On-the-Fly Model Checking of Security Protocols and Web Services

Luca Viganò

Department of Computer Science
University of Verona

Fosad 2009

Joint work with

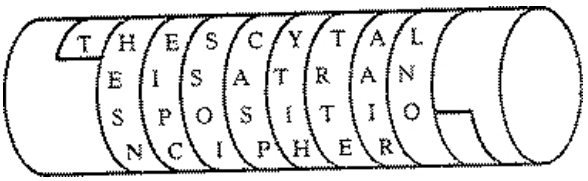
- Sebastian Mödersheim
- David Basin
- Paul Hankes Drielsma
- The AVISPA Project (and the AVISS Project)
- The AVANTSSAR Project

Outline

Security protocol

- A protocol consists of a set of rules (conventions) that determine the exchange of messages between two or more principals.
  In short, a distributed algorithm with emphasis on communication.
- Security (or cryptographic) protocols use cryptographic mechanisms to achieve security objectives.
  Examples: Entity or message authentication, key establishment, integrity, timeliness, fair exchange, non-repudiation, ...
- Small recipes, but nontrivial to design and understand.
  Analogous to **programming Satan's computer**.

Information Security — Past



Security primarily a military concern.

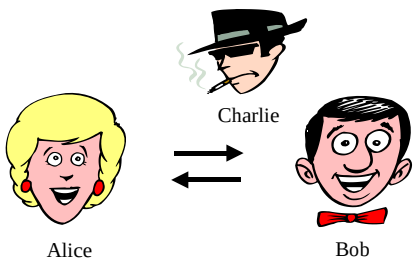
Information Security — Present

- 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.

Internet Security Protocols

- The world is distributed:
  - 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... ?
- Solutions involve protocols like IPSEC, KERBEROS, SSH, SSL, SET, PGP...

Internet Security Protocols

- 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

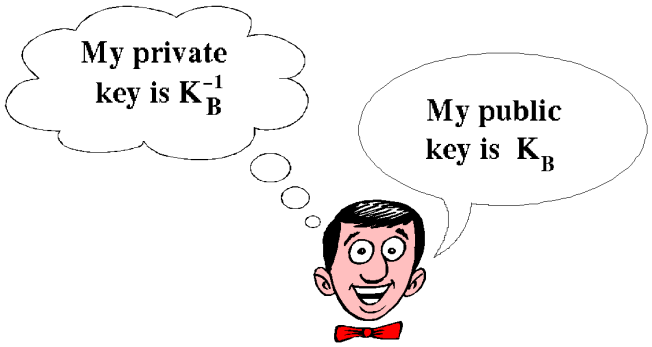
- tools that support the formal analysis of security protocols
- by either finding flaws or establishing their correctness.

- 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.

Outline

Building Blocks for Security Protocols

Cryptographic Procedures: encryption of messages.



$$\{\{M\}_{K_B}\}_{K_B^{-1}} = M$$

(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*.

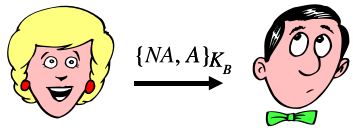
$$A \rightarrow B : \{A, M\}_{K_B}$$

An authentication protocol

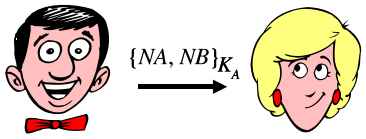
The Needham-Schroeder Public Key protocol (NSPK):

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

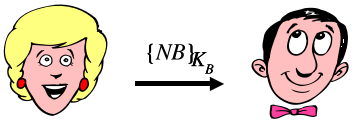
Goal: mutual authentication. Translation:



"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."

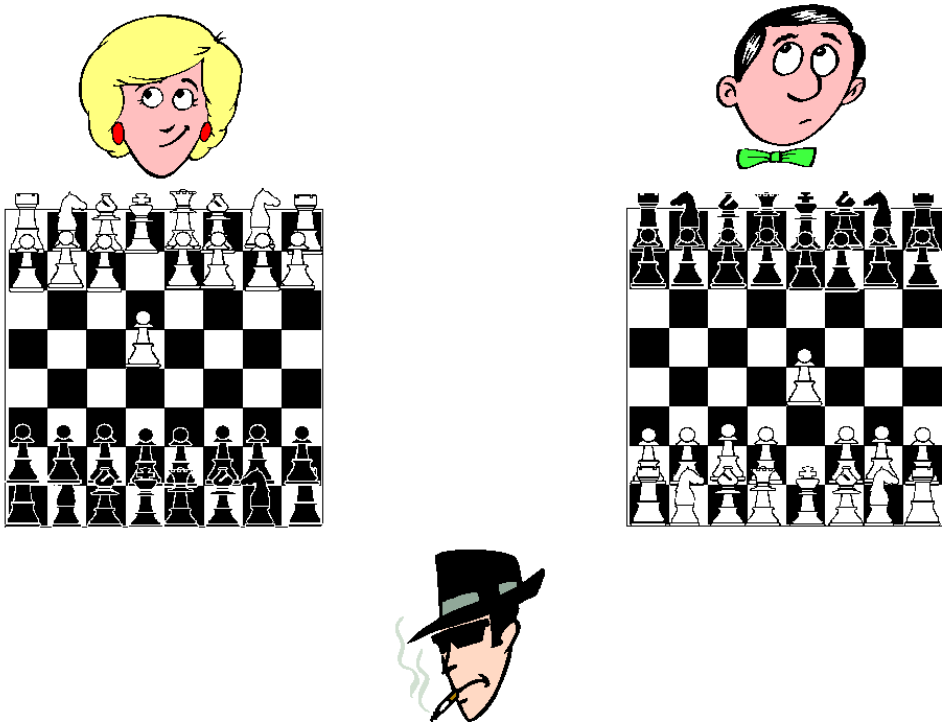


"You sent me *NB*. Since only Alice can read this and I sent it back, you must be Alice."

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

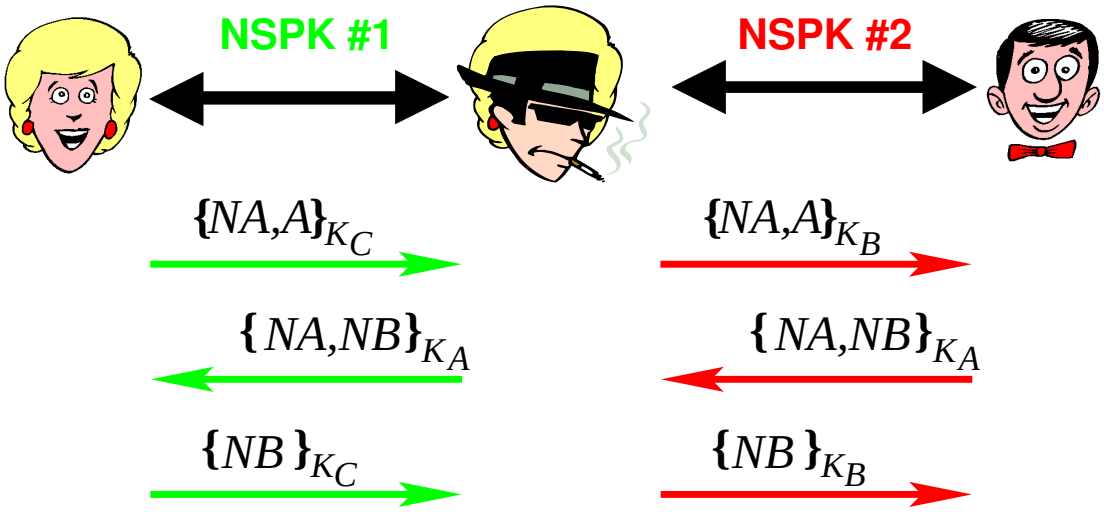
Protocols are typically small and convincing... and often wrong!

How to at least tie against a Chess Grandmaster



Man-in-the-middle attack on the NSPK

$$X \to Y : \{N1, X\}_{K_Y}$$
$$Y \to X : \{N1, N2\}_{K_X}$$
$$X \to Y : \{N2\}_{K_Y}$$



B believes he is speaking with *A*!

What went wrong?

- Problem in step 2:

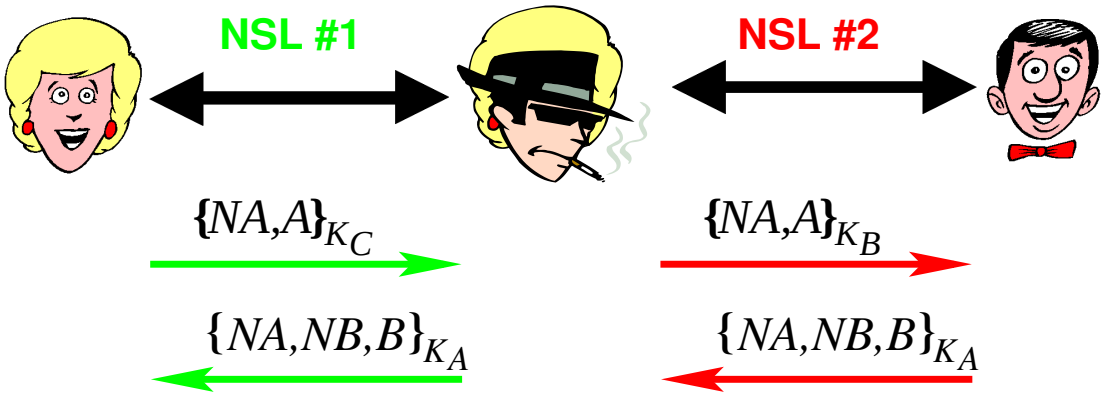
$$B \rightarrow A : \{NA, NB\}_{K_A}$$

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

- The improved version is called NSL protocol.
  Is the protocol now correct?

The NSL Protocol

$$X \rightarrow Y : \{N1, X\}_{K_Y}$$
$$Y \rightarrow X : \{N1, N2, Y\}_{K_X}$$
$$X \rightarrow Y : \{N2\}_{K_Y}$$



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: $\{NA, NB, B\}_{K_A}$.

- The improved version is called NSL protocol.
  Is the protocol now correct?

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

Let's take stock

Even simple protocols can lead to complex situations.

- 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*)

- Informal analysis can easily miss the attacks.

Formal analysis is required, automatic analysis is desirable.

- Formal analysis requires a formal model of protocol and its goals.
- Side-question: *Is Lowe's attack really an attack?*
  - If the goal of the protocol is *secrecy* or *authentication* then yes.
  - If the goal is only *aliveness* of the agents then no.

