Automated Reasoning for Security Protocol Analysis

The AVISPA Project

Luca Viganò
Department of Computer Science
ETH Zurich
Information Security — Past

Security primarily a military concern.
Information Security Present

- The world is distributed:
  - Our basic infrastructures are increasingly based on networked information systems.
  - Business, finance, communication, energy distribution, transportation, entertainment...
- Protocols essential to developing networked services and new applications.
- Security errors in protocol design are costly.

**Money**: security updates are costing hundreds of millions $/€.
**Time**: protocols are delayed by years.
**Acceptance**: eroding confidence in Internet Security and new applications.
Our basic infrastructures are increasingly based on networked information systems. Business, finance, communication, energy distribution, transportation, entertainment...

Alice → Bob@Bank: “Transfer $100 to account X”
Bob@Bank → Alice: “Transfer carried out”

How does Bob know that he is really speaking with Alice? How does Bob know Alice just said it? Confidentiality, integrity, accountability, non-repudiation, privacy...?
Information Security Present

- The number and scale of new security protocols under development is out-pacing the human ability to rigorously analyze and validate them.

- To speed up the development of the next generation of security protocols and to improve their security, it is of utmost importance to have

  - tools that support the formal analysis of security protocols
  - by either finding flaws or establishing their correctness.

- Optimally, these tools should be completely automated, robust, expressive, and easily usable, so that they can be integrated into the protocol development and standardization processes.
Road Map

- Introduction.

- Security protocols.
  - Formal Protocol Analysis.
  - OFMC (& the AVISPA Tool) in more detail.
  - Conclusions.
Building Blocks for Security Protocols

**Cryptographic Procedures:** encryption of messages.

\[ \{\{M\}_K\}_K^{-1} = M \]

**(Pseudo-)Random Number Generators:** to generate “nonces”, e.g. for “challenge/response”.

**Protocols:** recipe for exchanging messages.

Steps like: \( A \) sends \( B \) her name together with the message \( M \).

The pair \( \{A, M\} \) is encrypted with \( B \)'s public key.

\[ A \rightarrow B : \{A, M\}_{K_B} \]
An Authentication Protocol

The Needham-Schroeder Public Key Protocol (NSPK):

1. $A \rightarrow B : \{NA, A\}_{KB}$
2. $B \rightarrow A : \{NA, NB\}_{KA}$
3. $A \rightarrow B : \{NB\}_{KB}$

Goal: mutual authentication. Translation:

- $\{NA, A\}_{KB}$
  - “This is Alice and I have chosen a nonce $NA.$”

- $\{NA, NB\}_{KA}$
  - “Here is your nonce $NA.$ Since I could read it, I must be Bob. I also have a challenge $NB$ for you.”

- $\{NB\}_{KB}$
  - “You sent me $NB.$ Since only Alice can read this and I sent it back, you must be Alice.”
An Authentication Protocol

The Needham-Schroeder Public Key Protocol (NSPK):

1. $A \rightarrow B : \{NA, A\}_{KB}$
2. $B \rightarrow A : \{NA, NB\}_{KA}$
3. $A \rightarrow B : \{NB\}_{KB}$

Goal: mutual authentication. Translation:

- “This is Alice and I have chosen a nonce $NA$.”
- “Here is your nonce $NA$. Since I could read it, I must be Bob. I also have a challenge $NB$ for you.”
- “You sent me $NB$. Since only Alice can read this and I sent it back, you must be Alice.”

NSPK proposed in 1970s and used for decades, until...
An Authentication Protocol

The Needham-Schroeder Public Key Protocol (NSPK):

1. \( A \rightarrow B : \{NA, A\}_{KB} \)
2. \( B \rightarrow A : \{NA, NB\}_{KA} \)
3. \( A \rightarrow B : \{NB\}_{KB} \)

Goal: mutual authentication. Translation:

- “This is Alice and I have chosen a nonce \( NA \).”
- “Here is your nonce \( NA \). Since I could read it, I must be Bob. I also have a challenge \( NB \) for you.”
- “You sent me \( NB \). Since only Alice can read this and I sent it back, you must be Alice.”

NSPK proposed in 1970s and used for decades, until...

Protocols are typically small and convincing...
An Authentication Protocol

The Needham-Schroeder Public Key Protocol (NSPK):

1. \( A \rightarrow B : \{NA, A\}_{KB} \)
2. \( B \rightarrow A : \{NA, NB\}_{KA} \)
3. \( A \rightarrow B : \{NB\}_{KB} \)

Goal: mutual authentication. Translation:

- \( \{NA, A\}_{KB} \) → \( \text{“This is Alice and I have chosen a nonce } NA.\text{”} \)
- \( \{NA, NB\}_{KA} \) → \( \text{“Here is your nonce } NA. \text{ Since I could read it, I must be Bob. I also have a challenge } NB \text{ for you.”} \)
- \( \{NB\}_{KB} \) → \( \text{“You sent me } NB. \text{ Since only Alice can read this and I sent it back, you must be Alice.”} \)

NSPK proposed in 1970s and used for decades, until...
Protocols are typically small and convincing... and often wrong!
How to at Least Tie Against a Chess Grandmaster
How to at Least Tie Against a Chess Grandmaster
Man-in-the-Middle Attack

\[ A \rightarrow B : \{NA, A\}^{K_B} \]
\[ B \rightarrow A : \{NA, NB\}^{K_A} \]
\[ A \rightarrow B : \{NB\}^{K_B} \]

