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

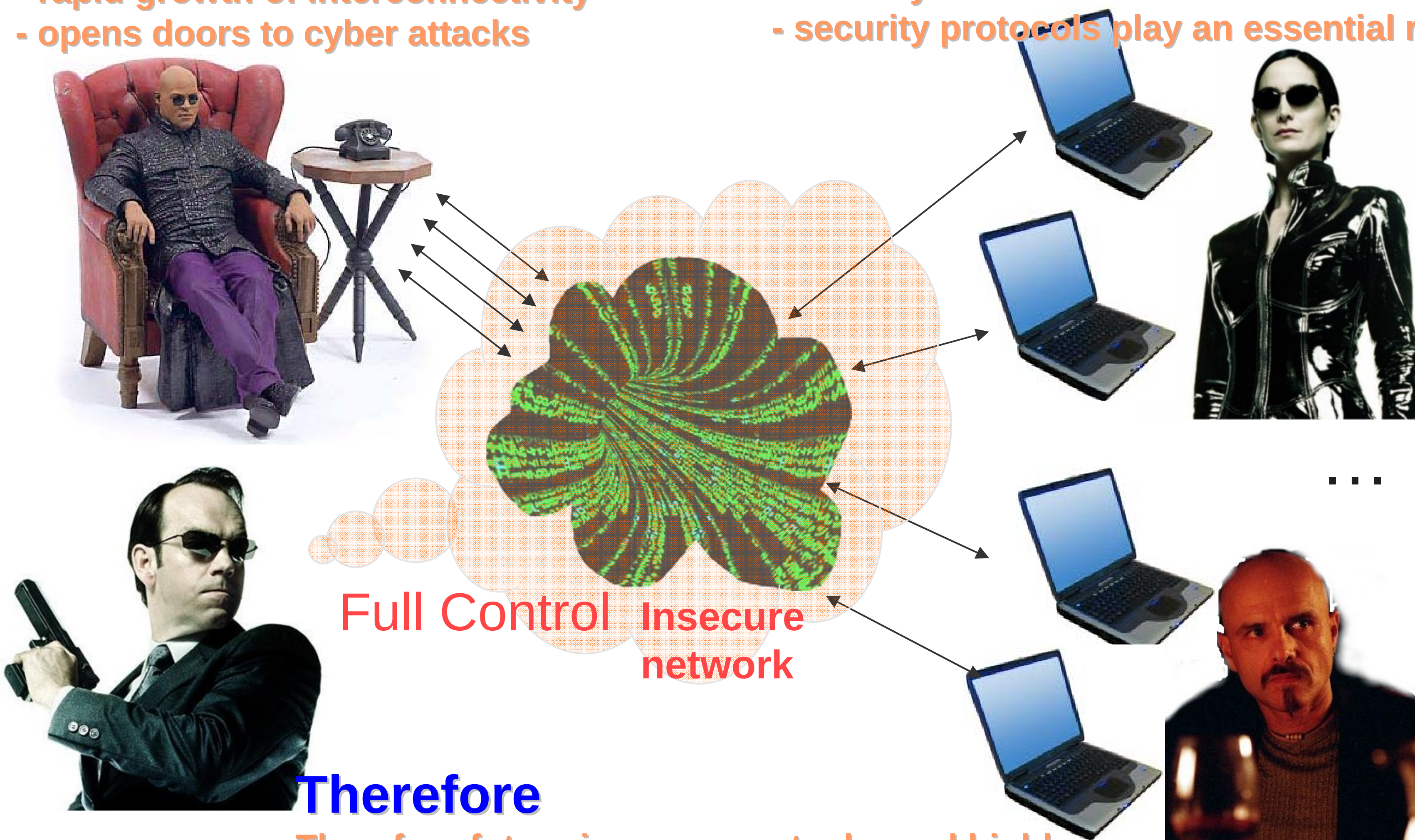
Big Picture

Biggest Problem

- rapid growth of interconnectivity
- opens doors to cyber attacks

Best tool

- security mechanisms
- security protocols play an essential role



Therefore

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 Alogrithm

- 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 → 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

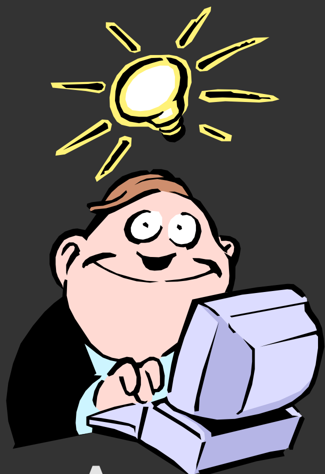
Attacker

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

Correctness

- Attacker cannot learn protected secret or cause incorrect protocol completion

A Simple Protocol



A

$\{ A, N_A \}_{K_B}$ →

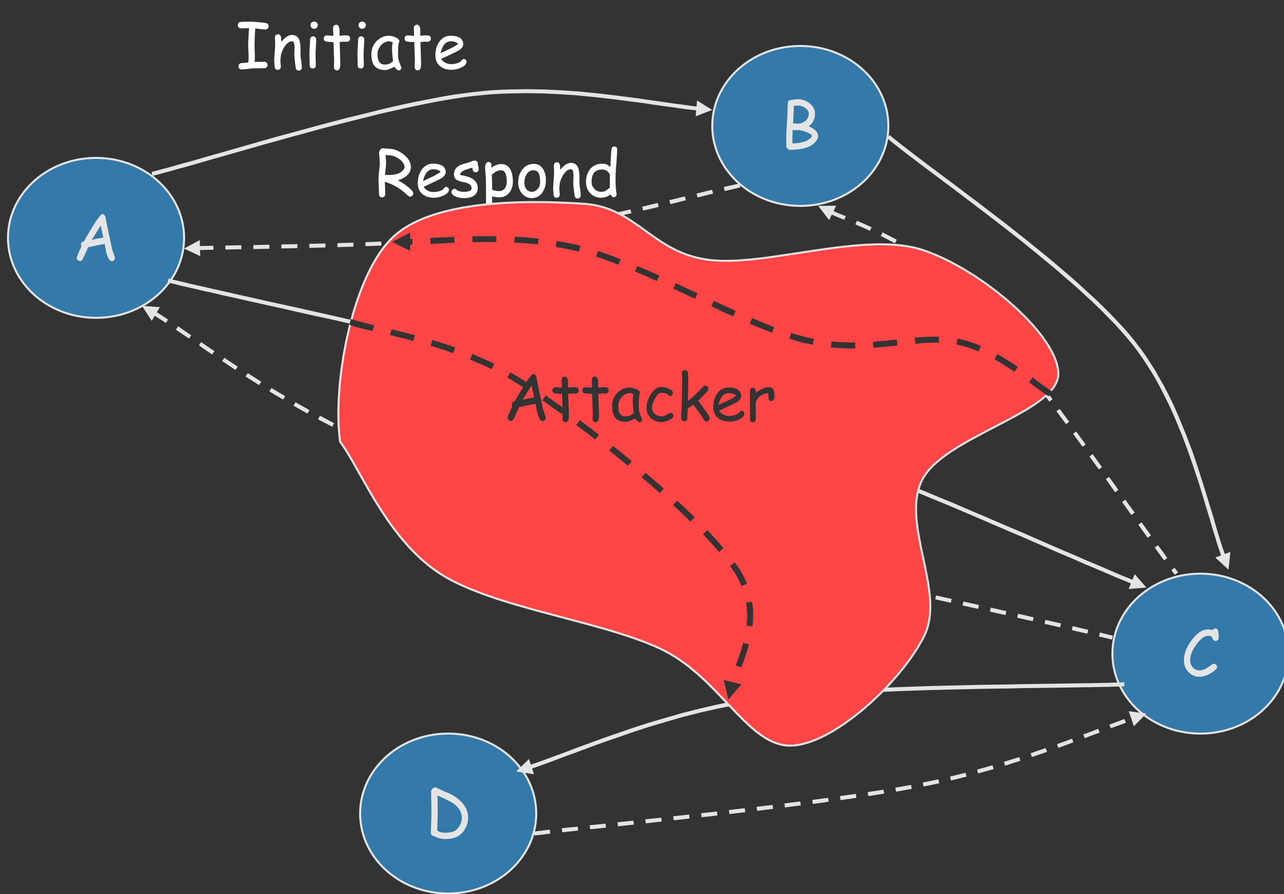
← $\{ N_A , N_B \}_{K_A}$

$\{ N_B \}_{K_B}$ →

B

- $K_B = pk(B)$, $K_B^{-1} = sk(B)$
- $K_A = pk(A)$, $K_A^{-1} = sk(A)$
- 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….

the answer is ….

