

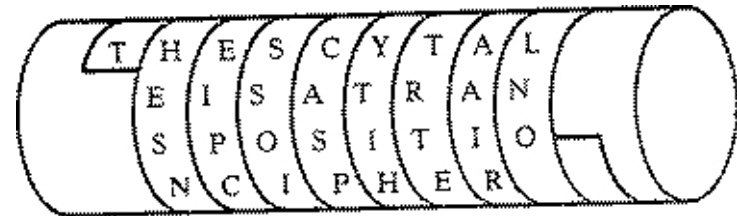
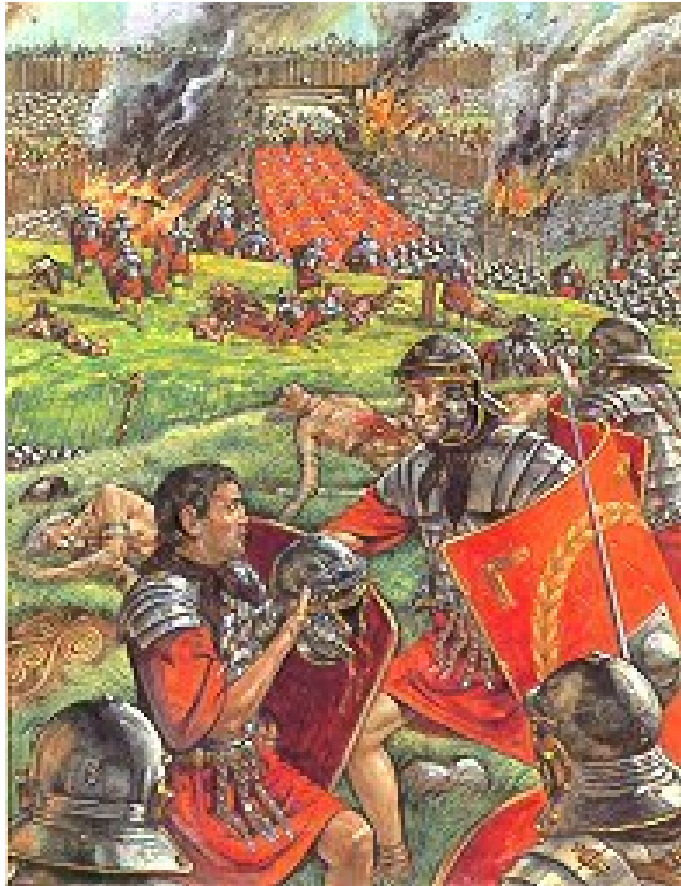
# **Automated Reasoning for Security Protocol Analysis**

**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 \$/€.

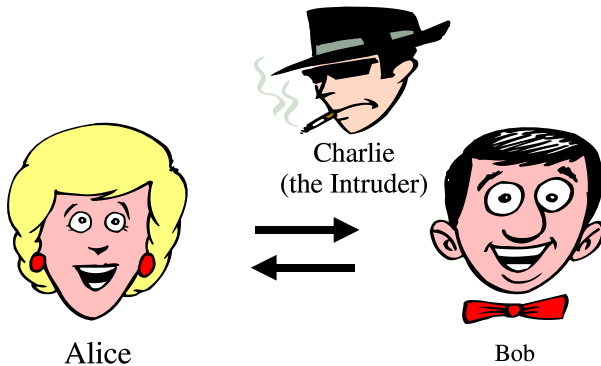
**Time:** protocols are delayed by years.

**Acceptance:** eroding confidence in Internet Security and new applications.



# Information Security Present

- 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... ?