NSPK #1

NSPK #2
Man-in-the-Middle Attack

\[ A \rightarrow B : \{NA, A\}_{KB} \]
\[ B \rightarrow A : \{NA, NB\}_{KA} \]
\[ A \rightarrow B : \{NB\}_{KB} \]
Man-in-the-Middle Attack

\[
\begin{align*}
A \to B : \{NA, A\}_K^B \\
B \to A : \{NA, NB\}_K^A \\
A \to B : \{NB\}_K^B
\end{align*}
\]
Man-in-the-Middle Attack

\[ A \rightarrow B : \{NA, A\}_{KB} \]
\[ B \rightarrow A : \{NA, NB\}_{KA} \]
\[ A \rightarrow B : \{NB\}_{KB} \]
Man-in-the-Middle Attack

\[ A \rightarrow B : \{ NA, A \}_{KB} \]
\[ B \rightarrow A : \{ NA, NB \}_{KA} \]
\[ A \rightarrow B : \{ NB \}_{KB} \]
Man-in-the-Middle Attack

\[
\begin{align*}
A \rightarrow B & : \{NA, A\}_{KB} \\
B \rightarrow A & : \{NA, NB\}_{KA} \\
A \rightarrow B & : \{NB\}_{KB}
\end{align*}
\]
Luca Viganò

**Man-in-the-Middle Attack**

\[ A \rightarrow B : \{ NA, A \}_{KB} \]
\[ B \rightarrow A : \{ NA, NB \}_{KA} \]
\[ A \rightarrow B : \{ NB \}_{KB} \]

\[ \{ NA, A \}_{KC} \quad \{ NA, NB \}_{KA} \quad \{ NB \}_{KC} \]
\[ \{ NA, A \}_{KB} \quad \{ NA, NB \}_{KA} \quad \{ NB \}_{KB} \]

\[ B \text{ believes he is speaking with } A! \]
What went wrong?

- Problem in step 2.

\[ B \rightarrow A : \{ NA, NB \}_{KA} \]

Agent \( B \) should also give his name: \( \{ NA, NB, B \}_{KA} \).

- Is the improved version now correct?
The NSL Protocol

$A \rightarrow B : \{NA, A\}_{KB}$

$B \rightarrow A : \{NA, NB, B\}_{KA}$

$A \rightarrow B : \{NB\}_{KB}$
The NSL Protocol

$A \rightarrow B : \{NA, A\}_{KB} \quad B \rightarrow A : \{NA, NB, B\}_{KA} \quad A \rightarrow B : \{NB\}_{KB}$
The NSL Protocol

\[ A \rightarrow B : \{NA, A\}_{KB} \]
\[ B \rightarrow A : \{NA, NB, B\}_{KA} \]
\[ A \rightarrow B : \{NB\}_{KB} \]
The NSL Protocol

\[ A \rightarrow B : \{ NA, A \}^\text{KB} \]
\[ B \rightarrow A : \{ NA, NB, B \}^\text{KA} \]
\[ A \rightarrow B : \{ NB \}^\text{KB} \]
The NSL Protocol

\[ A \rightarrow B : \{ NA, A \}_B \]
\[ B \rightarrow A : \{ NA, NB, B \}_A \]
\[ A \rightarrow B : \{ NB \}_B \]

\( A \) aborts the protocol execution!
(or ignores the message)
What went wrong?

- Problem in step 2.

\[ B \rightarrow A : \{ NA, NB \}_{K_A} \]

Agent \( B \) should also give his name: \( \{ NA, NB, B \}_{K_A} \).

- Is the improved version now correct?

Yes, it is secure against this attack but what about other kinds of attacks?
What went wrong?

- Problem in step 2.

\[ B \rightarrow A : \{NA, NB\}_{K_A} \]

Agent \( B \) should also give his name: \( \{NA, NB, B\}_{K_A} \).

- Is the improved version now correct?

Yes, it is secure against this attack but what about other kinds of attacks? **Use formal methods!**
Summary

• Security protocols can achieve properties that cryptographic primitives alone cannot offer, e.g. authentication, secrecy, ...

• The example is simple, but the ideas are general.

• Even three liners show how difficult the art of correct design is.

  Let every eye negotiate for itself  
  And trust no agent; for beauty is a witch  
  Against whose charms faith melteth into blood.  
  
  (William Shakespeare, Much ado about nothing)

• Formal analysis of protocols is required.

  However, formal analysis of protocols is nontrivial (even assuming perfect cryptography).
Road Map

- Introduction.
- Security protocols.

☞ Formal Protocol Analysis.

- OFMC (& the AVISPA Tool) in more detail.
- Conclusions.
Formal Modeling and Analysis of Protocols

Standard protocol notation is not formal!

**Goal:** formally model protocols and their properties and provide a mathematically sound means for reasoning about these models.

**Basis:** suitable abstraction of protocols and information flow.

**Analysis:** with formal methods based on mathematics and logic.
Formal Security Protocol Analysis

Security Protocol Analysis

Formal Models

Dolev–Yao (perfect cryptography)

Computational Models

Probabilistic cryptographic view
Cryptographically faithful proofs

Also: semi-formal (engineering) approaches.
Formal Methods for Security Protocol Analysis

- Security Protocol Analysis
  - Formal Models
    - Dolev–Yao (perfect cryptography)
  - Computational Models
    - Probabilistic cryptographic view
    - Cryptographically faithful proofs

Also: semi-formal (engineering) approaches.
Roger Needham & Michael Schroeder

*Using encryption for authentication in large networks of computers (CACM, 1978)*

- Early protocols for key distribution and authentication.
- First mention that formal methods could be useful for assuring protocol correctness; the last sentence of the paper is

  *Finally, protocols such as those developed here are prone to extremely subtle errors that are unlikely to be detected in normal operation. The need for techniques to verify the correctness of such protocols is great, and we encourage those interested in such problems to consider this area.*

The challenge has been taken up by many researchers!
Model by Dolev & Yao (& Even & Karp; early ’80s)

A protocol is an algebraic system operated by the intruder.

