On some Logic-Based Approaches to Security Protocol Verification

Bruno Blanchet

Google

On leave from Inria, Paris
bruno.blanchet@inria.fr

John Mitchell’s Festschrift
May 2016
Proceedings

14th IEEE Computer Security Foundations Workshop

11-13 June 2001
Cape Breton, Nova Scotia, Canada

Sponsored by
IEEE Computer Society Technical Committee on Security and Privacy
Many interesting papers

Table of Contents

Preface ................................................................. vii
Workshop Committee ................................................ viii

Non-interference & Information Flow

Noninterference Equations for Nondeterministic Systems .............................................. 3
S. Pinsky and E. Ziegler

Robust Declassification ................................................... 15
S. Edancevic and A. Myers

Access Control

A State-Transition Model of Trust Management and Access Control ............................ 27
A. Chanda, D. Dean, and J. Mitchell

Revocations — A Classification .............................................. 44
Å. Hagström, S. Jajodia, F. Parisi-Presicce, and D. Wijesekera

A Logical Reconstruction of SPKI .......................................... 59
J. Halpern and R. van der Meyden

Protocols I

A Security Analysis of the Cliques Protocols Suites ................................................. 73
O. Pereira and J-J. Quisquater

An Efficient Cryptographic Protocol Verifier Based on Prolog Rules .......................... 82
B. Blanchet

Proving Secrecy is Easy Enough ........................................... 97
V. Corrion, J. Millen, and H. Ruesl

Panel

Relating Cryptography and Cryptographic Protocols ............................................ 111
A. Scedrov (Moderator), R. Canetti, J. Gutman, D. Wagner, and M. Waidner

Information Flow & Multi-threading

A New Type System for Secure Information Flow ................................................. 115
G. Smith

A Generic Approach to the Security of Multi-threaded Programs ............................ 126
H. Mantel and A. Sabelfeld

Protocols II

Authenticity by Typing for Security Protocols ............................................. 145
A. Gordon and A. Jeffrey

Computing Symbolic Models for Verifying Cryptographic Protocols .......................... 160
M. Fiore and M. Abadi

Protocol Insecurity with Finite Number of Sessions is NP-Complete ......................... 174
M. Rusinowitch and M. Turarini

Intrusion Tolerance & Detection

Multi-phase Damage Confinement in Database Systems for Intrusion Tolerance .......... 191
P. Liu and S. Jajodia

Markov Chains, Classifiers, and Intrusion Detection .............................................. 206
S. Jha, K. Tan, and R. Maxion

Log Auditing through Model-Checking .................................................. 220
M. Roger and J. Goubault-Larrecq

Panel

Non-interference: Who Needs It? .................................................. 237
P. Ryan (Moderator), J. McLean, J. Millen, and V. Gligor

Logics for Protocol Verification

A Compositional Logic for Protocol Correctness ........................................... 241
N. Durgin, J. Mitchell, and D. Pavlovic

Logical Relations for Encryption .................................................. 256
E. Sumi and B. Pierce

Secrecy & Privacy

Privacy-Preserving Cooperative Scientific Computations .................................. 273
W. Du and M. Atallah

Confined Mobile Functions ..................................................... 283
Z. Kiri

Confidentiality-Preserving Refinement .................................................. 295
M. Heisel, A. Pfitzmann, and T. Santen

Author Index ................................................................. 307

Bruno Blanchet (Google - Inria) Logic-based approaches May 2016
Panel

Relating Cryptography and Cryptographic Protocols

A. Scedrov (Moderator), R. Canetti, J. Guttman, D. Wagner, and M. Waidner

(Martín Abadi may have replaced Michael Waidner.)
Model of protocols

- **The symbolic model** or “Dolev-Yao model”:
  - The cryptographic primitives are **blackboxes**.
  - The messages are **terms** on these primitives.
  - The adversary is restricted to compute only using these primitives.
    \[\Rightarrow \text{perfect cryptography assumption}\]

  This model facilitates automatic proofs.

- **The computational model**:
  - The messages are **bitstrings**.
  - The cryptographic primitives are **functions on bitstrings**.
  - The adversary is any **probabilistic polynomial-time Turing machine**.

  This model is more realistic than the symbolic model, but until recently proofs were only manual.


Access Control

A State-Transition Model of Trust Management and Access Control
A. Chander, D. Dean, and J. Mitchell

Logics for Protocol Verification

A Compositional Logic for Protocol Correctness
N. Durgin, J. Mitchell, and D. Pavlović
A **Hoare logic** for security protocol verification.

Based on a formal semantics of protocols.

Initially: **symbolic**.

Led to considerable developments (more than 20 papers):
- Protocol derivation
- **Computational** PCL
- Proof assistant in Isabelle

Applied to many protocols.
- Found and fixed problems in standardized protocols (IEEE 802.11i, IETF GDOI)
An Efficient Cryptographic Protocol Verifier Based on Prolog Rules ............................................. 82

B. Blanchet
Symbolic security protocol verifier.

Fully automatic.

Unbounded number of sessions and message space.

Cryptographic primitives: defined by rewrite rules or equations.

Security properties: secrecy, authentication, some equivalences.

