On-the-fly Model Checking of Security Protocols

Guoqiang LI

Japan Advanced Institute of Science and Technology

Supervisor: Prof. Mizuhito Ogawa

February 7, 2008
A Challenging Problem and A Legend...

- A security protocol, although only containing several flows, is easily caused attacks even without breaking cryptography.

- Analyzing security protocols proves to be a challenging problem over 30 years.

- Analyzing the Needham-Schroeder authentication protocol becomes a legend.
  - In 1978, the NSSK and NSPK protocols are proposed.
  - In 1981, Denning and Sacco found NSSK was flawed and proposed a refined protocol.
  - In 1994, Abadi found that the refined protocol was also flawed.
  - In 1995, Lowe found an attack on the NSPK protocol. (17 years after its publication)
A security protocol, although only containing several flows, is easily caused attacks even without breaking cryptography.

Analyzing security protocols proves to be a challenging problem over 30 years.

Analyzing the Needham-Schroeder authentication protocol becomes a legend.

- In 1978, the NSSK and NSPK protocols are proposed.
- In 1981, Denning and Sacco found NSSK was flawed and proposed a refined protocol.
- In 1994, Abadi found that the refined protocol was also flawed.
- In 1995, Lowe found an attack on the NSPK protocol.
  (17 years after its publication)
A Challenging Problem and A Legend...

- A security protocol, although only containing several flows, is easily caused attacks even without breaking cryptography.

- Analyzing security protocols proves to be a challenging problem over 30 years.

- Analyzing the Needham-Schroeder authentication protocol becomes a legend.
  - In 1978, the NSSK and NSPK protocols are proposed.
  - In 1981, Denning and Sacco found NSSK was flawed and proposed a refined protocol.
  - In 1994, Abadi found that the refined protocol was also flawed.
  - In 1995, Lowe found an attack on the NSPK protocol. (17 years after its publication)
A Challenging Problem and A Legend...

A security protocol, although only containing several flows, is easily caused attacks even without breaking cryptography.

Analyzing security protocols proves to be a challenging problem over 30 years.

Analyzing the Needham-Schroeder authentication protocol becomes a legend.

- In 1978, the NSSK and NSPK protocols are proposed.
- In 1981, Denning and Sacco found NSSK was flawed and proposed a refined protocol.
- In 1994, Abadi found that the refined protocol was also flawed.
- In 1995, Lowe found an attack on the NSPK protocol. (17 years after its publication)
A Challenging Problem and A Legend...

- A security protocol, although only containing several flows, is easily caused attacks even without breaking cryptography.
- Analyzing security protocols proves to be a challenging problem over 30 years.
- Analyzing the Needham-Schroeder authentication protocol becomes a legend.
  - In 1978, the NSSK and NSPK protocols are proposed.
  - In 1981, Denning and Sacco found NSSK was flawed and proposed a refined protocol.
  - In 1994, Abadi found that the refined protocol was also flawed.
  - In 1995, Lowe found an attack on the NSPK protocol.
    (17 years after its publication)
The NSPK Protocol

\[
\begin{align*}
A \rightarrow B : & \quad \{A, N_A\} + K_B \\
B \rightarrow A : & \quad \{N_A, N_B\} + K_A \\
A \rightarrow B : & \quad \{N_B\} + K_B \\
I(A) \rightarrow I : & \quad \{A, N_A\} + K_I \\
I(A) \rightarrow B : & \quad \{A, N_A\} + K_B \\
B \rightarrow I(A) : & \quad \{N_A, N_B\} + K_A \\
I \rightarrow A : & \quad \{N_A, N_B\} + K_A \\
A \rightarrow I : & \quad \{N_B\} + K_I \\
A \rightarrow I(A) : & \quad \{N_B\} + K_A \\
B \rightarrow I(A) : & \quad \{N_B\} + K_B
\end{align*}
\]
The Fixed NSPK Protocol

$A \rightarrow B : \{A, N_A\} + K_B$

$B \rightarrow A : \{B, N_A, N_B\} + K_A$

$A \rightarrow B : \{N_B\} + K_B$

$A \rightarrow I : \{A, N_A\} + K_I$

$I(A) \rightarrow B : \{A, N_A\} + K_B$

$B \rightarrow I(A) : \{B, N_A, N_B\} + K_A$

$I \not\rightarrow A : \{I, N_A, N_B\} + K_A$
Security Protocols

A security protocol is a finite sequence of flows taken between two or more protocol roles using cryptography to establish security properties in a hostile environment.

Protocol roles comprise sender, receiver, server, and so on. Instances of protocol roles are named principals.

A session of a security protocol is one execution of the protocol attended by minimal number of principals. A fresh session is usually distinguished by a nonce.

A message is a bit stream representing the information communicated between principals. A undecomposable message is named a name.
Process Calculus

\( \pi \)-calculus
\( \pi ::= \overline{x} y \mid x(z) \mid \tau \)
\( P ::= 0 \mid \pi. P \mid P | P \mid [x = y] P \mid \nu z P \mid P + P \mid !P \)
e.g.
\( (((\nu z)\overline{x} z.P) \mid x(y).Q \mid R \xrightarrow{+} (((\nu z)P \mid Q\{z/y\}) \mid R \)

- **Channel-based** Vs. **Environment-based**
  - **Communications:** exchange with a specific channel Vs. exchange with environment
  - **Freshness of names:** scopes of local names Vs. fresh public names
  - **Generation of messages by intruders and dishonest principals:** recursive processes Vs. deductive systems (Dolev-Yao model)

- **Replication** \( !P \) Vs. **Identifier** \( \Delta(x_1, \ldots, x_n) \)
  - Each identifier has the unique definition \( \Delta(x_1, \ldots, x_n) \triangleq P \).
  - Infinite process definitions: \( !P \equiv P \mid P \mid \ldots \) Vs. \( \Delta \triangleq E(\Delta) \).
Factors of Infinity

- **Unbounded number of messages** that intruders and dishonest principals can generate.
  - **Idea:** Variables are substituted only when needed.

- **Unbounded number of principals** that each principal contacts.
  - **Idea:** Names are extended from **constants** to **terms**, called **binders**. These binders are instantiated depending on the context.

- **Unbounded number of sessions** that each principal participates.
  - **Idea:** Restricting primitives due to difference assumptions.
    - **Bounded sessions:** contains no recursive processes (without identifiers), and uses **distinguished symbols** for fresh messages.
    - **Recursive protocols:** contains only one sequential recursive process, and uses **nested binders** for fresh messages.
Infinite Messages and Lazy Evaluation

A → B : NA, {A, M}K_{AB}

Sending a message

Lazy evaluation

Input requirement
Infinite Messages and Lazy Evaluation

\[ A \rightarrow B : \quad N_A, \{ A, M \}_{K_{AB}} \]

Sending a message

Lazy evaluation

Input requirement

\[ \bar{a}(N_A, \{ A, M \}_{K_{[A,B]}}) \quad b(x) \]
Infinite Messages and Lazy Evaluation

Sending a message
\[ \alpha(N_A, \{A, M\}_{k_{AB}}) \]

Lazy evaluation
\[ b(x) \]

Input requirement
\[ b(x_1, x_2) \]

Require pair splitting
Infinite Messages and Lazy Evaluation

\[
A \rightarrow B : N_A, \{A, M\}_{K_{AB}}
\]

Sending a message
\[
\overline{a}(N_A, \{A, M\}_{k[A,B]})
\]

Lazy evaluation
\[
b(x)
\]

Input requirement
\[
b(x_1, x_2)
\]

Require pair splitting
\[
b(x_1, \{y\}_{k[A,B]})
\]

Require decrypting
Infinite Messages and Lazy Evaluation

\[ A \rightarrow B : \quad N_A, \{A, M\}_K_{AB} \]

**Sending a message**
- \( \overline{a}(N_A, \{A, M\}_k_{A,B}) \)
- \( \overline{a}(N_A, \{A, M\}_k_{A,B}) \)
- \( \overline{a}(N_A, \{A, M\}_k_{A,B}) \)
- \( \overline{a}(N_A, \{A, M\}_k_{A,B}) \)

**Lazy evaluation**
- \( b(x) \)
- \( b(x_1, x_2) \)
- \( b(x_1, \{y\}_k_{A,B}) \)
- \( b(x_1, \{z, w\}_k_{A,B}) \)

**Input requirement**
- Require pair splitting
- Require decrypting
- Require pair splitting
Infinite Messages and Lazy Evaluation

\( A \rightarrow B : \quad N_A, \{A, M\}_{k_{AB}} \)

- **Sending a message**
  \( \overline{a}(N_A, \{A, M\}_{k_{A,B}}) \)
- **Lazy evaluation**
  \( b(x) \)
- **Input requirement**
  \( b(x_1, x_2) \)
  - Require pair splitting
  \( b(x_1, \{y\}_{k_{A,B}}) \)
  - Require decrypting
  \( b(x_1, \{z, w\}_{k_{A,B}}) \)
  - Require pair splitting
  \( b(x_1, \{A, w\}_{k_{A,B}}) \)
  - Require comparing
Infinite Messages and Lazy Evaluation

\[ A \rightarrow B : N_A, \{A, M\}_{k_{AB}} \]

Sending a message
\[ \overline{a}(N_A, \{A, M\}_{k_{AB}}) \]

Lazy evaluation
\[ b(x) \]

Input requirement
\[ b(x_1, x_2) \]

Require pair splitting
\[ b(x_1, \{y\}_{k_{AB}}) \]

Require decrypting
\[ b(x_1, \{z, w\}_{k_{AB}}) \]

Require pair splitting
\[ b(x_1, \{A, w\}_{k_{AB}}) \]