- Cryptoalgorithms behave like black-boxes that obey a limited set of algebraic properties (e.g. encryption and decryption operations cancel each other out $E_X(D_X(M)) = D_X(E_X(M)) = M$).
- Perfect cryptography (all $D_X$ private, decryption only with key, ...).
- The intruder can
  - read all traffic,
  - modify, delete and create traffic,
  - perform cryptographic operations available to legitimate users of the system,
  - and is in league with a subset of “corrupt” principals.

* A friend’s just an enemy in disguise. You can’t trust nobody. (C. Dickens, Oliver Twist)

- An arbitrary number of principals.
- Protocol executions may be interleaved (concurrent).

With some modifications, this is the most commonly used intruder model today for formal protocol analysis.
Semi-automated Reasoning for Security Protocol Analysis

Security Protocol Analysis

Formal Models

Dolev–Yao (perfect cryptography)

Semi-automated

Belief Logics

interleaving trace models
state-based models

Inductive Proofs

Model Checking

Computational Models

Probabilistic cryptographic view
Cryptographically faithful proofs
Interleaving Trace Models

- Modeling idea: model possible communication events.

\[ A \rightarrow B : M_1 \]

\[ B \rightarrow A : M_2 \]

\[ \vdots \]
Interleaving Trace Models

- Modeling idea: model possible communication events.

\[ A \to B : M_1 \]
\[ C \to D : P_1 \]
\[ B \to A : M_2 \]
\[ C \to D : P_2 \]
\[ \vdots \]
Interleaving Trace Models

- Modeling idea: model possible communication events.

\[
\begin{align*}
A \to B : M_1 & \quad A \to C : P_1 & \quad A \to B : M_1 \\
C \to D : P_1 & \quad A \to B : M_1 & \quad B \to A : M_2 \\
i \to A : M_2 & \quad \text{or} \quad B \to A : M_2 & \quad \text{or} \quad C \to D : P_1 \\
C \to D : P_2 & \quad C \to A : P_2 & \quad i \to C : P_1 \\
\vdots & \quad \vdots & \quad \vdots
\end{align*}
\]

- A trace is a sequence of events.

- Trace-based interleaving semantics:
  - A protocol denotes a set of traces.
  - Interleavings of (partial) protocol runs and intruder messages.

- Also: state-based models.

- Properties correspond to sets of traces/states, e.g. \(\text{secret}(NA, \{A, B\})\).
Modeling the Dolev-Yao Intruder

For a set $M$ of messages, let $\mathcal{DY}(M)$ (for Dolev-Yao) be the smallest set closed under the following generation ($G$) and analysis ($A$) rules, where $\{m_2\}_{m_1}$ denotes asymmetric encryption, $\{|m_2|\}_{m_1}$ and symmetric encryption:

$$
\frac{m \in M}{m \in \mathcal{DY}(M)} G_{\text{axiom}} \quad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\langle m_1, m_2 \rangle \in \mathcal{DY}(M)} G_{\text{pair}}
$$

$$
\frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{m_2\}_{m_1} \in \mathcal{DY}(M)} G_{\text{crypt}} \quad \frac{m_1 \in \mathcal{DY}(M) \quad m_2 \in \mathcal{DY}(M)}{\{|m_2|\}_{m_1} \in \mathcal{DY}(M)} G_{\text{scrypt}}
$$

$$
\frac{\langle m_1, m_2 \rangle \in \mathcal{DY}(M)}{m_i \in \mathcal{DY}(M)} A_{\text{pair}_i} \quad \frac{\{m_2\}_{m_1} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)} A_{\text{scrypt}}
$$

$$
\frac{\{m_2\}_{m_1} \in \mathcal{DY}(M) \quad m_1^{-1} \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)} A_{\text{crypt}} \quad \frac{\{m_2\}_{m_1^{-1}} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)} A_{\text{crypt}^{-1}}
$$
Formal Analysis of Security Protocols

- Challenging as general problem is **undecidable**.

- Several **sources of infinity** in protocol analysis:
  - Unbounded number of possible intruder messages (unbounded message depth).
  - Unbounded number of sessions or protocol steps (and agents).

- Possible approaches:
  - **Falsification** identifies attack traces but does not guarantee correctness.
  - **Verification** proves correctness but is difficult to automate (requires induction and often restrictions).
The State of the Art… “Yesterday”

- Several semi-automated tools have been developed to analyze protocols under the perfect cryptography assumption, but (in most cases) they are limited to small and medium-scale protocols.

  - For example, Clark/Jacob protocol library: NSPK, NSSK, Otway-Rees, Yahalom, Woo-Lam, Denning-Sacco, ...
The State of the Art… “Yesterday”

- Several semi-automated tools have been developed to analyze protocols under the perfect cryptography assumption, but (in most cases) they are limited to small and medium-scale protocols.

- For example, Clark/Jacob protocol library: NSPK, NSSK, Otway-Rees, Yahalom, Woo-Lam, Denning-Sacco, ...

- Scaling up to large-scale Internet security protocols is a considerable scientific and technological challenge.
The State of the Art… Today and Tomorrow

• Some tools (AVISPA, ProVerif, Casper/FDR, Scyther, NRL, ...) are taking up this challenge and
  ▶ developing languages for specifying industrial-scale security protocols and their properties,
  ▶ advancing analysis techniques to scale up to this complexity.