use automatic verification approach to analysis security protocol

Good domain for formal methods

Active research area since early 80's

Outline

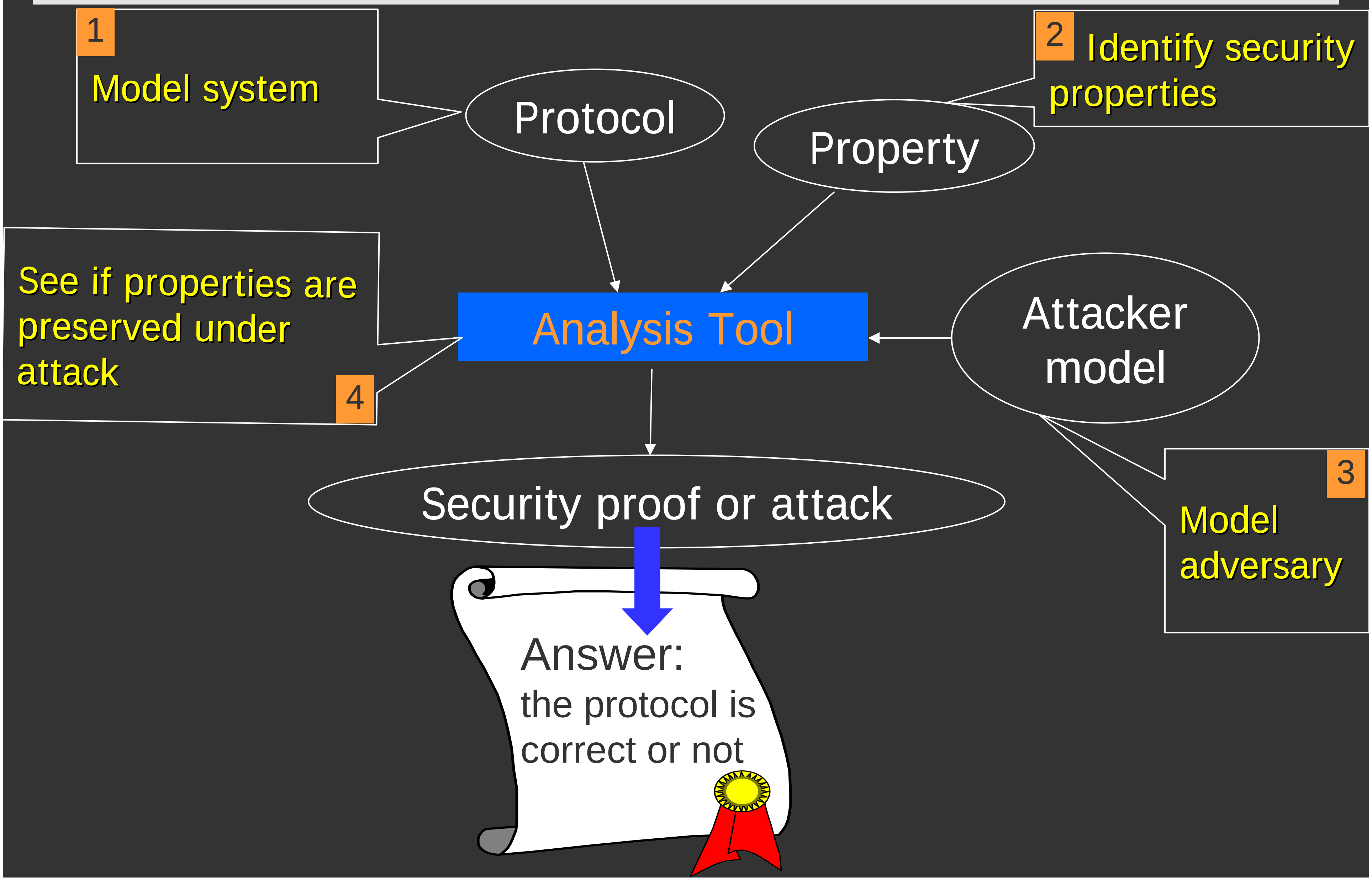
Part I: Overview

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

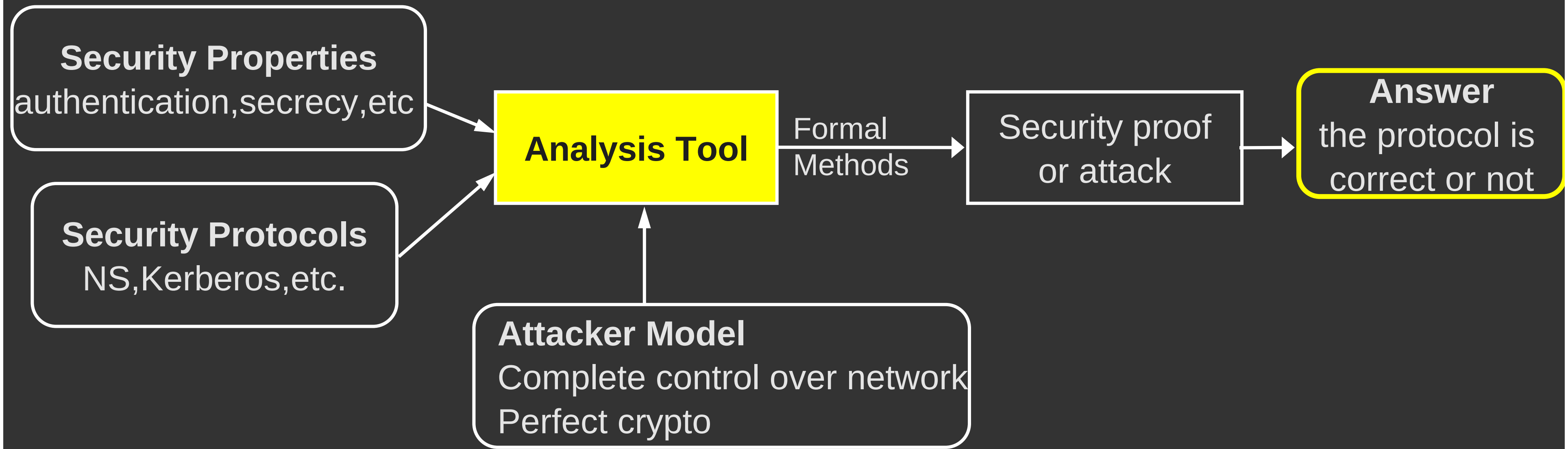
Part II: Athena Alogrithm

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

Central Problems



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 procol is flawed or it is correct

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

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

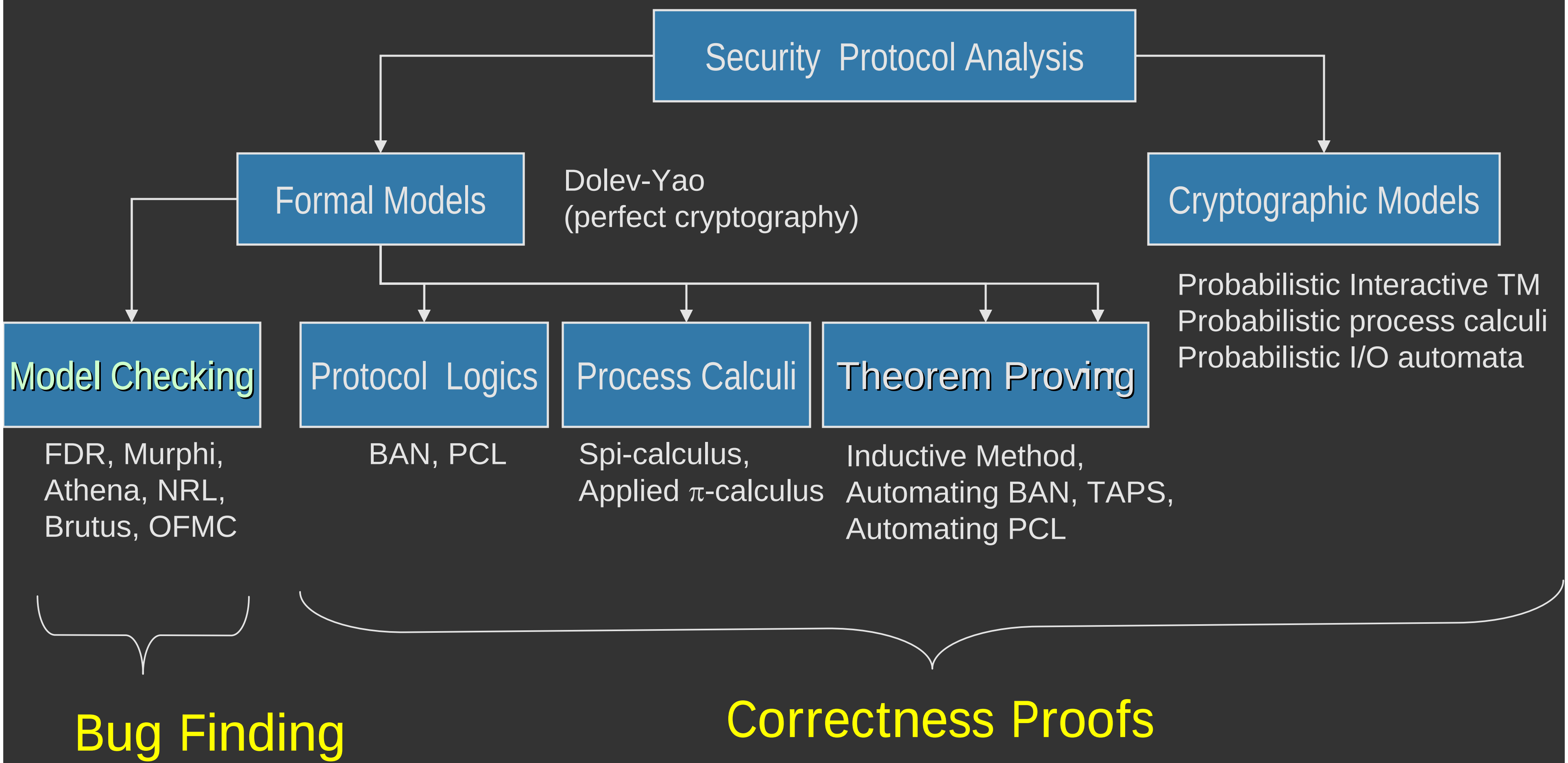
- Meadows: NRL Analyzer, Millen: Interrogator

