Michael Backes
Saarland University, Germany
joint work with Birgit Pfitzmann and Michael Waidner

Secure Reactive Systems, Day 3:
Reactive Simulatability – Property Preservation and Crypto. Examples

Tartu, 03/01/06
Recall the Big Picture

Idealized Crypto
- Signature
- Encryption
- Hashfunction
- Key establishment

But can we justify?

Designed by CAD
Verified by CAV
Recall the RS Framework

- Precise system model allowing cryptographic and abstract operations
- Reactive simulatability with composition theorem
- Preservation theorems for security properties
- Concrete pairs of idealizations and secure realizations
- Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support
- Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.
- Detailed Proofs (Poly-time, cryptographic bisimulations with static information flow analysis, ... )
Composition – One System

Given:

Then this holds:
Proof Idea (Single Composition)

\[ H \xrightarrow{\text{A}^\#} H_0 \xrightarrow{\geq} H_0 \xleftarrow{\text{A}'_0} \text{A}_0 = \text{A}^\# \]
Recall the RS Framework

- Precise system model allowing cryptographic and abstract operations
- Reactive simulatability with composition theorem
- **Preservation theorems for security properties**
- **Concrete pairs of idealizations and secure realizations**
- Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support
- Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.
- **Detailed Proofs** (Poly-time, cryptographic bisimulations with static information flow analysis, ... )
Abstraction of one-step Public-Key Encryption

• On the board…
Example: Encryption, passive

∀A₁, A₂ ∈ PPT:

\[ P( b^* = b :: ( Attacker success ) ) \]

(Attacker success)

\[ ( sk, pk ) \leftarrow gen(k); \]

(Keys)

(\( sk, pk \) \leftarrow gen(k);

\( ( m_0, m_1, v ) \leftarrow A_1(k, pk); \)

(Message choice)

\( b \in \mathbb{R} \{0, 1\}; \)

(Encrypt)

\( c := enc(pk, m_b); \)

(Guess)

\( b^* \leftarrow A_2(v, c) \)

(Negligible)

\( \leq 1/2 + 1/poly(k) \)
Cryptographic Idealization Layers

Symbolic abstractions

Larger abstractions

Small real abstractions

Low-level crypto (not abstract)

Normal cryptographic definitions

Dolev-Yao Model

VSS [GM95]

Secure channels [PW00, PW01, CK02, BJP02,...]

Encryption as $E(pk, 1^{len})$ [LMMS98, PW00, C01,...]

Certified mail [PSW00]

Authentication/signatures as statement database [BPW03 ...]

Real auth/sig's + integrity lookup [LMMS98, C01,...]

Certified mail [CL01]

Related: [SM93, P93]

Dolev-Yao Model
in_s: (send, m, r): enc_r(sign_s(s, m, r))

net_r,s: ( enc_r(sign_{s,c}(s, m, r)):

1. Decrypt, check signature, s, r \rightarrow abort at failure

2. Output (received, s, m)
Recall Naive Approach

E.g., secure channel

Not a good abstraction since not enough information for the simulator:

- Who is sender? Who is recipient?
- Length of $m$?
- No availability ...
Better Abstraction

in_s: (send, m, r):
msg_s,r := msg_s,r & m,
output (i, l, s, r) to Adversary
from_adv_r: (send,i,s):
m:= msg_s,r [i], output (received, s, m)
Proof Idea

Real Secure Channels \quad Ideal Secure Channels

$\text{view}_{\text{real}}(H) \approx \text{view}_{\text{ideal}}(H)$

1. Proof by probabilistic bisimulation possible for „most“ cases
2. Collect remaining traces in error sets (e.g., for forged signatures)
3. Show reduction proof of error sets against underlying crypto-primitives (e.g., against security of the signature scheme)
Explicit Security Requirements in the Model
Recall Prior Result

• “as secure as” (reactive simulatability)

• for certain versions of and
Specification Styles

- Is what people want?
  - Often yes, in particular together with
    - E.g., secure channels (see also spi calculus), certified mail
    - But not always ...
Alternative: Property-based spec.