• These technologies are migrating to companies and standardization organizations.

• Also: extensions to
  ▶ even more complex protocols and properties (group protocols, broadcast, ad-hoc networks, emerging properties, etc.)
  ▶ Web Services,
  ▶ and so on.
The AVISPA Tool

A state-of-the-art (for level of scope and performance), integrated environment for the automatic analysis and validation of Internet security protocols.

- A push-button integrated tool supporting the protocol designer in the debugging and validation of protocols.
  - Provides a role-based (& TLA-based) specification language for security protocols, properties, channels and intruder models.
  - Integrates different back-ends implementing a variety of state-of-the-art automatic analysis techniques.
- Assessed on a large collection of practically relevant, industrial protocols (the AVISPA Library).
- Large user base (the AVISPA users mailing list).
The Web Interface  www.avispa-project.org
The AVISPA Tool: Architecture

High-Level Protocol Specification Language (HLPSL)

Translator
HLPSL2IF

Intermediate Format (IF)

On-the-fly Model-Checker
OFMC

CL-based Attack Searcher
AtSe

SAT-based Model-Checker
SATMC

Tree Automata-based Protocol Analyser
TA4SP

Output Format (OF)
The AVISPA Tool: the Back-Ends

From protocol falsification to abstraction-based verification.

The On-the-fly Model-Checker (OFMC) employs several symbolic techniques to explore the state space in a demand-driven way.

CL-AtSe (Constraint-Logic-based Attack Searcher) applies constraint solving with simplification heuristics and redundancy elimination techniques.

The SAT-based Model-Checker (SATMC) builds a propositional formula encoding all the possible attacks (of bounded length) on the protocol and feeds the result to a SAT solver.

TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols) approximates the intruder knowledge by using regular tree languages and rewriting to produce under and over approximations.
The AVISPA Tool and the AVISPA Library: Results

- Beyond Clark/Jacob (few seconds for entire library, with new attacks).

- A library of 384 problems from 79 protocols that have recently been or are currently being standardized by the IETF (problem = protocol + property).

- Analysis:
  - 215 problems in 87 min.
  - Several new attacks (e.g. H.530 protocol).

<table>
<thead>
<tr>
<th>Protocol</th>
<th>#P</th>
<th>P</th>
<th>A</th>
<th>T</th>
<th>OFMC</th>
<th>A</th>
<th>CL-atse</th>
<th>P</th>
<th>A</th>
<th>T</th>
<th>SATMC</th>
</tr>
</thead>
<tbody>
<tr>
<td>UMTS_AKA</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0,02</td>
<td>0</td>
<td>0,01</td>
<td>3</td>
<td>0</td>
<td>0,11</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>AAA-MobileIP</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>0,75</td>
<td>0</td>
<td>0,20</td>
<td>7</td>
<td>0</td>
<td>1,32</td>
<td>0,01</td>
<td></td>
</tr>
<tr>
<td>ISO-PK1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0,02</td>
<td>0</td>
<td>0,00</td>
<td>1</td>
<td>1</td>
<td>0,05</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>ISO-PK2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0,05</td>
<td>0</td>
<td>0,00</td>
<td>1</td>
<td>0</td>
<td>1,62</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>ISO-PK3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0,04</td>
<td>2</td>
<td>0,01</td>
<td>2</td>
<td>2</td>
<td>0,27</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>ISO-PK4</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,54</td>
<td>0</td>
<td>0,03</td>
<td>2</td>
<td>0</td>
<td>1,153</td>
<td>1,16</td>
<td></td>
</tr>
<tr>
<td>LPD-MSR</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,02</td>
<td>2</td>
<td>0,02</td>
<td>2</td>
<td>2</td>
<td>0,17</td>
<td>0,02</td>
<td></td>
</tr>
<tr>
<td>LPD-IMSR</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,00</td>
<td>2</td>
<td>0,01</td>
<td>2</td>
<td>0</td>
<td>0,43</td>
<td>0,01</td>
<td></td>
</tr>
<tr>
<td>CHAPv2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0,32</td>
<td>3</td>
<td>0,01</td>
<td>3</td>
<td>0</td>
<td>0,55</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>EKE</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>0,19</td>
<td>3</td>
<td>0,04</td>
<td>3</td>
<td>2</td>
<td>0,22</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>TLS</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2,20</td>
<td>0</td>
<td>0,32</td>
<td>3</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DHCP-delayed</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,07</td>
<td>2</td>
<td>0,00</td>
<td>2</td>
<td>0</td>
<td>0,19</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>Kerb-Cross-Realm</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>11,86</td>
<td>8</td>
<td>0,41</td>
<td>8</td>
<td>0</td>
<td>113,60</td>
<td>1,69</td>
<td></td>
</tr>
<tr>
<td>Kerb-Ticket-Cache</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>2,43</td>
<td>6</td>
<td>0,38</td>
<td>6</td>
<td>0</td>
<td>495,66</td>
<td>7,75</td>
<td></td>
</tr>
<tr>
<td>Kerb-V</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>3,08</td>
<td>8</td>
<td>0,42</td>
<td>8</td>
<td>0</td>
<td>139,56</td>
<td>2,95</td>
<td></td>
</tr>
<tr>
<td>Kerb-Forwardable</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>30,34</td>
<td>6</td>
<td>0,89</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Kerb-PKINIT</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>4,41</td>
<td>7</td>
<td>0,64</td>
<td>7</td>
<td>0</td>
<td>640,33</td>
<td>11,65</td>
<td></td>
</tr>
<tr>
<td>Kerb-preauth</td>
<td>7</td>
<td>7</td>
<td>0</td>
<td>1,86</td>
<td>7</td>
<td>0,62</td>
<td>7</td>
<td>0</td>
<td>373,72</td>
<td>2,57</td>
<td></td>
</tr>
<tr>
<td>CRAM-MD5</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,71</td>
<td>2</td>
<td>0,74</td>
<td>2</td>
<td>0</td>
<td>0,40</td>
<td>0,00</td>
<td></td>
</tr>
<tr>
<td>PKB</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0,25</td>
<td>1</td>
<td>0,01</td>
<td>1</td>
<td>1</td>
<td>0,34</td>
<td>0,02</td>
<td></td>
</tr>
<tr>
<td>PKB-fix</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>4,06</td>
<td>2</td>
<td>0,42</td>
<td>2</td>
<td>0</td>
<td>0,86</td>
<td>0,02</td>
<td></td>
</tr>
<tr>
<td>SRP_siemens</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>2,86</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>EKE2</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>0,16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>SPEKE</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>3,11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IKEv2-CHILD</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1,19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IKEv2-DS</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>5,22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IKEv2-DSx</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>42,56</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IKEv2-MAC</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>8,03</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>IKEv2-MACx</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>40,54</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>h.530</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>0,64</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>h.530-fix</td>
<td>3</td>
<td>3</td>
<td>0</td>
<td>4,278</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>lipkey-spkm-known</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0,23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>lipkey-spkm-unknown</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>7,33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>