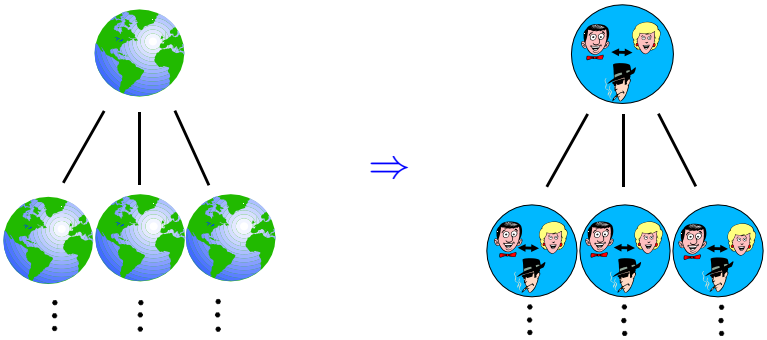
Use formal methods to clarify all this!
And be careful: there are provably secure protocols that become insecure when combined with other protocols.

Outline

Formal modeling and analysis of protocols

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

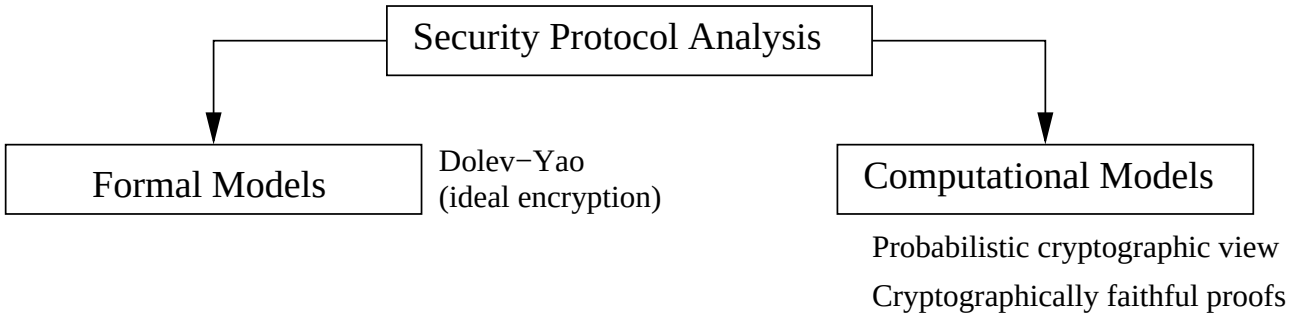
Basis: suitable abstraction of protocols and information flow.



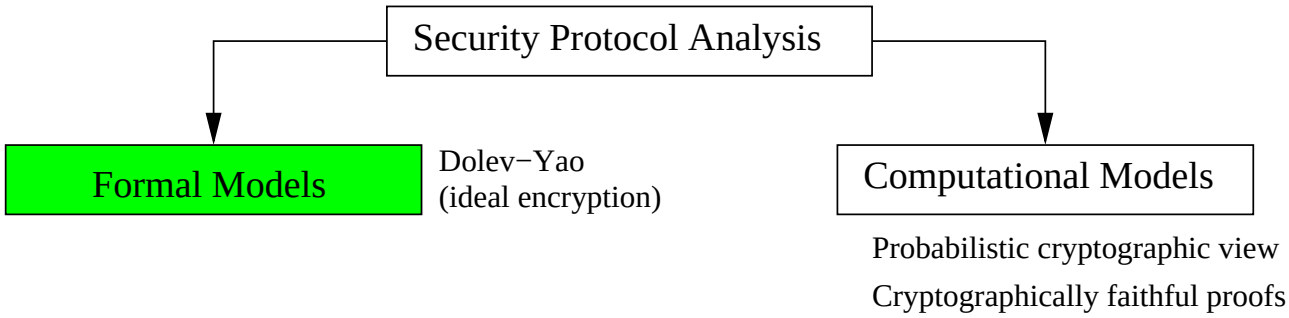
Analysis: with formal methods based on mathematics and logic.

A **language** is **formal** when it has a well-defined syntax and semantics. Additionally there is often a deductive system for determining the truth of statements.

Formal security protocol analysis



Formal Methods for Security Protocol Analysis



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.

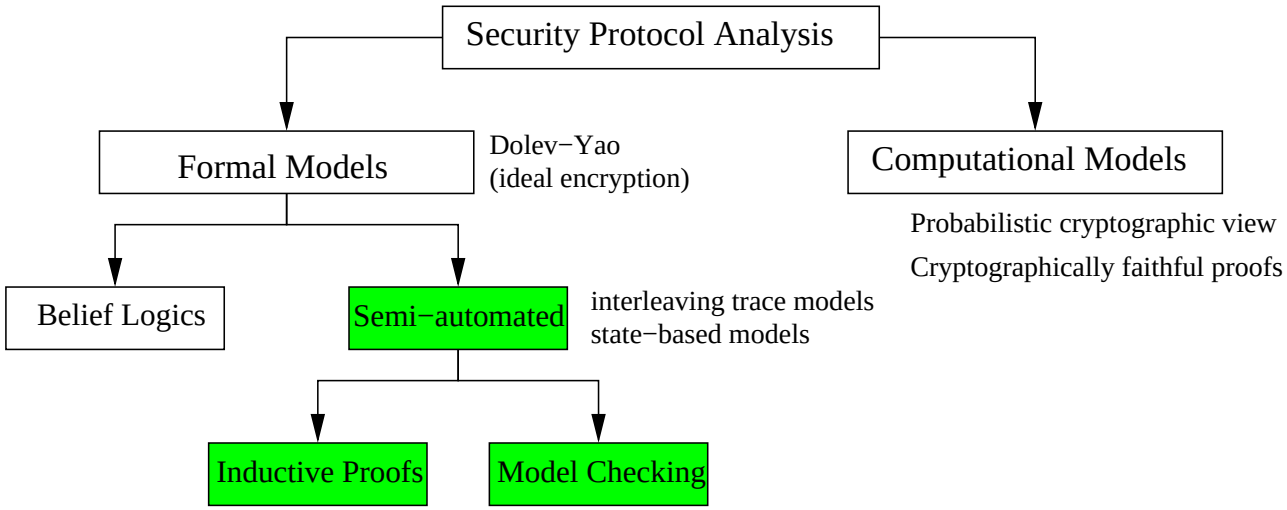
- Cryptoalgorithms behave like black-boxes that obey a limited set of algebraic properties (e.g. encryption and decryption operations cancel each other out $E_X(D_X(M)) = D_X(E_X(M)) = M$).
- Perfect cryptography (all D_X private, decryption only with key,...).
- The intruder can
  - read all traffic,
  - 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



Interleaving Trace Models

- Modeling idea: model possible communication events.

$$\begin{array}{llll} A \to B : M_1 & C \to D : P_1 & A \to B : M_1 & A \to i : M_1 \\ B \to A : M_2 & A \to B : M_1 & C \to D : P_1 & i \to A : M_2 \\ \vdots & B \to A : M_2 & i \to A : M_2 & A \to i : M_3 \quad \ldots \\ & C \to D : P_2 & C \to D : P_2 & \vdots \\ & \vdots & \vdots & \end{array}$$

- 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. $secret(NA, \{A, B\})$.

Modeling: properties

- A property also corresponds to set of traces (or set of states).
  Authentication for A: If (1) A used NA to start a protocol run with B, and (2) received NA back, then B sent NA back.

$$\begin{array}{lcll} AauthenticatesB(t) & \equiv & \textbf{If} & A \rightarrow B : \{NA, \texttt{Agent}A\}_{K_B} \in t \ \textbf{and} \\ & & & B' \rightarrow A : \{NA, NB\}_{K_A} \in t \\ & & \textbf{then} & B \rightarrow A : \{NA, NB\}_{K_A} \in t \\ iattacksA(t) & \equiv & & \neg AauthenticatesB(t) \end{array}$$

  Secrecy: Intruder cannot discover NB, e.g.

  For all $t \in P$, if $B \rightarrow A : \{NA, NB\}_{K_A} \in t$ then $NB \notin analyze(sees\ t)$

- Hence, the correctness of protocols has an exact meaning.
  Every [no] trace of the protocol P has property X.

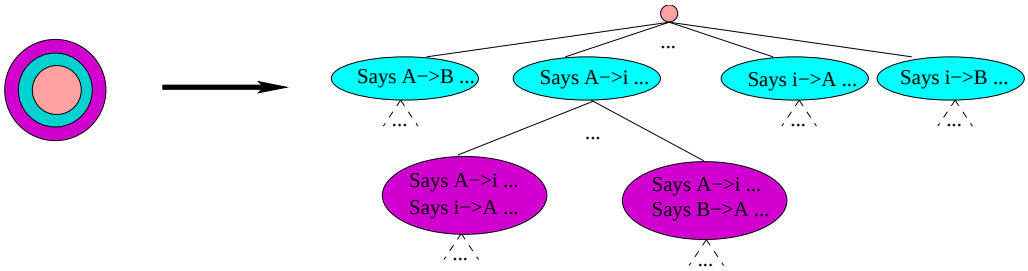
- Every proposition is either true or false.
  How do we determine which holds?

Model provides a basis for analysis

- Trace-based (state-based) model provides a basis for protocol analysis.
- Challenging as general problem is undecidable.
- Possible approaches:
  - Verification proves correctness but is difficult to automate.
    - Use interactive (semi-automated) theorem-proving approach based on induction.
    - This often requires restrictions and/or simplifications, and only semi-automation.
  - Falsification identifies attack traces but does not guarantee correctness.

Falsification using state enumeration

- Inductive definition corresponds to an infinite tree.

