

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

	CSFW-14
Preface	vii
Workshop Committee	viii
Non-interference & Information Flow	
Noninterference Equations for Nondeterministic Systems.....	3
<i>S. Pinsky and E. Ziegler</i>	
Robust Declassification.....	15
<i>S. Zdancewic and A. Myers</i>	
Access Control	
A State-Transition Model of Trust Management and Access Control.....	27
<i>A. Chander, D. Dean, and J. Mitchell</i>	
Revocations — A Classification.....	44
<i>Å. Hagström, S. Jajodia, F. Parisi-Presicce, and D. Wijesekera</i>	
A Logical Reconstruction of SPKI.....	59
<i>J. Halpern and R. van der Meyden</i>	
Protocols I	
A Security Analysis of the Cliques Protocols Suites.....	73
<i>O. Pereira and J.-J. Quisquater</i>	
An Efficient Cryptographic Protocol Verifier Based on Prolog Rules.....	82
<i>B. Blanchet</i>	
Proving Secrecy is Easy Enough.....	97
<i>V. Cortier, J. Millen, and H. Rueß</i>	
Panel	
Relating Cryptography and Cryptographic Protocols.....	111
<i>A. Scedrov (Moderator), R. Canetti, J. Guttman, D. Wagner, and M. Waidner</i>	
Information Flow & Multi-threading	
A New Type System for Secure Information Flow.....	115
<i>G. Smith</i>	
A Generic Approach to the Security of Multi-threaded Programs.....	126
<i>H. Mantel and A. Sabelfeld</i>	

Protocols II	
Authenticity by Typing for Security Protocols.....	145
<i>A. Gordon and A. Jeffrey</i>	
Computing Symbolic Models for Verifying Cryptographic Protocols.....	160
<i>M. Fiore and M. Abadi</i>	
Protocol Insecurity with Finite Number of Sessions is NP-Complete.....	174
<i>M. Rusinowitch and M. Turuani</i>	
Intrusion Tolerance & Detection	
Multi-phase Damage Confinement in Database Systems for Intrusion Tolerance.....	191
<i>P. Liu and S. Jajodia</i>	
Markov Chains, Classifiers, and Intrusion Detection.....	206
<i>S. Jha, K. Tan, and R. Maxion</i>	
Log Auditing through Model-Checking.....	220
<i>M. Roger and J. Goubault-Larrecq</i>	
Panel	
Non-interference: Who Needs It?.....	237
<i>P. Ryan (Moderator), J. McLean, J. Millen, and V. Gligor</i>	
Logics for Protocol Verification	
A Compositional Logic for Protocol Correctness.....	241
<i>N. Durgin, J. Mitchell, and D. Pavlovic</i>	
Logical Relations for Encryption.....	256
<i>E. Sumii and B. Pierce</i>	
Secrecy & Privacy	
Privacy-Preserving Cooperative Scientific Computations.....	273
<i>W. Du and M. Atallah</i>	
Confined Mobile Functions.....	283
<i>Z. Krlh</i>	
Confidentiality-Preserving Refinement.....	295
<i>M. Heisel, A. Pfizmann, and T. Santen</i>	
Author Index	307

Panel

Relating Cryptography and Cryptographic Protocols.....	111
<i>A. Scedrov (Moderator), R. Canetti, J. Guttman, D. Wagner, and M. Waidner</i>	

(Martín Abadi may have replaced Michael Waidner.)

- 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.
⇒ **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.

Relating Cryptography and Cryptographic Protocols.....	111
<i>A. Scedrov (Moderator), R. Canetti, J. Guttman, D. Wagner, and M. Waidner</i>	

- Lincoln, **Mitchell**, Mitchell, and Scedrov. A probabilistic poly-time framework for protocol analysis. CCS'98.
- Pfitzmann, Schunter and Waidner. Cryptographic security of reactive systems. Workshop on secure architectures and information flow, 1999.
- Abadi and Rogaway. Reconciling two views of cryptography (the computational soundness of encryption). IFIP TCS 2000.
- Canetti. A unified framework for analysing security of protocols. Cryptology ePrint Archive, report 2000/067.

Access Control

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

Logics for Protocol Verification

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

First paper on protocol composition logic (PCL)

- 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
<i>B. Blanchet</i>	

- **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.

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 $f(M_1, \dots, M_n)$

```
fun  $f(T_1, \dots, T_n) : T.$ 
```

Example: shared-key encryption

```
fun encrypt(bitstring, key) : bitstring.
```

- **Destructors** g manipulate terms

Destructors are defined by rewrite rules $g(M_1, \dots, M_n) \rightarrow M.$

```
reduc forall  $x_1 : T_1, \dots, x_k : T_k; g(M_1, \dots, M_n) = M.$ 
```

Example: shared-key decryption $\text{decrypt}(\text{encrypt}(m, k), k) \rightarrow m$

```
reduc forall  $x : \text{bitstring}, y : \text{key}; \text{decrypt}(\text{encrypt}(x, y), y) = x.$ 
```

Syntax of the process calculus

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

$M, N ::=$	terms
x, y, z	variable
a, b, c, k, s	name
$f(M_1, \dots, M_n)$	constructor application

$P, Q ::=$	processes
out (M, N); P	output
in ($M, x : T$); P	input
let $x = g(M_1, \dots, M_n)$ in P else Q	destructor application
if $M = N$ then P else Q	conditional
new $a : T$; P	restriction
$0 \quad P \mid Q \quad !P$	

Example: The Denning-Sacco protocol

Message 1. $A \rightarrow B : \{\{k\}_{sk_A}\}_{pk_B}$ k fresh

Message 2. $B \rightarrow A : \{s\}_k$

new $sk_A : \text{sskey}$; **let** $pk_A = \text{spk}(sk_A)$ **in**

new $sk_B : \text{skey}$; **let** $pk_B = \text{pk}(sk_B)$ **in**

out(c, pk_A); **out**(c, pk_B);

- (A) ! **in**($c, x_{pk_B} : \text{pkey}$);
 new $k : \text{key}$; **out**($c, \text{pencrypt}(\text{sign}(\text{k2b}(k), sk_A), x_{pk_B})$);
 in($c, x : \text{bitstring}$); **let** $s = \text{decrypt}(x, k)$ **in** 0
- (B) | ! **in**($c, y : \text{bitstring}$); **let** $y' = \text{pdecrypt}(y, sk_B)$ **in**
 let $\text{k2b}(k) = \text{checksign}(y', pk_A)$ **in** **out**($c, \text{encrypt}(s, k)$)

- **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 Horn clause representation

- 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 $\text{attacker}(M)$ *cannot* be derived from the clauses, then M is secret.

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

1 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)
- ...

2 Extensions

3 ProVerif as back-end

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?