Also: TA4SP establishes in a few minutes that a number of protocols (EKE, EKE2, IKEv2-CHILD, IKEv2-MAC, TLS, UMTS_AKA, MS-ChapV2) guarantee secrecy.
A Simple Example: the H.530 Protocol

1. MT -> VGK : MT, VGK, NIL, CH1, G^DHX, 
   F(ZZ, MT, VGK, NIL, CH1, G^DHX)

2. VGK -> AuF : MT, VGK, NIL, CH1, G^DHX, 
   F(ZZ, MT, VGK, NIL, CH1, G^DHX), 
   VGK, G^DHX XOR G^DHY, 
   F(ZZ, MT, VGK, NIL, CH1, G^DHX), 
   VGK, G^DHX XOR G^DHY)

3. AuF -> VGK : VGK, MT, F(ZZ, VGK), 
   F(ZZ, G^DHX XOR G^DHY), 
   F(ZZ, MT, VGK, F(ZZ, VGK), 
   F(ZZ, G^DHX XOR G^DHY))

4. VGK -> MT : VGK, MT, CH1, CH2, G^DHY, 
   F(ZZ, G^DHX XOR G^DHY), 
   F(ZZ, VGK), 
   F((G^DHX)^DHY, VGK, MT, CH1, CH2, G^DHY, 
   F(ZZ, G^DHX XOR G^DHY), F(ZZ, VGK))

5. MT -> VGK : MT, VGK, CH2, CH3, 
   F((G^DHX)^DHY, MT, VGK, CH2, CH3)

6. VGK -> MT : VGK, MT, CH3, CH4, 
   F((G^DHX)^DHY, VGK, MT, CH3, CH4)

Protocol proposed (and patented) by Siemens. Modeling time, ca. 1 day. Analysis time, ca. 1 second. New patent filed, ca. 1 year.
Summary: the Present and the Future

• The AVISPA Tool is a state-of-the-art, integrated environment for the automatic validation of Internet security protocols.

AVISPA package (& web-interface): www.avispa-project.org

• Current work:
  ► Extending the AVISPA library with further protocols and properties.
  ► Unbounded verification using abstractions.
  ► Algebraic properties.
  ► Guessing intruder and other intruder models (and channels).
  ► Web-services.
  ► Combining cryptographic and formal proof techniques.

• Integration of other tools via HLPSL/IF (e.g. translator from HLPSL to Applied Pi Calculus to then apply ProVerif).
Road Map

- Introduction.
- Security protocols.
- Formal Protocol Analysis.
- OFMC (& the AVISPA Tool) in more detail.
- Conclusions.
Formal Analysis of Security Protocols

- Challenging as general problem is undecidable.

- Several sources of infinity in protocol analysis:
  - Unbounded number of possible intruder messages (unbounded message depth).
  - Unbounded number of sessions or protocol steps (and agents).

- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate (requires induction and often restrictions).
Formal Analysis of Security Protocols

- Challenging as general problem is **undecidable**.

- Several **sources of infinity** in protocol analysis:
  - Unbounded **number of possible intruder messages** (unbounded **message depth**).
  - Unbounded **number of sessions** or protocol steps (and **agents**).

- **OFMC**: an On-the-Fly Model-Checker for Security Protocols.
  - Falsification.
  - (Bounded and abstraction-based unbounded) verification.

Symbolic techniques to reduce the search space without excluding or introducing attacks.
Graphical Overview of some Symbolic Reductions

- The Lazy Intruder

- Compressions

- Symbolic Sessions

- Constraint Differentiation

- Abstractions (data and control)
Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.

2. Concurrency: number of parallel sessions executed by honest agents.
Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
   - No bound on the messages the intruder can compose.
   - Lazy Intruder: symbolic representation of intruder.
     “Often just as if there were no intruder!”

2. Concurrency: number of parallel sessions executed by honest agents.
Road Map

- Introduction.
- Security protocols.
- Formal Protocol Analysis.

☞ OFMC (& the AVISPA Tool) in more detail.
  - Lazy Intruder.
  - Constraint Differentiation.

- Conclusions.
Protocol Model

- Protocol modeled as an **infinite-state transition system**.
  - States: local states of honest agents and current knowledge of the intruder.
  - Transitions: actions of the honest agents and the intruder.