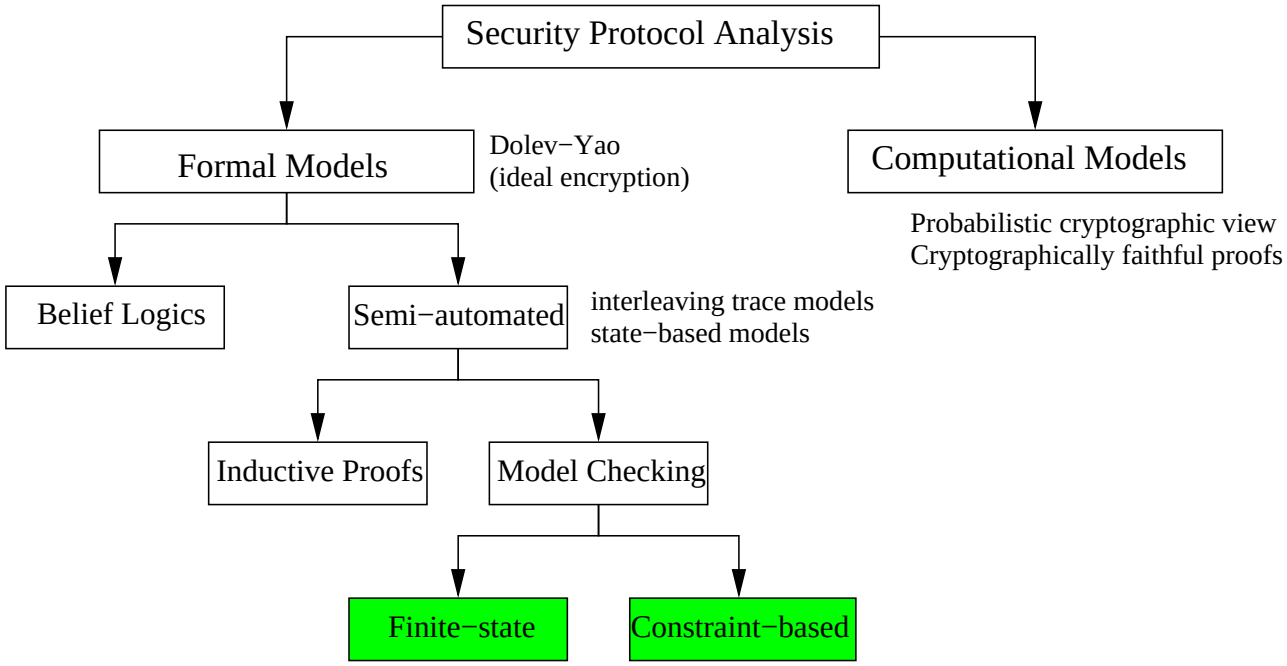


- Properties correspond to a subset of nodes, e.g., i*attacks*$A(t)$.
- State enumeration can be used to find attack in the infinite tree.
- But naive search is hopeless! Challenges:

  Tree too wide: the intruder is extraordinarily prolific!
  Too many interleavings: much “redundant” information.

We will see some ideas for tackling these problems.

Model-checking



Model-checking security protocols

- In a nutshell:
  - System behavior modeled as a (finite) state transition system.
  - System properties expressed by state satisfaction relations.
  - State space exploration to check whether certain properties will or will not be satisfied (yields attack trace).
- Model checking of security protocols focuses on safety properties.
  - Safety: check that certain undesirable properties never occur.
  - Liveness: check that certain desirable properties do eventually occur.
- Very effective at finding flaws, but no guarantee of correctness, due to “artificial” finite bounds.
  - Problem can be partially solved by infinite-state model checking (e.g. based on symbolic methods and abstractions).

Model checking (cont.)

- More in detail, a model checking approach can be described as follows:
  - The operational behavior of a finite state system is modeled by a finite state labeled transition system (LTS), which can make state transitions by interacting with its environment on a set of events.
  - Each state of an LTS is interpreted mechanically into (or assigned with) a logical formula.
  - A system property which is the target of an analysis is also explicitly interpreted into a logical formula.
  - An LTS is symbolically executed to produce a trace.
  - A mechanical procedure can check whether or not a target formula is satisfiable by any formula in any trace (i.e. whether or not the formula is a logical formula in a trace).

Model checking (cont.)

- **Theorem proving:** a theorem is an assertion of a desired goal of the system.
- **Model checking:** a target formula can model a desirable property of the system as well as an undesirable one,
  - e.g. i knows the newly distributed session key K.

  In this case, the result of a satisfiable checking produces a trace that provides an explicit description of a system error,
  - e.g. the trace where i gets hold of K.
- Model checking very effective at finding flaws, but no guarantee of correctness, due to “artificial” finite bounds.
  - Problem can be partially solved by infinite-state model checking (e.g. based on symbolic methods).

Finiteness concerns for model checking

- How many protocol sessions must be executed to ensure coverage?
  - E.g.: man-in-the-middle attack on NSPK needs 2 sessions, but sometimes more are needed.
- No algorithmic determined bound is possible for all cases (because of undecidability for the model).
- Possible bounds for limited classes of protocols (e.g. small system result).
- Only need $n + 1$ distinct principals (in roles, multiple sessions).
  - Only 2 if an honest principal can talk to itself in all roles.
  - The extra principal is the intruder.

N.B.: a *role* is a procedure specified for each party in a protocol. A, B, ... are protocol variables corresponding to roles; their values (e.g. *Alice*, *Bob*, ...) are principals.

Restricted classes of protocols

Different restrictions can be applied in isolation or in conjunction, e.g.

- Bounded number of sessions.
- Bounded number of principals.
- Bounded message size.
  - Fixed number of fields in a message.
  - Fixed set of message constants.
  - Fixed depth restriction.
  - Allow nonces (but only “create new nonce” and $=?$).
- Everything constant, except for number of roles and number of new nonces.

These restrictions can also be applied to finitize model-checking.

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:

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

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

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

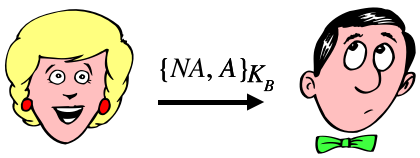
$$\frac{\{m_2\}_{m_1} \in \mathcal{DY}(M) \quad m_1^{-1} \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)}\ A_{\text{crypt}} \qquad \frac{\{m_2\}_{m_1^{-1}} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)}\ A^{-1}_{\text{crypt}}$$

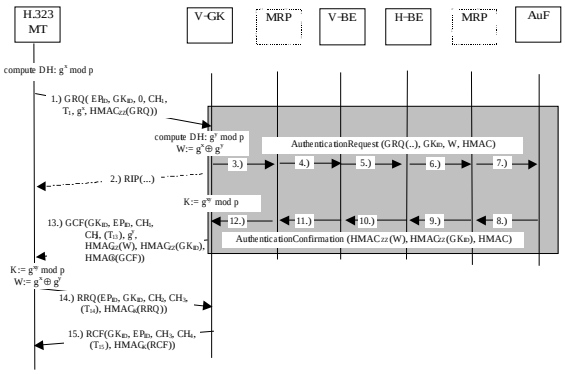
Formal Analysis of Security Protocols

- 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, ...
  - 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, ...) have been taking up this challenge and have been
  - 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,
  - protocol/service composition,
  - and so on.

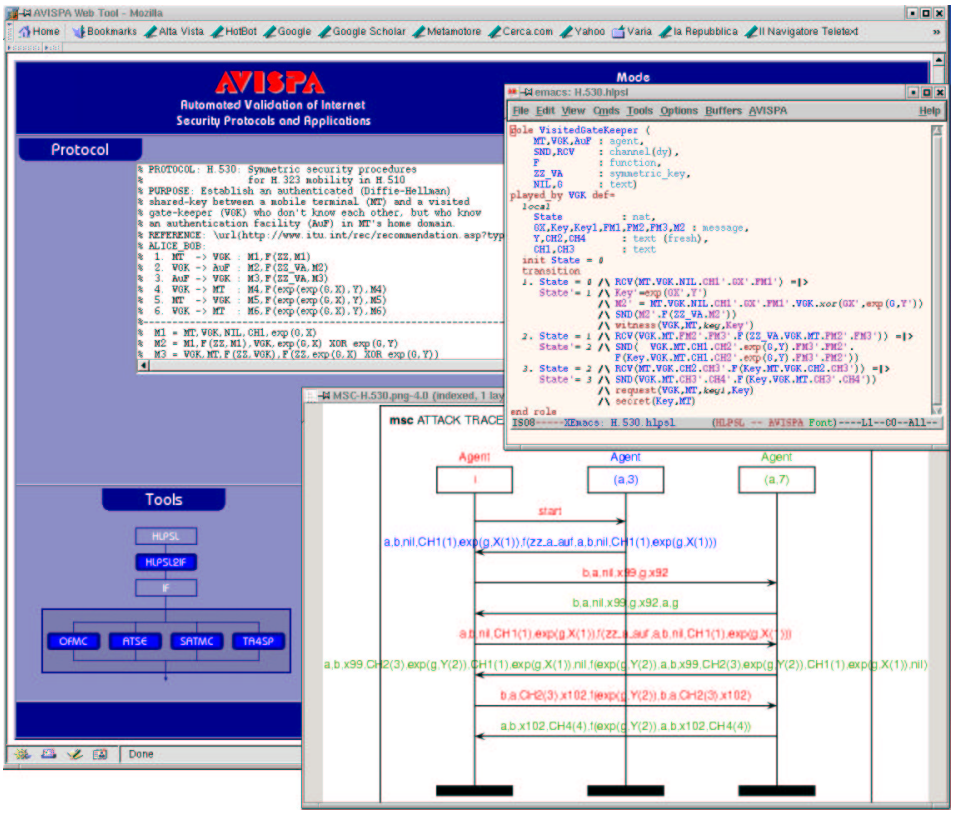
The AVISPA Tool

The AVISPA Tool (soon to be part of the AVANTSSAR Platform)

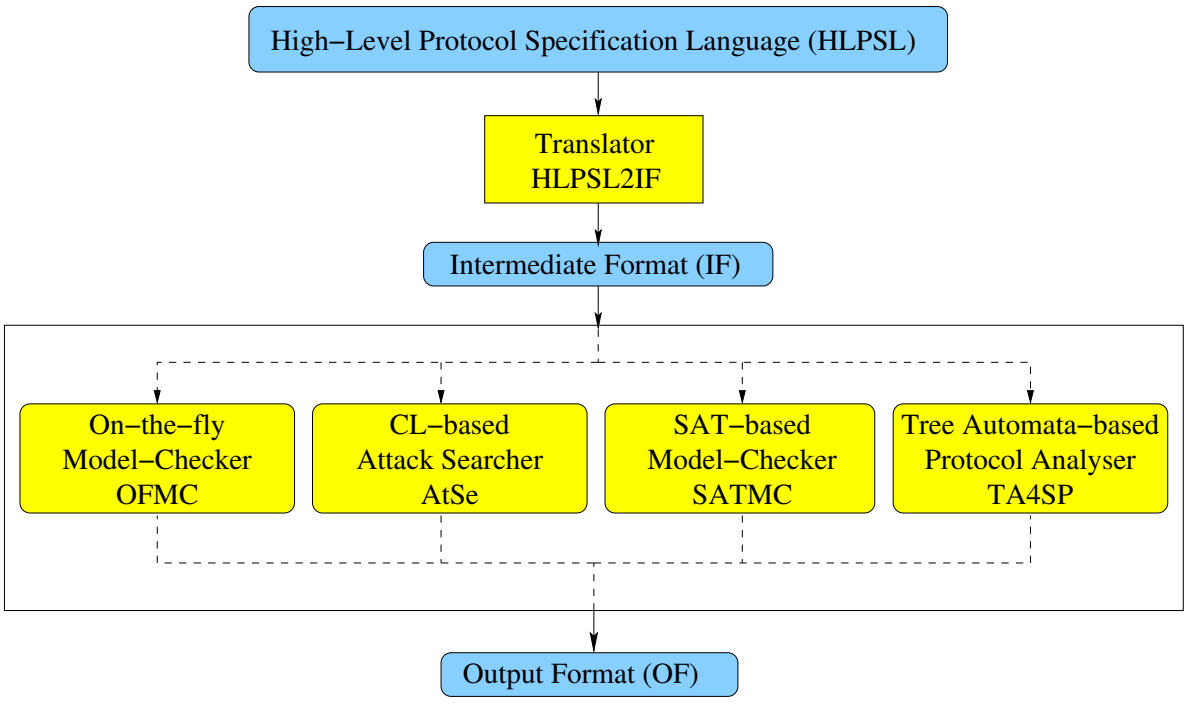
A state-of-the-art (for level of scope and performance), integrated environment for the automatic analysis and validation of Internet security protocols.