- E.g., “I want a tight roof on top”: integrity
  - Preserved by “≥”:

\[
\begin{align*}
\text{⇒} & \quad \text{Roof on top} \\
\text{ﬄ} & \quad \text{…}
\end{align*}
\]

\[
\begin{align*}
\text{ﬄ} & \quad \text{Roof on top} \\
\text{ﬄ} & \quad \text{…}
\end{align*}
\]
Characterization

Integrity (e.g., temporal logic)
Privacy (e.g., information flow, non-interference)

Liveness: (Something good eventually happens)
  • Termination
  • Starvation freedom
  • Guaranteed service
Integrity
Integrity

Abstract formulation: e.g., temporal logic over the interface of a system (ports to the user)

Cryptographic semantics: For all with linear-time semantics (set of permitted traces)

Example: “If m is input at p? at time t, then there exists a future time s such that m is output at port q!” (≈ Reliability)

A trace tr is contained in Req if
\[ \forall t: p?m \rightarrow \exists s > t: s: q!m \]
Fulfillment of Integrity

Different kinds of fulfillment:

- **Perfect**: Requirement always holds
- **Computational**: For polynomial-time adversary and users only and up to negligible error probability

**Integrity Preservation Theorem**: Simulatability preserves “⩾”: $\text{Sys}_1 \geq \text{Sys}_2$ and $\text{Sys}_2 \models \text{Req}$ implies $\text{Sys}_1 \models^{\text{poly}} \text{Req}$
Example: Ordered Secure Channels over Unordered Ones

\[ \square (\text{receivelist}_{u,v} \subseteq \text{sendlist}_{u,v}) \]

Preservation theorem

\[ \text{TH}_{\text{ord}} \]

\[ \text{A} \]

\[ \text{H} \]

\[ \text{M}_1 \]

\[ \text{M}_2 \]

\[ \text{A'} \]

\[ \text{TH}_{\text{SecMess}} \]

\[ \text{Real SecMess} \]

1. Application: Formally verified Bisimulation

2. Application: Composition theorem

Preservation

Transitivity
Cryptographic Non-Interference (Transitive)
Privacy

- No single well-established type of privacy properties in formal methods
- Most common type here: Non-interference
- Lots of application areas:
  - Secure operating systems [De76,De77]
  - Confinement: trusted program leaks information through covert channels
  - Renewed importance with extensible systems: applets, kernel extensions, mobile agents, etc.
Some Prior Approaches

Non-probabilistic Reactive systems: [Many]
- Based on process calculi
- Definitions are the main issue, different types of non-interference.
- Main problem here: refinement

Probabilistic Reactive systems [Gr92]
- Gray‘s definition „Probabilistic Non-Interference“ stands out
  - For all high-level environment behaviours same probability distribution of the low-events.
  - Perfect fulfillment only, not yet suited for real cryptography → introduce error probabilities, etc.
## Prior work (cont’d)

<table>
<thead>
<tr>
<th>Non-Interference</th>
<th>Deterministic</th>
<th>Non-deterministic</th>
<th>Probabilistic</th>
<th>Cryptographic</th>
</tr>
</thead>
<tbody>
<tr>
<td>GM 82</td>
<td>Many</td>
<td>Gray 92</td>
<td>New</td>
<td></td>
</tr>
</tbody>
</table>
Cryptographic Non-Interference

Want to express: No information can flow from H to L

Idea: Whatever H does, L will not recognize it

+ Now error probabilities, computational restrictions
+ “Guessing a bit“ is a typical concept in cryptography
  → Closely related to cryptographic definitions
Preservation under Simulatability

• **Preservation Theorem (Informal):**
  Whenever an abstraction fulfills a cryptographic non-interference requirement, then every secure implementation of it also fulfills this requirement.

• **Formally:**
  \[
  \text{Sys}_1 \geq \text{Sys}_2 \land \text{Sys}_2 \models \text{NIReq}_{H,L} \Rightarrow \text{Sys}_1 \models \text{NIReq}_{H,L}
  \]
Cryptographic Non-Interference (Intransitive)
A Scenario for Intransitive Non-Interference
<table>
<thead>
<tr>
<th></th>
<th>Deterministic</th>
<th>Non-deterministic</th>
<th>Probabilistic</th>
<th>Cryptographic</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Non-Interference</strong></td>
<td>GM 82</td>
<td>Many</td>
<td>Gray 92</td>
<td>New</td>
</tr>
<tr>
<td><strong>Intransitive</strong></td>
<td>GM 84</td>
<td>Rushby 92, Pinsky 95, RG 99, SRS+ 00</td>
<td>New</td>
<td>New</td>
</tr>
</tbody>
</table>
Definition 1: Blocking Non-Interference