- The **Dolev-Yao intruder**:
  - Controls the entire network.
  - Perfect cryptography.
  - Unbounded composition of messages.

- **Security properties**: attack predicates on states.

- Also: protocol-independent declarations (operator symbols, algebraic properties, intruder model,...)
Lazy Intruder: Overview

- Many different approaches based on different formalisms, e.g.:
  - Process calculi (e.g. [Amadio & Lugiez], [Boreale & Buscemi])
  - Strand spaces (e.g. [Millen & Shmatikov], [Corin & Etalle])
  - Rewriting (e.g. [Chevalier & Vigneron], [BMV])

- But they all share the same basic ideas:
  - Avoid the naïve enumeration of possible messages the intruder can send.
  - Use variables and constraints for messages sent by the intruder.
The Lazy Intruder: Idea

1. $A \rightarrow B : M, A, B, \{N A, M, A, B\} K_{AS}$
The Lazy Intruder: Idea

1. \( i(A) \rightarrow B : M, A, B, \{NA, M, A, B\}^{K_{AS}} \)
The Lazy Intruder: Idea

1. \( i(A) \rightarrow B : M, A, B, \{\|NA, M, A, B\|\}_{KAS} \)

Which concrete value is chosen for these parts makes a difference only later.
The Lazy Intruder: Idea

1. \( i(A) \rightarrow B : M, A, B, \{N A, M, A, B\} \)_{K_{AS}}

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

1. \( i(A) \rightarrow B : x_1, x_2, B, x_3 \) from(\( \{x_1, x_2, x_3\} \), \( IK \))

\( IK \): current Intruder Knowledge
The Lazy Intruder: Idea

1. $i(A) \rightarrow B : M, A, B, \{NA, M, A, B\}^{KAS}$

Which concrete value is chosen for these parts makes a difference only later.

Idea: postpone this decision.

1. $i(A) \rightarrow B : x_1, x_2, B, x_3 \quad \text{from} \{x_1, x_2, x_3\}, IK$

$IK$: current Intruder Knowledge

$from$-constraints are evaluated in a demand-driven way, hence lazy intruder.
The Lazy Intruder: Formally

- Constraints of the lazy intruder:

  \[ \text{from}(T, IK) \]

  \[ [\text{from}(T, IK)] = \{\sigma \mid \text{ground}(T\sigma \cup IK\sigma) \land (T\sigma \subseteq DY(IK\sigma))\} \]

  where \( DY(IK) \) is the closure of \( IK \) under Dolev-Yao rules.

- Semantics hence relates \textit{from}-constraints to the Dolev-Yao model.
The Lazy Intruder: Formally

- Constraints of the lazy intruder:

\[ \text{from}(T, IK) \]

- \[ \mathcal{E}[\text{from}(T, IK)] = \{ \sigma \mid \text{ground}(T\sigma \cup IK\sigma) \land (T\sigma \subseteq \mathcal{D}\mathcal{Y}(IK\sigma)) \} \]

  where \( \mathcal{D}\mathcal{Y}(IK) \) is the closure of \( IK \) under Dolev-Yao rules.

- Semantics hence relates \textit{from}-constraints to the Dolev-Yao model.

- Theorem. Satisfiability of (well-formed) \textit{from}-constraints is decidable.

- A restriction on the depth of messages is not necessary.

- Non-atomic keys can easily be handled.
Integration: Symbolic Transition System

- **Symbolic state** = term with variables + constraint set

- $\llbracket (t, C) \rrbracket = \{ t\sigma \mid \sigma \in \llbracket C \rrbracket \}$ (a set of ground states).

- Two layers of search:
  
  **Layer 1:** search in the symbolic state space
  
  **Layer 2:** constraint reduction
We allow messages to contain variables and employ unification.

\[
\begin{align*}
A \rightarrow B : & \quad \{ NA, A \}_{KB} \\
B \rightarrow A : & \quad \{ NA, NB \}_{KA} \\
A \rightarrow B : & \quad \{ NB \}_{KB}
\end{align*}
\]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[ a \rightarrow I : \{na, a\}_{K_I} \]

\[ A \rightarrow B : \{NA, A\}_{K_B} \]
\[ B \rightarrow A : \{NA, NB\}_{K_A} \]
\[ A \rightarrow B : \{NB\}_{K_B} \]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[ a \rightarrow I : \{na, a\}_{K_I} \]
\[ I \rightarrow b : X_1 \]

\[ A \rightarrow B : \{NA, A\}_{K_B} \]
\[ B \rightarrow A : \{NA, NB\}_{K_A} \]
\[ A \rightarrow B : \{NB\}_{K_B} \]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[ a \rightarrow I : \{na, a\}_{K_I} \]
\[ I \rightarrow b : X_1 \]
\[ X_1 = \{X_2, X_3\}_{K_b} \]
\[ b \rightarrow I : \{X_2, nb\}_{K_{X_3}} \]
\[ A \rightarrow B : \{NA, A\}_{K_B} \]
\[ B \rightarrow A : \{NA, NB\}_{K_A} \]
\[ A \rightarrow B : \{NB\}_{K_B} \]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[
\begin{align*}
\text{a} & \rightarrow I : \{na, a\}_{K_I} \\
I & \rightarrow b : \{X_2, X_3\}_{K_b} \\
b & \rightarrow I : \{X_2, nb\}_{K_{X_3}} \\
I & \rightarrow a : \{X_2, nb\}_{K_{X_3}} \\
X_1 & = \{X_2, X_3\}_{K_b} \\
A & \rightarrow B : \{NA, A\}_{K_B} \\
B & \rightarrow A : \{NA, NB\}_{K_A} \\
A & \rightarrow B : \{NB\}_{K_B}
\end{align*}
\]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[ a \rightarrow I : \{na, a\}_{K_I} \]
\[ I \rightarrow b : \{X_2, X_3\}_{K_b} \quad X_1 = \{na, a\}_{K_b} \]
\[ b \rightarrow I : \{X_2, nb\}_{K_{X_3}} \]
\[ I \rightarrow a : \{X_2, nb\}_{K_{X_3}} \]
\[ a \rightarrow I : \{nb\}_{K_I} \quad X_2 = na, \quad X_3 = a \]
\[ A \rightarrow B : \{NA, A\}_{K_B} \]
\[ B \rightarrow A : \{NA, NB\}_{K_A} \]
\[ A \rightarrow B : \{NB\}_{K_B} \]
NSPK and the Lazy Intruder

