A Machine Checkable Logic of Knowledge for Specifying Security Properties of Electronic Commerce Protocols

Written by:

Edmund Clarke, Somesh Jha and Will Marrero, June 1998

Presented by:

Ali Ebnenasir

Oct. 14, 2002

Outline

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

Automatic Analysis of Security Properties - Approaches

- Using General-Purpose Model Checkers (e.g., Casper):
  - Specify bad traces of execution and check if they belong to the model.
- Theorem-Proving (e.g., Isabelle):
  - Define a set of rules that define *valid* traces (modeling).
  - Define events that must hold in a *correct* trace (axioms).
  - Prove that all traces of the model are correct.
- Special-Purpose Model Checkers:
  - Built in adversary constructs new messages to subvert the protocol.
  - The work of the authors in †

† E. Clarke, S. Jha, W. Marrero. Using state space exploration and a natural deduction style message derivation engine to verify security protocols. In *Proceedings of the IFIP PROCOMET*, 1998.

What is missing?

- A logic that has a precise semantics and the expressive power to express:
  - Relationships between events and the variable bindings in different participants of a protocol.
  - Properties involving knowledge (the manipulation of knowledge is built-in).
  - The evolution of an adversary's knowledge.
- The language used for protocol specification may not be appropriate for the specification of security requirements.
- The basic components of a semantics model: Messages and Computations.

How to Model Messages?

- Atomic Message: not decomposable (e.g., principal name, keys, Nonces, Data).
- The space of atomic messages is denoted $\mathcal{A}$.
- Inductively define the set of messages $\mathcal{M}$ over $\mathcal{A}$:
  - $a \in \mathcal{A} \Rightarrow a \in \mathcal{M}$
  - $m_1 \in \mathcal{M} \ \wedge \ m_2 \in \mathcal{M} \Rightarrow m_1 \cdot m_2 \in \mathcal{M}$ (Concatenation).
  - $m \in \mathcal{M} \ \wedge \ k \in \mathcal{A} \Rightarrow \{m\}_k \in \mathcal{M}$ (Encryption/Decryption).
- Generalization: Message template contains one or more message variables.
- If v is a message variable then $v \in \mathcal{M}$.

How to Model Messages? - cntd.

- Encryption assumption
  - $\forall m, m', m_1, m_2 \in \mathcal{M}, k \in \mathcal{A}$ ::
    - $\{m\}_k \neq m_1 \cdot m_2$
    - $\{m\}_k = \{m'\}_{k'} \Rightarrow (m = m') \ \wedge \ (k = k')$
- Derivability relation ($\vdash$): $m \in \mathcal{I} \Rightarrow \mathcal{I} \vdash m$, where $\mathcal{I}$ is the initial set of information.
- Common capabilities of an adversary in the literature:
  - $\mathcal{I} \vdash m_1 \ \wedge \ \mathcal{I} \vdash m_2 \Rightarrow \mathcal{I} \vdash (m_1 \cdot m_2)$ (Pairing).
  - $\mathcal{I} \vdash (m_1 \cdot m_2) \Rightarrow \mathcal{I} \vdash m_1 \ \wedge \ \mathcal{I} \vdash m_2$ (Projection).
  - $\mathcal{I} \vdash m \ \wedge \ \mathcal{I} \vdash k \Rightarrow \mathcal{I} \vdash \{m\}_k$ (Encryption).
  - $\mathcal{I} \vdash \{m\}_k \ \wedge \mathcal{I} \vdash k^{-1} \Rightarrow \mathcal{I} \vdash m$ (Decryption), where k^{-1} is an inverse key of k.

How to Model Messages? - cntd.

- The closure of $\mathcal{I}$ under the above rules ($\bar{\mathcal{I}}$) can be infinite.
- Checking $m \in \bar{\mathcal{I}}$, for a given m, can still remain decidable.

How to Model the Computations of a Protocol?

- Asynchronous composition of the actions of honest agents and the adversary (i.e., a **run** of the protocol).
- Assumptions:
  - All communications go through the adversary.
  - Adversary is allowed to create new messages from its acquired information.
  - Each agent can be involved in multiple sessions simultaneously (this assumption may blow the model up).
- How to make the model finite?
  - Impose a bound on the number of involved sessions.
- How to model a session?
  - Personify a principal's role (Incarnation).

How to Model the Computations of a Protocol?