Require comparing
\[ b(x_1, \{A, M\}_{k_{AB}}) \]

Require encrypted message

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

\[ A \rightarrow B : \{A, N_A\} + K_B \]

Existing method

\[ \{A, N_A\} + K_B \]

\[ A \rightarrow B \]

\( a_1 \{A, N_A\} + K_A \)

\( a_2 \{A, N'_A\} + K_B \)

\( a_3 \{A, N''_A\} + K_C \)

(\( new \ x \ I \))

\( I \) : set of principals' names

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

\[ A \rightarrow B : \{ A, N_A \} + K_B \]

Existing method

\[ \{ A, N_A \} + K_B \]

\[ \{ A, N_A \} + K_C \]

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

\[ A \rightarrow B : \{A, N_A\} + K_B \]

Existing method

\[ \{A, N_A\} + K_B \]

\[ \{A, N_A'\} + K_C \]

\[ \{A, N_A''\} + K_I \]

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

\[ A \rightarrow B : \{A, N_A\} + K_B \]

Existing method

\[
\begin{align*}
\{A, N_A\} + K_B \\
\{A, N'_A\} + K_C \\
\{A, N''_A\} + K_I \\
\end{align*}
\]

\[
\begin{align*}
\overline{a_1}\{A, N_A\} _{K_A} & \mid \overline{a_2}\{A, N'_A\} _{K_B} & \mid \overline{a_3}\{A, N''_A\} _{K_I}
\end{align*}
\]

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

Existing method

\[ A \rightarrow B : \{A, N_A\} + K_B \]

Our approach

\[ A \rightarrow B : \{A, N_A\} + K_B \]

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

Existing method

\[
A \rightarrow B : \{A, N_A\} + K_B
\]

Our approach

\[
\bar{a}_1\{A, N_A\}_K_A \mid \bar{a}_2\{A, N'_A\}_K_B \mid \bar{a}_3\{A, N''_A\}_K_I
\]
Infinite Principals and Binders

Existing method

\[ A \rightarrow B : \{A, N_A\}_B \]

Our approach

\[ \overline{a_1}\{A, N_A\}_A | \overline{a_2}\{A, N'_A\}_B | \overline{a_3}\{A, N''_A\}_I \]

On-the-fly Model Checking of Security Protocols
Infinite Principals and Binders

\[ A \rightarrow B : \{A, N_A\} + K_B \]

Existing method

\[ \overline{a_1} \{A, N_A\}_K \mid \overline{a_2} \{A, N'_A\}_K \mid \overline{a_3} \{A, N''_A\}_K \]

Our approach

\[ (\text{new } x : \mathcal{I}) \overline{a_1} \{A, N_A\}_K[x] \]

\( \mathcal{I} \): set of principals' names

On-the-fly Model Checking of Security Protocols
Infinite Sessions and Freshness of Messages

- Freshness of sessions is usually captured by fresh nonces.
- Only one sequential recursive process is allowed in the system. It is later encoded by the pushdown system.
- Unbounded number of nonces are represented by nested applications of binders. (e.g. \( N[\text{null}] \), \( N[N[\text{null}]] \), ...)

\[
A \rightarrow B : \quad \{A, N_A\}_{+K_B}
\]

- \( A \) recursively sending messages is encoded by \( \langle \{A, N[]\}_{+K[]}, * \rangle \). (by \( N^*[[]] \), a group of distinguished messages is represented.)
- Recursive protocols can be analyzed within the restriction.
| Environment-based process calculus + Environmental deductive system |
|---|---|---|
| Allows finite processes Sec. & Auth. in bounded sessions | Extended with a recursive process Recursive Protocols | Extended with $P$-deductive system Fair. & Non-r. in bounded sessions |
| Protocol-independent spec. transf. to a reachability problem | Protocol-independent spec. transf. to a reachability problem | Protocol-dependent spec. transf. to a reachability problem |
| Parametrization Refinement step | Parametrization Refinement step | Parametrization Two-phase refinement steps |
| Encoded by a parametric model | Encoded by the PDS | Encoded by a parametric model |
| Implemented by Maude | Implemented by Maude | Implemented by Maude |
## Thesis Outline

| Environment-based process calculus + Environmental deductive system |
|---|---|---|
| **Allows finite processes**  
Sec. & Auth. in bounded sessions | **Extended with a recursive process**  
Recursive Protocols | **Extended with P-deductive system**  
Fair. & Non-r. in bounded sessions |
| Protocol-independent spec. transf.  
to a reachability problem | Protocol-independent spec. transf.  
to a reachability problem | Protocol-dependent spec. transf.  
to a reachability problem |
| Parametrization  
Refinement step | Parametrization  
Refinement step | Parametrization  
Two-phase refinement steps |
| Encoded by a parametric model | Encoded by the PDS | Encoded by a parametric model |
| Implemented by Maude | Implemented by Maude | Implemented by Maude |

On-the-fly Model Checking of Security Protocols
Environment-based process calculus + Environmental deductive system

Allows finite processes
Sec. & Auth. in bounded sessions

Extended with a recursive process
Recursive Protocols

Extended with $P$-deductive system
Fair. & Non-r. in bounded sessions

Protocol-independent spec. transf.
to a reachability problem

Protocol-independent spec. transf.
to a reachability problem

Protocol-dependent spec. transf.
to a reachability problem

Parametrization
Refinement step

Parametrization
Refinement step

Parametrization
Two-phase refinement steps

Encoded by a parametric model

Encoded by the PDS

Encoded by a parametric model

Implemented by Maude

Implemented by Maude

Implemented by Maude

On-the-fly Model Checking of Security Protocols
# Thesis Outline

| Environment-based process calculus + Environmental deductive system |
|---|---|---|
| Allows finite processes Sec. & Auth. in bounded sessions | Extended with a recursive process Recursive Protocols | Extended with $P$-deductive system Fair. & Non-r. in bounded sessions |
| Protocol-independent spec. transf. to a reachability problem | Protocol-independent spec. transf. to a reachability problem | Protocol-dependent spec. transf. to a reachability problem |
| Parametrization Refinement step | Parametrization Refinement step | Parametrization Two-phase refinement steps |
| Encoded by a parametric model | Encoded by the PDS | Encoded by a parametric model |
| Implemented by Maude | Implemented by Maude | Implemented by Maude |

On-the-fly Model Checking of Security Protocols
Environment-based process calculus + Environmental deductive system

- Allows finite processes
  Sec. & Auth. in bounded sessions
- Protocol-independent spec. transf. to a reachability problem
- Parametrization Refinement step
- Encoded by a parametric model
- Implemented by Maude

Extended with a recursive process
Recursive Protocols
Extended with P-deductive system
Fair. & Non-r. in bounded sessions

Protocol-independent spec. transf. to a reachability problem
Parametrization Refinement step
Encoded by the PDS
Implemented by Maude

Protocol-independent spec. transf. to a reachability problem
Parametrization Refinement step
Parametrization
Two-phase refinement steps
Encoded by a parametric model
Implemented by Maude
Thesis Outline

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  Sec. & Auth. in bounded sessions
- Protocol-independent spec. transf. to a reachability problem
- Parametrization
  Refinement step
- Encoded by a parametric model
- Implemented by Maude

Extended with a recursive process
Recursive Protocols

- Protocol-independent spec. transf. to a reachability problem
- Parametrization
  Refinement step
- Encoded by the PDS
- Implemented by Maude

Extended with $P$-deductive system
Fair. & Non-r. in bounded sessions

- Protocol-dependent spec. transf. to a reachability problem
- Parametrization
  Two-phase refinement steps
- Encoded by a parametric model
- Implemented by Maude

On-the-fly Model Checking of Security Protocols
**Thesis Outline**

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  - Sec. & Auth. in bounded sessions
- Protocol-independent spec. transf.
  - to a reachability problem
- Parametrization
  - Refinement step
- Encoded by a parametric model
- Implemented by Maude

Extended with a recursive process
- Recursive Protocols
- Protocol-independent spec. transf.
  - to a reachability problem
- Parametrization
  - Refinement step
- Encoded by the PDS
- Implemented by Maude

Extended with P-deductive system
- Fair. & Non-r. in bounded sessions
- Protocol-dependent spec. transf.
  - to a reachability problem
- Parametrization
  - Two-phase refinement steps
- Encoded by a parametric model
- Implemented by Maude

On-the-fly Model Checking of Security Protocols
Thesis Outline

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  - Sec. & Auth. in bounded sessions

- Protocol-independent spec. transf.
  - to a reachability problem

- Parametrization
  - Refinement step
  - Encoded by a parametric model
  - Implemented by Maude

- Extended with a recursive process
  - Recursive Protocols

- Protocol-independent spec. transf.
  - to a reachability problem

- Parametrization
  - Refinement step

- Extended with P-deductive system
  - Fair. & Non-r. in bounded sessions

- Protocol-dependent spec. transf.
  - to a reachability problem

- Parametrization
  - Two-phase refinement steps

- Encoded by the PDS
  - Implemented by Maude

On-the-fly Model Checking of Security Protocols
Thesis Outline

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  - Sec. & Auth. in bounded sessions
- Extended with a recursive process
  - Recursive Protocols
- Extended with $P$-deductive system
  - Fair. & Non-r. in bounded sessions

- Protocol-independent spec. transf.
  - to a reachability problem
- Protocol-dependent spec. transf.
  - to a reachability problem
- Protocol-dependent spec. transf.
  - Two-phase refinement steps

- Parametrization
  - Refinement step
- Parametrization
  - Refinement step
- Parametrization
  - Two-phase refinement steps

- Encoded by a parametric model
- Encoded by the PDS
- Encoded by a parametric model