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



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

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 Alogrithm

- 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?*

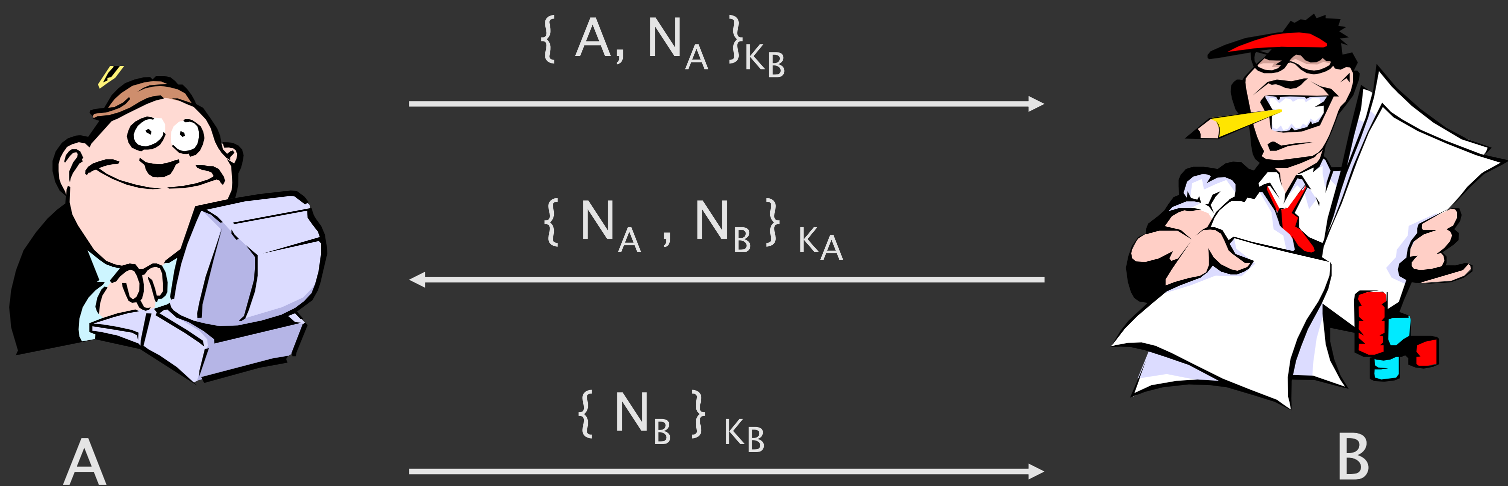
F. Javier Thayer Fábrega   Jonathan C. Herzog
Joshua D. Guttman
The MITRE Corporation
{jt, jherzog, guttman}@mitre.org

Thayer, Herzog, Guttman [THG98]

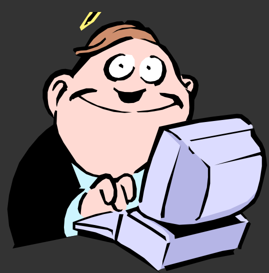
Defining Strand Spaces

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)



$\{ A, N_A \}_{KB}$ →

← $\{ N_A , N_B \}_{KA}$

$\{N_B\}_{KB}$ →

A's view of the protocol

B's view of the protocol

$\{ A, N_A \}_{KB}$ →

← $\{ N_A, N_B \}_{KA}$

$\{N_B \}_{KB}$ →

$+ \{ A, N_A \}_{KB}$

⇓

$- \{ N_A , N_B \}_{KA}$

⇓

$+ \{N_B \}_{KB}$

A's (trace of his) strand

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

+ means principal <u>sends</u> this message

– means principal <u>receives</u> this message

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

Security Properties in this Logic

- Answer Problem 2:
  Specifies the desired security properties

- Authentication

$$\forall C.Resp(\vec{x}) \in C \Longrightarrow Init(\vec{x}) \in C$$

- 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∈C , such that *term(n)=v*

$$\neg\exists C.(Resp(\vec{x}) \in C \wedge node(+v) \in C)$$

For NS Authentication

The formula that needs to be checked is

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

gamma

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

protocol

The NS protocol

$$\left\{ \begin{array}{c} +\{A, N_A\}_{K_B} \\ \Downarrow \\ -\{N_A, N_B\}_{K_A} \\ \Downarrow \\ +\{N_B\}_{K_B} \end{array} , \begin{array}{c} -\{A, N_A\}_{K_B} \\ \Downarrow \\ +\{N_A, N_B\}_{K_A} \\ \Downarrow \\ -\{N_B\}_{K_B} \end{array} \right\}$$

delta

- n_1: $+\{A_0, N_{a0}\}_{K_{b0}}$
- $\Downarrow$
- n_2: $-\{N_{a0}, N_{b0}\}_{K_{a0}}$
- $\Downarrow$
- 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 Alogrithm

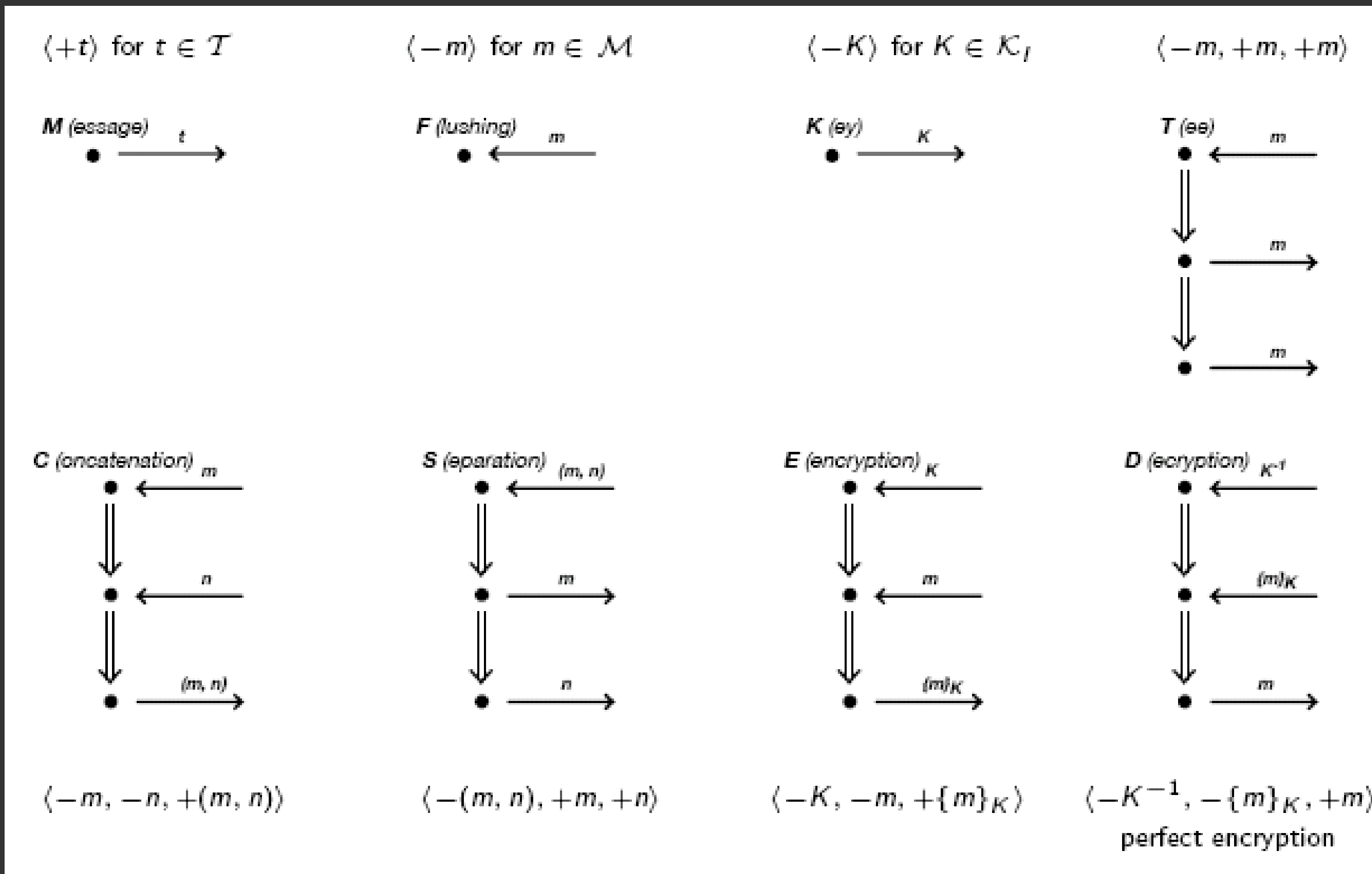
- Strand Spaces Model
- Security Properties
- Penetrator Strands
- Design of Model Checker
- Experiment Results

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”

