trivial proof, as “By Definition”, any unencrypted message is in the set  $\text{analz } H$ .

# OR Proof Forwarding

## Case #2—Encrypted Data

- If the message remains encrypted, Spy gains nothing.
- If encrypted  $X$  is decrypted either through
  - (1) the forwarding party stripping off encryption, or
  - (2) by revealing the session key (Oops)
  - $X$  is now considered to exist in the set parts  $H$ .

$$\frac{\text{Says } S \text{ B: } \{Na, X, \text{Crypt } K'\{Nb, X, K\}\} \in \text{set\_of\_list evs}}{K' \in \text{parts ( sees lost Spy evs)}}$$
$$X \in \text{analz } H$$

- First, the Server sends a message where  $X$  is no longer protected through encryption, thus becoming part of parts  $H$ .
- Second, if  $K'$  is within parts (sees Lost Spy evs), then the message can be decrypted, and  $X$  revealed.

# OR “Regularity” Properties #4

## Regularity

- Regularity focuses on where particular messages and Keys, can only occur.
  - For the OR protocol, in what sets can message  $X$ , and Key  $k$  exist.
  - Case #1, if message  $X$  is encrypted
  - Case #2, if Key  $K$  has not been discovered

Note: Regularity identifies where  $X$ ,  $K$  can exist, this in turn, defines where it **cant** exist.

# OR Proof Regularity

## Case #1—Encrypted X

- Proof Establishes that the only time an encrypted message is available to a spy, is if it can be decrypted.

If Say  $A B \{X\}K \in \text{set\_of\_list evs}$   
 $K \in \text{analz ( sees lost Spy evs)} \leftrightarrow K \in \text{analz H}$   
 $X \in \text{analz H}$

- If Encrypted message X exists
- And, the key is available to the spy (lost server key, oops)
- Then X exists within  $\text{analz H}$ , and is available to the spy.
  
- Conversely if those conditions are not true, then X should not be available to the spy.

Restated: if X is part of an encrypted message, (and not decrypted), it should never appear in parts H, or  $\text{analz H}$ , and be available the Spy.

# OR Proof Regularity

## Case #2—Key K

- if a shared Server/Agent key has not been discovered then it is not part of set parts H the spy can work with.

Key  $Kas \in \text{parts}(\text{sees lost Spy evs}) \leftrightarrow Kas \in \text{Lost}$

- If a Shared Server/Agent key is available to the spy, then that key must be part of the 'Lost' Set.
- If a  $Kas$  is within the 'Lost' set, then it is available to the spy.

Restated: The only time a shared server/agent key should be available is if its part of the 'lost' set.

# OR Regularity Properties #5

## Unicity

### ● Unicity (Unique Messages)

- For the OR protocol, messages should be unique
- This is related to our freshness principles, spoofing, as well as mutual authentication.
- Case #1, Session keys to provide Unicity
- Case #2, Nonces to provide Unicity

# OR Proof Unicity

## Case #1—Session Keys

- Session Keys: Assumes that the message originated from the server. The message is unique if everything is the same, however the Key is new

If Says S B:  $\{Na, X, \text{Crypt}(Kbs)\{Nb, K\} \in \text{set\_of\_list evs}$   
And  $B=B'$  and  $Na=Na'$  and  $Nb'=Nb'$  and  $X=X'$   
Message X is unique

- If a message exists
- And is compared to another, and only the Key K is different
- then the message is unique (non-spoofed).

- Restated: If only the session key is different between two messages, and the message originated from the server, then the messages are unique.

# OR Proof Unicity

## Case #2--Nonces

- Nonces used to prevent spoofing
- Assume the message is encrypted with a secure Key.

If  $\text{Crypt}(K_A) \{N_A, A, B\}$  parts (sees lost Spy evs)  
However,  $K_A \notin \text{Lost}$

- If encrypted message exists within parts (sees lost Spy evs)
- Nonce can be used to determine uniqueness.
- If  $K_A$  is not lost (available to the spy), then the spy cannot perform the encryption (spoofer the message), guaranteeing that the Nonce  $N_A$  can be used to identify uniqueness.

$N_A \notin \text{Parts H}$  and  $N_A \notin \text{Analz H}$

- Since Spy cannot decrypt message, then the Nonce not known, and cannot be reused by the spy to duplicate messages.

Restated: If a message is encrypted, and the spy is unable to decrypt the message, then a nonce can be used to determine if the message is unique or not.

# Protocol OR Protocol: Regularity Properties--#5 Secrecy

- How can a Spy have access to encrypted information?
- Secrecy is approached from 2 directions
- Case #1: (Between Agents) If a session key is compromised, no matter the cause, it does not imply all keys are compromised.
- Case #2: (Server to Agent) If there is communication between the Server and an Agent, where a session key is provided, there is secrecy.