- Implemented by Maude
- Implemented by Maude
- Implemented by Maude

On-the-fly Model Checking of Security Protocols
Thesis Outline

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  Sec. & Auth. in bounded sessions

- Protocol-independent spec. transf.
  to a reachability problem

- Parametrization
  Refinement step

- Encoded by a parametric model

- Implemented by Maude

- Extended with a recursive process
  Recursive Protocols

- Protocol-independent spec. transf.
  to a reachability problem

- Parametrization
  Refinement step

- Encoded by the PDS

- Implemented by Maude

- Extended with $P$-deductive system
  Fair. & Non-r. in bounded sessions

- Protocol-dependent spec. transf.
  to a reachability problem

- Parametrization
  Two-phase refinement steps

- Encoded by a parametric model

- Implemented by Maude
Thesis Outline

Environment-based process calculus + Environmental deductive system

- Allows finite processes
  - Sec. & Auth. in bounded sessions
- Protocol-independent spec. transf. to a reachability problem
- Parametrization
  - Refinement step
  - Encoded by a parametric model
  - Implemented by Maude

- Extended with a recursive process
  - Recursive Protocols
- Protocol-independent spec. transf. to a reachability problem
  - Parametrization
  - Refinement step
  - Encoded by the PDS
  - Implemented by Maude

- Extended with $P$-deductive system
  - Fair. & Non-r. in bounded sessions
- Protocol-dependent spec. transf. to a reachability problem
- Parametrization
  - Two-phase refinement steps
  - Encoded by a parametric model
  - Implemented by Maude

On-the-fly Model Checking of Security Protocols
Introduction

Modeling Protocols for Authentication and Secrecy
- Description of Security Protocols
- Environment Ability
- Specification of Security Protocols

Authentication and Secrecy in Bounded Sessions

Authentication for Recursive Protocols

Non-repudiation and Fairness in Bounded Sessions

Conclusion
### Process Calculus

#### Syntax:

- $pr ::= x$
- $m[pr_1, \ldots, pr_n]$
- $\bar{a}M.P$
- $a(x).P$
- $[M = N]P$
- $(\text{new } x : A)P$
- $(\nu n)P$
- $let (x, y) = M \text{ in } P$
- $\text{case } M \text{ of } \{x\}_L \text{ in } P$
- $P \parallel Q$
- $P + Q$
- $P; Q$
- $\Delta(pr_1, \ldots, pr_n)$

#### Semantics:

- $\mathcal{H}(M)$
- $\text{output}$
- $\text{input}$
- $\text{match}$
- $\text{new}$
- $\text{restriction}$
- $\text{pair splitting}$
- $\text{decryption}$
- $\text{composition}$
- $\text{summation}$
- $\text{sequence}$
- $\text{identifier}$

---

**On-the-fly Model Checking of Security Protocols**
Description of the NSPK Protocol

\[\begin{align*}
A \rightarrow B : & \quad \{A, N_A\} + K_B \\
B \rightarrow A : & \quad \{N_A, N_B\} + K_A \\
A \rightarrow B : & \quad \{N_B\} + K_B
\end{align*}\]

\[A \triangleq (\text{new } x_a : \mathcal{T})(\nu N_A)
\overline{a1}\{A, N_A\} + _{k[x_a]} a2(y_a).
case y_a of \{y'_a\} - {k}[A] \text{ in}
let (z_a, z'_a) = y'_a \text{ in}
[z_a = N_A] \overline{a3}\{z'_a\} +_{k[x_a]} 0\]

\[B \triangleq (\nu N_B) b1(x_b).
case x_b of \{x'_b\} - _{k}[B] \text{ in}
let (y_b, y'_b) = x'_b \text{ in } [y_b = A]
\overline{b2}\{y'_b, N_B\} +_{k[A]} b3(z_b). \text{ case } z_b
of \{u_b\} - {k}[B] \text{ in } [u_b = N_B] 0\]

\[SYS^{NSPK} \triangleq A \parallel B\]
Environment Ability

- Environment memorizes all communicated messages.
- Environment can produce, encrypt/decrypt, compose/split, and hash messages, following Dolev-Yao model (described as an environmental deductive system $\vdash$).
- Environment produces infinitely many messages based on a bounded number of messages.
Specifications

- **Trace semantics** is chosen for the calculus. Each possible run of a protocol can be represented explicitly by a **concrete trace**.

- Processes for a given property will be inserted into a description of a protocol, to represent the security property.

- Security properties will be defined as reachability problems on a set of traces.

\[
\begin{aligned}
A \rightarrow B : & \quad \{A, N_A\} + K_B \\
B \rightarrow A : & \quad \{N_A, N_B\} + K_A \\
A \rightarrow B : & \quad \{N_B\} + K_B
\end{aligned}
\]

**Authentication as reachability**

\[
a3 \ x \leftarrow \overline{\text{acc}} \ x
\]

\[
B \triangleq (\nu \ N_B) \ b1(x_b).
\]

\[
\text{case } x_b \text{ of } \{x'_b\} - k_B \text{ in } \]

\[
\text{let } (y_b, y'_b) = x'_b \text{ in } [y_b = A]
\]

\[
\overline{b2}\{y'_b, N_B\} + k_A \cdot b3(z_b) \cdot \text{case } z_b \]

\[
\text{of } \{u_b\} - k_B \text{ in } [u_b = N_B] \overline{\text{acc}} z_b.0
\]
1 Introduction

2 Modeling Protocols for Authentication and Secrecy

3 Authentication and Secrecy in Bounded Sessions
   - Parametrization and Refinement
   - Experimental Results

4 Authentication for Recursive Protocols

5 Non-repudiation and Fairness in Bounded Sessions

6 Conclusion
Parametrization

- A receiver may receive an unbounded number of messages.
- A sender may send messages to an unbounded number of principals.
- Traces are abstracted to \textit{parametric traces} by keeping fresh variables un-instantiated.

\[
\langle \epsilon, a(x).0 \rangle \\
\langle a(M_i), 0 \rangle \cdot \cdot \cdot \langle a((M_i, M_i)), 0 \rangle \cdot \cdot \cdot \langle a([M_i]_+ k[I]), 0 \rangle
\]

\[
\langle \epsilon, (\textbf{new } x : I) b_I\{M\} + k[x].0 \rangle
\]

\[
\langle b_I\{M\} + k[B], 0 \rangle \cdot \cdot \cdot \langle b_I\{M\} + k[C], 0 \rangle \cdot \cdot \cdot \langle b_I\{M\} + k[I], 0 \rangle
\]

\[
\langle \epsilon, (\textbf{new } x : I) b_I\{M\} + k[x].0 \rangle
\]

\[
\langle b_I\{M\} + k[x], 0 \rangle
\]
Refinement Step

- Unifiers are applied when generating new parametric traces.
  \[
  \langle \epsilon, a_1(x).\text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \rightarrow_p \\
  \langle a_1(x), \text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \rightarrow_p \langle a_1(\{y\}_{k[A,S]}), P\{\{y\}_{k[A,S]} / x\} \rangle
  \]

- An input action can be regarded as a required pattern. Some are easily satisfied (e.g. \(a_1(x)\)), some are not (e.g. \(a_1(\{y\}_{k[A,S]})\)).

- A required pattern like \(a_1(\{y\}_{k[A,S]})\) is named a rigid message. It should be satisfied by messages in the prefix trace (by unifications).

\[
\cdots \rightarrow \tilde{s} \{A, M\}_{k[A,S]} \rightarrow \cdots \rightarrow a(\{A, x\}_{k[A,S]})
\]
Refinement Step

- Unifiers are applied when generating new parametric traces.
  \[
  \langle \epsilon, a_1(x) \text{ . } \text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a_1(x), \text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a_1(\{y\}_{k[A,S]}), P\{\{y\}_{k[A,S]} / x\} \rangle
  \]

- An input action can be regarded as a required pattern. Some are easily satisfied (e.g. \(a_1(x)\)), some are not (e.g. \(a_1(\{y\}_{k[A,S]})\)).

- A required pattern like \(a_1(\{y\}_{k[A,S]})\) is named a rigid message. It should be satisfied by messages in the prefix trace (by unifications).

\[
\text{Unification}
\]

\[
\begin{align*}
\text{x} & \mapsto M \\
\cdots & \mapsto \tilde{s} \{ A, M \}_{k[A,S]} \\
\cdots & \mapsto a(\{ A, x \}_{k[A,S]})
\end{align*}
\]
Refinement Step

- Unifiers are applied when generating new parametric traces.
  \[ \langle \epsilon, a_1(x).\text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a_1(x), \text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a_1(\{y\}_{k[A,S]}), P\{\{y\}_{k[A,S]} / x\} \rangle \]

- An input action can be regarded as a required pattern. Some are easily satisfied (e.g. \(a_1(x)\)), some are not (e.g. \(a_1(\{y\}_{k[A,S]})\)).