- 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



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. Now: **The Open-source Fixed-point Model Checker (OFMC)**

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).

Problems		OFMC			CL-atse			SATMC			
Protocol	#P	P	A	T	P	A	T	P	A	TE	TS
UMTS_AKA	3	3	0	0,02	3	0	0,01	3	0	0,11	0,00
AAAMobileIP	7	7	0	0,75	7	0	0,20	7	0	1,32	0,01
ISO-PK1	1	1	1	0,02	1	1	0,00	1	1	0,05	0,00
ISO-PK2	1	1	0	0,05	1	0	0,00	1	0	1,62	0,00
ISO-PK3	2	2	2	0,04	2	2	0,01	2	2	0,27	0,00
ISO-PK4	2	2	0	0,54	2	0	0,03	2	0	1.153	1,16
LPD-MSR	2	2	2	0,02	2	2	0,02	2	2	0,17	0,02
LPD-IMSR	2	2	0	0,08	2	0	0,01	2	0	0,43	0,01
CHAPv2	3	3	0	0,32	3	0	0,01	3	0	0,55	0,00
EKE	3	3	2	0,19	3	2	0,04	3	2	0,22	0,00
TLS	3	3	0	2,20	3	0	0,32	3	0	-	0,00
DHCP-delayed	2	2	0	0,07	2	0	0,00	2	0	0,19	0,00
Kerb-Cross-Realm	8	8	0	11,86	8	0	4,14	8	0	113,60	1,69
Kerb-Ticket-Cache	6	6	0	2,43	6	0	0,38	6	0	495,66	7,75
Kerb-V	8	8	0	3,08	8	0	0,42	8	0	139,56	2,95
Kerb-Forwardable	6	6	0	30,34	6	0	10,89	0	0	-	-
Kerb-PKINIT	7	7	0	4,41	7	0	0,64	7	0	640,33	11,65
Kerb-preauth	7	7	0	1,86	7	0	0,62	7	0	373,72	2,57
CRAM-MD5	2	2	0	0,71	2	0	0,74	2	0	0,40	0,00
PKB	1	1	1	0,25	1	1	0,01	1	1	0,34	0,02
PKB-fix	2	2	0	4,06	2	0	44,25	2	0	0,86	0,02
SRP_siemens	3	3	0	2,86	0	0	-	0	0	-	-
EKE2	3	3	0	0,16	0	0	-	0	0	-	-
SPEKE	3	3	0	3,11	0	0	-	0	0	-	-
IKEv2-CHILD	3	3	0	1,19	0	0	-	0	0	-	-
IKEv2-DS	3	3	1	5,22	0	0	-	0	0	-	-
IKEv2-DSx	3	3	0	42,56	0	0	-	0	0	-	-
IKEv2-MAC	3	3	0	8,03	0	0	-	0	0	-	-
IKEv2-MACx	3	3	0	40,54	0	0	-	0	0	-	-
h.530	3	1	1	0,64	0	0	-	0	0	-	-
h.530-fix	3	3	0	4.278	0	0	-	0	0	-	-
lipkey-spkm-known	2	2	0	0,23	0	0	-	0	0	-	-
lipkey-spkm-unknown	2	2	0	7,33	0	0	-	0	0	-	-

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.

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`
  - Soon to be available as part of the AVANTSSAR Platform: `www.avantssar.eu`
- Recent and 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).

Outline

Formal Analysis of Security Protocols

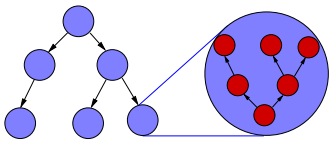
- Challenging as general problem is undecidable.
- Several sources of infinity in protocol analysis:
  - Unbounded number of possible intruder messages (unbounded message depth).
  - Unbounded number of sessions or protocol steps (and agents).
- Possible approaches:
  - Falsification identifies attack traces but does not guarantee correctness.
  - Verification proves correctness but is difficult to automate (requires induction and often restrictions).

Formal Analysis of Security Protocols

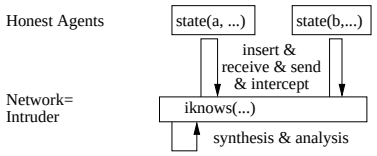
- Challenging as general problem is undecidable.
- Several sources of infinity in protocol analysis:
  - Unbounded number of possible intruder messages (unbounded message depth).
  - Unbounded number of sessions or protocol steps (and agents).
- OFMC: an On-the-Fly Model Checker for Security Protocols.
  - Symbolic techniques to reduce the search space without excluding or introducing attacks.
    - Falsification.
    - (Bounded and abstraction-based unbounded) verification.
  - Now: Open-source Fixed-point Model Checker

Graphical Overview of some Symbolic Reductions

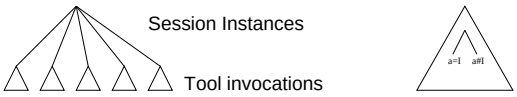
- The Lazy Intruder



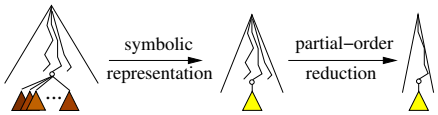
- Compressions



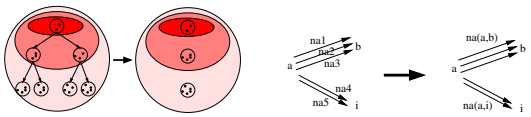
- Symbolic Sessions



- 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.

But first let us see how we model protocols.

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,...)

AnB: Secure pseudonymous channels

OFMC supports three input languages

- HLPSL
- IF
- AnB: an *Alice and Bob* notation that allows for the specification of (algebraic properties and) secure pseudonymous channels for message transmission and protocol/service composition

Formal definition of different kinds of channels (e.g. authentic, confidential, secure) in security protocols and web services

- specify properties of message exchanges over different kinds of channels,
- either as an assumption or as a goal,
- where agents can use their real names or some pseudonyms.

Some channel types

We borrow the bullet notation of Maurer et al., but use it to denote

- the transmission of messages over channels, (i.e. to describe protocols that assume particular kinds of channels),
- authentication and secrecy goals.

Authentic channel: $A \bullet\!\!\rightarrow B : M$

- B can rely that A has sent the message M.
- Moreover, an authentic channel should ensure that A meant to say M to B (i.e. receiver name part of what should be authenticated).

Confidential channel: $A \rightarrow\!\!\bullet\, B : M$

A can rely that only B can receive M.

Secure channel: $A \bullet\!\!\rightarrow\!\!\bullet\, B : M$

A channel that is both authentic and secure.

Some channel types

Standard Channels:

- Authentic channel: $A \bullet\!\!\rightarrow B : M$.
- Confidential channel: $A \rightarrow\!\!\bullet\, B : M$.
- Secure Channel: $A \bullet\!\!\rightarrow\!\!\bullet\, B : M$.

Variant with Freshness:

- Fresh-Authentic channel: $A \bullet\!\!\twoheadrightarrow B : M$.
- Fresh-Secure Channel: $A \bullet\!\!\twoheadrightarrow\!\!\bullet\, B : M$.

Variants with Pseudonyms:

- $[A]_p \bullet\!\!\rightarrow B : M$
- $A \rightarrow\!\!\bullet\, [B]_P : M$
- $[A]_P \bullet\!\!\rightarrow\!\!\bullet\, B : M$
- ...

An example: Diffie-Hellman key exchange

$$
\begin{array}{lll}
1. & A \to B: & g^X \bmod p \\
2. & B \to A: & g^Y \bmod p
\end{array}
$$

$$
\begin{array}{lccc}
 & A: & & B: \\
key = & (g^Y)^X & \approx & (g^X)^Y
\end{array}
$$

$$
3. \quad A \leftrightarrow B: \quad \{\!|\ldots|\!\}_{g^{XY}}
$$

- Need commutativity of exponentiation to represent this protocol.
- Minimum: the algebraic properties necessary for legal protocol execution.
- Affects also authentication/agreement goals.
- Degree of abstraction and aspects to model:

$$\mathrm{dec}(k, \{\!|m|\!\}_k) \approx m \qquad X\|(Y\|Z) \approx (X\|Y)\|Z \qquad X \oplus Y \oplus X \approx Y$$

An example: AnB specification of an authenticated Diffie-Hellman key exchange

`Protocol` : *Authenticated Diffie-Hellman key exchange*

`Types` :

`Agent` A, B;

`Number` g, X, Y, Msg;

`Knowledge` :

A : A, B, g;

B : B, g;

`Actions` :

$A \bullet\!\!\to B : \exp(g, X)$

$B \bullet\!\!\to A : \exp(g, Y)$

$A \to B : \{\!|A, Msg|\!\}_{\exp(\exp(g,X),Y)}$

`Goals` :

$A \bullet\!\!\to\!\!\bullet B : Msg$

Abstraction of the message layer with channels

Formally define channels from different points of view:

1. the ideal functionality of a channel (as an assumption),
2. the implementation of channels by cryptographic means (implementing the ideal functionality),
3. the channel/secure transmission of a message as a goal of a protocol (so that any protocol that fullfills the goal can be used as an implementation in an ideal channel).