We allow messages to contain variables and employ unification.

\[
\begin{align*}
  a &\rightarrow I : \{na, a\}_{K_I} \\
  I &\rightarrow b : \{na, a\}_{K_b} \\
  b &\rightarrow I : \{na, nb\}_{K_a} \\
  I &\rightarrow a : \{na, nb\}_{K_a} \\
  a &\rightarrow I : \{nb\}_{K_I} \\
  I &\rightarrow b : \{nb\}_{K_b}
\end{align*}
\]

\[
X_1 = \{na, a\}_{K_b}
\]

\[
A \rightarrow B : \{NA, A\}_{K_B}
\]

\[
B \rightarrow A : \{NA, NB\}_{K_A}
\]

\[
A \rightarrow B : \{NB\}_{K_B}
\]

\[
X_2 = na, \ X_3 = a
\]
Road Map

- Introduction.
- Security protocols.
- Formal Protocol Analysis.
  - OFMC (& the AVISPA Tool) in more detail.
    - Lazy Intruder.
    - Constraint Differentiation.
- Conclusions.
Two Key Challenges and their Solutions

Two key challenges of model-checking security protocols:

1. The prolific Dolev-Yao intruder model.
   - No bound on the messages the intruder can compose.
   - Lazy Intruder: symbolic representation of intruder.
     “Often just as if there were no intruder!”

2. Concurrency: number of parallel sessions executed by honest agents.
   - Often addressed using Partial-Order Reduction (POR).
   - POR is limited when using the lazy intruder technique.
   - Constraint Differentiation: general, POR-inspired reduction technique extending the lazy intruder — correct and complete.
**Constraint Differentiation: Idea**

Typical situation: 2 independent actions executable in either order:

\[
\begin{array}{c|c|c|c}
   s & t & IK & C \\
\end{array}
\]
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:

- **i** sends $m_1$ to $a$ and receives $m_2$ from $a$.
- **s** sends $m_2$ to **IK** and receives $(m_1, IK)$ from **C**.

Diagram:

```
  s  t  IK  C

  s1
  t1  IK  C  from(m1, IK)

m1  m2
```
**Constraint Differentiation: Idea**

Typical situation: 2 independent actions executable in either order:

- $s_1$: $i$ sends $m_1$ to $a$ and receives $m_2$ from $a$
  
  - $t_1$: \( \text{IK} \quad C \)
  
  - $m_2$: from($m_1, \text{IK}$)

- $s_2$: $i$ sends $m_3$ to $b$ and receives $m_4$ from $b$
  
  - $t_2$: \( \text{IK} \quad C \)
  
  - $m_2$: from($m_1, \text{IK}$)
  
  - $m_4$: from($m_3, \text{IK} \cup m_2$)
Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:

\[ (where \ t_2 = t_4) \]
**Constraint Differentiation: Idea**

Typical situation: 2 independent actions executable in either order:

\[
\begin{align*}
&i \text{ sends } m_1 \text{ to } a \text{ and } \\
&\quad \text{receives } m_2 \text{ from } a\
&i \text{ sends } m_3 \text{ to } b \text{ and } \\
&\quad \text{receives } m_4 \text{ from } b \\
&i \text{ sends } m_3 \text{ to } b \text{ and } \\
&\quad \text{receives } m_4 \text{ from } b \\
&i \text{ sends } m_1 \text{ to } a \text{ and } \\
&\quad \text{receives } m_2 \text{ from } a
\end{align*}
\]

(where \( t_2 = t_4 \))

Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.
**Constraint Differentiation: Idea**

Typical situation: 2 independent actions executable in either order:

Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.
Constraint Differentiation (1)

- New kind of constraints: $D\text{-from}(T, IK, NIK)$.
- Intuition:
  - Intruder has just learned some new intruder knowledge $NIK$. 
Constraint Differentiation (1)

- New kind of constraints: \( D\text{-from}(T, IK, NIK) \).
- Intuition:
  - Intruder has just learned some new intruder knowledge \( NIK \).
  - All solutions \( \{ \text{from}(T, IK \cup NIK) \} \) are “correct”
• New kind of constraints: \(D\text{-from}(T, IK, NIK)\).

• Intuition:
  
  ▶ Intruder has just learned some new intruder knowledge \(NIK\).