- A required pattern like \(a_1(\{y\}_{k[A,S]})\) is named a rigid message. It should be satisfied by messages in the prefix trace (by unifications).
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

\[ \cdots \rightarrow s1\{A, M\}_{k[A,S]} \rightarrow \cdots \rightarrow a2(\{A, x\}_{k[A,S]}) \rightarrow \cdots \]
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

\[ s_1 \{ A, M \}_{k[A,S]} \rightarrow \cdots \rightarrow a_2(\{ A, x \}_{k[A,S]}) \rightarrow \cdots \]
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

\[ \vdots \rightarrow s_1\{A, M\}_{k[A, S]} \rightarrow \cdots \rightarrow a_2(\{A, x\}_{k[A, S]}) \rightarrow \cdots \]

Unification

\[ \theta \]

\[ \vdots \rightarrow s_1\{A, M\}_{k[A, S]} \rightarrow \cdots \theta \rightarrow a_2(\{A, M\}_{k[A, S]}) \rightarrow \cdots \theta \]
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

\[ \cdots \rightarrow \overline{s_1} \{ A, M \}_{k[A,S]} \rightarrow \cdots \rightarrow a_2(\{ A, x \}_{k[A,S]}) \rightarrow \cdots \]

\[ \cdots \rightarrow \overline{s_1} \{ A, M \}_{k[A,S]} \rightarrow \cdots \theta \rightarrow a_2(\{ A, M \}_{k[A,S]}) \rightarrow \{ M(x) \}_{k[N]} \rightarrow \cdots \theta \]
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

\[ \ldots \rightarrow s_1 \{A, M\}_{k[A,S]} \rightarrow \ldots \rightarrow a_2(\{A, x\}_{k[A,S]}) \rightarrow \ldots \]

\[ \ldots \rightarrow s_1 \{A, M\}_{k[A,S]} \rightarrow \ldots \rightarrow a_2(\{A, M\}_{k[A,S]}) \rightarrow \{M(x)\}_{k[N]} \rightarrow \ldots \]
Satisfiable Normal Form

Given a parametric trace with bounded length, the refinement step will terminate and the set of normal forms of a parametric trace is finite.

Unification

\[ \cdots \rightarrow \overline{s}T\{A, M\}_{k[A, S]} \rightarrow \cdots \rightarrow a2(\{A, x\}_{k[A, S]}) \rightarrow \cdots \]

Unification

\[ \cdots \rightarrow \overline{s}T\{A, M\}_{k[A, S]} \rightarrow \cdots \theta \rightarrow a2(\{A, M\}_{k[A, S]}) \rightarrow \{M(x)\}_{k[N]} \rightarrow \cdots \theta \]

Unification

Normal form

The set of satisfiable normal forms (each rigid message is satisfiable) is sound and complete representatives to the set concrete traces.
Experimental Results

<table>
<thead>
<tr>
<th>protocols</th>
<th>session</th>
<th>protocol spec.</th>
<th>states</th>
<th>times(s)</th>
<th>flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSPK protocol</td>
<td>1</td>
<td>20</td>
<td>46</td>
<td>0.13</td>
<td>detected</td>
</tr>
<tr>
<td>Woo-Lam protocol*</td>
<td>1</td>
<td>25</td>
<td>168</td>
<td>0.16</td>
<td>detected</td>
</tr>
<tr>
<td>fixed NSPK protocol</td>
<td>1</td>
<td>20</td>
<td>164</td>
<td>6.37</td>
<td>secure</td>
</tr>
<tr>
<td>fixed NSPK protocol ‡</td>
<td>2</td>
<td>29</td>
<td>16,468</td>
<td>243.4</td>
<td>secure</td>
</tr>
<tr>
<td>Abadi-Gordon protocol</td>
<td>1</td>
<td>20</td>
<td>238</td>
<td>0.71</td>
<td>secure</td>
</tr>
<tr>
<td>Abadi-Gordon protocol †</td>
<td>2</td>
<td>30</td>
<td>4,802</td>
<td>30.5</td>
<td>secure</td>
</tr>
<tr>
<td>Yahalom protocol</td>
<td>1</td>
<td>26</td>
<td>279</td>
<td>2.11</td>
<td>secure</td>
</tr>
<tr>
<td>Yahalom protocol ‡</td>
<td>2</td>
<td>36</td>
<td>536</td>
<td>1.04</td>
<td>detected</td>
</tr>
<tr>
<td>Otway-Rees protocol</td>
<td>1</td>
<td>25</td>
<td>461</td>
<td>8.19</td>
<td>secure</td>
</tr>
<tr>
<td>Otway-Rees protocol ‡</td>
<td>2</td>
<td>34</td>
<td>2,164</td>
<td>22.3</td>
<td>detected</td>
</tr>
<tr>
<td>Woo-lam protocol</td>
<td>1</td>
<td>25</td>
<td>552</td>
<td>2.46</td>
<td>secure</td>
</tr>
<tr>
<td>Woo-lam protocol †</td>
<td>2</td>
<td>51</td>
<td>105,423</td>
<td>476.5</td>
<td>detected</td>
</tr>
</tbody>
</table>

- The tests were performed on a Pentium M 1.4 GHz, 1.5 GB memory PC, under Windows XP.
- The common part is about 330 lines (for structures and functions).
- A two-session protocol in which each principal acts as the same role is labeled by †, and as different roles is labeled by ‡.
- * represents a variation of the Woo-Lam protocol.
1. Introduction

2. Modeling Protocols for Authentication and Secrecy

3. Authentication and Secrecy in Bounded Sessions

4. Authentication for Recursive Protocols
   - Recursive Protocols
   - Abstractions in the pushdown system
   - Experimental Results

5. Non-repudiation and Fairness in Bounded Sessions

6. Conclusion
Recursive Protocols

- operate over an arbitrarily long chain of protocol principals, terminating with a key-generated server.
- intend to generate session-keys between each pair of two adjacent principals by contacting the key-generated server once.
- each principal has two choices: either contacts the server to terminate the protocol, or communicates with its next principal to continue the protocol.
Recursive Authentication Protocol [Bull&Otway 97]

\[ \mathcal{H}_K(X) = (\mathcal{H}(K, X), X) \]

- \( A_i \rightarrow A_{i+1} : \mathcal{H}_{K_{A_i}}(A_i, A_{i+1}, N_{A_i}, X) \)
- \( A_0 \rightarrow A_1 : \mathcal{H}_{K_{A_0}}(A_0, A_1, N_{A_0}, \text{Null}) \)
- \( A_1 \rightarrow A_2 : \mathcal{H}_{K_{A_1}}(A_1, A_2, N_{A_1}, \mathcal{H}_{K_{A_0}}(A_0, A_1, N_{A_0}, \text{Null})) \)
- \( \ldots \)

- \( A_n \rightarrow S : \mathcal{H}_{K_{A_n}}(A_n, S, N_{A_n}, \mathcal{H}_{K_{A_{n-1}}}(... \text{Null} \ldots)) \)

- \( S \rightarrow A_n : \{K_n, S, N_{A_n}\}_{K_{A_n}} \cdot \{K_{n-1}, A_{n-1}, N_{A_n}\}_{K_{A_n}} \cdot \{K_{n-1}, A_n, N_{A_{n-1}}\}_{K_{A_{n-1}}} \cdot \{K_{n-2}, A_{n-2}, N_{A_{n-1}}\}_{K_{A_{n-1}}} \cdot \ldots \)
- \( \{K_1, A_2, N_{A_1}\}_{K_{A_1}} \cdot \{K_0, A_0, N_{A_1}\}_{K_{A_1}} \)
- \( \{K_0, A_1, N_0\}_{K_{A_0}} \)

- \( A_i \rightarrow A_{i-1} : \{K_{i-1}, A_i, N_{i-1}\}_{K_{A_{i-1}}} \cdot \{K_{i-2}, A_{i-2}, N_{i-1}\}_{K_{A_{i-1}}} \cdot \ldots \)

- \( A_1 \rightarrow A_0 : \{K_0, A_1, N_0\}_{K_{A_0}} \)
Abstracting Unbounded Number of Messages
A Usage of Binders

In bounded sessions, messages that principals generated are explicitly represented by distinguished symbols.

- Principals’ names, \( A, B, C, I, \ldots \)
- Nonces, \( N_A, N_B, \ldots \)

For a recursive protocol, unbounded number of messages are encoded by nested applications of binders.

- Names: \( A[Null], A[A[Null]], \ldots \)
- Nonces: \( N[Null], N[N[Null]], \ldots \)
A pushdown system $\mathcal{P} = (Q, \Gamma, \Delta, c_0)$ is a transition system with carrying an unbounded stack.

- $Q$ is a set of control locations. A control location is a pair $(R, \hat{t}r)$, in which $R$ is a finite set of messages, and $\hat{t}r$ is a finite parametric trace.

- $\Gamma$ is a finite set of stack alphabet.
  - $\Gamma = \{\star\}$.
  - Nested applications of binders are encoded with binder markers, and nested times are captured by the length of the stack. (e.g. $A[Null], A[A[Null]], \ldots$ will be encoded by $\langle \ldots A[\ldots, \star\rangle$.)

- $c_0 = (q_0, \omega_0)$, is an initial configuration, where $q_0 \in Q$ and $\omega_0 \in \Gamma^*$. $\langle (\emptyset, \epsilon), \epsilon \rangle$.

- $\Delta : (Q \times \Gamma) \times (Q \times \Gamma^*)$ is a finite subset of transitions ($\hookrightarrow$). It encodes the parametric transitions and the refinement step.
Recall the Parametric Transitions

- Traces are abstracted to **parametric traces** by keeping fresh variables un-instantiated.

\[
\langle \epsilon, a(x).0 \rangle \xrightarrow{\epsilon, (\text{new } x : I)\overline{\text{b}}\{M\}_+k[x].0 \rangle
\]

- Unifiers are applied when generating new parametric traces.

\[
\langle \epsilon, a1(x).\text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a1(x), \text{case } x \text{ of } \{y\}_{k[A,S]} \text{ in } P \rangle \xrightarrow{p} \langle a1(\{y\}_{k[A,S]}), P\{\{y\}_{k[A,S]}/x\} \rangle
\]

- Unbounded length of parametric traces should be represented.
Nonces are encoded by $N[n], N[N[n]], \ldots$

$A_0$ is defined as $A \triangleq a_1 N[n].0$

An identifier for receivers is defined as: $R \triangleq b_1(x).b_2 N[x].R$

The finite parametric trace in a control location is a compaction of the original trace.

