Oct. 14, 2002

Ali Ebuenasir

Presented by:

Edmund Clarke, Somesh Jha and Vijil Narzaro, June 1998

Written by:

Commerce Protocols

Specifying Security Properties of Electronic

A Machine Checkable Logic of Knowledge for
Discussion - Questions •
Discussion - A Wrap Up on Mini-Tutorial •
Conclusions •
Example •
Specification Language - Semantics •
Specification Language - Syntax •
What is predicate logic? •
How to Model the Computations of a Protocol? •
How to Model Messages? •
What is missing? •
Automating Analysis of Security Properties - Approaches •

Outline

† E. Clarke, S. Jha, W. Mcarro. Using state space exploration and a natural deduction style

† The work of the authors in †

Protocol:

— Built in adversary constructs new messages to subvert the

Special-Purpose Model Checkers:

— Prove that all traces of the model are correct.

— Define events that must hold in a correct trace (axioms).

— Define a set of rules that define valid traces (modeling).

Theorem-Proving (e.g., Isabelle):

— Theorem.

— Specify bad traces of execution and check if they belong to

Using General-Purpose Model Checkers (e.g., Casper):

Automated Analysis of Security Properties — Approaches
Computations.

The basic components of a semantics model: Messages and

appropriate for the specification of security requirements.

The language used for protocol specification may not be

— The evolution of an adversary’s knowledge.

Knowledge (the manipulation of)

— Properties involving knowledge (the manipulation of)

different participants of a protocol.

— Relationships between events and the variable bindings in

to express:

A logic that has a precise semantics and the expressive power

What is missing?
If $v$ is a message variable then $v \in \mathcal{M}$.

message variables.

Generalization: Message template contains one or more

Encryption/Decryption $\forall w \in \{m\} \Rightarrow \forall v \in w \land \mathcal{M} \ni w$.

Concatenation $\forall w \in m_1 \cdot m_2 \iff \forall \mathcal{M} \ni w \land \forall \mathcal{M} \ni w$.

Inductively define the set of messages over $\mathcal{M}$.

The space of atomic messages is denoted $\mathcal{A}$.

Nonces, replied Data.

Atomic Messages: not decomposable (e.g., principal name, keys).

How to Model Messages?
an inverse key of \( \mathcal{K} \).

\[
\text{KeyDecryption} \quad \mathcal{I} \models m \vdash \mathcal{I} \iff \forall \mathcal{I} \models \mathcal{I} \vee \exists \mathcal{I} \models \mathcal{I} -
\]

\[
\text{KeyEncryption} \quad \forall \mathcal{I} \models \mathcal{I} \iff \forall \mathcal{I} \models \mathcal{I} \vee \mathcal{I} \models \mathcal{I} -
\]

\[
\text{Projection} \quad \forall \mathcal{I} \models \mathcal{I} \iff \exists \mathcal{I} \models \mathcal{I} \vee \mathcal{I} \models \mathcal{I} -
\]

\[
\text{Pairing} \quad (\exists \mathcal{I} \models \mathcal{I} \vee \mathcal{I} \models \mathcal{I}) \iff \mathcal{I} \models \mathcal{I} -
\]

Common capabilities of an adversary in the literature.

- Set of information.

\[
\text{Derivability Relation} \quad \mathcal{I} \models \mathcal{I} \models m \iff \mathcal{I} \models \mathcal{I} \models m \iff \mathcal{I} \models \mathcal{I} \models m \iff \mathcal{I} \models \mathcal{I} \models m \iff \mathcal{I} \models \mathcal{I} \models m
\]

- Equality assumptions.

- Equality assumptions.

- Equality assumptions.

- Equality assumptions.

How to Model Messages? - cont.
Checking $m \in I$, for a given $m$, can still remain decidable.

The closure of $I$ under the above rules ($I$) can be infinite.

Perspectives on Principal's Role (Incarnation)

- How to model a session?
  - Impose a bound on the number of involved sessions.
  - How to make the model finite?