  ▶ All solutions \([\text{from}(T, IK \cup NIK)]\) are “correct” but a solution is interesting only if it requires \(NIK\).

\[
[D\text{-from}(T, IK, NIK)] = [\text{from}(T, IK \cup NIK)] \setminus [\text{from}(T, IK)].
\]
Constraint Differentiation (2)

- \([D\text{-}from(T, IK, NIK)] = [\text{from}(T, IK \cup NIK)] \setminus [\text{from}(T, IK)]\)

- **Theorem.** *Satisfiability of (well-formed) $D$-from constraints is decidable.*

- **Theorem.** \([s_2] \cup [s_4] = [s_2] \cup [s'_4]\)
# Constraint Differentiation: Experimental Results

IKE Aggressive Mode Pre-Shared Key without and with CD: the nodes for each ply of the search tree and search time

<table>
<thead>
<tr>
<th>Mode:</th>
<th>without CD</th>
<th>with CD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario:</td>
<td>[a, b], [a, i]</td>
<td>[a, b], [a, i], [i, a]</td>
</tr>
<tr>
<td>Ply</td>
<td>s1</td>
<td>s2</td>
</tr>
<tr>
<td>-----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
<td>14</td>
</tr>
<tr>
<td>4</td>
<td>17</td>
<td>27</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>53</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td>101</td>
</tr>
<tr>
<td>7</td>
<td>191</td>
<td>483</td>
</tr>
<tr>
<td>8</td>
<td>410</td>
<td>420</td>
</tr>
<tr>
<td>9</td>
<td>720</td>
<td>29783</td>
</tr>
<tr>
<td>10</td>
<td>960</td>
<td>79939</td>
</tr>
<tr>
<td>11</td>
<td>990</td>
<td>201861</td>
</tr>
<tr>
<td>12</td>
<td>990</td>
<td>467533</td>
</tr>
<tr>
<td>13</td>
<td>929500</td>
<td>TO</td>
</tr>
<tr>
<td>14</td>
<td>1583582</td>
<td>TO</td>
</tr>
<tr>
<td>15</td>
<td>2132130</td>
<td>TO</td>
</tr>
<tr>
<td>16</td>
<td>1801800</td>
<td>TO</td>
</tr>
<tr>
<td>17</td>
<td>10531</td>
<td>TO</td>
</tr>
<tr>
<td>18</td>
<td>7857</td>
<td>TO</td>
</tr>
<tr>
<td>19</td>
<td>2371</td>
<td>TO</td>
</tr>
<tr>
<td>Nodes</td>
<td>71</td>
<td>4467</td>
</tr>
<tr>
<td>Time</td>
<td>0.16s</td>
<td>13.66s</td>
</tr>
</tbody>
</table>

IKE Aggressive Mode Pre-Shared Key without CD: the nodes for each ply of the search tree and search time

IKE Aggressive Mode Pre-Shared Key with CD: the nodes for each ply of the search tree and search time
Lazy Intruder and Constraint Reduction
Graphical Overview of some Symbolic Reductions

- The Lazy Intruder
- Compressions
- Symbolic Sessions
- Constraint Differentiation
- Abstractions (data and control)
Road Map

- Introduction.
- Security protocols.
- Formal Protocol Analysis.
- OFMC (& the AVISPA Tool) in more detail.

✍️ Conclusions.
Conclusions and Outlook

- The AVISPA Tool is a state-of-the-art, integrated environment for the automatic validation of Internet security protocols.

**AVISPA package (& web-interface):** [www.avispa-project.org](http://www.avispa-project.org)

- Current work:
  - Extending the AVISPA library with further protocols and properties.
  - Unbounded verification using abstractions.
  - Algebraic properties.
  - Guessing intruder and other intruder models (and channels).
  - Web-services.
  - Combining cryptographic and formal proof techniques.

- Integration of other tools via HLPSL/IF (e.g. translator from HLPSL to Applied Pi Calculus to then apply ProVerif).
Web Services

- **Web Services (WS):** a series of standards that add higher-layer semantics and quality of service to web-based and XML-based communication, in particular among enterprises.

- Structure is far more complex than standard security protocols.
  - Requires model simplifications, approximations, and abstractions (and showing that these do not exclude attacks).

- **Case study:** Secure WS-ReliableMessaging Scenario.
  1. an automated analysis based on symbolic protocol analysis techniques under the assumption of perfect cryptography,
  2. an analysis closer to real cryptography based on explicit cryptographic assumptions on the underlying crypto-algorithms.

Both analyses have positive results: they demonstrate that at the abstraction level of each analysis, the protocol is error-free.

- **Future work:** link the 2 kinds of analysis for WS in the style of previous proofs of soundness of Dolev-Yao models.
Computational Models

Security Protocol Analysis

Formal Models
Dolev–Yao
(perfect cryptography)

Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs
The formal methods and cryptography communities have both developed formal techniques for protocol analysis.

However, they have quite different points of view.

- Cryptographers apply complexity and probability theory to reduce protocol security to security of underlying cryptosystem.
- Security of cryptosystem: an attacker with polynomial computing power can break it only with negligible probability.
- Typically, cryptographic proofs are long and difficult, and error prone (done by hand).

Unsatisfying: standard Dolev-Yao abstraction used in formal methods analysis lacks cryptographic justification.

There are protocols that are secure in the DY model, but become insecure if implemented with some provably secure crypto-primitives.
Closing the Gap

- **Goal**: cryptographically faithful verification of security protocols.
  - Considerable amount of research on tool-supported formal verification using cryptographically sound abstractions.
  - IBM & ETH, Abadi, Bellare, Canetti, Cortier, Mitchell, Rogaway, Scedrov, Warinschi, ....
  - For example: crypto library of Backes, Pfitzmann, Waidner (IBM) (and also Basin and Sprenger, ETH)
  - Real library reflects probabilistic cryptographic view.
  - Ideal library reflects non-probabilistic formal methods view.
  - Procedure:
    * Prove that real library securely implements ideal library.
    * Use ideal library for tool-supported analysis of ideal protocol.
    * Conclude real protocol is secure by preservation results.