- $\langle (\emptyset, \epsilon), \epsilon \rangle \leftrightarrow \langle (\emptyset, a_1 N[n]), \epsilon \rangle$
- $\langle (\emptyset, a_1 N[n]), \epsilon \rangle \leftrightarrow \langle (\emptyset, a_1 N[n].b_1(x).b_2 N[N[[]]], \epsilon \rangle$
- $\langle (\emptyset, \epsilon), \epsilon \rangle \leftrightarrow \langle (\emptyset, b_1(x).b_2 N[N[[]]], \epsilon \rangle$
- $\langle (\emptyset, b_1(x).b_2 N[N[[]]], \epsilon \rangle \leftrightarrow \langle (\emptyset, b_1(x).b_2 N[N[[]]], \star \rangle$
- $\langle (\emptyset, a_1 N[n].b_1(x).b_2 N[N[[]]], \epsilon \rangle \leftrightarrow \langle (\emptyset, a_1 N[n].b_1(x).b_2 N[N[[]]], \star \rangle$
- $\ldots$
Recall the Refinement Step

- An input action can be regarded as a required pattern. Some are easily satisfied (e.g. \( a^1(x) \)), some are not (e.g. \( a^1(\{y\}_{k[A,S]}) \)).

- A required pattern like \( a^1(\{y\}_{k[A,S]}) \) is named a rigid message. It needs to be satisfied by messages in the prefix trace (by unifications).

Unbounded number of rigid messages should be unified.

The set of messages to be unified may also be unbounded.
Context-insensitive Rigid Messages

A is defined as $\mathbb{A}(N)$, where
\[\mathbb{A}(x) \triangleq \overline{a^1} \{A, x\}_{k[A, B]} \cdot \mathbb{A}(N[x])\]

B is defined as
\[B \triangleq b1(x).\text{case } x \text{ of } \{y\}_{k[A, B]} \text{ in let } (z, w) = y \text{ in } [z = A] 0\]

A rigid message $\{A, w\}_{k[A, B]}$ is insensitive to the current context.

A rigid message is context-insensitive if it does not contain any binder markers.

\[
\begin{align*}
\langle (\emptyset, \epsilon), \epsilon \rangle & \leftrightarrow \langle (\emptyset, \overline{a^1} \{A, N[\ ]\}_{k[A, B]}), \epsilon \rangle \\
\langle (\emptyset, \overline{a^1} \{A, N[\ ]\}_{k[A, B]}), \epsilon \rangle & \leftrightarrow \langle (\emptyset, \overline{a^1} \{A, N[\ ]\}_{k[A, B]}), \ast \rangle \\
\ldots \\
\langle (\emptyset, \overline{a^1} \{A, N[\ ]\}_{k[A, B]}), b1(\{A, w\}_{k[A, B]}), \epsilon \rangle & \leftrightarrow \\
\langle \langle\{\{A, N[\ ]\}_{k[A, B]}\}, a^1 \{A, \top\}_{k[A, B]} \rangle \cdot b1(\{A, \top\}_{k[A, B]}), \epsilon \rangle
\end{align*}
\]
Context-sensitive Rigid Messages

- A is defined as \( A(N) \), where
  \[
  A(x) \triangleq \overline{a_1} \cdot x \cdot a_2(y).
  \]
- \[ \text{case } y \text{ of } \{ z \}_{k[A,B]} \text{ of } \]
  \[ \text{let } (z_1, z_2) = z \text{ in } \]
  \[ [z_2 = x] A(N[x]) \]

- A rigid message \( \{ z_1, N[\_] \}_{k[A,B]} \) is sensitive to the current context.
- A rigid message is context-sensitive if it contains at least a binder marker.
- A context-sensitive rigid message can only be unified with bounded number of messages (in current context).

\[
\langle (\emptyset, \overline{a_1} N[\_]).b_1(x).b_2 \{ B, x \}_{k[A,B]} \cdot a_2(\{ z_1, N[\_] \}_{k[A,B]}), \varepsilon \rangle \leftrightarrow \langle (\emptyset, \overline{a_1} N[\_]).b_1(N[\_]).b_2 \{ B, N[\_] \}_{k[A,B]} \cdot a_2(\{ B, N[\_] \}_{k[A,B]}), \varepsilon \rangle
\]
Freshness of Sessions and Principals

- Freshness of sessions is captured by fresh nonces; freshness of principals is captured by fresh names of principals.

- Generally, a stack can only represent one type of fresh messages by the context.
  - When push, enters a new context (new session).
  - When pop, returns to the previous context (previous session).

- In recursive protocols, two type of fresh messages coincide with each other. Thus by one stack, two type of fresh messages can be described.
An Attack for the RA Protocol

- There are two different views of the authentication property.
- L. Paulson took one view of authentication. He proved the correctness of the RA protocol by Isabelle/HOL.
- An attack, which violates authentication (a la Abadi et. al.) is found in the RA protocol by our method.

This attack does not violate secrecy.
Corrected Version of the RA Protocol

\[ S \rightarrow A_n : \]
\[
\{ K_n, S, N_{A_n} \} K_{A_n S}, \{ K_{n-1}, A_{n-1}, N_{A_n} \} K_{A_n S}, \\
\{ K_{n-1}, A_n, N_{A_n-1} \} K_{A_{n-1} S}, \{ K_{n-2}, A_{n-2}, N_{A_n-1} \} K_{A_{n-1} S}, \\
\ldots \\
\{ K_1, A_2, N_{A_1} \} K_{A_1 S}, \{ K_0, A_0, N_{A_1} \} K_{A_1 S}, \\
\{ K_0, A_1, N_{A_0} \} K_{A_0 S} 
\]
Experimental Results

<table>
<thead>
<tr>
<th>protocols</th>
<th>protocol spec.</th>
<th>states</th>
<th>times(s)</th>
<th>flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>recursive authentication protocol</td>
<td>32</td>
<td>416</td>
<td>0.82</td>
<td>detected</td>
</tr>
<tr>
<td>fixed recursive authentication protocol</td>
<td>32</td>
<td>416</td>
<td>1.07</td>
<td>secure</td>
</tr>
</tbody>
</table>

- The tests were performed on a Pentium M 1.4 GHz, 1.5 GB memory PC, under Windows XP.
- The common part is about 500 lines.
1. Introduction

2. Modeling Protocols for Authentication and Secrecy

3. Authentication and Secrecy in Bounded Sessions

4. Authentication for Recursive Protocols

5. Non-repudiation and Fairness in Bounded Sessions
   - Extended with Dishonest Principals
   - Two-Phase Refinement Steps
   - Experimental Results

6. Conclusion
Honest Principal VS. Dishonest Principal

\[ A \rightarrow B : N_A \]
\[ B \rightarrow A : \{B, N_A\}_{K_{BS}} \]

Each principal follows the rules of a protocol.

Each principal disobeys the rules of a protocol.
Honest Principal VS. Dishonest Principal

Each principal follows the rules of a protocol.

- \( A \rightarrow B : N_A \)
- \( B \rightarrow A : \{B, N_A\}_{K_{BS}} \)

\( N_A \)

\( \{B, N_A\}_{K_{BS}} \)
Honest Principal VS. Dishonest Principal

- Each principal follows the rules of a protocol.
- Each principal disobeys the rules of a protocol.

A → B : $N_A$
B → A : $\{B, N_A\}_{K_{BS}}$
Honest Principal VS. Dishonest Principal

\[ A \rightarrow B : N_A \]
\[ B \rightarrow A : \{B, N_A\}_{K_{BS}} \]

- Each principal follows the rules of a protocol.
- Each principal disobeys the rules of a protocol.
Honest Principal VS. Dishonest Principal

Each principal follows the rules of a protocol.

Each principal disobeys the rules of a protocol.
Honest Principal VS. Dishonest Principal

Each principal follows the rules of a protocol.

Each principal disobeys the rules of a protocol.

\[ A \rightarrow B : N_A \]
\[ B \rightarrow A : \{ B, N_A \}_{K_{BS}} \]
Model a Network
Dishonest Principals

- Environment can memorize, produce, encrypt/decrypt, and compose/split messages (described as an environmental deductive system $\vdash$).

- A dishonest principal may send unbounded number of messages during the communication (represented as $P$-deductive system $\vdash_P$).

- A dishonest principal $P$ can generate each message the environment generates ($s \vdash M \Rightarrow s \vdash_P M$).

- A dishonest principal $P$ can sign(encrypt with its private key), or encrypt with shared key a message it generates. ($s \vdash_P M \Rightarrow s \vdash_P \{M\}_{K_P}$, $s \vdash_P M \Rightarrow s \vdash_P \{M\}_{K_{PS}}$)
New Infinity Factor and its Parametrization

- A sender may send infinitely many messages to the environment due to the $P$-deductive system.

- Traces are abstracted to parametric traces by keeping fresh variables un-instantiated.

\[ \langle \varepsilon, (\text{new } x) a_1\{x\} \rangle \rightarrow_{k[A]} 0 \]

\[ \langle a_1\{M\} \rangle \rightarrow_{k[A]} 0 \]

\[ \langle a_1\{M, B\} \rangle \rightarrow_{k[A]} 0 \]

\[ \langle a_1\{\{M\} + k[B]\} \rangle \rightarrow_{k[A]} 0 \]

\[ \langle a_1\{x\} \rangle \rightarrow_{k[A]} 0 \]
Recall the Refinement Step

- An input action can be regarded as a required pattern. Some are easily satisfied (e.g. \(a^1(x)\)), some are not (e.g. \(a^1(\{y\}_{k[A,S]})\)).