- In particular, two ways to encode channels as assumptions:
  - the **Ideal Channel Model ICM** describes channel functionality in an ideal way,
  - the **Cryptographic Channel Model CCM** employs concrete cryptographic messages on insecure channels.
- Proof of "equivalence" of ICM and CCM: cryptographic channels correctly implement ideal channels.

A compositionality result, relating channels as assumptions and as goals

A compositionality result

Given that we have verified that a protocol P' provides its goals under the assumption of a particular kind of channel, can we then replace the assumed channel with an arbitrary protocol P'' that provides such a channel?

In general, the answer is negative, while we prove that under certain restrictions such **a compositionality result** is possible.

Composability result

P_1
Protocol
Goal $[A] \bullet\!\!\rightarrow\!\!\bullet\ B : Payload$

P_2
Part1 $[A] \bullet\!\!\rightarrow\!\!\bullet\ B : Message$ Part2
Goal: G

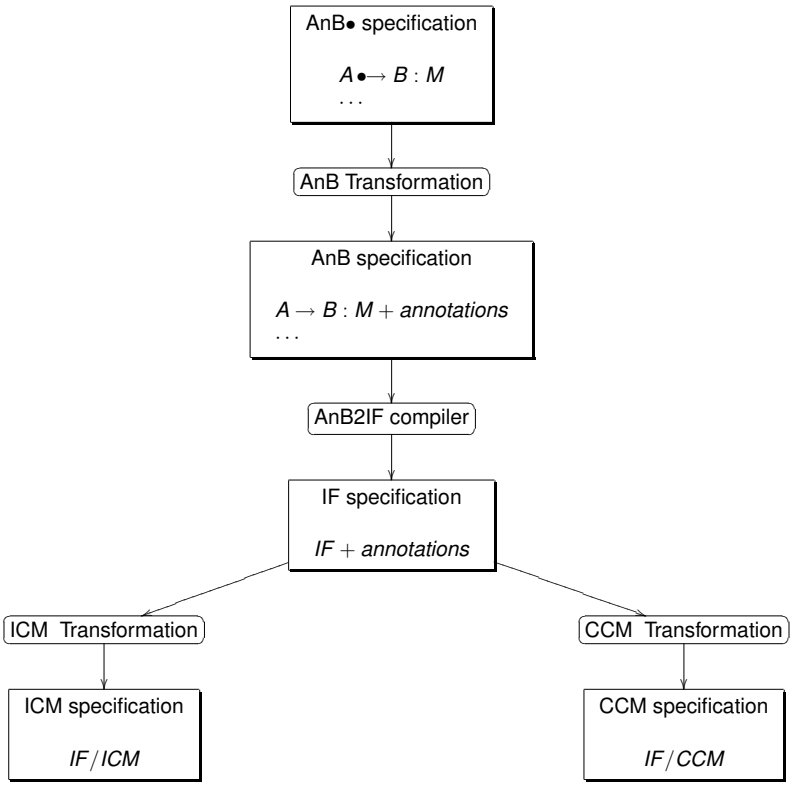
$P_2[P_1]$
Part1 $P_1[Payload \mapsto Message]$ Part2
Goal: G

Idea: verify P_1 and P_2 individually, derive $P_2[P_1]$ is safe — under certain conditions...

From channels to IF (via AnB• and AnB)

- We define an extension AnB• of AnB to specify properties of message exchanges over different kinds of channels.
- Given an AnB specification that uses our notion of channels, we translate it into one that uses only syntactic expressions of the existing language (without using channel bullets) with additional annotations that we will need later on in the translation process.
- We then use an AnB2IF compiler to translate the resulting AnB specification into an IF specification with annotations.
- We then further transform the IF file exploiting the annotations we introduced on the AnB level.

The translations between the models



From AnB• to AnB

To exploit our AnB2IF compiler as a black box:

- We transform a specification written in AnB• notation into a specification in AnB notation where messages are annotated in a particular way that reflects the channels (assumptions/goals).
- We introduce 9 new symbols to annotate the messages of the AnB specification for the channels
  - as assumptions (those ending on A)
  - as goals for the sender (those ending on GS) and the receiver (those ending on GR).

Channel type	AnB•	AnB (+ annotations) for channels as assumptions	AnB (+ annotations) for channels as goals (sender/receiver side)
Authentic	$A \bullet\!\!\rightarrow B : M$	$A \rightarrow B : aAnnA, A, B, M$	$aAnnGS, A, B, h(M)$ and $aAnnGR, A, B, h(M)$
Confidential	$A \rightarrow\!\!\bullet\, B : M$	$A \rightarrow B : cAnnA, B, M$	$cAnnGS, A, B, h(M)$ and $cAnnGR, A, B, h(M)$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet\, B : M$	$A \rightarrow B : sAnnA, A, B, M$	$sAnnGS, A, B, h(M)$ and $sAnnGR, A, B, h(M)$

The Intermediate Format IF: facts

- IF: a more low-level (with respect to AnB) language designed to specify transition systems for protocol analysis tools.
- **IF specification** $P = (I, R, G)$**:**
  - an initial state I (a finite set of ground terms),
  - a set R of rules that induces a transition relation on states,
  - a set G of "attack rules" (i.e. goals) that specify which states count as an attack state.

  A protocol is **safe** when no attack state is reachable from the initial state with the transition relation.
- A state is a set of **facts**:
  - $iknows(m)$ represents persistent intruder knowledge,
  - $state_{\mathcal{R}}(m_1, \ldots, m_n)$ characterizes the local state of an honest agent during the protocol execution by the messages $m_1, \ldots, m_n$ (the local knowledge of that agent), where $\mathcal{R}$ identifies the role of that agent.
  - ...

The Intermediate Format IF: rules

- The IF has **rules** of the form

$$L \mid EQ \;=[V]\Rightarrow\; R$$

  - L and R are sets of facts,
  - EQ is a set of equations on terms,
  - V is a list of variables that do not occur in L or EQ.
  - R may only contain variables that also occur in L, EQ or V.
- Semantics defined by the state transitions the rule allows: we can get from a state S to a state S' with this rule iff there is a substitution σ of all rule variables such that
  - $L\sigma \subseteq S$,
  - $S' = (S \setminus L\sigma) \cup R\sigma$,
  - $V\sigma$ are fresh constants (that do not appear in S), and
  - all equations of EQ are satisfied under σ.
- Equalities between terms/facts are modulo considered algebra.
- Conditions on variables ensure that S' is ground whenever S is.

The Intermediate Format IF: rule examples

- Symmetric decryption:

$$iknows(\{|M|\}_K).iknows(K) \Rightarrow iknows(M).iknows(\{|M|\}_K).iknows(K)$$

  By **persistency** of $iknows(\cdot)$ facts it reduces to:

$$iknows(\{|M|\}_K).iknows(K) \Rightarrow iknows(M)$$

- Symmetric encryption:

$$iknows(M).iknows(K) \Rightarrow iknows(\{|M|\}_K)$$

- and so on

IF: attack states

- Goals of a protocol described by attack states, i.e. states that violate the goals, which are in turn described by attack rules.
- Attack states described by rules without a right-hand side: a state at which the attack rule can fire is thus an attack state.
- Example: secrecy goal

$$secret(M, B).iknows(M).honest(B)$$

  where transition rules contain $secret(M, B)$ whenever an honest agent A generates a message M that is supposed to be secret with another, not necessarily honest, agent B.

  Thus, it is an attack if the intruder finds out M but B is honest.
- Other standard goals, e.g. authentication, described similarly.

IF: transition rules of honest agents

$$state_{\mathcal{R}}(m_0, \ldots, m_k).iknows(m_{k+1})$$
$$=[V]\Rightarrow$$
$$state_{\mathcal{R}}(m_0, \ldots, m_{k+2}, V).iknows(m_{k+2})$$

- m_0 is the initial knowledge of role $\mathcal{R}$
- $m_1, \ldots, m_k$ is the sequence of messages that $\mathcal{R}$ has sent and received so far,
- m_{k+1} is the message that $\mathcal{R}$ receives in this transition,
- m_{k+2} is the message that $\mathcal{R}$ replies with,
- V is the set of fresh variables in m_{k+2}.

Rule applicable whenever an agent playing role $\mathcal{R}$ is in an appropriate state and receives an appropriate message from the intruder.

IF: transition rules of honest agents

$$state_{\mathcal{R}}(m_0, \ldots, m_k).iknows(m_{k+1})$$
$$=[V]\Rightarrow$$
$$state_{\mathcal{R}}(m_0, \ldots, m_{k+2}, V).iknows(m_{k+2})$$

- Rule applicable whenever an agent playing role $\mathcal{R}$ is in an appropriate state and receives an appropriate message from the intruder.
- Reflects an optimization for insecure channels: identify intruder and network for insecure channels (that are controlled by intruder).
- If we apply the rule, then the agent creates the new variables V and sends the outgoing message m_{k+2} to the “network”, and also updates its local state by the received message and the sent one, and by the fresh variables.

Example: transition rules of A in authenticated DH

$$state_{\mathcal{A}}(A, B, g)$$
$$=[X]\Rightarrow$$
$$state_{\mathcal{A}}(A, B, g, \exp(g, X), X).iknows(\exp(g, X))$$

$$state_{\mathcal{A}}(A, B, g, \exp(g, X), X).iknows(\exp(g, Y))$$
$$=[Msg]\Rightarrow$$
$$state_{\mathcal{A}}(\ldots).iknows(\{|A, Msg|\}_{\exp(\exp(g,X),Y)})$$

- Weak points of this naive schema:
  - A will accept only messages of the form $\exp(g, Y)$, while in reality, nobody can check this for an unknown Y.
  - A should accept any incoming message GY here and build the Diffie-Hellman key for the outgoing message as $\exp(GY, X)$.
- See paper for a description of how such a translation is computed in general: appropriate check of incoming messages and correct construction of outgoing messages.

Channels as assumptions: ICM and CCM

- The AnB specification with annotations that results from the "AnB Transformation" is fed into the AnB2IF compiler, which translates it into an IF specification with annotations.

  We can then derive the actual IF specification corresponding to the given AnB• specification in 2 variants, the ICM and the CCM.