# OR Proof Secrecy

## Case #1—Session Keys (Between Agents)

- Earlier, 3.7 Stated that although a session key is compromised, this does not imply other session keys are compromised.
- $K'$  is an arbitrary set of known keys.

$$K \in \text{analz}(K' \cup \text{sees lost Spy evs}) \\ \Leftrightarrow \text{either } K \in K' \text{ or } K \in \text{analz}(\text{sees lost Spy evs})$$

- If a session key  $K$  is available to a Spy
- Either, it is part of a set of keys the Spy already Knows
- Or  $K$  is able to be gained through decryption (server/agent)
  - Can be Oops event, or decrypting Server/Agent Messages
- Essentially this rules out the possibility that any other Key  $K$  is available to the Spy

$$K \notin \text{analz}(\text{sees lost Spy evs})$$

- Restated: The only time a session key is available to the spy is if its already known, or part of  $\text{Analz } H$  (Oops/Lost)
- The loss of a given session key does not imply all session keys are lost.

# OR Proof Secrecy

## Case #2—Server Provided Session Key

- If a server provides a shared key to Agent A and B, if no oops events, the key is never available to the spy.
- Oops Event form, Says B Spy {Na, Nb, K}

---

Says S B { Na, Crypt(Kas){Na, Kab}, Crypt(Kbs){Nb, Kab} }  $\notin$  set\_of\_list evs  
Says B Spy {Na, Nb, Kab}  $\notin$  set\_of\_list evs  
Kab  $\notin$  analz (sees lost Spy evs)

the server issues shared key to both A and B (forwarding proved already)  
and there is no 'Oops' event where the key is 'lost' (i.e. shared with the spy)  
then the session key is never available to the spy

---

Restated: If a shared key is issued from the Server to an Agent, and that key is never revealed via 'Oops', then this ensures session secrecy.

# OR Proof Expanding OR Version 2

- Authors wanted to expand on the OR protocol, how to address identified weaknesses, and to show how induction/Isabelle can show the impact of protocol changes.
- Original protocol allows attacks between the agents and the server.

## Otway-Rees Protocol (Original Version)

(1)  $A \rightarrow B : Na, A, B, \{Na, A, B\}_{Ka}$

(2)  $B \rightarrow S : Na, A, B, \{Na, A, B\}_{Ka}, Nb, \{Na, A, B\}_{Kb}$

New (2)  $B \rightarrow S : Na, A, B, \{Na, A, B\}_{Ka}, Nb, \{Na, Nb, A, B\}_{Kb}$

(3)  $S \rightarrow B : Na, \{Na, Kab\}_{ka}, \{Nb, Kab\}_{Kb}$

(4)  $B \rightarrow A : Na, \{Na, Kab\}_{Ka}$

- The additional nonce (2) provides a means to ensure that each respective Agent can verify the other agent participating in the conversation.
- Author steps through the proof to show that the new message ensures continuity between the Server and Agents, as well as between the agents themselves.
- Drawback, proof is considerably more complex, requiring 1466 Steps

# OR Proof Expanding

## OR Version 3--Simplified

### Abadi Needham suggest simplified protocol

#### Otway-Rees Protocol (Original Version)

- (1)  $A \rightarrow B : Na, A, B, \{Na, A, B\}_{K_a}$
- (2)  $B \rightarrow S : Na, A, B, \{Na, A, B\}_{K_a}, Nb, \{Na, A, B\}_{K_b}$
- (3)  $S \rightarrow B : Na, \{Na, K_{ab}\}_{K_a}, \{Nb, K_{ab}\}_{K_b}$
- (4)  $B \rightarrow A : Na, \{Na, K_{ab}\}_{K_a}$

#### Otway-Rees Protocol (Version #3)

- (1)  $A \rightarrow B : Na, A, B$  (Simplified)
- (2)  $B \rightarrow S : A, B, Na, Nb$  (Simplified)
- (3)  $S \rightarrow B : Na, \{Na, A, B, K_{ab}\}_{K_a}, \{Nb, A, B, K_{ab}\}_{K_b}$  (Changed)
- (4)  $B \rightarrow A : Na, \{Na, A, B, K_{ab}\}_{K_a}$  (Changed)

- Author supports the claim that this protocol is about as secure, but much simpler to prove.
- New proof requires only 57 steps.
- They do state that it is slightly weaker than the original.

# Conclusion:

## Proving Otway-Rees

- Authors show that induction can be used to examine protocols, in this case Otway-Rees.
- The OR flaws that other analysis have identified, Isabelle can identify them as well.
- Isabelle can go beyond identifying flaws, but “prove” that the OR protocol security principles hold true.
- By Modifying Otway-Rees, it is easy to show the impact of a given set of changes (Original verses Version #2, and Version #3)

# Conclusion: Disclaimer

- Proofs can only certify correct design, but are not immune from other practical implementation vulnerabilities.
  - Strong Encryption needed to prevent modification of messages.
  - The encoding of the messages is not subject to type confusion.
  - There are no flaws in the mathematics behind the encryption methods.

# Conclusion:

## Combination of Methods Best

- Last, the author suggests that all three methods (BAN, Model checking, and Inductive methods) need to be used for best results.
- BAN for early design and verification.
- Model checking to identify and find obvious attacks quickly.
- Inductive methods to conclusively prove the needed properties of the security protocol in question.

# End

🌐 Questions?