- A required pattern like \(a^1(\{y\}_{k[A,S]})\) is named a rigid message. It needs to be satisfied by messages in the prefix trace (by unifications).

\[
\begin{align*}
\cdots & \rightarrow \overline{s} \{A, M\}_{k[A,S]} \rightarrow \cdots \rightarrow a(\{A, x\}_{k[A,S]}) \\
\cdots & \rightarrow \overline{s} \{A, M\}_{k[A,S]} \rightarrow \cdots \rightarrow a(\{A, M\}_{k[A,S]})
\end{align*}
\]
Further Refinement Step

- A refinement step may cause an inconsistency when a principal sends messages.

- A parametric trace is $\overline{a_1}x.a_2\{y\}_{-k[B]}$. ($\{y\}_{-k[B]}$: rigid message).

- After unification, $\overline{a_1}\{y\}_{-k[B]}a_2\{y\}_{-k[B]}$. Can principal $A$ sign with $B$’s signature? (Need further refinement!)

$$\ldots \rightarrow \overline{b_1}\{B, y\}_{-k[B]} \rightarrow \ldots \rightarrow \overline{a_1}z \rightarrow a_2(\{B, x\}_{-k[B]})$$
Further Refinement Step

- A refinement step may cause an inconsistency when a principal sends messages.

- A parametric trace is $\overline{a_1 x}.a_2 \{y\}_{-k[B]}$. ($\{y\}_{-k[B]}$: rigid message).

- After unification, $\overline{a_1 \{y\}}_{-k[B]} . a_2 \{y\}_{-k[B]}$. Can principal $A$ sign with $B$’s signature? (Need further refinement!)

\[ \begin{align*}
A & \rightarrow B : \quad N_A \\
B & \rightarrow A : \quad \{N_B\}_{-k_B}
\end{align*} \]
Further Refinement Step

- A refinement step may cause an inconsistency when a principal sends messages.

- A parametric trace is $\overline{a_1} x . a_2 \{y\} - k_B$. ($\{y\} - k_B$: rigid message).

- After unification, $\overline{a_1} \{y\} - k_B . a_2 \{y\} - k_B$. Can principal $A$ sign with $B$’s signature? (Need further refinement!)
Further Refinement Step

- A refinement step may cause an inconsistency when a principal sends messages.

- A parametric trace is $\overline{a_1} x . a_2 \{y\}_{-k[B]}$. ($\{y\}_{-k[B]}$: rigid message).

- After unification, $\overline{a_1} \{y\}_{-k[B]} . a_2 \{y\}_{-k[B]}$. Can principal $A$ sign with $B$’s signature? (Need further refinement!)

\[
A \rightarrow B : N_A \\
B \rightarrow A : \{N_B\}_{-k_B}
\]
Further Refinement Step

- A refinement step may cause an inconsistency when a principal sends messages.

- A parametric trace is $\overline{a_1}x.a_2\{y\}_{-k[B]}$. ($\{y\}_{-k[B]}$: rigid message).

- After unification, $\overline{a_1}\{y\}_{-k[B]}.a_2\{y\}_{-k[B]}$. Can principal A sign with B’s signature? (Need further refinement!)

Unification

\[ ... \rightarrow \overline{b_1}\{B,y\}_{-k[B]} \rightarrow ... \rightarrow \overline{a_1}z \rightarrow a_2(\{B,x\}_{-k[B]}) \]

Unification

\[ ... \rightarrow \overline{b_1}\{B,x\}_{-k[B]} \rightarrow ... \rightarrow \overline{a_1}\{B,y\}_{-k[B]} \rightarrow a_2(\{B,y\}_{-k[B]}) \]

Unification

\[ ... \rightarrow \overline{b_1}\{B,x\}_{-k[B]} \rightarrow ... \rightarrow \overline{a_1}\{B,x\}_{-k[B]} \rightarrow a_2(\{B,x\}_{-k[B]}) \]
Satisfiable Normal Form

The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.

\[ \cdots \to b1\{B, y\}_{-k[B]} \to \cdots \to a1z \to a2(\{B, x\}_{-k[B]}) \]
Satisfiable Normal Form

The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.

\[ ... \rightarrow \overline{b1}\{B, y\}_{k[B]} \rightarrow \cdots \rightarrow \overline{a1}z \rightarrow a2(\{B, x\}_{k[B]}) \]

Unification
Satisfiable Normal Form

The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.

\[
\cdots \rightarrow b_1 \{B, y\}_{-k[B]} \rightarrow \cdots \rightarrow a_1 z \rightarrow a_2 (\{B, x\}_{-k[B]})
\]

Unification
Satisfiable Normal Form

The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.
Satisfiable Normal Form

The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.
The refinement steps will also terminate and the set of normal forms of a parametric trace is finite.

The set of satisfiable normal forms (each rigid message is satisfiable) is sound and complete representatives to concrete traces.
### Experimental Results

- **Parametric system are implemented by Maude.**

<table>
<thead>
<tr>
<th>protocols</th>
<th>property</th>
<th>protocol spec.</th>
<th>states</th>
<th>times(s)</th>
<th>flaws</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified ZG protocol</td>
<td>NRO</td>
<td>50</td>
<td>513</td>
<td>3.95</td>
<td>detected</td>
</tr>
<tr>
<td></td>
<td>NRR</td>
<td>50</td>
<td>780</td>
<td>3.90</td>
<td>secure</td>
</tr>
<tr>
<td></td>
<td>FAIRO</td>
<td>55</td>
<td>770</td>
<td>2.96</td>
<td>detected</td>
</tr>
<tr>
<td></td>
<td>FAIRR</td>
<td>55</td>
<td>846</td>
<td>3.90</td>
<td>secure</td>
</tr>
<tr>
<td></td>
<td>FAIRM</td>
<td>50</td>
<td>4,109</td>
<td>45.5</td>
<td>detected</td>
</tr>
<tr>
<td>Full ZG protocol</td>
<td>NRO</td>
<td>50</td>
<td>633</td>
<td>7.40</td>
<td>secure</td>
</tr>
<tr>
<td></td>
<td>FAIRO</td>
<td>55</td>
<td>788</td>
<td>3.39</td>
<td>secure</td>
</tr>
<tr>
<td></td>
<td>FAIRM</td>
<td>60</td>
<td>788</td>
<td>3.490</td>
<td>secure</td>
</tr>
<tr>
<td>ISO/IEC13888-2 M2</td>
<td>NRO</td>
<td>50</td>
<td>1,350</td>
<td>7.71</td>
<td>detected</td>
</tr>
<tr>
<td></td>
<td>FAIRO</td>
<td>65</td>
<td>1,977</td>
<td>6.83</td>
<td>detected</td>
</tr>
<tr>
<td></td>
<td>FAIRR</td>
<td>65</td>
<td>2,131</td>
<td>7.51</td>
<td>secure</td>
</tr>
<tr>
<td>ISO/IEC13888-3 M-h</td>
<td>FAIRO</td>
<td>60</td>
<td>295</td>
<td>0.92</td>
<td>detected</td>
</tr>
<tr>
<td></td>
<td>FAIRR</td>
<td>60</td>
<td>305</td>
<td>1.04</td>
<td>secure</td>
</tr>
</tbody>
</table>

(Non-repudiation is composed of NRO and NRR; Fairness is composed of FAIRO, FAIRR, FAIRM)

- **Pentium M 1.4 GHz, 1.5 GB memory PC, under Windows XP.**
- **The common part is about 400 lines.**
1 Introduction

2 Modeling Protocols for Authentication and Secrecy

3 Authentication and Secrecy in Bounded Sessions

4 Authentication for Recursive Protocols

5 Non-repudiation and Fairness in Bounded Sessions

6 Conclusion
Related Work

Process Calculi

- Bisimulation (M. Abadi et. al. 97, 98)
- Typed-based static analysis (M. Abadi et. al. 99, 01)
- Horn clause+ resolution (M. Abadi et. al. 03, 04, 05)
- Trace analysis
  - Theorem proving
    - CSP+PVS (S. Schneider et. al. 98-01)
  - Model checking
    - CSP+FDR G. Lowe et. al. 96, Spi+STA M. Boreale 01
David Basin, et al. proposed an On-the-fly model checking method (OFMC). (Basin et.al. 05)

They use a high-level language HLPSL to represent a protocol, then translate automatically to a low-level one, IF.

The knowledge of intruder will be refined to satisfy the requirement of a principal (Lazy intruder).

An intruder’s role is explicitly assigned, thus flexible and efficient.
Future Works

- More on the same direction:
  - Analyzing other properties.
    - anonymity, and other authentication variations.
    - security properties for multiparty protocols and broadcast protocols.
  - A translator that translates a formal protocol description to a Maude source file is designable.
  - Adopt bisimulation as specifications.

- Affiliate to the resolution method, used as a heuristic guide to extract a counterexample when finding a flaw by resolution.

- Analyze from source codes of security protocols.
Conclusions

- Proposed security protocol analysis based on a sound and complete model checking method to analyze various security properties under certain assumptions.

- When flaws are not detected, it guarantees that a protocol is secure under the given assumptions.