Does not always terminate and is not complete. In practice:

- Efficient: small examples verified in less than 0.1 s; complex ones from a few minutes to hours.
- Precise: no false attack in 19 protocols of the literature tested for secrecy and authentication.
ProVerif

Protocol: Pi calculus + cryptography
Primitives: rewrite rules, equations

Properties to prove: Secrecy, authentication, process equivalences

Automatic translator

Horn clauses

Derivability queries

Resolution with selection

Non-derivable: the property is true
Derivation

Attack: the property is false
False attack: I don’t know
Definition of cryptographic primitives

Two kinds of operations:

- **Constructors** $f$ are used to build terms $f(M_1, \ldots, M_n)$
  
  \[
  \text{fun } f(T_1, \ldots, T_n) : T. \]

- **Destructors** $g$ manipulate terms
  
  Destructors are defined by rewrite rules $g(M_1, \ldots, M_n) \rightarrow M$.
  
  \[
  \text{reduc forall } x_1 : T_1, \ldots, x_k : T_k; g(M_1, \ldots, M_n) = M. \]

**Example: shared-key encryption**

\[
\text{fun encrypt(bitstring, key) : bitstring.} \]

**Example: shared-key decryption**

\[
\text{decrypt(encrypt(m, k), k) \rightarrow m} \]

\[
\text{reduc forall } x : \text{bitstring}, y : \text{key}; \text{decrypt(encrypt(x, y), y)} = x. \]
Syntax of the process calculus

A dialect of the applied pi calculus (Abadi, Fournet, POPL’01):
Pi calculus + cryptographic primitives

\[ M, N ::= \]
\[ x, y, z \quad \text{variable} \]
\[ a, b, c, k, s \quad \text{name} \]
\[ f(M_1, \ldots, M_n) \quad \text{constructor application} \]

\[ P, Q ::= \]
\[ \text{output} (M, N); P \quad \text{output} \]
\[ \text{input} (M, x : T); P \quad \text{input} \]
\[ \text{let} \ x = g(M_1, \ldots, M_n) \text{ in } P \text{ else } Q \quad \text{destructor application} \]
\[ \text{if} \ M = N \text{ then } P \text{ else } Q \quad \text{conditional} \]
\[ \text{new} \ a : T; P \quad \text{restriction} \]
\[ 0 \ P \ | \ Q \ | \ !P \]
Example: The Denning-Sacco protocol

Message 1. \( A \rightarrow B : \{\{ k \}_sk_A \}_pk_B \quad k \text{ fresh} \)
Message 2. \( B \rightarrow A : \{s\}_k \)

\[
\text{new } sk_A : \text{sskey}; \text{let } pk_A = \text{spk}(sk_A) \text{ in } \\
\text{new } sk_B : \text{skey}; \text{let } pk_B = \text{pk}(sk_B) \text{ in } \\
\text{out}(c, pk_A); \text{out}(c, pk_B);
\]

\((A)\) \quad ! \text{in}(c, x \_ pk_B : \text{pkey}); \\
\text{new } k : \text{key}; \text{out}(c, \text{pencrypt}(\text{sign}(k2b(k), sk_A), x \_ pk_B)); \\
\text{in}(c, x : \text{bitstring}); \text{let } s = \text{decrypt}(x, k) \text{ in } 0

\((B)\) \quad | \quad ! \text{in}(c, y : \text{bitstring}); \text{let } y' = \text{pdecrypt}(y, sk_B) \text{ in } \\
\text{let } k2b(k) = \text{checksign}(y', pk_A) \text{ in out}(c, \text{encrypt}(s, k))
Security properties

- **Secrecy**: the adversary cannot obtain the secret $s$.

**query** attacker(s).

- **Correspondence assertions**: (authentication)
  If an event has been executed, then some other events must have been executed.

- **Process equivalences**: the adversary cannot distinguish between two processes.
  - **Strong secrecy**: the adversary cannot see when the value of the secret changes.
  - Equivalences between processes that differ only by terms they contain (joint work with Martín Abadi and Cédric Fournet)

In particular, proof of protocols relying on weak secrets.
The main predicate used by the Horn clause representation of protocols is attacker:

\[ \text{attacker}(M) \] means “the adversary may have } M\)."

We can model actions of the adversary and of the protocol participants thanks to this predicate.

The term } M} is secret when it cannot be built by an adversary.

**Theorem (Secrecy)**

*If attacker(*} M\) cannot be derived from the clauses, then } M\) is secret.*

The resolution algorithm determines whether a fact is derivable from the clauses.
Applications

Case studies:
- 19 protocols of the literature
- Certified email (with Martín Abadi)
- JFK (with Martín Abadi and Cédric Fournet)
- Plutus (with Avik Chaudhuri)
- Avionic protocols (ARINC 823)

Case studies by others:
- E-voting protocols (Delaune, Kremer, and Ryan; Backes et al)
- Zero-knowledge protocols, DAA (Backes et al)
- Shared authorisation data in TCG TPM (Chen and Ryan)
- Electronic cash (Luo et al)
- ...
Applications

1. **Case studies**

2. **Extensions:**
   - Extensions to **XOR** and **Diffie-Hellman** (Küsters and Truderung), to **bilinear pairings** (Pankova and Laud)
   - StatVerif: extension to **mutable state** (Arapinis et al)
   - Set-Pi: extension to **sets with revocation** (Bruni et al)

3. **ProVerif as back-end:**
   - TulaFale: **Web service** verifier (Bhargavan et al)
   - FS2PV: F# to ProVerif, applied to TLS and TPM (Bhargavan et al)
   - JavaSpi: Java to ProVerif (Avalle et al)
   - Web-spi: web security mechanisms (Bansal et al)
Conclusion

- Power of logic for modeling and reasoning on protocols.
- Used in very different ways.
- Any relation between these approaches?