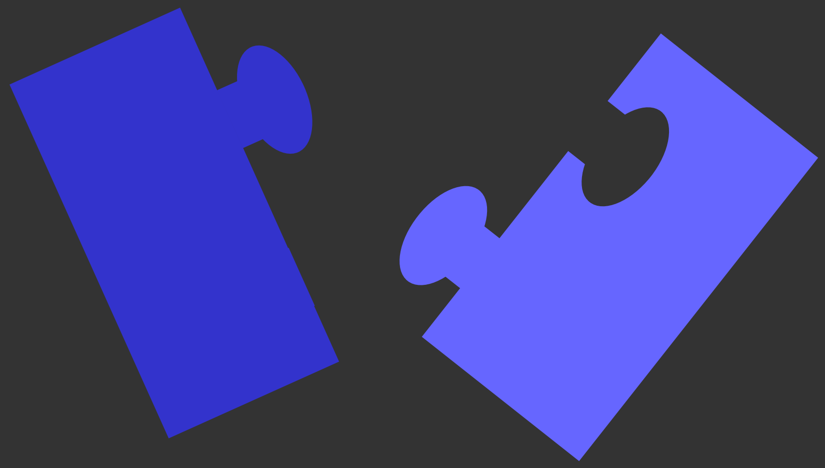
The Penetrator

Description of the intruder

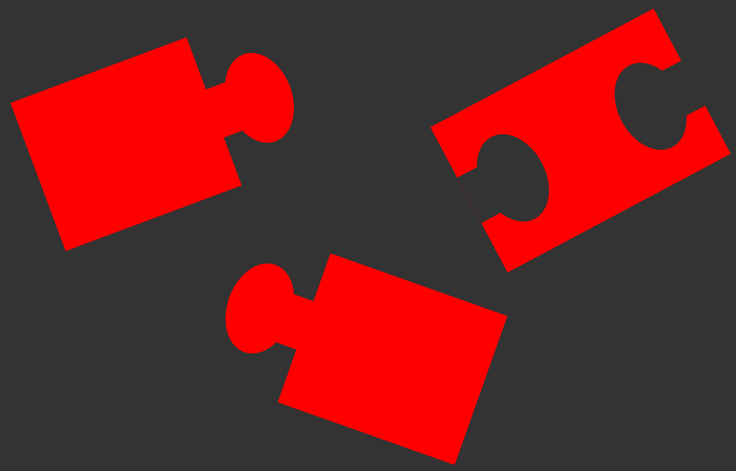


Composing Strands to Bundles

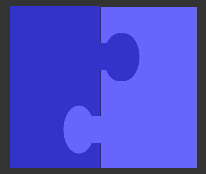
Regular strands



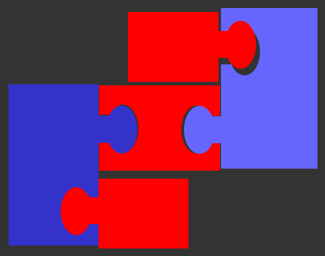
Penetrator strands



Intended protocol

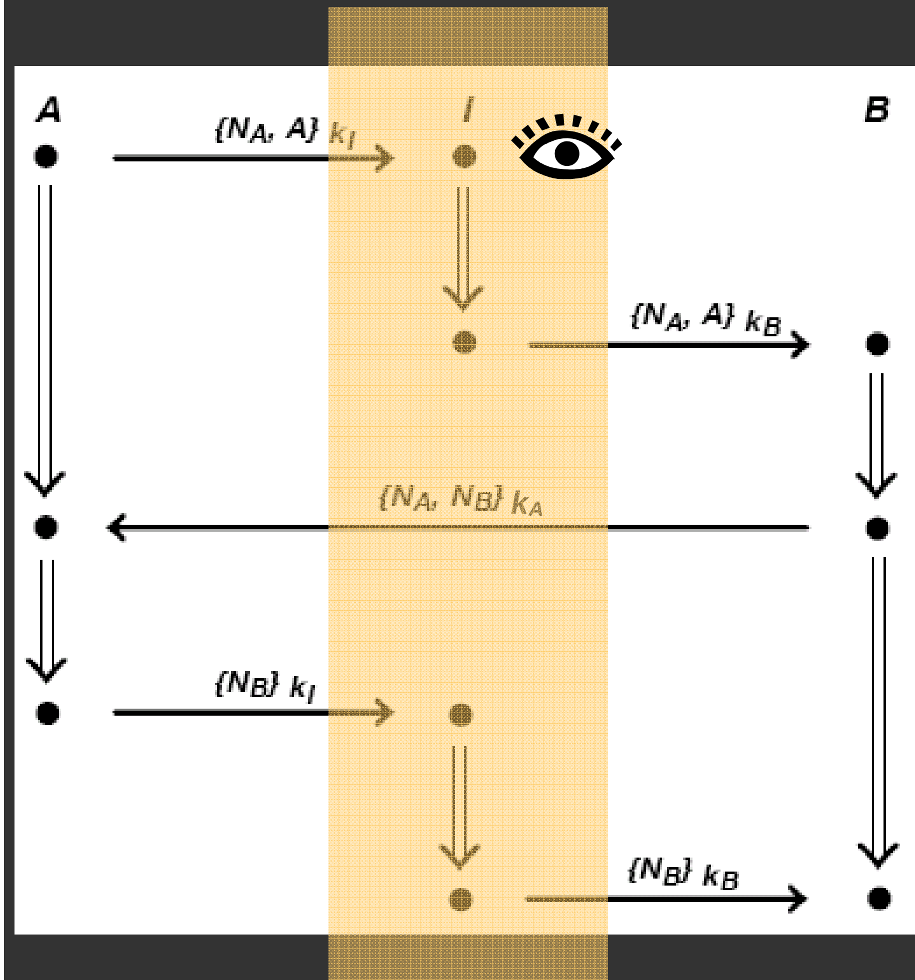


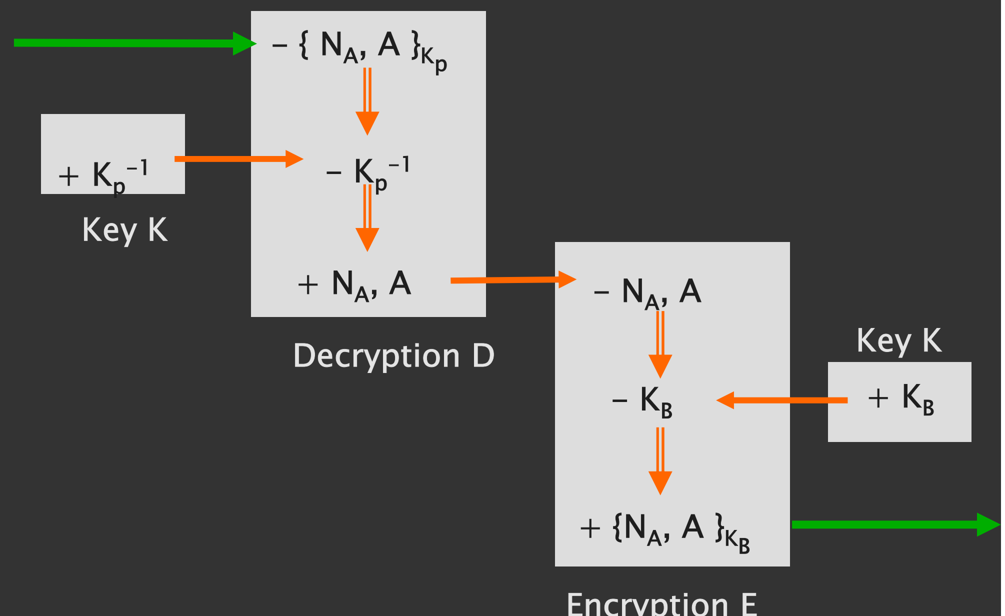
Attacker protocol



Penetrator‘s Work – An Example

Breaking into Needham–Schroeder protocol





Man-in-the-middle attack

pass messages through to another session A ↔ X ↔ B