- An incarnation includes:
  - A separate instantiation of a principal, and
  - A single thread of execution, and
  - All the variable bindings and acquired knowledge.
- An incarnation of an agent is denoted $\Psi = \langle N, S, I, B, P \rangle$.
  - $N \in names$.
  - S is a unique session ID.
  - I is the set of messages known to the principal in session S ($I \subseteq \mathcal{M}$).
  - $B : vars(N) \rightarrow \mathcal{M}$.
  - P is the set of actions N.
- An agent can have multiple incarnation (involved in multiple sessions).

How to Model the Computations of a Protocol? - Adversary

- Incarnation of an adversary (denoted $\Omega = \langle N_\Omega, S_\Omega, I_\Omega, \emptyset, \emptyset \rangle$):
  - Does not follow the protocol.
  - Does not include a predefined set of actions.
  - Includes a set of known messages.
  - At any time can generate a message from I_Ω.

The Global Model of a Protocol

- The Global Model: The set of incarnations of honest principals plus the incarnations of the adversary.
- Trace: A finite alternating sequence of global states and actions annotated as $\pi = \sigma_0\alpha_1\sigma_1\alpha_2 \cdots \alpha_n\sigma_n$, where σ_i is a global state and α_i is an action.
- Actions: SEND and RECEIVE, and user defined actions.
- Result: transition from a global state s_0 to another state s_1.
- Formal Definition of an Action: $\Sigma \times S \times A \times \mathcal{M} \times \Sigma$, where
  - Σ is the set of global states.
  - S is the set of session IDs.
  - A is the set of action names.
  - $\mathcal{M}$ is the set of all possible messages.

The Global Model of a Protocol - Actions

- An incarnation s sends a message m in global state σ and reaches state σ' (denoted $\sigma \rightarrow^{s.Send.m} \sigma'$).
- An incarnation s receives a message m in σ and reaches state σ' (denoted $\sigma \rightarrow^{s.Rec.m} \sigma'$).
- An incarnation s performs some user defined action A with argument m in state σ and reaches state σ' (denoted $\sigma \rightarrow^{s.A.m} \sigma'$).

First-Order Logic

- To reason about the objects in a domain of discourse with predicates.
- Atomic Proposition: identifies a basic relation on a group of objects (e.g., $(2 < 3)$ in the domain of natural numbers).
- Term: A constant, variable, or a function over objects.
- Connectives can connect different propositions (e.g., Boolean connectives $\wedge$ and $\vee$).
- Formula: is built by an inductive definition on propositions.
  - Each atomic proposition is a formula.
  - If α is a formula then $\neg\alpha$ is also a formula.
  - If α and β are formulae then $\alpha \ \wedge \ \beta$ is also a formula.
  - If α is a formula then $\forall x :: \alpha$ is a formula, where x is a free variable in α.

Specification Language - Syntax

- Formal definition of terms:
  - If S is a session ID then S is an incarnation term.
  - If s is an incarnation variable then s is an incarnation term.
  - If M is a message then M is a message term.
  - If m is a message variable then m is a message term.
  - If s is an incarnation term then $pr(s)$ is a message, where $pr : S \rightarrow N$.
  - If s is an incarnation term and m is a message variable then $s.m$ is a message term.
  - If m_1 and m_2 are message terms then $m_1 \cdot m_2$ is a message term.
  - If m_1 and m_2 are message terms then $\{m_1\}_{m_2}$ is a message term.

Specification Language - Syntax

- Terms represent incarnations and messages.
- How to build atomic propositions (AP)?
  - Equality relation: If m_1, m_2 are message terms then $(m_1 = m_2)$ is an AP.
  - Knowledge relation: If s is an incarnation and m is a message term then (s **Knows** m) is an AP.
  - Action relation: If s is an incarnation and m is a message term then (s **Act** m) is an AP.

Specification Language - Syntax

- How to build well-formed formula?
  - If f is an atomic proposition then f is a formula.
  - If f is a formula then $\neg f$ is a formula.
  - If f_1 and f_2 are formulae then $f_1 \ \wedge \ f_2$ is a formula.
  - If f is a formula and s is an incarnation variable then $\forall s :: f$ is a formula.

