Formal Methods in Security Protocols Analysis

Li Zhiwei
Aidong Lu
Weichao Wang

Department of Computer Science
Department of Software and Information Systems
University of North Carolina at Charlotte
Big Picture

**Biggest Problem**
- rapid growth of interconnectivity
- opens doors to cyber attacks

**Best tool**
- security mechanisms
- security protocols play an essential role

Therefore future improvements depend highly on our ability to analyse security protocols
This talk is about...

Network security protocols
- Become a central concern

And formal methods for their security analysis
- Security proof in some model; or
- Identify attacks
Outline

Part I: Overview

- **Motivation**
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm

- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Security Protocol

- Security Protocols use **cryptographic primitives** as building blocks to achieve security goals such as authentication, confidentiality, and integrity.

- consists of a set of rules which determine the exchange of messages between two or more participants.

- Protocol steps
  
  \[ n : A \rightarrow B : M \]
  
  “A sends M to B according to the n’th protocol step.”
  
  A, B principals, M message
Security Protocol

Security Protocol

- Program distributed over network
- Use cryptography to achieve goal

Attacker

- Read, intercept, replace messages, and remember their contents

Correctness

- Attacker cannot learn protected secret or cause incorrect protocol completion
A Simple Protocol

- $K_B = \text{pk}(B), K_B^{-1} = \text{sk}(B)$
- $K_A = \text{pk}(A), K_A^{-1} = \text{sk}(A)$
- Needham–Schroeder Public Key Protocol
Run of a protocol

Correct if no security violation in any run
Many other security protocols

Challenge-response
- ISO 9798-1,2,3; Needham-Schroeder, ...

Authentication
- Kerberos

Key Exchange
- SSL handshake, IKE, JFK, IKEv2,

Wireless and mobile computing
- Mobile IP, WEP, 802.11i

Electronic commerce
- Contract signing, SET, electronic cash, ...
Motivation

Why do we need security protocol analysis even after we constructed security protocols with so much care?

- Error–prone
  - Security protocols are intricate and attackers are powerful
- Non–optimal
  - may contain unnecessary operations
Examples of protocol flaws

IKE [Meadows; 1999]
- Reflection attack; fix adopted by IETF WG

IEEE 802.11i [He, Mitchell; 2004]
- DoS attack; fix adopted by IEEE WG

GDOI [Meadows, Pavlovic; 2004]
- Composition attack; fix adopted by IETF WG

Kerberos V5 [Scedrov et al; 2005]
- Identity misbinding attack; fix adopted by IETF WG

How to addresses these shortcomings….
use automatic verification approach to analysis security protocol

Good domain for formal methods

Active research area since early 80’s
Part I: Overview

- Motivation
  - Why Security Protocols Analysis?
- **Central problems**
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm

- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Central Problems

1. Model system
2. Identify security properties
3. Model adversary
4. See if properties are preserved under attack

Protocol

Property

Analysis Tool

Attacker model

Security proof or attack

Answer: the protocol is correct or not
Security Analysis Methodology

1. Specifies the security protocols to be verified as input
2. Specifies the desired security properties as requirement
3. Use Attacker Model to model adversary
4. The protocol analysis tool analyzes the input protocols using formal methods
   See if the security properties are preserved under attack
5. Output the result: the protocol is flawed or it is correct
Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Four “Stanford” approaches

Finite-state analysis
- Case studies: find errors, debug specifications

Symbolic execution model: Multiset rewriting
- Identify basic assumptions
- Study optimizations, prove correctness
- Complexity results

Process calculus with probability and complexity
- More realistic intruder model
- Interaction between protocol and cryptography
- Equational specification and reasoning methods

Protocol logic
- Axiomatic system for modular proofs of protocol properties
Some other projects and tools

Exhaustive finite-state analysis
  - FDR, based on CSP

Search using symbolic representation of states
  - Meadows: NRL Analyzer, Millen: Interrogator

Prove protocol correct
  - Paulson’s “Inductive method”, others in HOL, PVS, …
  - MITRE -- Strand spaces
  - Process calculus approach: Abadi-Gordon spi-calculus, applied pi-calculus, …
  - Type-checking method: Gordon and Jeffreys, …

Many more – this is just a small sample
Protocol Analysis Techniques

Security Protocol Analysis

Formal Models
- Dolev-Yao (perfect cryptography)

Model Checking
- FDR, Murphi, Athena, NRL, Brutus, OFMC

Protocol Logics
- BAN, PCL

Process Calculi
- Spi-calculus, Applied $\pi$-calculus

Theorem Proving
- Inductive Method, Automating BAN, TAPS, Automating PCL

Cryptographic Models
- Probabilistic Interactive TM
- Probabilistic process calculi
- Probabilistic I/O automata

Bug Finding

Correctness Proofs
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
The Athena aims to provide a method to:

- generates a proof *if the protocol is actually RIGHT*
- generates a counterexample *if the protocol is actually WRONG*
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Strand Space Model