Secretary can prevent the flow

∀ Bad ∀ CEO ∃ Sec: Bad ↛ CEO

all poly-time

\[ \text{Prob}(b^* = b :: r \leftarrow \text{run}_{conf}; b := r\lceil b_{in} \ldots; b^* := r\lceil b_{out}) \leq \frac{1}{2} + \varepsilon \]

\[ \left\{ \begin{array}{ll} 0 & \text{Small} \\ \text{Negl} & \end{array} \right. \]
Definition 2: Recognition Non-interference

Secretary sees what’s going on

\[ \forall \text{Bad} \forall \text{CEO} \forall \text{Sec} \exists D \]

CEO gets \( b \) \( \Rightarrow \) Sec gets \( b \).
Arbitrary Flow Graphs

∀ Bad ∀ CEO ∀ cuts ∃ Cut-Distinguisher
Preservation under Simulatibility

Theorem:

\[ \text{Sys} \xrightarrow{\text{sec}} \text{IdealSys} \]

Diagram:

- **Bad** \(\xrightarrow{\text{Sec}}\) **CEO**
- **Bad** \(\xrightarrow{\text{Sec}}\) **CEO**
Implementation with Cryptographic Firewall

Filtering rules

Secure channels

Ideal Firewall

Prove recognition NI
Michael Backes
Saarland University, Germany
joint work with Birgit Pfitzmann and Michael Waidner

Secure Reactive Systems, Day 4:
Justifying Symbolic Abstractions of Cryptography

Tartu, 03/02/06
Recall the RS Framework

• Precise system model allowing cryptographic and abstract operations
• Reactive simulatability with composition theorem
• Preservation theorems for security properties
• Concrete pairs of idealizations and secure realizations
• Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support
• Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.
• Detailed Proofs (Poly-time, cryptographic bisimulations with static information flow analysis, …)
Recall the RS Framework

- Precise system model allowing cryptographic and abstract operations
- Reactive simulatability with composition theorem
- Preservation theorems for security properties
- Concrete pairs of idealizations and secure realizations
- **Sound symbolic abstractions (Dolev-Yao models) that are suitable for tool support**
- Sound security proofs of security protocols: NSL, Otway-Rees, iKP, etc.
- Detailed Proofs (Poly-time, cryptographic bisimulations with static information flow analysis, ... )
Automatic Proofs of Security
Why Formal Methods?

- Automation if
  - Repetitive
  - Tedious
  - Prone to human errors
  - Critical application
- A top candidate: Distributed protocols
- Security variants for 20 years
Protocol Proof Tools

- HOL Provers
  - Theory 1
  - Theory n
  - Almost anything
  - Much human interaction

- Special security provers
  - Special logic fragments for security
  - Approximations: correct, not complete

- Model Checkers
  - ∞ state
  - Data indep/
  - Fully automatic
  - State exploration
Automating Security Protocol Proofs

• Even simple protocol classes & properties undecidable
  • Robust protocol design helps
• Full arithmetic is out
• Probability theory just developing

So how do current tools handle cryptography?
Dolev-Yao Model

- Idea [DY81]
  - Abstraction as term algebras, e.g., $D_x(E_x(E_x(m)))$
  - Cancellation rules, e.g., $D_xE_x = \varepsilon$
- Well-developed proof theories
  - Abstract data types
  - Equational 1st-order logic
- Important for security proofs:
  - Inequalities! (Everything that cannot be derived.)
  - Known as “initial model”

Important goal: Justify or replace
Dolev-Yao Model – Variants [Ours]

- **Operators and equations**  [EG82, M83, EGS85 ...]
  - pub enc, sym enc, nonce, payload, pairing, sigs, ...
  - Inequalities assumed across operators!
- **Untyped or typed**
- **Destructors explicit or implicit**
- **Abstraction from probabilism**
  - Finite selection, counting, multisets
- **Surrounding protocol language**
  - Special-purpose, CSP, pi calculus, ... [any]
The BPW Model
(Ideal Dolev-Yao Style Library)
**Dolev-Yao-style Crypto Abstractions**

- **Recall:** Term algebra, inequalities
- **Major tasks:**
  - Represent ideal and real library in the same way to higher protocols
  - Prevent honest users from stupidity with real crypto objects, but don’t restrict adversary
    - E.g., sending a bitstring that’s almost a signature
  - What imperfections are tolerable / must be allowed?
Ideal Cryptographic Library

Term 1  Term 2  Term 3  Not globally known

For U:  $T_{u,1}$  $T_{u,2}$  $T_{u,3}$
For V:  -  $T_{v,1}$  -
For A:  -  $T_{a,1}$  -

Payloads / test results, handles
Terms?