- Several properties, such as non-repudiation, and authentication for recursive protocols, are first analyzed by us using model checking.
Thank you!
Sub-calculi

\( P, Q ::= \)

0
\( aM.P \)
\( [M = N] P \)
\( (\text{new } x : A)P \)
\( (\nu n)P \)
let \((x, y) = M\) in \(P\)
case \(M\) of \(\{x\}_L\) in \(P\)
\(P \parallel Q\)
\(P + Q\)
\(P ; Q\)
\(A(\tilde{r})\)}
Difference of the New Operators

- In the sub-calculi for authentication and secrecy properties, the ranges of variables bounded by New operators are given statically: \((\text{new} \ x : A)P\)

- In the sub-calculus for non-repudiation and fairness, the ranges of variables bounded by New operators are given dynamically by the \(P\)-deductive system: \((\text{new} \ x)P\)

- The main difference, a process in the former calculi has a **prenex normal form**, while in the latter one has not.

\[
(\text{new} \ x : A)a_1 \ x .(\text{new} \ y : B)a_2 \ y .0 \equiv (\text{new} \ x : A)(\text{new} \ y : B)a_1 \ x .a_2 \ y .0
\]

\[
(\text{new} \ x)a_1 \ x .(\text{new} \ y)a_2 \ y .0 \not\equiv (\text{new} \ x)(\text{new} \ y)a_1 \ x .a_2 \ y .0
\]
Restricting Infinite Processes

- Adopt **identifier** to represent infinite processes. For any identifier $\Delta(x_1, \ldots, x_n)$, there must be a unique definition, that is, $\Delta(x_1, \ldots, x_n) \triangleq P$.

- A **process expression** $E$ is like a process, but may contain **identifier variables**. (eg: $E(X) \triangleq a(x).X$)

- A recursive process is defined as an identifier. (eg: $\Delta \triangleq E(\Delta)$)

- For recursive protocols, there are two restrictions:
  - A system allows only **one** recursive process.
  - The $X$ in $E$ for the recursive process is **sequential**. ($X$ is sequential in $E$ if $X$ does not occur with composition combinators in $E$)
Sequence Primitive

- Sequence primitive, $P; Q$ are usually a redundant operator in process calculus.
  - $P; Q$ has sub-expressiveness of $P||Q$.
  - If there is no identifiers, $P; Q$ is equal to replace all $0$ in $P$ to $Q$.
    (eg: $(\alpha.0 + \beta.0); Q \equiv \alpha.Q + \beta.Q$)

- Sequence operator is required, since the sequential restriction is adopted to identifier variables in expression $E$. while $P; Q$ is needed when defining a recursive protocol.

- Without $P; Q$, when entering the another context, there are no actions in previous context.
Representing the RA Protocol: originator

\[ \otimes(x_1, x_2) \overset{a_1}{=} H_{\bot k[x_1, s]}(x_1, x_2, N[Null], Null). \]

\[ a_2(x).case \ x \ of \ \{ y_1, y_2, y_3 \}_{\bot k[x_1, s]} \cdot \]

\[ [y_3 = N[Null]] 0 \]

\[ \otimes(A[Null], A[A[Null]]) \]
Representing the RA Protocol: recipient

$\mathbb{R}(x_1, x_2) \triangleq (b1(x).let (y_1, y_2, y_3, y_4, y_5) = x in [y_2 = x_1] \cdot$

$(b2 \mathcal{H}_{lk[x_1,s]}(x_1, A[x_1], N[y_3], x).\mathbb{R}(A[x_1], x_1) +$ $b3 \mathcal{H}_{lk[x_1,s]}(x_1, S, N[y_3], x).0)) ;$

$(b4(x).let (z_1, z_2, z_3) = x in$ $\text{case } z_1 \text{ of } \{z_4, z_5, z_6\}_{lk[x_1,s]} \text{ in } [z_6 = N[y_3]]$ $\text{case } z_2 \text{ of } \{z_7, z_8, z_9\}_{lk[x_1,s]} \text{ in}$ $[z_8 = x_2] [z_9 = N[y_3]] b5z_3.0)$

Representing the RA Protocol: server

\[ S \triangleq s1(x).s2(F(x)).0 \]

\[ F(x) = \begin{align*}
&\text{let } (y_1, y_2, y_3, y_4, y_5) = x; \\
&\text{let } t = \epsilon; \\
&\text{while } (y_1 = \mathcal{H}(y_2, y_3, y_4, y_5, \text{lk}[y_2, S]) && y_5 != \text{Null}) \\
&\quad \text{let } (z_1, z_2, z_3, z_4, z_5) = y_5; \\
&\quad \text{if } (z_1 = \mathcal{H}(z_2, z_3, z_4, z_5, \text{lk}[z_2, S]) && z_3 == y_2) \\
&\quad \quad \text{then } t = (t, \{k[y_4], y_3, y_4\}, \{k[y_3], z_2, z_4\}); \\
&\quad \text{else } \text{raise error} \\
&\quad \text{endif} \\
&\quad (y_1, y_2, y_3, y_4, y_5) := (z_1, z_2, z_3, z_4, z_5); \\
&\text{endwhile} \\
&t := (t, \{k[y_4], y_3, y_4\}); \\
&\text{return } t;
\]

\[ SYS^{RA} \triangleq O(A[\text{Null}], A[A[\text{Null}]])) || R(A[A[\text{Null}]], A[\text{Null}])) || S \]
Environmental Deductive System

\[
\begin{align*}
S \vdash M \quad & M \in E \\
S \vdash M \quad & M \in S \\
S \vdash M \quad S \vdash N \\
\hline
S \vdash (M, N) \\
S \vdash (M, N) \\
S \vdash M \\
S \vdash N \\
S \vdash \{M\}_k[A, B] \\
S \vdash k[A, B] \\
\hline
S \vdash M \\
S \vdash k[A, B] \\
S \vdash \{M\}_k[A, B] \\
S \vdash \{M\}_\pm k[A] \\
S \vdash \mp k[A] \\
S \vdash \{M\}_\pm k[A] \\
S \vdash \{M\}_\mp k[A] \\
S \vdash H(M)
\end{align*}
\]
### The Axioms of Structural Congruence

<table>
<thead>
<tr>
<th>Axiom</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong><code>SC-COMP-ASSOC</code></strong></td>
<td>$P \parallel (Q \parallel R) \equiv (P \parallel Q) \parallel R$</td>
</tr>
<tr>
<td><strong><code>SC-COMP-COMM</code></strong></td>
<td>$P \parallel Q \equiv Q \parallel P$</td>
</tr>
<tr>
<td><strong><code>SC-COMP-INACT</code></strong></td>
<td>$P \parallel 0 \equiv P$</td>
</tr>
<tr>
<td><strong><code>SC-SUM-ASSOC</code></strong></td>
<td>$P + (Q + R) \equiv (P + Q) + R$</td>
</tr>
<tr>
<td><strong><code>SC-SUM-COMM</code></strong></td>
<td>$P + Q \equiv Q + P$</td>
</tr>
<tr>
<td><strong><code>SC-RES</code></strong></td>
<td>$(\nu m)(\nu n)P \equiv (\nu n)(\nu m)P$</td>
</tr>
<tr>
<td><strong><code>SC-RES-INACT</code></strong></td>
<td>$(\nu n)0 \equiv 0$</td>
</tr>
<tr>
<td><strong><code>SC-RES-COMP</code></strong></td>
<td>$(\nu n)(P \parallel Q) \equiv P \parallel (\nu n)Q$ if $n \not\in f_n(P)$</td>
</tr>
<tr>
<td><strong><code>SC-NEW</code></strong></td>
<td>$(\text{new } x : \mathcal{A})(\text{new } y : \mathcal{B})P \equiv (\text{new } y : \mathcal{B})(\text{new } x : \mathcal{A})P$</td>
</tr>
<tr>
<td><strong><code>SC-NEW-INACT</code></strong></td>
<td>$(\text{new } x : \mathcal{A})0 \equiv 0$</td>
</tr>
<tr>
<td><strong><code>SC-NEW-COMP</code></strong></td>
<td>$(\text{new } x : \mathcal{A})(P \parallel Q) \equiv P \parallel (\text{new } x : \mathcal{A})Q$ if $x \not\in f_v(P)$</td>
</tr>
<tr>
<td><strong><code>SC-SEQ-INACT</code></strong></td>
<td>$0; P \equiv P$</td>
</tr>
</tbody>
</table>
Concrete Transition Rules in Bounded Sessions

<table>
<thead>
<tr>
<th>Rule</th>
<th>Transition</th>
</tr>
</thead>
<tbody>
<tr>
<td>(INPUT)</td>
<td>( \langle s, a(x).P \rangle \rightarrow \langle s.a(M), P{M/x} \rangle )</td>
</tr>
<tr>
<td>(OUTPUT)</td>
<td>( \langle s, \overline{a}M.P \rangle \rightarrow \langle s.\overline{a}M, P \rangle )</td>
</tr>
<tr>
<td>(DEC)</td>
<td>( \langle s, \text{case} \ M \text{ of } {x}_L \text{ in } P \rangle \rightarrow \langle s, P{M/x} \rangle )</td>
</tr>
<tr>
<td></td>
<td>( L' = \text{Opp}(L) )</td>
</tr>
<tr>
<td>(PAIR)</td>
<td>( \langle s, \text{let} (x, y) = (M, N) \text{ in } P \rangle \rightarrow \langle s, P{M/x, N/y} \rangle )</td>
</tr>
<tr>
<td>(NEW)</td>
<td>( \langle s, (\text{new } x : \mathcal{A})P \rangle \rightarrow \langle s, P{m/x} \rangle )</td>
</tr>
<tr>
<td></td>
<td>( m \in \mathcal{A} )</td>
</tr>
<tr>
<td>(RESTRICTION)</td>
<td>( \langle s, (\nu n)P \rangle \rightarrow \langle s, P{m/n} \rangle )</td>
</tr>
<tr>
<td></td>
<td>( m = \text{freshN}(V) )</td>
</tr>
<tr>
<td>(MATCH)</td>
<td>( \langle s, [M = M]P \rangle \rightarrow \langle s, P \rangle )</td>
</tr>
<tr>
<td></td>
<td>( \langle s, P \rangle \rightarrow \langle s', P' \rangle )</td>
</tr>
<tr>
<td>(LCOM)</td>
<td>( \langle s, P\parallel Q \rangle \rightarrow \langle s', P'\parallel Q \rangle )</td>
</tr>
<tr>
<td></td>
<td>( \langle s, Q \rangle \rightarrow \langle s', Q' \rangle )</td>
</tr>
<tr>
<td>(RCOM)</td>
<td>( \langle s, P\parallel Q \rangle \rightarrow \langle s', P\parallel Q' \rangle )</td>
</tr>
<tr>
<td></td>
<td>( P \equiv P' \rightarrow \langle s, P' \rangle \rightarrow \langle s', Q' \rangle )</td>
</tr>
<tr>
<td></td>
<td>( Q' \equiv Q )</td>
</tr>
<tr>
<td>(STR)</td>
<td>( \langle s, P \rangle \rightarrow \langle s', Q \rangle )</td>
</tr>
</tbody>
</table>
Parametric Transition Rules in Bounded Sessions