- Answer Problem 1: Specifies the security protocols
- model security protocols and their execution
- graphical representation of execution
  - Simple and intuitive
- A suitable framework for
  - ... formal specifications of security properties
  - ... proving correctness of protocols

Strand Spaces: Why is a Security Protocol Correct?*

F. Javier Thayer Fábrega  Jonathan C. Herzog
Joshua D. Guttman
The MITRE Corporation
{jt, jherzog, guttman}@mitre.org

*Thayer, Herzog, Guttman [THG98]*
Defining Strand Spaces

Message: ground term in free algebra
Strand: sequence of nodes    Node is labeled with +/− message
Bundle: causal partial ordering of nodes in strands
strand space:is a collection of strands

Example:
(The global view)
Defining Strand Spaces (cont’d)

A’s view of the protocol

\[ \{ A, N_A \}_{KB} \]
\[ \{ N_A, N_B \}_{KA} \]
\[ \{ N_B \}_{KB} \]

B’s view of the protocol

\[ \{ A, N_A \}_{KB} \]
\[ \{ N_A, N_B \}_{KA} \]
\[ \{ N_B \}_{KB} \]

Strand represents a sequence of actions (i.e., signed messages \( \pm m \)) of an instance of a role

+ means principal sends this message

- means principal receives this message

A’s (trace of his) strand
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Security Properties in this Logic

- Answer Problem 2: Specifies the desired security properties

- Authentication
  \[ \forall C. \text{Resp}(\vec{x}) \in C \iff \text{Init}(\vec{x}) \in C \]

- Secrecy
  - A value \( v \) is secret in a strand space \( S \) if, for every bundle \( C \) that contains \( S \), there does not exist a node \( n \in C \), such that \( \text{term}(n) = v \)
  \[ \neg \exists C. (\text{Resp}(\vec{x}) \in C \land \text{node}(+v) \in C') \]
For NS Authentication

The formula that needs to be checked is:

\[ \forall C. \text{Resp}[A_0, B_0, N_{a0}, N_{b0}] \in C \implies \text{Init}[A_0, B_0, N_{a0}, N_{b0}] \in C. \]

\[ n_1: - \{ A_0, N_{a0} \}_{K_{b0}} \]
\[ n_2: + \{ N_{a0}, N_{b0} \} \}_{K_{a0}} \]
\[ n_3: - \{ N_{b0} \} \}_{K_{b0}} \]

\[ \text{The NS protocol} \]

\[ n_1: + \{ A_0, N_{a0} \}_{K_{b0}} \]
\[ n_2: - \{ N_{a0}, N_{b0} \} \}_{K_{a0}} \]
\[ n_3: + \{ N_{b0} \} \}_{K_{b0}} \]
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
The Penetrator

- Answer Problem 3: Model adversary
- participates in protocols via penetrator strands
- reflect the potentials of the penetrator
- penetrator is able to
  - intercept and create messages (independent of the protocol)
  - read and memorize all (not encrypted) message parts
  - synthesize new messages built from his “knowledge”
The Penetrator

Description of the intruder

\[
\begin{align*}
\langle -t \rangle & \quad \text{for} \quad t \in T \\
\langle -m \rangle & \quad \text{for} \quad m \in M \\
\langle -K \rangle & \quad \text{for} \quad K \in K_I \\
\langle -m, +m, +m \rangle & \\
M (message) \quad t & \quad \text{to} \quad F (lushing) \quad m & \quad K (key) \quad K & \quad T (tag) \quad m \\
\langle -m, -n, +(m, n) \rangle & \quad \langle -(m, n), +m, +n \rangle & \quad \langle -K, -m, +\{m\}_K \rangle & \quad \langle -K^{-1}, -\{m\}_K, +m \rangle \\
\quad \text{perfect encryption} & \\
C (concatenation) \quad m & \quad S (separation) \quad (m, n) & \quad E (encryption) \quad K & \quad D (decryption) \quad k^{-1} \\
\quad \text{to} \quad n & \quad \text{to} \quad m & \quad \text{to} \quad m & \quad \text{to} \quad (m)_K & \quad \text{to} \quad m
\end{align*}
\]
Composing Strands to Bundles

Regular strands

Penetrator strands

Intended protocol

Attacker protocol
Penetrator’s Work – An Example

Breaking into Needham–Schroeder protocol

Man-in-the-middle attack
pass messages through to another session $A \leftrightarrow X \leftrightarrow B$
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Algorithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
Answer Problem 4:
See if properties are preserved under attack

- Verification Algorithm
  - Given a model P, check if it satisfies formula F
  - It turns out to be state search problem
Design of Model Checker

- Verification Algorithm
  - Given a model $P$, check if it satisfies formula $F$
  - It turns out to be state search problem

Athena

If provable

Obtain a formal proof of the protocol

If not provable, then counter-example

Obtain concrete attacks on the protocol

Axioms

\[ \vdash \text{Agreement formula} \]