- We do this by introducing new facts:

Channel	AnB	ICM	CCM
Authentic	$A \bullet\!\!\rightarrow B : M$	$athCh_{A,B}(M)$	$iknows(\{atag, B, M\}_{inv(ak(A))})$
Confidential	$A \rightarrow\!\!\bullet\, B : M$	$cnfCh_{B}(M)$	$iknows(\{ctag, M\}_{ck(B)})$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet\, B : M$	$secCh_{A,B}(M)$	$iknows(\{\{stag, B, M\}_{inv(ak(A))}\}_{ck(B)})$

ICM

Channel	AnB	ICM	CCM
Authentic	$A \bullet\!\!\rightarrow B : M$	$athCh_{A,B}(M)$	$iknows(\{atag, B, M\}_{inv(ak(A))})$
Confidential	$A \rightarrow\!\!\bullet\, B : M$	$cnfCh_B(M)$	$iknows(\{ctag, M\}_{ck(B)})$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet\, B : M$	$secCh_{A,B}(M)$	$iknows(\{\{stag, B, M\}_{inv(ak(A))}\}_{ck(B)})$

Rules for the intruder behavior (sending / receiving on new channels):

$$iknows(B).iknows(M).dishonest(A) \Rightarrow athCh_{A,B}(M) \quad (1)$$
$$athCh_{A,B}(M) \Rightarrow iknows(M) \quad (2)$$
$$iknows(B).iknows(M) \Rightarrow cnfCh_B(M) \quad (3)$$
$$cnfCh_B(M).dishonest(B) \Rightarrow iknows(M) \quad (4)$$
$$iknows(B).iknows(M).dishonest(A) \Rightarrow secCh_{A,B}(M) \quad (5)$$
$$secCh_{A,B}(M).dishonest(B) \Rightarrow iknows(M) \quad (6)$$

(1) intruder can send messages on an authentic channel to any agent B but only under the name of a dishonest agent A.
(2) intruder can receive any message on an authentic channel.

CCM

Channel	AnB	ICM	CCM
Authentic	$A \bullet\!\!\rightarrow B : M$	$athCh_{A,B}(M)$	$iknows(\{atag, B, M\}_{inv(ak(A))})$
Confidential	$A \rightarrow\!\!\bullet\, B : M$	$cnfCh_B(M)$	$iknows(\{ctag, M\}_{ck(B)})$
Secure	$A \bullet\!\!\rightarrow\!\!\bullet\, B : M$	$secCh_{A,B}(M)$	$iknows(\{\{stag, B, M\}_{inv(ak(A))}\}_{ck(B)})$

- Employs concrete cryptographic messages on insecure channels (as opposed to the channel facts of the specification in the ICM).
  Here, public key cryptography (MACs would also be possible).
- **Tags**: public constants that determine meaning of a message (from point of view of agent that generated the message), and the receiver will only accept messages that have correct tag according to protocol.
- *ak* and *ck* are tables of public keys, for signing and encrypting.
- Pseudonymous channels: pseudonym = public key, e.g.

$$[A]_P \bullet\!\!\rightarrow B : M \quad A \rightarrow B : \{atag, B, M\}_{inv(P)}$$

An example

$$\begin{array}{lcll} A & \bullet\!\!\rightarrow & B & : \exp(g, X) \\ B & \bullet\!\!\rightarrow & A & : \exp(g, Y) \\ A & \rightarrow & B & : \{|A, Msg|\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : Msg \end{array}$$

- A's second transition when using insecure channels

$$\begin{aligned} & state_{\mathcal{A}}(A, B, g, \exp(g, X), X).iknows(GY) \\ & =[Msg]\Rightarrow \\ & state_{\mathcal{A}}(\ldots).iknows(\{|A, Msg|\}_{\exp(GY,X)}) \end{aligned}$$

- ICM's rule uses authentic-channel predicate on incoming msg:

$$\begin{aligned} & state_{\mathcal{A}}(A, B, g, \exp(g, X), X).athCh_{B,A}(GY) \\ & =[Msg]\Rightarrow \\ & state_{\mathcal{A}}(\ldots).iknows(\{|A, Msg|\}_{\exp(GY,X)}) \end{aligned}$$

- CCM's rule:

$$\begin{aligned} & state_{\mathcal{A}}(A, B, ak(A), inv(ak(A)), ak(B), g, \exp(g, X), X).iknows(\{atag, A, GY\}_{inv(ak(B))}) \\ & =[Msg]\Rightarrow \\ & state_{\mathcal{A}}(\ldots).iknows(\{|A, Msg|\}_{\exp(GY,X)}) \end{aligned}$$

Relating the two models

- Attacks in ICM can be simulated in CCM.
- Typed attacks in ICM can be simulated in CCM.

$\Rightarrow$ CCM is a correct implementation of the ICM

- Relating the two models has revealed subtle details of their realization.

Relating the two models (details)

Theorem

Consider an ICM specification and the corresponding CCM specification. For a reachable state S_1 of the ICM specification, there is a reachable state S_2 of the CCM specification such that $S_1 \sim S_2$.

Theorem

Consider an ICM specification and the corresponding CCM specification, both with full receiver decryption, and consider a well-typed attack on the CCM specification that leads to the attack state S_2. Then there is a reachable attack state S_1 of the ICM specification such that $S_1 \sim S_2$.

Two assumptions: typed model and that a message can be fully analyzed by an honest receiver in the sense that its message pattern contains only variables of an atomic type.

Channels as goals

- First use standard “weak” (non-injective) authentication and secrecy goals.
- We need however more for composability. Consider

$$\frac{A \rightarrow B : \{M\}_{pk(B)}, \{h(M)\}_{inv(pkA)}}{A \bullet\!\!\rightarrow B : M}$$

- Standard authentication goal holds, but actually this is not an authentic channel:

$$\begin{array}{ll} a \rightarrow i(b): & \{M\}_{pk(B)}, \{h(M)\}_{inv(pkA)} \\ i \rightarrow b: & \{M\}_{pk(B)}, \{h(M)\}_{inv(pki)} \end{array}$$

- These subtle things come up when you try to prove compositionality!

Channels as goals

Channel	Goal Facts
Authentic/Sending	$witness(A, B, P, M)$
Confidential/Sending	$whisper(B, P, M)$
Secure/Sending	$witness(A, B, P, M).whisper(B, P, M)$
Authentic/Receiving	$request(A, B, P, M)$
Confidential/Receiving	$hear(B, P, M)$
Secure/Receiving	$request(A, B, P, M).hear(B, P, M)$

- Specify goals of a protocol using the different kinds of channels.

  Intuition: protocol should ensure the authentic, confidential, or secure transmission of the respective message.

Composability result

P_1
Protocol
Goal $[A] \bullet\!\!\rightarrow\!\!\bullet\ B : Payload$

P_2
Part1 $[A] \bullet\!\!\rightarrow\!\!\bullet\ B : Message$ Part2
Goal: G

$P_2[P_1]$
Part1 $P_1[Payload \mapsto Message]$ Part2
Goal: G

Idea: verify P_1 and P_2 individually, derive $P_2[P_1]$ is safe — under certain conditions...

— under certain conditions. . .

- Arbitrary payload message, ensure freshness.
- Additional goals on intruder knowledge:
  When the intruder sends a message on an authentic or confidential channel, then he must know it. Not covered by previous goals.
- Composability properties of the P_i.

Example

- P_2:

$$\begin{array}{llll} A & \bullet\!\!\rightarrow & B & : \exp(g,X) \\ B & \bullet\!\!\rightarrow & A & : \exp(g,Y) \\ A & \rightarrow & B & : \{|Payload|\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload \end{array}$$

- Implement first authentic channel by P_1:

$$\begin{array}{llll} A' & \rightarrow & B' & : \{B', M'\}_{inv(pk(A'))} \\ \hline A' & \bullet\!\!\rightarrow & B' & : M' \end{array}$$

- The composition $P_2[P_1]$ is safe:

$$\begin{array}{llll} A & \rightarrow & B & : \{B, \exp(g,X)\}_{inv(pk(A))} \\ B & \bullet\!\!\rightarrow & A & : \exp(g,Y) \\ A & \rightarrow & B & : \{|Payload|\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload \end{array}$$

Example

- Variant P_2' breaks

$$\begin{array}{lcll} A & \rightarrow & B & : \{B, g\}_{inv(pk(A))} \\ A & \bullet\!\!\rightarrow & B & : \exp(g, X) \\ B & \bullet\!\!\rightarrow & A & : \exp(g, Y) \\ A & \rightarrow & B & : \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload \end{array}$$

when composing it with P_1:

$$\begin{array}{lcll} A' & \rightarrow & B' & : \{B', M'\}_{inv(pk(A'))} \\ \hline A' & \bullet\!\!\rightarrow & B' & : M' \end{array}$$

- While P_2' is also correct in isolation, $P_2'[P_1]$ has an authentication attack: first message of P_2' has the same format as the message of P_1:

$$\{B', M'\}_{inv(pk(A'))}$$

Pseudonymous channels

- Example: Diffie-Hellman with unauthenticated A (also TLS/SSL):

$$\begin{array}{lcll} A & \rightarrow & B & : \exp(g, X) \\ B & \bullet\!\!\rightarrow & A & : \exp(g, Y) \\ A & \rightarrow & B & : \{\!|Payload|\!\}_{\exp(\exp(g,X),Y)} \\ \hline [A] & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload \end{array}$$

- Such a channel is good enough for a login protocol, e.g.

$$\begin{array}{lcll} [A] & \bullet\!\!\rightarrow\!\!\bullet & B & : A, password(A) \\ [A] & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload' \\ \hline A & \bullet\!\!\rightarrow\!\!\bullet & B & : Payload' \end{array}$$

Pseudonyms and freshness

- Definitions and results carry over also when one (or both) ends of a channel are identified by a pseudonym rather than the real name.
- We can also consider a further variant of channels that ensure freshness of messages, i.e. suppress the replay of messages.
- Ongoing work on other kinds of channels.

Let us now return to the two challenges for the analysis.

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.

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

$$1. \quad A \rightarrow B : \; M, A, B, \{|NA, M, A, B|\}_{K_{AS}}$$

The Lazy Intruder: Idea

$$1.\quad \mathrm{i}(A) \rightarrow B:\ M, A, B, \{|NA, M, A, B|\}_{K_{AS}}$$

The Lazy Intruder: Idea

$$1.\quad \mathrm{i}(A) \rightarrow B:\ M, A, B, \{|NA, M, A, B|\}_{K_{AS}}$$

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

Idea: postpone this decision.