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

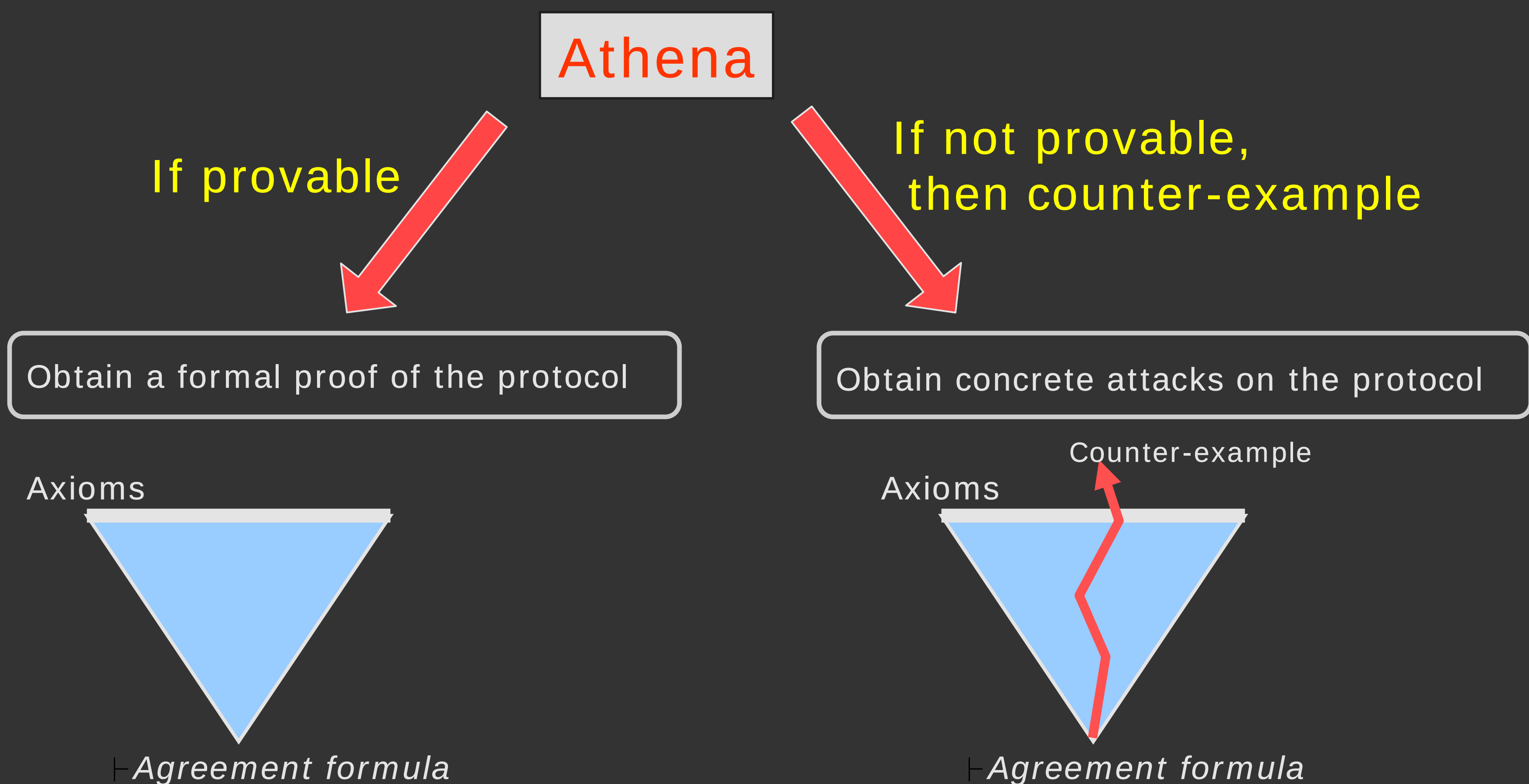
Design of Model Checker

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



Outline

Part I: Overview

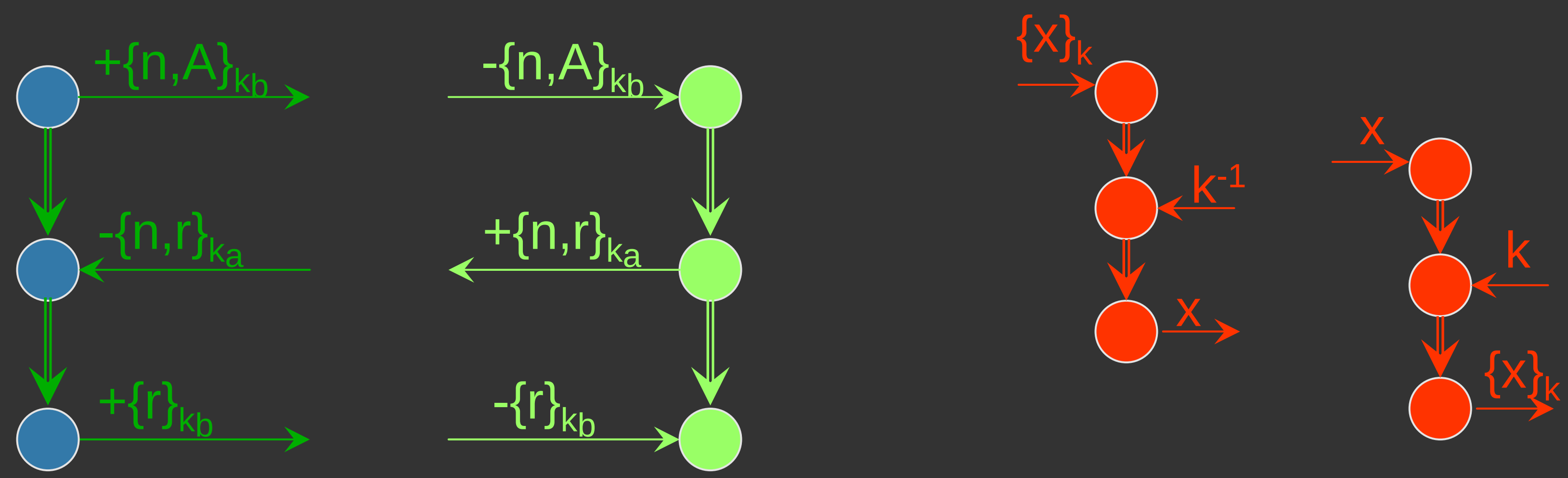
- Motivation
  - Why Security Protocols Analysis?
- Central problems
  - Security Analysis Methodology
- Current Analysis Techniques

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

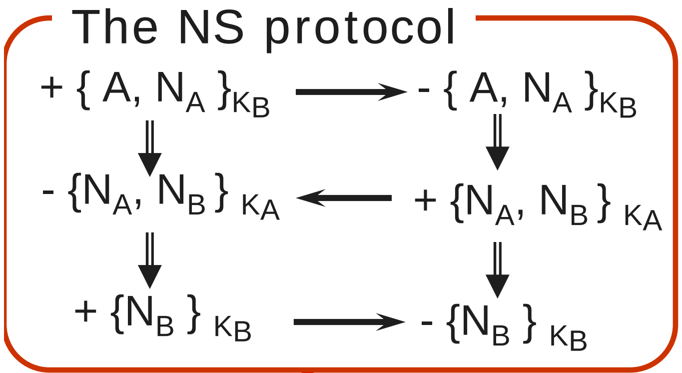
Same set of attacker strands for every protocol