Commands, payloads, handles
Terms?

No crypto outputs! Deterministic!

Animals: U, A, V

TH

pk

E

pk

TH

m

E

pk

TH

m
Ideal Cryptographic Library (2)

For U:
- $T_{u,1}$
- $T_{u,2}$
- $T_{u,3}$

For V:
- $T_{v,1}$
- $T_{v,3}$

For A:
- $T_{a,1}$

Term 1 | Term 2 | Term 3 | Term 4
--- | --- | --- | ---
$T_{u,1}$ | $T_{u,2}$ | $T_{u,3}$ | ...

$T_{u,4} \leftarrow \text{encrypt}(T_{u,1}, T_{u,3})$

send($V, T_{u,4}$)

received($U, T_{v,2}$)

get_type($T_{v,2}$)

$T_{v,3} := \text{decrypt}(...)$

U sends $T_{u,4}$ to V, which is encrypted with $T_{u,1}$ and decrypted using $T_{u,3}$.
Main Differences to Dolev-Yao

Tolerable imperfections:

- Lengths of encrypted messages cannot be kept secret
- Adversary may include incorrect messages inside encryptions
- Signature schemes can have memory
- Slightly restricted key usage for symmetric encryption

Most imperfections avoidable for more restricted cases
Main Additions to Given Cryptosystems

- Type tags
- Tagging with keys
- Additional randomization (e.g., needed when correct machines use A’s keys)
Proof of Correct Simulation (1)

Rewrite

Idealize, comp/theorem
Proof of Correct Simulation (2)

Combined system

Probabilistic bisimulations

Reduction proofs for collisions, guesses, forgeries

- With error sets (of runs)
- With info-flow analysis
## Related Work (until first half of 2005)

<table>
<thead>
<tr>
<th>Attacks</th>
<th>Operators</th>
<th>Protocols</th>
<th>Properties</th>
<th>DY version &amp; impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>AR00, AJ01, L01</td>
<td>Passive</td>
<td>1 (pke or ske)</td>
<td>differs</td>
<td>Equivalences</td>
</tr>
<tr>
<td>BPW02, BPW03, BP04</td>
<td>Active</td>
<td>Many</td>
<td>Arbitrary</td>
<td>Simulatability, ⇒ Int., non-interf, now nonce, key &amp; payload secrecy</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>More complex but see L05, BB06</td>
</tr>
<tr>
<td>MW04</td>
<td>Active</td>
<td>1 (pke)</td>
<td>Restricted</td>
<td>Integrity</td>
</tr>
<tr>
<td>L04</td>
<td>Active</td>
<td>1 (ske)</td>
<td>Restricted</td>
<td>Equivalences</td>
</tr>
<tr>
<td>CW05</td>
<td>Active</td>
<td>pke, sig</td>
<td>Restricted</td>
<td>Nonce secrecy</td>
</tr>
<tr>
<td>CH05</td>
<td>Active</td>
<td>1 (pke)</td>
<td>Restricted</td>
<td>Key secrecy</td>
</tr>
</tbody>
</table>

All simple ones come with tool: Specific for “equivalences”, any standard DY tool otherwise.
New General Framework for Symbolic Analysis

Automata framework

Different Logics

Lemmas and Theories

Our Sound

DY-Model

Protocol Instantiations

[Diagram showing the relationships between various components such as protocols, CLWpcProps, CryptoLib, EvaluationLogic, TemporalLogic, StateExceptMonad, TransitionSystems, ProtoVerifTools, CLVerifTools, CLInvariants, CLInvariantsII, CLInvariantsIII, NISLVerifTools, VerifToolbox, and ProtoVerifTools]