Specification Language - Semantics

- Interpret each formula over the traces of a model.
- Inductively define the satisfaction relation $\models$.
  - $\langle \pi, i \rangle \models f$ means the ith state of π satisfies f
  - $\langle \pi, i \rangle \models (m_1 = m_2)$ iff $\sigma_i(m_1) = \sigma_i(m_2)$
  - $\langle \pi, i \rangle \models (s$ **Knows** $m)$ iff $\sigma_i(m) \in \bar{I}_j$ for some Ψ_j in σ_i such that $S_j = s$
  - $\langle \pi, i \rangle \models (s$ **A** $m)$ iff $\alpha_i = s.\mathbf{A}.m$
  - $\langle \pi, i \rangle \models \neg f$ iff $\langle \pi, i \rangle \not\models f$
  - $\langle \pi, i \rangle \models (f_1 \ \wedge \ f_2)$ iff $\langle \pi, i \rangle \models f_1$ and $\langle \pi, i \rangle \models f_2$
  - $\langle \pi, i \rangle \models (\forall s :: f)$ iff for all sessions s_0 in the model $\langle \pi, i \rangle \models [f/s \mapsto s_0]$, where $[f/s \mapsto s_0]$ means the substitution of free occurrences of s with s_0 in f

Example - 1KP Protocol for E-Commerce

- Protocol Flow:
  1. Initiate: Customer → Merchant: *rand, Hash(nonce, CCNum)*
  2. Invoice: Merchant → Customer: ID_M, $Trans_M$, *Date*, $nonce_M$, *Hash(Common)*.
  3. Payment: Customer → Merchant: $\{SLIP\}_{pk_A}$.
  4. Auth-Request: Merchant → Authority: ClearanceRequest.
  5. Auth-Response: Authority → Merchant: Y/N, $\{Y/N, Hash(Common)\}_{sk_A}$.
  6. Confirm: Merchant → Customer: : Y/N, $\{Y/N, Hash(Common)\}_{sk_A}$.
- Common: includes transaction info., the price, and a description of goods.
- SLIP: includes Common, customer's credit card number, and customer's nonce.

Example - Security Properties

- How to analyze a protocol?
  - Protocol Specification.
  - Model Creation.
  - The Specification of Security Protocols.
- Examples: Privacy and Anonymity.
- Privacy:
  - $\forall S :: \forall C_0 :: (pr(C_0) = C) \ \wedge \ (S \ \textbf{Knows} \ C_0 \cdot DESC) \rightarrow [pr(S) = C \ \vee \ pr(S) = Mer]$.
  - $\forall S :: \forall C_0 :: (pr(C_0) = C) \ \wedge \ (S \ \textbf{Knows} \ C_0 \cdot CCNum) \rightarrow [pr(S) = C \ \vee \ pr(S) = Auth]$.
- Anonymity:
  - $\forall S :: (S \ \textbf{Knows} \ \ C) \rightarrow [pr(S) = C \ \vee \ pr(S) = Auth]$.

Conclusions

- Precise semantics with respect to a well-defined model of computation.
- Specify security requirement in a different language with appropriate expressive power.
- Reasoning about knowledge is required.
- The more expressive language, the more difficult to build a tool.
- Limitations:
  - Learning curve.
  - The efficiency of the decision procedure for message derivation.
  - The size of the model grows exponentially in the number of runs.
  - The complexity of modeling knowledge (e.g., $\langle \pi, i \rangle \models (s$ **Knows** $m)$).

Discussion - A Wrap Up on Mini-Tutorial

	Knowledge	CM	Tool	Application	Built-In Adversary
Spi-Calculus	No	No	No	Manual Analysis of Auth. and Secrecy	No
NRL PA	No	Unification	Yes	Analysis of Auth.	specified operations
BAN	Proof of trust	No	No	Analysis of Auth. using state of knowledge	No
CAPSL	No	Rewrite Rules	Yes	Analysis of Auth.	specified scenario
Strands	No	Strands	Yes (Athena)	Analysis of Auth.	specified generic strands
Murϕ	limited to adversary	Rule-Based	Yes	Analysis of Auth.	Yes

CM: Computational Model

Questions

- What is (are) the problem(s) that this paper attempts to address? Also, what is its solution? (Slide 4)
  - The same language for the specification of protocol and security properties.
- What is the difference between the knowledge-based specification language proposed by this paper and BAN?
  - BAN does not introduce a model of computation.
  - BAN only provides a proof of trust (no reasoning on the relations between the events and the values of the variables).
  - BAN is not completely formal (idealization is based on the heuristics and experience).
  - We have to separately analyze each run of the protocol.
- What are the shortcomings and the limitations of the method presented in this paper? (Slide 20)