The NS protocol

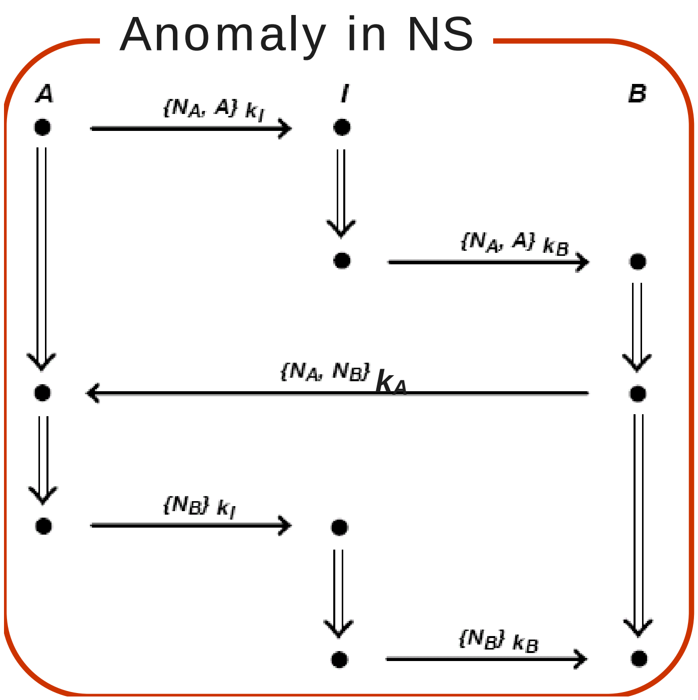
$1.\ A \rightarrow B : \{N_A, A\}_B$

$2.\ B \rightarrow A : \{N_A, N_B\}_A$

$3.\ A \rightarrow B : \{N_B\}_B$

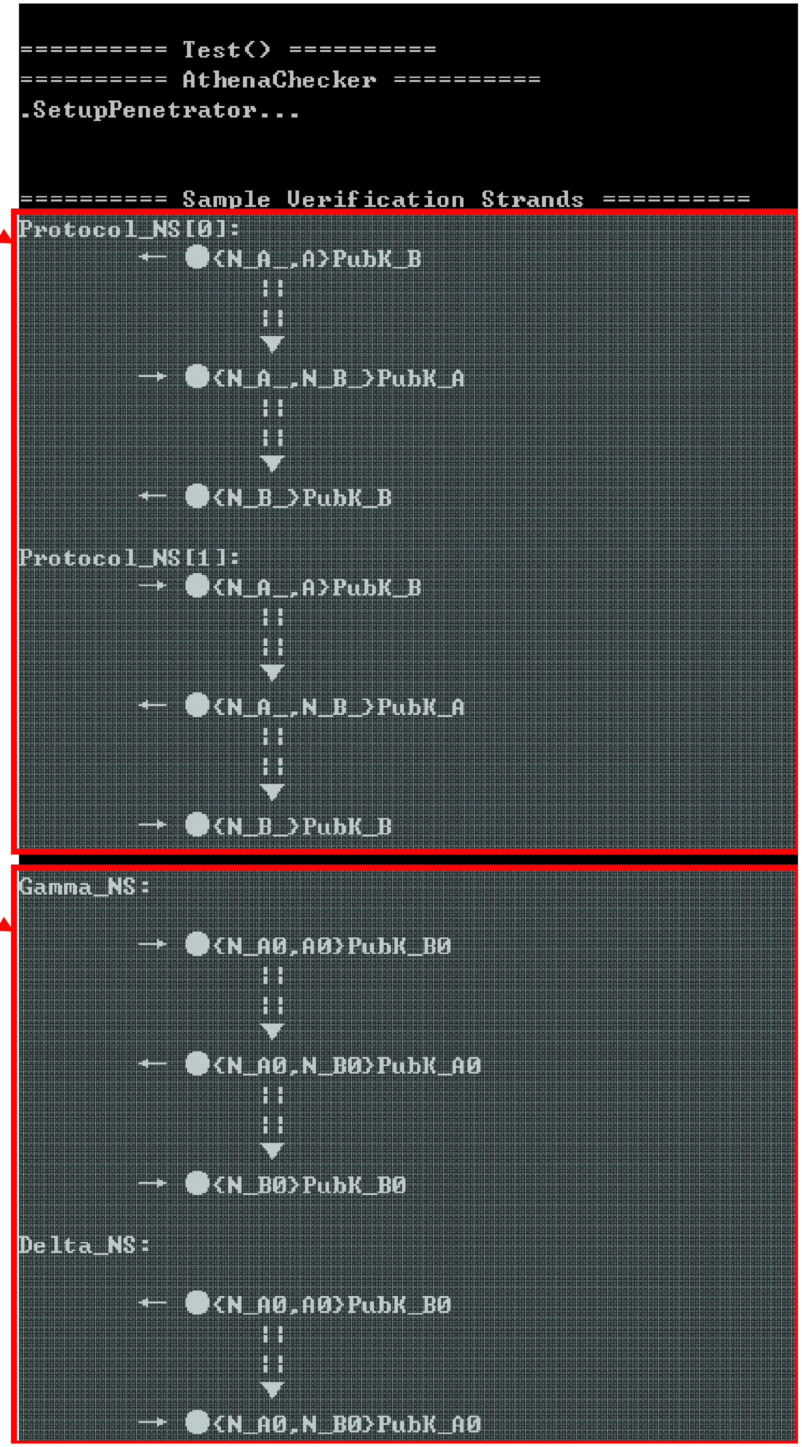


Try to discover the flaw in NS



NS Protocol

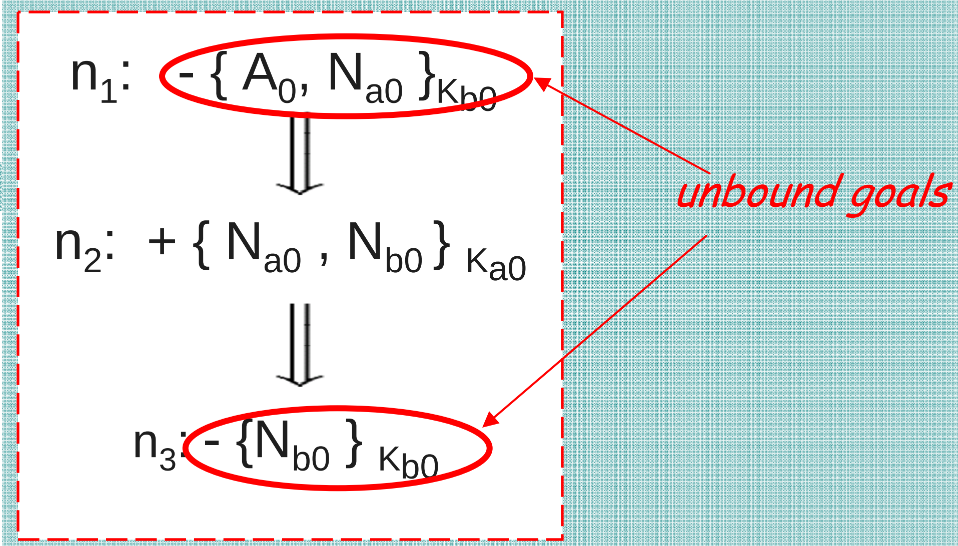
gamma,delta



NS Example

proc searchI(initialState){
    L = {initialState};
    while($\neg empty(L)$) *do*{
        $l = choose(L)$;
        $L = L \setminus l$;
        $L' = \mathcal{F}(l)$;
        if $\neg empty(L')$ *then*{
            for each $l' = \langle S', G', \twoheadrightarrow', \precsim' \rangle \in L'${
                if $s_2 \notin l'$ *then*{
                    if $empty(G')$ *then return* false;
                    else $L = add(l', L)$;
                }
            }
        }
    }
    return true;
}

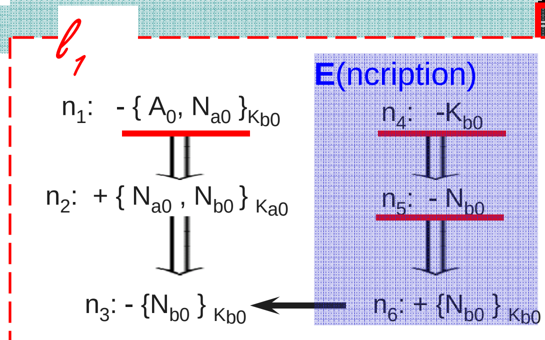
l_0=gamma $Resp[A_0, B_0, N_{a0}, N_{b0}]$




```
0.2) Commpute initial state l_0.

== State:l0
 = S(semi-bundle)
Strand[0]:
        ●N1:-<N_A0,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(l0).********************
chosen state:l0
>> 1.2 Generate next states.
===================== Finding NextStates ....
```

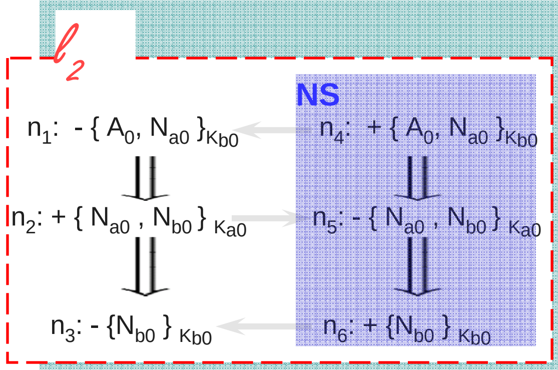
```
proc searchI(initialState){
    L = {initialState};
    while(¬empty(L)) do{
        l = choose(L);
        L = L \ l;
        L' = F(l);
        if ¬empty(L') then{
            for each l' = ⟨S', G', ↠', ≾'⟩ ∈ L'{
                if s2 ∉ l' then{
                    if empty(G') then return false;
                    else L = add(l', L);
                }
            }
        }
    }
    return true;
}
```



3 unbound goals

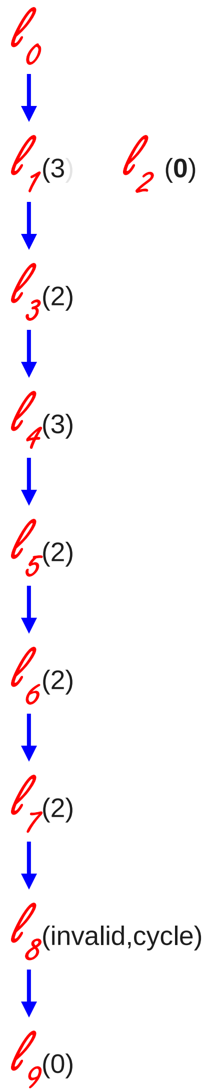
$\ell_0 \rightarrow \ell_1$ (3 unbound goal)

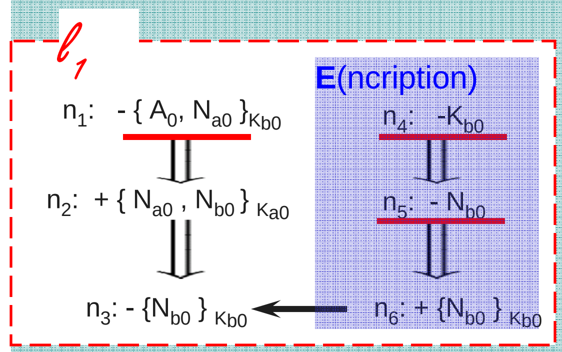
ℓ_2 (0 unbound goal)

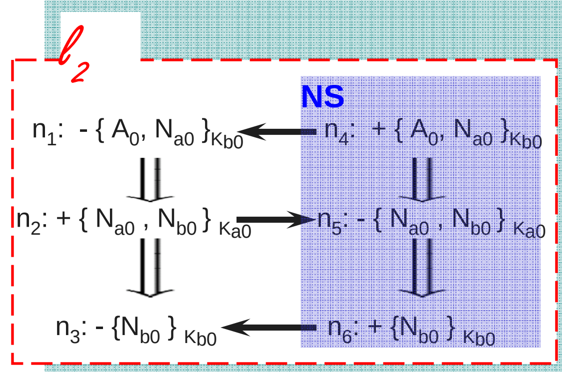



```
>> 1.2 Generate next states.
===================== Finding NextStates ....
1.pick up a goal from G:N3      {N_B0}PubK_B0
Generate 2 next states:
NextStates[0]:
== State:l1
= S(semi-bundle)
trand[0]:
        N1:-{N_A0,A0}PubK_B0
        N2:+{N_A0,N_B0}PubK_A0
        N3:-{N_B0}PubK_B0
trand[1]:
        N4:-PubK_B0
        N5:-N_B0
        N6:+{N_B0}PubK_B0
= G(unbound goal set)
        N1      {N_A0,A0}PubK_B0
        N4      PubK_B0
        N5      N_B0
 = B(goal binding set)
Binder[0]:
        Goal:N3 {N_B0}PubK_B0
        Node:N6   +{N_B0}PubK_B0
NextStates[1]:
== State:l2
 = S(semi-bundle)
Strand[0]:
        N1:-{N_A0,A0}PubK_B0
        N2:+{N_A0,N_B0}PubK_A0
        N3:-{N_B0}PubK_B0
Strand[1]:
        N4:+{N_A0,A0}PubK_B0
        N5:-{N_A0,N_B0}PubK_A0
        N6:+{N_B0}PubK_B0
 = G(unbound goal set)
 = B(goal binding set)
Binder[0]:
        Goal:N3 {N_B0}PubK_B0
        Node:N6   +{N_B0}PubK_B0
Binder[1]:
        Goal:N1 {N_A0,A0}PubK_B0
        Node:N4   +{N_A0,A0}PubK_B0
Binder[2]:
        Goal:N5 {N_A0,N_B0}PubK_A0
        Node:N2   +{N_A0,N_B0}PubK_A0
```