Simultaneously (this assumption may blow the model up)

- Each agent can be involved in multiple sessions
  - Acquitted information
- Adversary is allowed to create new messages from its adversaries
  - All communications go through the adversary

Assumptions:
- the adversary (i.e., a run of the protocol)
- Asynchronous composition of the actions of honest agents and

---

How to Model the Computations of a Protocol?
An agent can have multiple incarnations involved in multiple sessions.

\[ \mathcal{W} \leftarrow (N) \text{vars} : \mathcal{B} \leftarrow \text{}(\mathcal{W} \supset I) \]

\( S \) is the set of messages known to the principal in session \( S \).

\( I \) is a unique session ID.

\[ \langle p, I, S \rangle \in \Phi \text{ denotes an agent's entire knowledge and acquired bindings.} \]

\[ \forall \text{ the variable bindings acquired and execution thread of } S \text{ and a principal, and} \]

\[ \forall \text{ separate instantiation of a principal and} \]

How to Model the Computations of a Protocol?
Adversary

- How to Model the Computations of a Protocol?
\( \mathcal{W} \) is the set of all possible messages.

\( \mathcal{A} \) is the set of action names.

\( \mathcal{S} \) is the set of session IDs.

\( \mathcal{Z} \) is the set of global states.

Formal Definition of an Action: Action: \( \text{SEND and RECEIVE} \), and \( \text{user-defined actions} \).

Result: Transition from a global state \( s_0 \) to another state \( s_1 \).

Trace: A finite alternating sequence of global states and actions annotated as \( \tau = o_0 o_1 o_2 \ldots \), where \( o_i \) is a global state and \( o_i \) is an action.

The Global Model: The set of interactions of honest principals and the adversary.

The Global Model of a Protocol
An incarntation s performs some user defined action A with

\[ \rho \xleftarrow{s \cdot \text{HeC}} \rho \]

An incarntation s receives a message m in and reaches state \( \rho \) reached state \( \rho \) and sends a message m in global state \( \rho \)
variable in a.

- If φ is a formula, then Ax : φ is also a formula.
- If φ and ψ are formulas, then φ ∨ ψ is also a formula.
- If φ is a formula, then ¬φ is also a formula.
- Each atomic proposition is a formula.

Formula: is built by an inductive definition on propositions.

- Connectives of ∧ and ∨.

- Connectives can connect different propositions (e.g., Boolean).

- Term: a constant, variable, or a function over objects.

- Objects (e.g., \( z > 3 \)) in the domain of natural numbers.

- Atomic Proposition: identifies a basic relation on a group of predicates.

To reason about the objects in a domain of discourse with

First-Order Logic
term.

- If \( m_1 \) and \( m_2 \) are message terms then \( m_1 \{ m_2 \} \) is a message term.

- If \( m_1 \) and \( m_2 \) are message terms then \( m_1 \cdot m_2 \) is a message term.

- If \( s \) is an information term and \( m \) is a message variable then \( s \{ m \} \) is a message term.

- If \( s \) is an information term then \( pr(s) \) is a message where

- If \( m \) is a message variable then \( m \) is a message term.

- If \( m \) is a message then \( \bar{m} \) is a message term.

- If \( s \) is an information variable then \( s \) is an information term.

- If \( S \) is a session ID then \( S \) is an information term.

\[ N \leftarrow S: pr \]

Formal definition of terms: Specifications Language - Syntax
Term $s$ acts on $m$, $m$ is a message

- **Action Relation:** If $s$ is an incarntation and $m$ is a message

Term $s$ knows $m$, $m$ is an AP

- **Knowledge Relation:** If $s$ is an incarntation and $m$ is an AP

$\exists m \in M_1$. $m$ is an AP

- **Equality Relation:** If $m_1$, $m_2$ are message terms then $m_1 = m_2$

How to build atomic propositions (AP)

Terms represent incarntations and messages

**Specification Language - Syntax**
is a formula.
\[ f :: \text{is a formula} \wedge s \text{ is an intensional variable then } \forall s \]
- If \( f \) is a formula and \( s \) is an intensional variable then \( \forall s \) is a formula.
- If \( f_1 \) and \( f_2 \) are formulæ then \( f_1 \lor f_2 \) is a formula.
- If \( f \) is a formula then \( \neg f \) is a formula.
- If \( f \) is an atomic proposition then \( f \) is a formula.

How to build well-formed formulæ?