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

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:

$$from(T; IK)$$

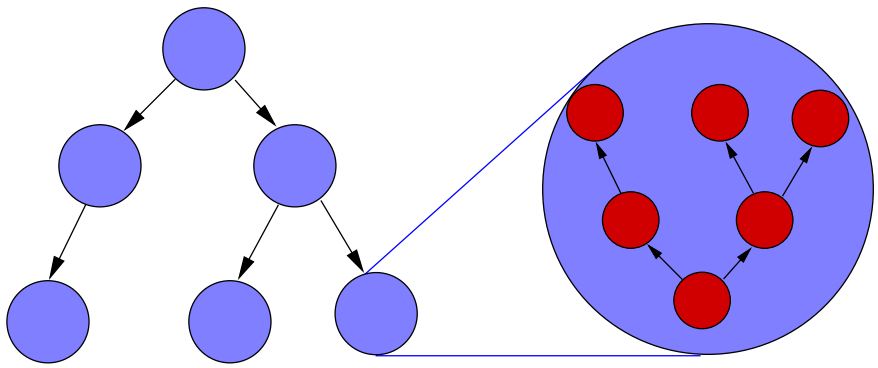
- $[\![from(T; IK)]\!] = \{\sigma \mid \text{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$
- Simple constraint always satisfiable: terms to be generated are variables.
- Reduce a given constraint to an equivalent set of simple constraints, via constraint reduction rules that are
  - correct: they maintain the set of models,
  - terminating: we arrive, after finitely many steps, at a finite set of simple constraints.

  Rules of three kinds:
  - generation: intruder can compose messages from known ones,
  - analysis: intruder can decompose messages,
  - a unification rule: intruder can use unifiable messages from his knowledge to fulfill the constraint.

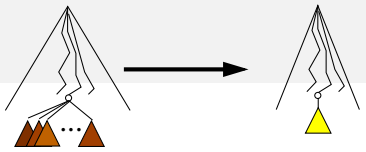
Integration: Symbolic Transition System

- Symbolic state = term with variables + constraint set
- $[\![(t, C)]\!] = \{t\sigma \mid \sigma \in [\![C]\!]\}$ (a set of ground states).
- Two layers of search:

  Layer 1: search in the symbolic state space
  Layer 2: constraint reduction



Lazy Intruder: an example



- Non-authenticated Diffie-Hellman key exchange (half keys are sent on insecure channels)

$$\begin{array}{lll} 1. & a \rightarrow i(b): & \exp(g,x) \\ 2. & i(b) \rightarrow a: & M_1 \\ 3. & a \rightarrow i(b): & \{|msg|\}_{\exp(M_1,x)} \\ & & secret(msg,b) \end{array}$$

  where a parses $i(b)$'s reply as the Diffie-Hellman half key from b and declares the payload as a secret with b.

- Can intruder find out this secret, assuming $IK_0 = \{g\}$?

$$\begin{array}{ll} from(\{M_1\}; IK_1) & \text{where } IK_1 = IK_0 \cup \{\exp(g,x)\} \\ from(\{msg\}; IK_2) & \text{where } IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\} \end{array}$$

Lazy Intruder: an example

$$\begin{array}{lll} 1. & a \rightarrow i(b): & \exp(g,x) \\ 2. & i(b) \rightarrow a: & M_1 \\ 3. & a \rightarrow i(b): & \{|msg|\}_{\exp(M_1,x)} \\ & & secret(msg,b) \end{array}$$

$$IK_0 = \{g\}$$

$$\begin{array}{ll} from(\{M_1\}; IK_1) & \text{where } IK_1 = IK_0 \cup \{\exp(g,x)\} \\ from(\{msg\}; IK_2) & \text{where } IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\} \end{array}$$

One reduction sequence leads to solution of constraint set (procedure considers also other sequences, leading to unsatisfiable constraints):

- intruder successfully decrypts encrypted message in IK_2:

$$\begin{array}{ll} from(\{M_1\}; IK_1) & \text{where } IK_1 = IK_0 \cup \{\exp(g,x)\} \\ from(\{msg\}; IK_2 \cup \{msg\}) & \text{where } IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\} \\ from(\{\exp(M_1,x)\}; IK_2) & \text{where } IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\} \end{array}$$

- second constraint can be solved by unification rule and removed

$$\begin{array}{ll} from(\{M_1\}; IK_1) & \text{where } IK_1 = IK_0 \cup \{\exp(g,x)\} \\ from(\{\exp(M_1,x)\}; IK_2) & \text{where } IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\} \end{array}$$

Lazy Intruder: an example

1. $a \rightarrow i(b)$: $\exp(g, x)$
2. $i(b) \rightarrow a$: M_1
3. $a \rightarrow i(b)$: $\{|msg|\}_{\exp(M_1,x)}$ $secret(msg, b)$

$IK_0 = \{g\}$

$from(\{M_1\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

$from(\{\exp(M_1, x)\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- intruder cannot directly compose key as he does not know a's secret value x, but he can compose this term if M_1 has the form $\exp(M_2, M_3)$ for two new variables M_2 and M_3.
  The resulting key $\exp(\exp(M_2, M_3), x)$ is equivalent to $\exp(\exp(M_2, x), M_3)$ according to algebraic properties of exponentiation:

$from(\{\exp(M_2, M_3)\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

$from(\{\exp(\exp(M_2, x), M_3)\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- This latter representation can indeed be composed.

Lazy Intruder: an example

1. $a \rightarrow i(b)$: $\exp(g, x)$
2. $i(b) \rightarrow a$: M_1
3. $a \rightarrow i(b)$: $\{|msg|\}_{\exp(M_1,x)}$ $secret(msg, b)$

$IK_0 = \{g\}$

$from(\{\exp(M_2, M_3)\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

$from(\{\exp(\exp(M_2, x), M_3)\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- By an application of a generation rule:

  $from(\{\exp(M_2, M_3)\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

  $from(\{\exp(M_2, x), M_3\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- We can unify $\exp(M_2, x)$ with a's half key, (i.e. $M_2 = g$):

  $from(\{\exp(g, M_3)\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

  $from(\{M_3\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- As $g \in IK_1$, we can generate and leave a set of simple constraints:

  $from(\{M_3\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

  $from(\{M_3\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

Lazy Intruder: an example

1. $a \rightarrow i(b)$: $\exp(g, x)$
2. $i(b) \rightarrow a$: M_1
3. $a \rightarrow i(b)$: $\{|msg|\}_{\exp(M_1,x)}$
   $secret(msg, b)$

$$IK_0 = \{g\}$$

$from(\{M_3\}; IK_1)$ where $IK_1 = IK_0 \cup \{\exp(g, x)\}$

$from(\{M_3\}; IK_2)$ where $IK_2 = IK_1 \cup \{\{|msg|\}_{\exp(M_1,x)}\}$

- Hence,
  - $M_1 = \exp(M_2, M_3) = \exp(g, M_3)$
  - $M_2 = g$

  and intruder can perform the attack for any value M_3 that he knows

1. $a \rightarrow i(b)$: g^x
2. $i(b) \rightarrow a$: g^{M_3}
3. $a \rightarrow i(b)$: $\{|msg|\}_{(g^{M_3})^x}$

since $(g^{M_3})^x = g^{M_3 x} = (g^x)^{M_3}$.
For instance, $M_3 = g$.

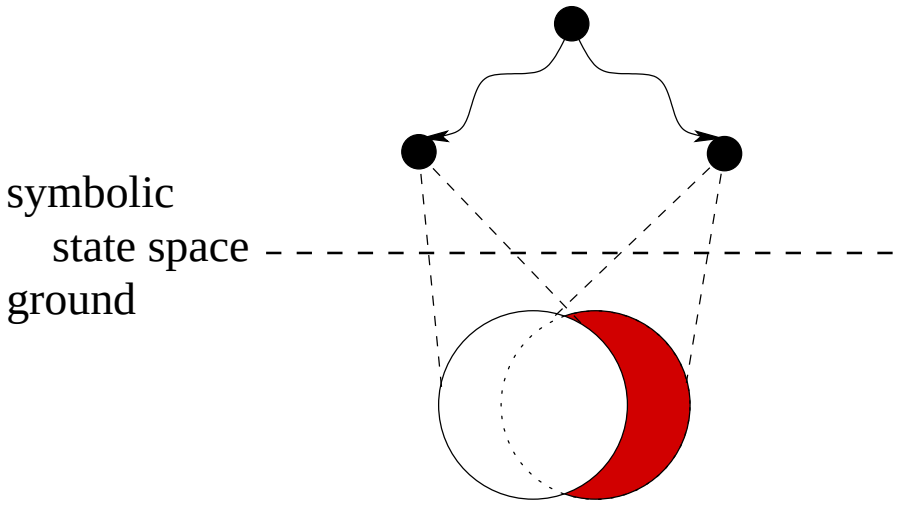
Two Key Challenges and their Solutions

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

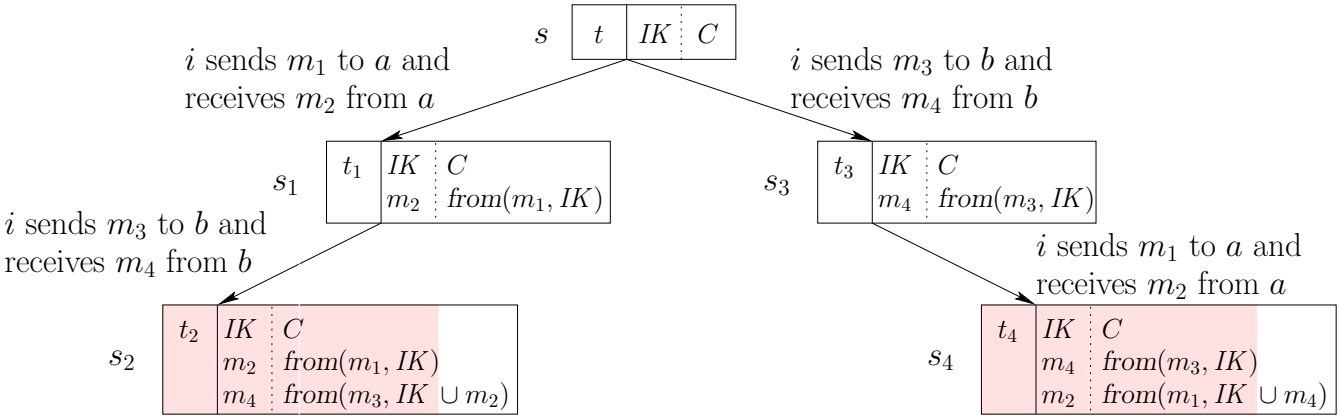
Typical situation: 2 independent actions executable in either order:



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

Constraint Differentiation: Idea

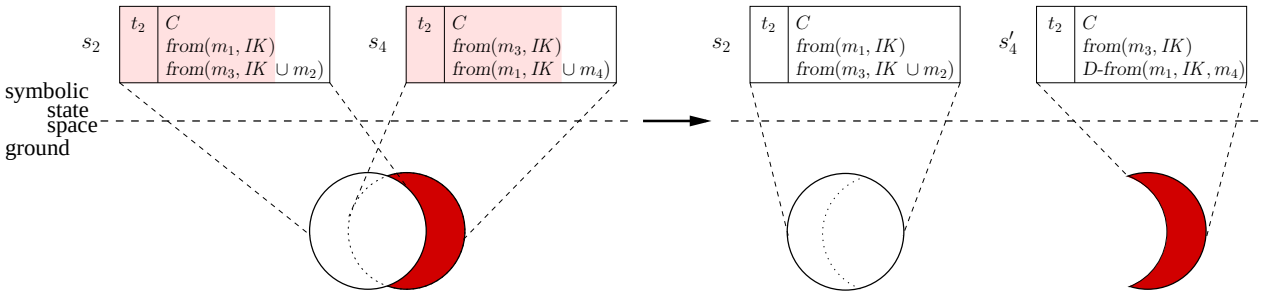
Typical situation: 2 independent actions executable in either order:



where $t_2 = t_4$

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

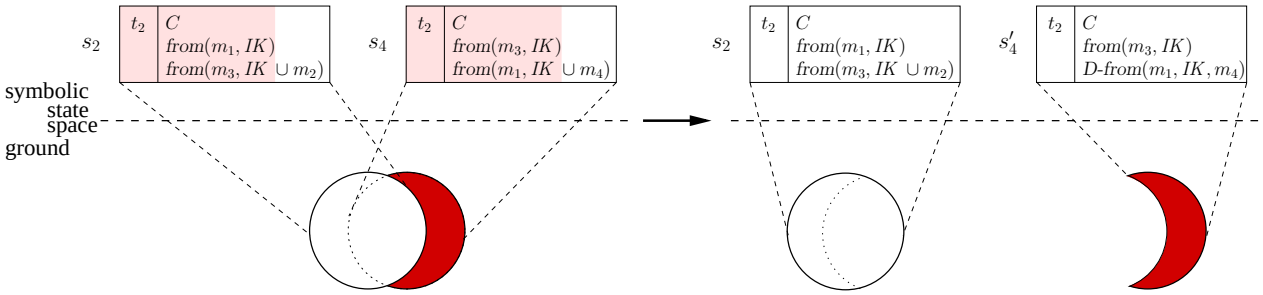
Constraint Differentiation (1)



- New kind of constraints: *D-from*(T, IK, NIK).
- Intuition:
  - Intruder has just learned some new intruder knowledge NIK.
  - All solutions $[\![from(T; IK \cup NIK)]\!]$ are "correct" but a solution is interesting only if it requires NIK.

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

Constraint Differentiation (2)



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

Theorem

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

Theorem

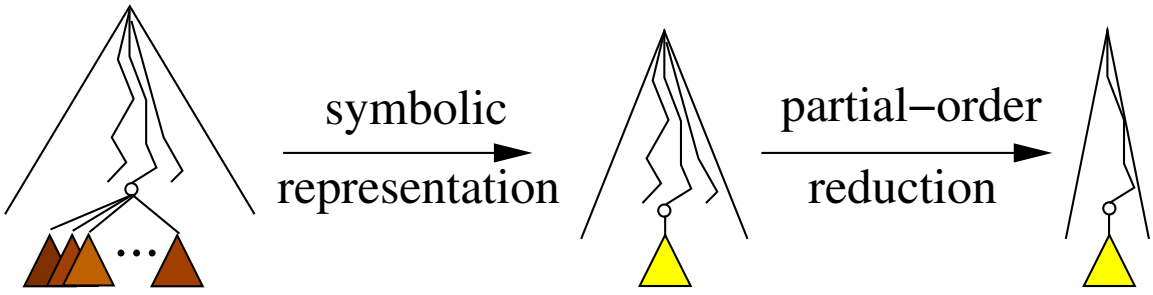
$$[\![s_2]\!] \cup [\![s_4]\!] = [\![s_2]\!] \cup [\![s'_4]\!]$$

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

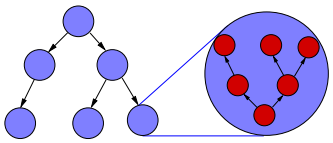
IKE Aggressive Mode Pre-Shared Key												
Mode:	without CD						with CD					
Scenario:	$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$		$[a,b],[a,i]$		$[a,b],[a,i],[i,a]$		$[a,b],[a,i],[i,a],[b,i]$	
Ply	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2	s1	s2
1	3	3	4	4	5	5	3	3	4	4	5	5
2	7	7	14	14	23	23	5	5	10	10	16	16
3	13	14	43	45	97	100	7	8	19	21	40	43
4	17	27	112	139	368	420	6	12	30	44	86	111
5	15	53	238	422	1228	1727	5	17	35	81	150	261
6	15	101	393	1262	3501	6989	3	18	31	139	218	578
7		191	483	3699	8232	27835		20	22	215	241	1174
8		410	420	10637	15288	108927		23	8	319	203	2290
9		720		29783	21168	417862		22		436	136	4112
10		960		79939	18900	1565354		12		527	48	7025
11		990		201861		5695140		9		602		11062
12		990		467533		TO		5		576		16390
13				929500		TO				428		22544
14				1583582		TO				233		27443
15				2132130		TO				177		31024
16				1801800		TO				53		29595
17						TO						10531
18						TO						10531
19						TO						7857
20						TO						2371
Nodes	71	4467	1708	7242353	68811	TO	30	155	160	3866	1144	197426
Time	0.16s	13.66s	4.64s	40655.50s	3m41s	TO	0.08s	0.49s	0.49s	21.60s	4.17s	26m30s

Lazy Intruder and Constraint Reduction

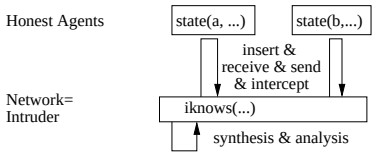


Graphical Overview of some Symbolic Reductions

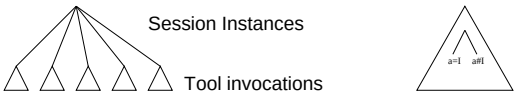
- The Lazy Intruder



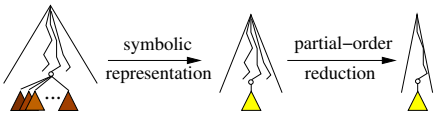
- Compressions



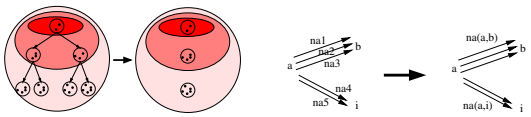
- Symbolic Sessions



- Constraint Differentiation



- Abstractions (data and control)

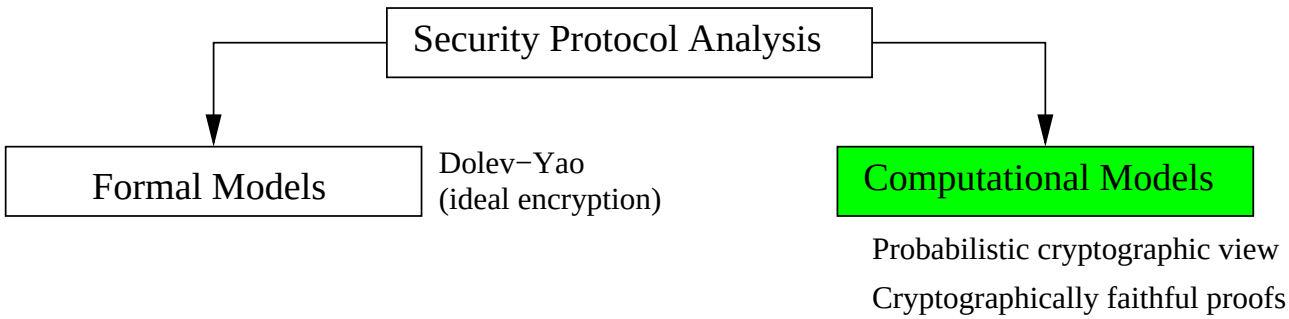


Outline

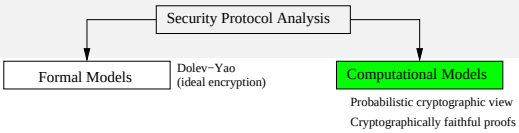
Conclusions and outlook

- 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`
  - Soon to be available as part of the AVANTSSAR Platform: `www.avantssar.eu`
- Recent and 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).

Computational Models

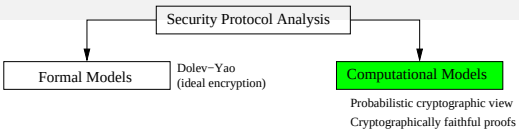


Computational Models



- 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 underling 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.

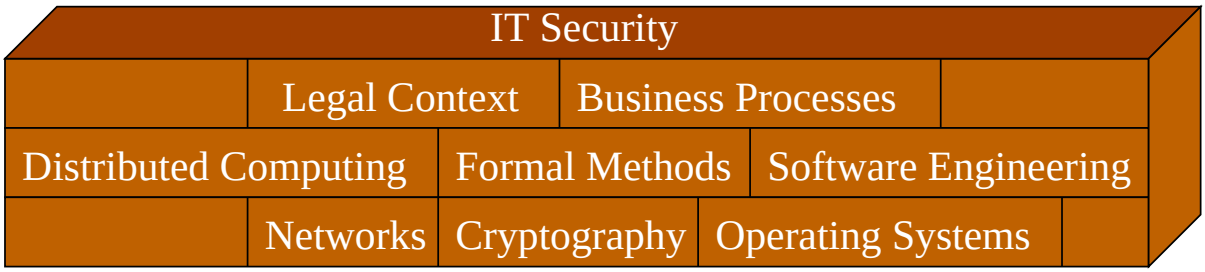
Closing the Gap



- 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.

Security is power!

- Security is an enabling technology.
- Security is power! E.g., in e-government:
  IT processes are used to model and realize government processes. The ability to access and modify data/processes is equal to the ability spy on the most private details of government and its citizens as well as to change the working of the government itself!
- Security is interdisciplinary



and therein lies, in part, the challenge, excitement, and reward!