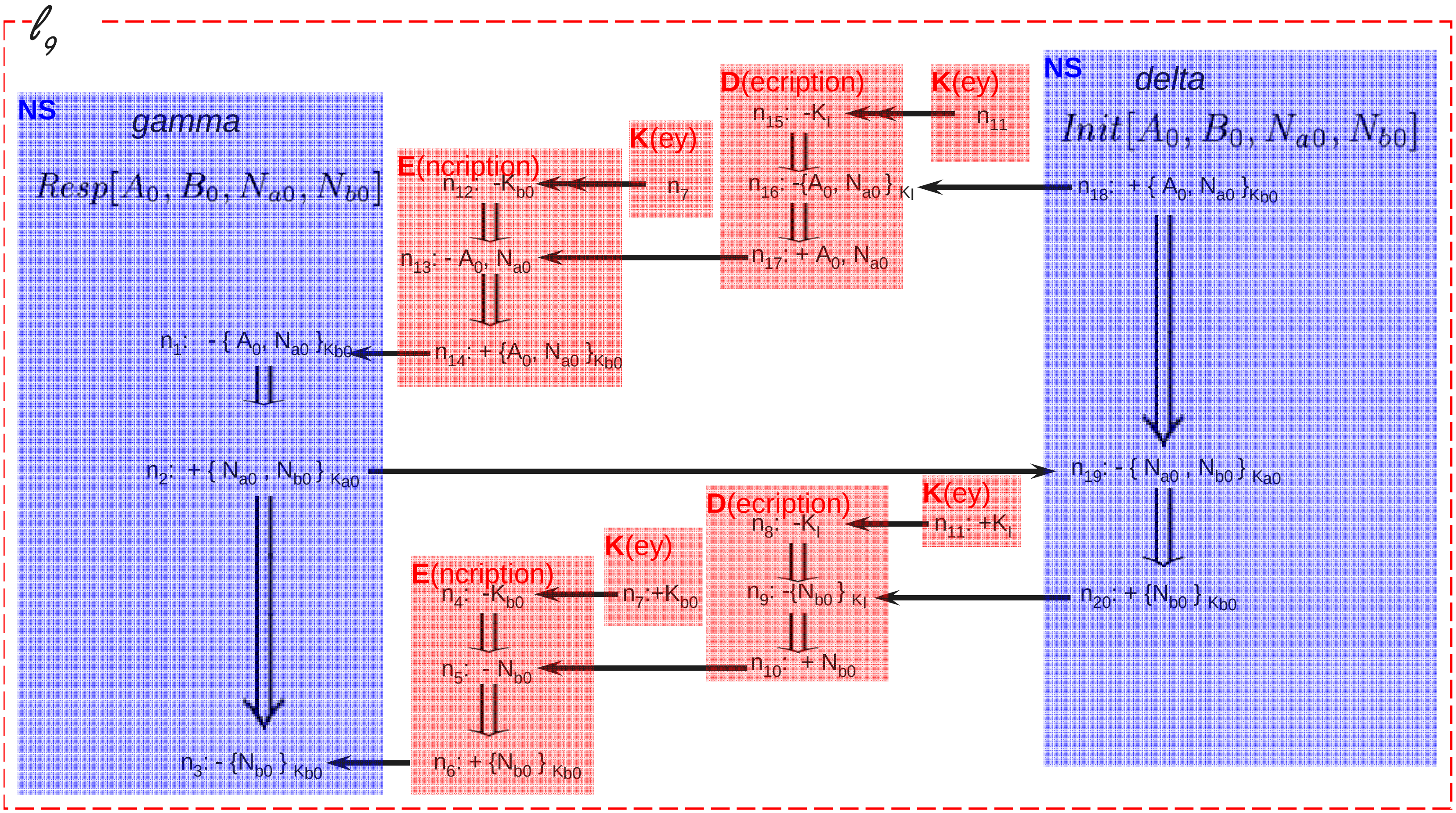
State Search for Needham-Shroeder



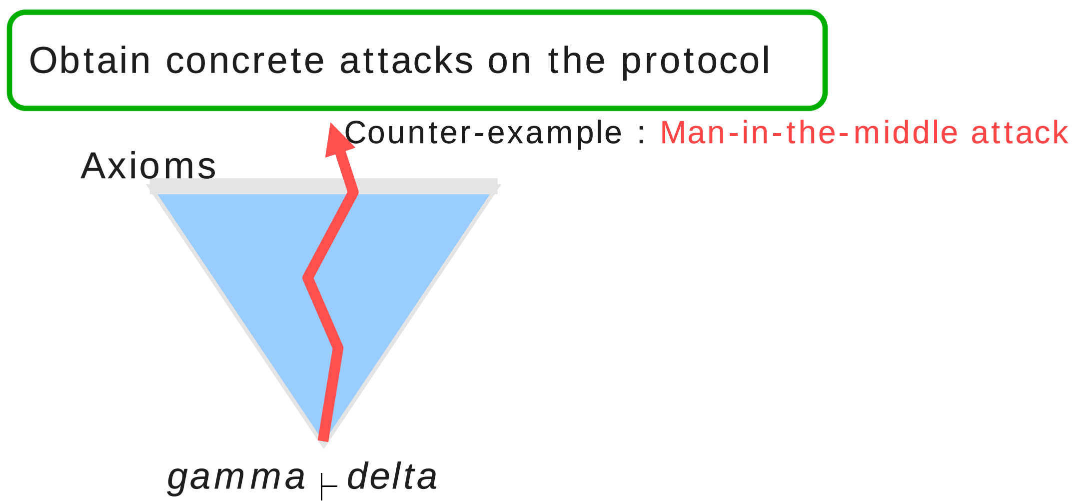




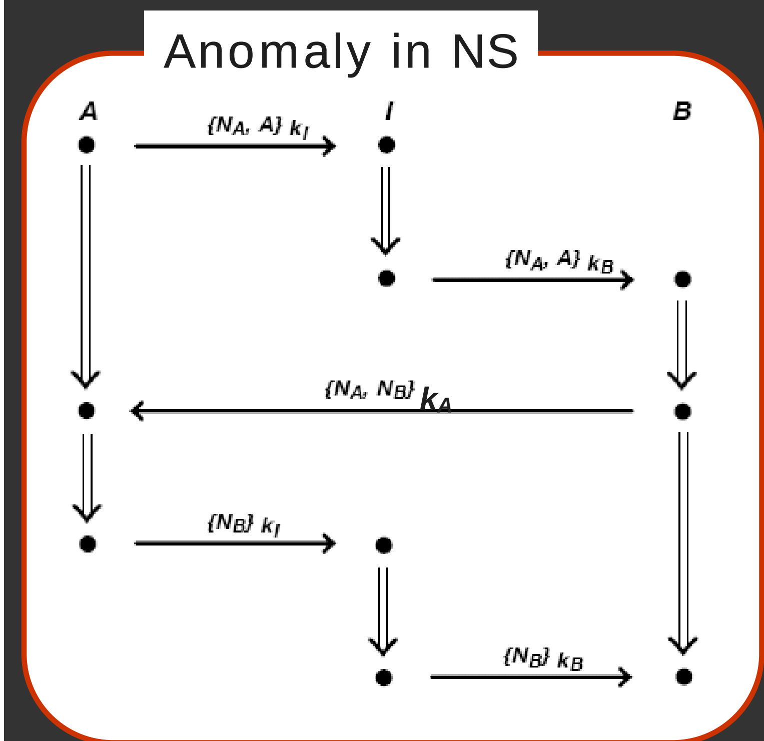

```
Strand[6]:
        N15:-PriK_I
        N16:-<N_A0,A0>PubK_I
        N17:+N_A0,A0,
Strand[7]:
        N18:+<N_A0,A0>PubK_I
        N19:-<N_A0,N_B0>PubK_A0
        N20:+<N_B0>PubK_I
 = G(unbound goal set)
 = B(goal binding set)
Binder[0]:
        Goal:N3 <N_B0>PubK_B0
        Node:N6  +<N_B0>PubK_B0
Binder[1]:
        Goal:N4 PubK_B0
        Node:N7  +PubK_B0
Binder[2]:
        Goal:N5 N_B0
        Node:N10  +N_B0,
Binder[3]:
        Goal:N8 PriK_I
        Node:N11  +PriK_I
Binder[4]:
        Goal:N1 <N_A0,A0>PubK_B0
        Node:N14  +<N_A0,A0>PubK_B0
Binder[5]:
        Goal:N12        PubK_B0
        Node:N7  +PubK_B0
Binder[6]:
        Goal:N13        N_A0,A0
        Node:N17  +N_A0,A0,
Binder[7]:
        Goal:N15        PriK_I
        Node:N11  +PriK_I
Binder[8]:
        Goal:N9 <N_B0>PubK_I
        Node:N20  +<N_B0>PubK_I
Binder[9]:
        Goal:N16        <N_A0,A0>PubK_I
        Node:N18  +<N_A0,A0>PubK_I
Binder[10]:
        Goal:N19        <N_A0,N_B0>PubK_A0
        Node:N2  +<N_A0,N_B0>PubK_A0
>> 1.3.1.1 delta is NOT a sub set of l' state
>> 1.3.1.1.1 G' is empty, verify WRONG
```