Specification Language - Syntax


\[ f \text{ substitution of free occurrences of } s \text{ with } s^0 \text{ in } [0s \leftarrow s/f] \text{ where } [0s \leftarrow s/f] \models \langle \nu, \nu \rangle \text{ for all sessions } s^0 \text{ in the model} \]

\[ \exists f = \langle \nu, \nu \rangle \text{ and } f = \langle \nu, \nu \rangle \text{ if } (\exists f \vee \neg f) = \langle \nu, \nu \rangle \]

\[ f \not\models \langle \nu, \nu \rangle \text{ if } \neg f = \langle \nu, \nu \rangle \]

\[ \forall s = \nu \text{ s.t. } \nu \text{ in } \forall \text{ such } \forall s \models (\nu)\exists s \models (\exists s) \models \langle \nu, \nu \rangle \]

\[ (\forall s = \nu)\nu \models (\nu)\nu \models (\exists s = \nu) \models \langle \nu, \nu \rangle \]

\[ f \text{ satisfies the } i-th \text{ state of the model} \]

\[ = \models \text{Inductively define the satisfaction relation} \]

\[ \models \text{ Interpret each formula over the traces of a model} \]

**Specication Language - Semantics**
Example - IKE Protocol for E-Commerce

**Initiate:** Customer → Merchant ← Merchant: rand, Hash(nonce)

**Invoice:** Merchant → Customer: ID, Trans, Date

**Payment:** Request: Merchant ← Merchant: SIPP

**Authorization Request:** Authority ← Authority: ClearanceRequest

**Authorization Response:** Authority ← Authority: Merchant: Authority: Authorization: Clearance

**Common:** Merchant ← Merchant: Y/N

**Common:** Includes transaction info, the price, and a description of goods.

**Common:** Includes credit card number, and customer’s nonce.

\[ \text{Hash(COMPON)}(\text{N, Y}) \]

\[ \text{Hash(COMPON)}(\text{N, Y}) \]

**Initiate:** Customer ← Customer: Y/N, Common: …
\[ \text{Auth} = (S)_{id} \land C = (S')_{id} \leftarrow (C \text{ knows } S) :: SA - \]

\textbf{Anonymity} •

\[ \text{Auth} = (S)_{id} \land C = (S')_{id} \leftarrow (m \cap C) \cdot 0C \text{ knows } S) \lor (C = (0C)_{id}) :: 0C_A :: SA - \]

\textbf{Privacy} •

\[ \text{Auth} = (S)_{id} \land C = (S')_{id} \leftarrow (C \text{ knows } S) \lor (C = (0C)_{id}) :: 0C_A :: SA - \]

\textbf{Examples: Privacy and Anonymity} •

  - Model Creation.
  - Protocol Specification.

\textbf{Example - Security Properties}
The complexity of modeling knowledge increases exponentially in the number of runs.

- The size of the model grows exponentially in the number of runs.

- The efficiency of the decision procedure for message derivation.

- Learning curve.

Limitations:

- The more expressive language, the more difficult to build a tool.

- Reasoning about knowledge is required.

- Expressive power.

- Security requirement in a different language with appropriate computation.

- Precise semantics with respect to a well-defined model of

Conclusions
<table>
<thead>
<tr>
<th>Yes</th>
<th>of Auth. Analysis</th>
<th>Yes</th>
<th>Rule-Based Analysis</th>
<th>adversary limited to Simple</th>
<th>φurp</th>
<th>STRANDS</th>
</tr>
</thead>
<tbody>
<tr>
<td>yes</td>
<td>Generic</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>CST</td>
</tr>
<tr>
<td>yes</td>
<td>Specific</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>NTL PA</td>
</tr>
<tr>
<td>no</td>
<td>Specific</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>SPI-Calculus</td>
</tr>
</tbody>
</table>

Discussion - A Wrap-Up on Mini-Tutorial
presented in this paper? (Slide 20)

What are the shortcomings and the limitations of the method?

What have to separately analyze each run of the protocol.

heuristics and experience.

BAN is not completely formal (idealization is based on the

relations between the events and the values of the variables).

BAN only provides a proof of trust (no reasoning on the

BAN does not introduce a model of computation.

Language proposed by this paper and BAN?

What is the difference between the knowledge-based specification

properties.

The same language for the specification of protocol and security

Also, what is its solution? (Slide 4)

What is (are) the problem(s) that this paper attempts to address?

Questions