Axioms

\[ \lnot \vdash \text{Agreement formula} \]
Outline

Part I: Overview
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

Part II: Athena Alogrithm
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results
NSPK in Strand Space Model

“A” strand

“B” strand

“Penetrator” strands

Each primitive capability of the attacker is a “penetrator” strand

Same set of attacker strands for every protocol

The NS protocol

1. $A \rightarrow B : \{N_A, A\}_B$
2. $B \rightarrow A : \{N_A, N_B\}_A$
3. $A \rightarrow B : \{N_B\}_B$
The NS protocol

\[
\begin{align*}
+ \{A, N_A\}_K_B & \quad \longrightarrow \quad - \{A, N_A\}_K_B \\
- \{N_A, N_B\}_K_A & \quad \longrightarrow \quad + \{N_A, N_B\}_K_A \\
+ \{N_B\}_K_B & \quad \longrightarrow \quad - \{N_B\}_K_B \\
\end{align*}
\]

Try to discover the flaw in NS

Anomaly in NS

\[\gamma, \delta\]
NS Example

proc searchL(initialState){
    L = {initialState};
    while(¬empty(L)) do{
        l = choose(L);
        L = L \ l;
        L' = F(l);
        if ¬empty(L') then{
            for each l' = (S', G', →', ≤') ∈ L'{
                if s2 ∉ l' then{
                    if empty(G') then return false;
                    else L = add(l', L);
                }
            }
        }
    }
    return true;
}

\[ l_{\text{c}} = \text{gamma} \]
\[
Resp[A_0, B_0, N_{a0}, N_{b0}]
\]

\[ n_1: -\{A_0, N_{a0}\}_{K_{b0}} \]
\[
\quad \downarrow
\]

\[ n_2: +\{N_{a0}, N_{b0}\}_{K_{a0}} \]
\[
\quad \downarrow
\]

\[ n_3: -\{N_{b0}\}_{K_{b0}} \]

unbound goals

0.2) Compute initial state l_0.

-- State: l_0
S(semi-handles)
Strand[l_0]:
    N1: <N_{B0}, A0> PubK_B0
    N2: <N_{A0}, N_{B0}> PubK_A0
    N3: <N_{B0}> PubK_B0
G(unbound goal set)
    N1: <N_{A0}, A0> PubK_B0
    N3: <N_{B0}> PubK_B0
B(goal binding set)

>> 1. Athena<Search Procedure>.
>> 1.1 Choose a state from set L(l_0).**************
chosen state:l_0
>> 1.2 Generate next states.

------------------------ Finding NextStates ....
proc search(initialState) {
    L = {initialState};
    while (¬empty(L)) do {
        l = choose(L);
        L = L \ l;
        L' = F(l);
        if ¬empty(L') then {
            for each l' = <S', G', →', ζ'> ∈ L' {
                if s_2 ∉ l' then {
                    if empty(G') then return false;
                    else L = add(l', L);
                }
            }
        }
    }
    return true;
}

l_0 → l_1 (3 unbound goal)

l_2 (0 unbound goal)
State Search for Needham-Shroeder

$\text{State Search for Needham-Shroeder}$
Search procedure terminate 
and counter example found !!!
NSPK Attack

Anomaly in NS

A
\{N_A, A\} k_I
\{N_A, A\} k_B
{N_A, N_B} k_A
{N_B} k_I
{N_B} k_B

I
\{N_A, A\} k_I

B
+{n,A}_{ke}
-k_e^{-1}
{n,A}
k_b
+{n,r}_{ka}
-k_b
-{r}_{kb}

-{n,r}_{ka}
+{n,A}_{kb}

+{n,A}_{ke}
The NSL protocol

1. $A \rightarrow B : \{N_1, A\}_B$
2. $B \rightarrow A : \{N_1, N_2, B\}_A$
3. $A \rightarrow B : \{N_2\}_B$

>> 1.3.1.1 delta is NOT a sub set of l' state
>> 1.3.1.2 C' is not empty, add l' state to L
>> 1.4 L_next(next states) empty.

>> 1.1 Choose a state from set L(19).***************************
chosen state:19
>> 1.2 Generate next states.  
=========== Finding NextStates ....
1.pick up a goal from G:N16 \langle Var_u26,N_A0,A0,Var_u27\rangle PubK_I 
Generate 0 next states:
>> 1.4 L_next(next states) empty.

>> 2.3 L is empty; return correct. 
Verify return:1
Conclusions

Practical protocols may contain errors
- Automated formal methods find bugs that humans overlook

Variety of tools
- Model checking can find errors
- Proof method can show correctness

Athena
- Closing gap between the model checking and proof method
  - Strand Spaces Model
  - Security Properties
  - Penetrator Strands
  - Design of Model Checker

Security protocol analysis is a challenge
- Some subtleties are hard to formalize
- No “absolute security”
- Security means: under given assumptions about system, no attack of a certain form will destroy specified properties.
References