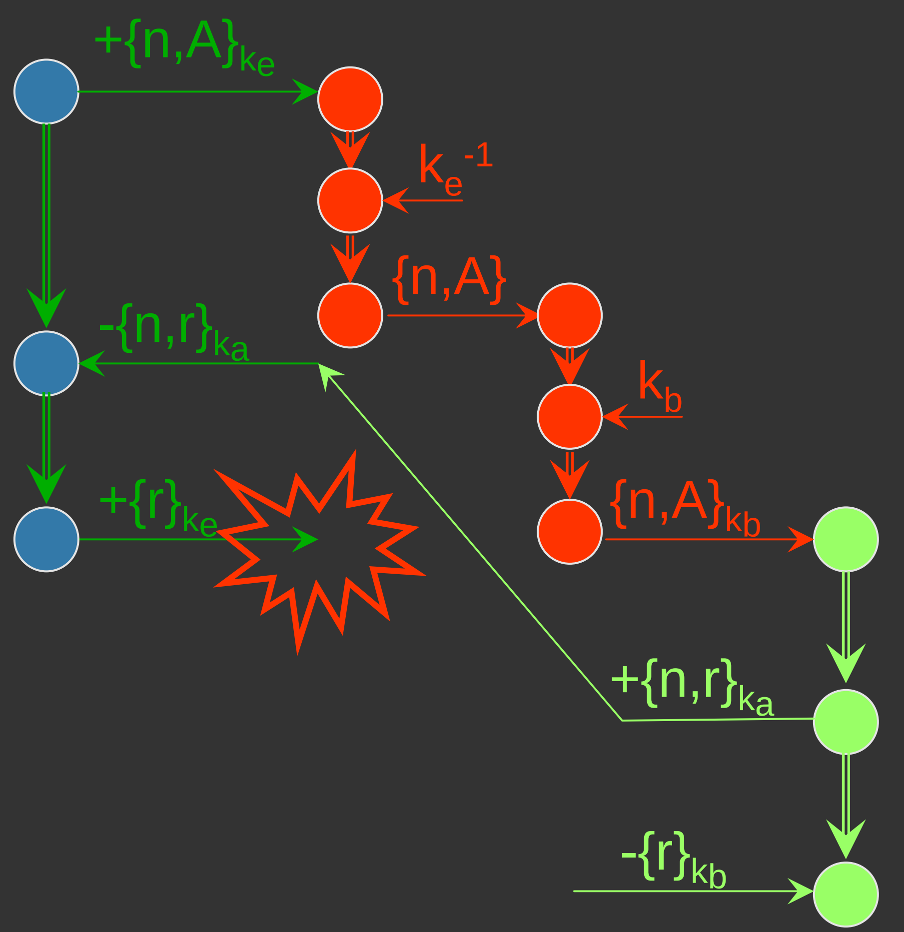


Search procedure terminate and counter example found !!!



NSPK Attack

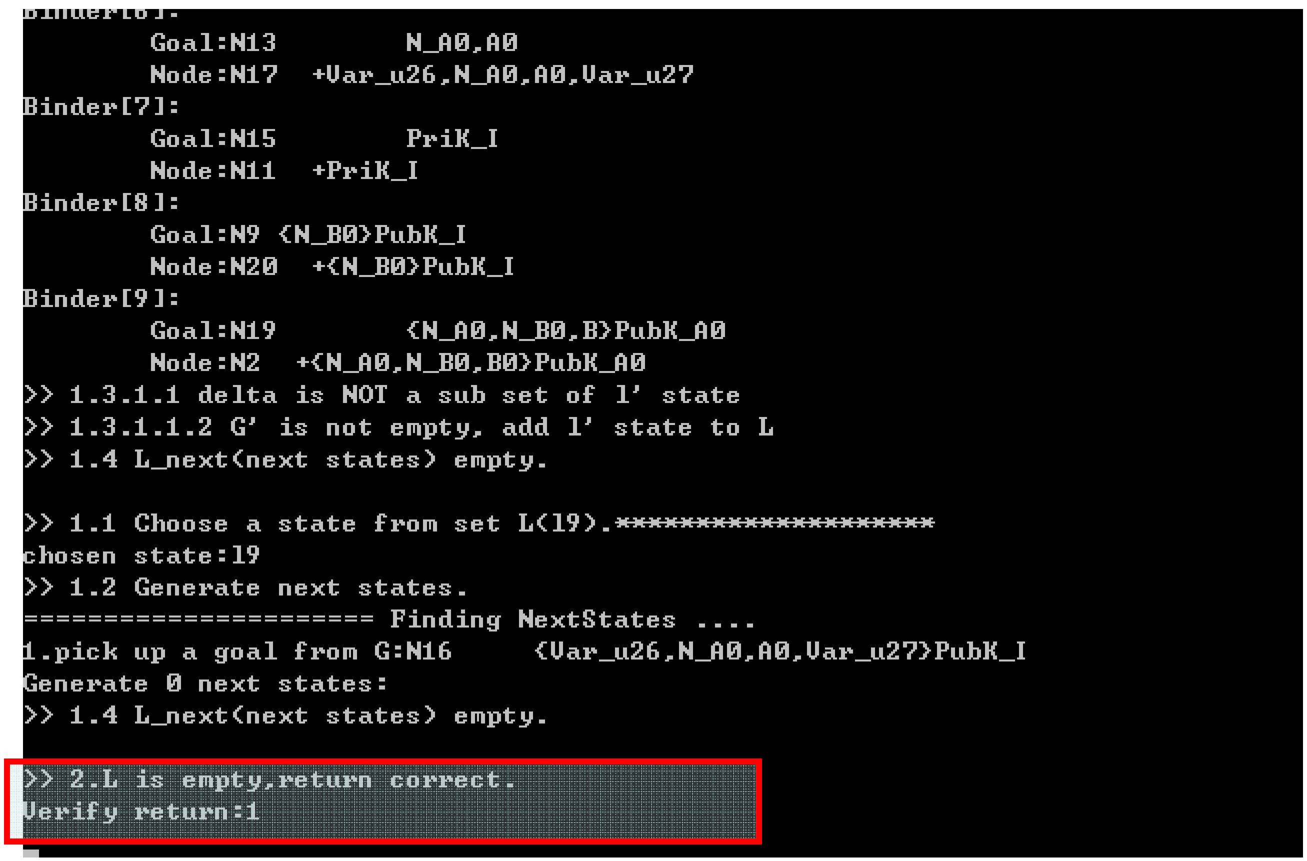

Anomaly in NS
A
I
B
{NA, A} kI
{NA, A} kB
{NA, NB} kA
{NB} kI
{NB} kB


+{n,A}ke
ke-1
{n,A}
-{n,r}ka
kb
+{r}ke
{n,A}kb
+{n,r}ka
-{r}kb

NSL verification

The NSL protocol

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



```
        Goal:N13        N_A0,A0
        Node:N17  +Var_u26,N_A0,A0,Var_u27
Binder[7]:
        Goal:N15        PriK_I
        Node:N11  +PriK_I
Binder[8]:
        Goal:N9 {N_B0}PubK_I
        Node:N20  +{N_B0}PubK_I
Binder[9]:
        Goal:N19        {N_A0,N_B0,B}PubK_A0
        Node:N2  +{N_A0,N_B0,B0}PubK_A0
>> 1.3.1.1 delta is NOT a sub set of l' state
>> 1.3.1.1.2 G' 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      {Var_u26,N_A0,A0,Var_u27}PubK_I
Generate 0 next states:
>> 1.4 L_next(next states) empty.

>> 2.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
- Secrity 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

[THG98] F.Javier Thayer, Jonathan C. Herzog, and Joshua D. Guttman. Strand spaces: Why is a security protocol correct? In Proceedings of 1998 IEEE Symposium on Security and Privacy, 1998.

[Low96] G. Lowe. Breaking and fixing the Needham-Schroeder public-key protocol using FDR. In Tools and Algorithms for the Construction and Analysis of Systems, volume 1055 of Lecture Notes in Computer Science, pages 147– 166. Springer-Verlag, 1996.

[Mil95] J. Millen. The Interrogator model. In Proceedings of the 1995 IEEE Symposium on Security and Privacy, pages 251–260. IEEE Computer Society Press, 1995.

[MMS97] J. C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using mur. In Proceedings of the 1997 IEEE Symposium on Security and Privacy. IEEE Computer Society Press, 1997.

[RN95] Stuart Russell and Peter Norvig. Artificial Intelligence: A Modern Approach. Prentice Hall Series in Artificial Intelligence, 1995.

[HT96] N. Heintze and J. Tygar. A model for secure protocols and their compositions. IEEE Transactions on Software Engineering, 22(1):16–30, January 1996.