\[(PINPUT) \quad \langle \hat{s}, a(x).P \rangle \rightarrow_p \langle \hat{s}.a(x), P \rangle\]

\[(POUTPUT) \quad \langle \hat{s}, \overline{a}M.P \rangle \rightarrow_p \langle \hat{s}.\overline{a}M, P \rangle\]

\[(PDEC) \quad \langle \hat{s}, \text{case } \{M\}_L \text{ of } \{x\}_{L'} \text{ in } P \rangle \rightarrow_p \langle \hat{s}\theta, P\theta \rangle\]

\[\theta = \text{Mgu} \left( \{M\}_L, \{x\}_{\text{opp}(L')} \right)\]

\[(PPAIR) \quad \langle \hat{s}, \text{let } (x, y) = M \text{ in } P \rangle \rightarrow_p \langle \hat{s}\theta, P\theta \rangle\]

\[\theta = \text{Mgu} ((x, y), M)\]

\[(PNEW) \quad \langle \hat{s}, (\text{new } x : A)P \rangle \rightarrow_p \langle \hat{s}, P\{y/x\} \rangle\]

\[y \notin f_v(P) \cup b_v(P)\]

\[(PRESTRICTION) \quad \langle \hat{s}, (\nu n)P \rangle \rightarrow_p \langle \hat{s}, P\{m/n\} \rangle\]

\[m = \text{freshN}(V)\]

\[(PMATCH) \quad \langle \hat{s}, [M = M']P \rangle \rightarrow_p \langle \hat{s}\theta, P\theta \rangle\]

\[\theta = \text{Mgu}(M, M')\]

\[(PLCOM) \quad \langle \hat{s}, P \parallel Q \rangle \rightarrow_p \langle \hat{s}', P' \parallel Q' \rangle\]

\[Q' = Q\theta \text{ if } \hat{s}' = \hat{s}\theta \text{ else } Q' = Q\]

\[P \equiv P' \quad \langle s, P' \rangle \rightarrow_p \langle s', Q' \rangle\]

\[Q' \equiv Q\]

\[(PSTR) \quad \langle s, P \rangle \rightarrow_p \langle s', Q \rangle\]
Q&A: Defining Security Properties

Action Terms

\[ T ::= \alpha \mid \neg T \mid T \land T \mid T \lor T \]
\[ \sigma ::= T \mid T \leftrightarrow T \mid T \leftrightarrow_F T \]

- \( \langle s, P \rangle \models T_1 \leftrightarrow T_2 \): for each concrete trace \( s' \) generated by \( \langle s, P \rangle \), if there is a ground substitution \( \rho \) such that \( T_2\rho \) occurs in \( s' \), then so do \( T_1\rho \), and \( T_1\rho \) occurs before \( T_2\rho \) in \( s' \).

- \( \langle s, P \rangle \models T_1 \leftrightarrow_F T_2 \): for each concrete configuration \( \langle s', P' \rangle \) reached by \( \langle s, P \rangle \), if there is a ground substitution \( \rho \) such that \( T_1\rho \) occurs in \( s' \), then for every path starting from \( \langle s', P' \rangle \), there exists a concrete trace \( s'' \) such that \( T_2\rho \) occurs in \( s'' \).
The testing equivalence $P \simeq Q$ between $P$ and $Q$ is defined as follow: For all test $(R,\beta)$, $(P \parallel R) \downarrow \beta$ if and only if $(Q \parallel R) \downarrow \beta$. 

\[
\begin{align*}
    m(x).Q \downarrow m \\
    \frac{P \downarrow \beta}{P \downarrow \beta}
\end{align*}
\]

\[
\begin{align*}
    \overline{m}\langle M \rangle.Q \downarrow \overline{m} \\
    \frac{P \rightarrow Q Q \downarrow \beta}{P \downarrow \beta}
\end{align*}
\]
The Zhou-Gollmann fair non-repudiation protocol

<table>
<thead>
<tr>
<th>A $\rightarrow$ B</th>
<th>${B, N_A, {M}_K} - K_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B $\rightarrow$ A</td>
<td>${A, N_A, {M}_K} - K_B$</td>
</tr>
<tr>
<td>A $\rightarrow$ S</td>
<td>${B, N_A, K} - K_A$</td>
</tr>
<tr>
<td>S $\rightarrow$ A</td>
<td>${A, B, N_A, K} - K_S$</td>
</tr>
<tr>
<td>S $\rightarrow$ B</td>
<td>${A, B, N_A, K} - K_S$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A $\rightarrow$ B</th>
<th>${F_{NRO}, B, N_A, {M}_K} - K_A$</th>
</tr>
</thead>
<tbody>
<tr>
<td>B $\rightarrow$ A</td>
<td>${F_{NRR}, A, N_A, {M}_K} - K_B$</td>
</tr>
<tr>
<td>A $\rightarrow$ S</td>
<td>${F_{SUB}, B, N_A, K} - K_A$</td>
</tr>
<tr>
<td>S $\rightarrow$ A</td>
<td>${F_{CON}, A, B, N_A, K} - K_S$</td>
</tr>
<tr>
<td>S $\rightarrow$ B</td>
<td>${F_{CON}, A, B, N_A, K} - K_S$</td>
</tr>
</tbody>
</table>
Two reasons to use Maude:

- A new parametric trace generation is decided dynamically by trying to unify rigid message (it may fail).
- A property is a reachability problem, which can be checked at the same time while a model is being generated.

The way of implementation by Maude,

- Each elementary definition and function in the parametric model is implemented to functional modules.
- A trace generating system is represented in a system module.
- `search` command is used to find whether the negation of a specification is reachable.
Trace Generating System

- A state of the trace generating system is a 3-tuple: $\langle tr, S, k \rangle$, where
  - $tr$ is a parametric trace.
  - $S$ is a list of substitutions.
  - $k$ is a type of $tr$, where $k \in \{ot, st, pt\}$. $ot$ represents an **original** trace, $st$ represents a **satisfiable** trace and $pt$ represents a **pending** trace.

```
\[ A \rightarrow B : \{A, M\}_{k[A,B]} \]
\[ \langle a1\{A, M\}_{k[A,B]} , Nil, ot \rangle \]
\[ \langle \epsilon, Nil, ot \rangle \]
\[ \langle b1\{A, x\}_{k[A,B]} , Nil, ot \rangle \]
\[ \langle a1\{A, M\}_{k[A,B]} . b1\{A, x\}_{k[A,B]} , Nil, ot \rangle \]
\[ \langle b1\{A, x\}_{k[A,B]} , Nil, pt \rangle \]
\[ \langle a1\{A, M\}_{k[A,B]} . b1\{A, M\}_{k[A,B]} , Nil, st \rangle \]
\[ \langle a1\{A, M\}_{k[A,B]} . b1\{A, x\}_{k[A,B]} \{x \mapsto M\} , pt \rangle \]
```

On-the-fly Model Checking of Security Protocols
The Fixed NSPK Protocol

\[
\begin{align*}
A \rightarrow B : & \quad \{A, N_A\} + K_B \\
B \rightarrow A : & \quad \{B, N_A, N_B\} + K_A \\
A \rightarrow B : & \quad \{N_B\} + K_B \\
B \rightarrow A : & \quad \{N'_B\}_{N_B} \\
A \rightarrow B : & \quad \{N'_B - 1\}_{N_B}
\end{align*}
\]
The Fixed NSPK Protocol

\[ A \rightarrow B : \{ A, N_A \} + K_B \]
\[ B \rightarrow A : \{ B, N_A, N_B \} + K_A \]
\[ A \rightarrow B : \{ N_B \} + K_B \]

\[ A \rightarrow B : \{ A, N_A \} + K_B \]
\[ B \rightarrow A : \{ B, N_A, N_B \} + K_A \]
\[ A \rightarrow I(B) : \{ N_B \} + K_B \]
\[ I(A) \rightarrow B : \{ N_B \} + K_B \]
The Original Otway-Rees Protocol

\[
\begin{align*}
A \rightarrow B &: M, A, B, \{N_A, M, A, B\}_{K_{AS}} \\
B \rightarrow S &: M, A, B, \{N_A, M, A, B\}_{K_{AS}}, \{N_B, M, A, B\}_{K_{BS}} \\
S \rightarrow B &: M, \{N_A, K_{AB}\}_{K_{AS}}, \{N_B, K_{AB}\}_{K_{BS}} \\
B \rightarrow A &: M, \{N_A, K_{AB}\}_{K_{AS}}
\end{align*}
\]
Different Views of the Attack

“It is interesting to note that this protocol does not make use of $K_{AB}$ as an encryption key, so neither principal can know whether the key is known to the other”. (BAN logic paper)

“We refute the claim, showing that there exists a protocol similar to Otway-Rees that does not use the session key as an encryption key but informs one agent that his peer does know the session key.” (G. Bella’s thesis)

Integrity but rather authentication.