# 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.

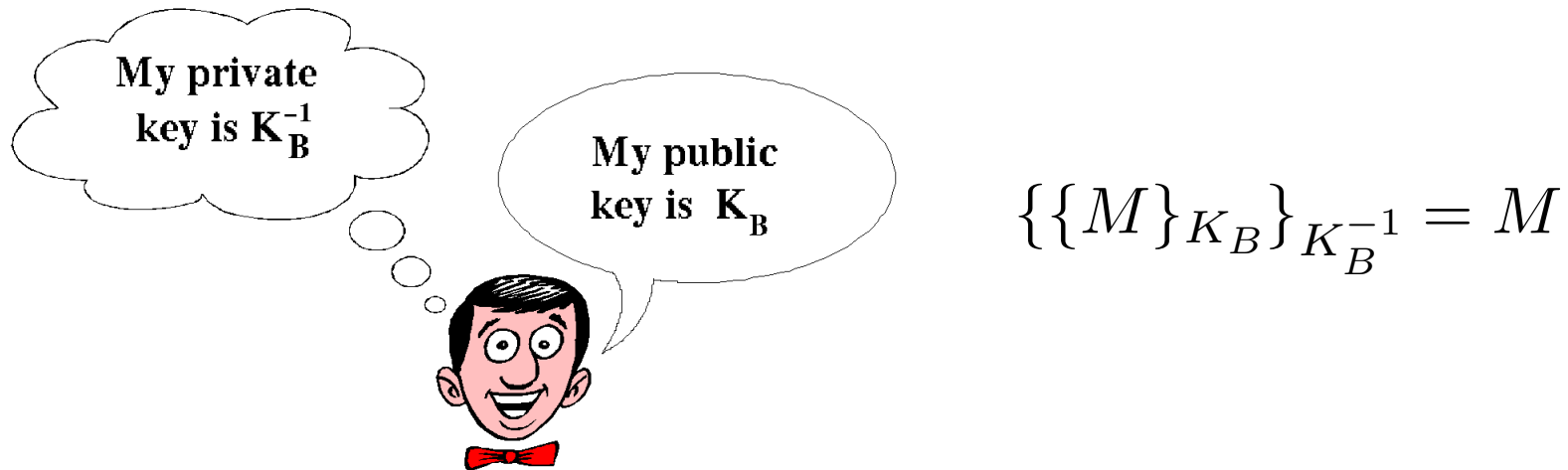


# Road Map

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

# Building Blocks for Security Protocols

**Cryptographic Procedures:** encryption of messages.



**(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.*

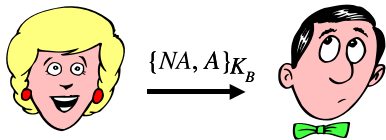
$$A \rightarrow B : \{A, M\}_{K_B}$$

# An Authentication Protocol

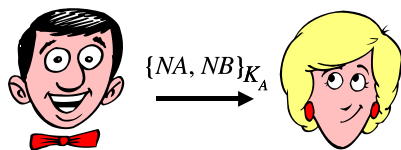
## The Needham-Schroeder Public Key Protocol (NSPK):

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

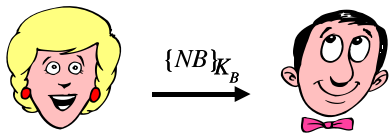
Goal: mutual authentication. Translation:



“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.”



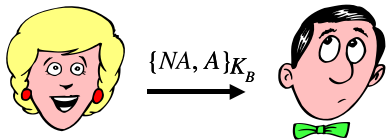
“You sent me  $NB$ . Since only Alice can read this and I sent it back, you must be Alice.”

# An Authentication Protocol

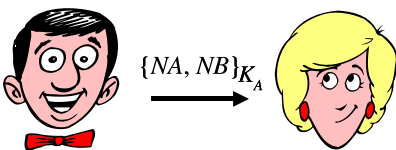
## The Needham-Schroeder Public Key Protocol (NSPK):

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

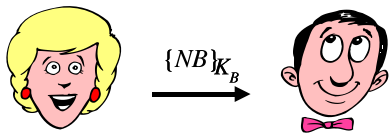
Goal: mutual authentication. Translation:



“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.”

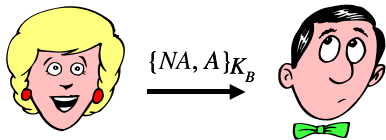
NSPK proposed in 1970s and used for decades, until...

# An Authentication Protocol

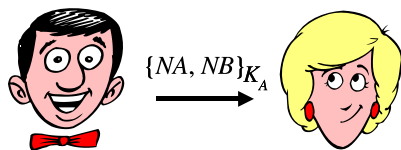
## The Needham-Schroeder Public Key Protocol (NSPK):

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

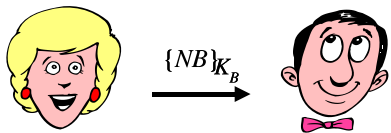
Goal: mutual authentication. Translation:



“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.”

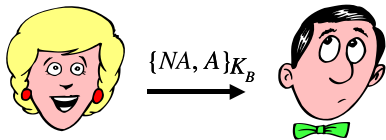
NSPK proposed in 1970s and used for decades, until...  
 Protocols are typically small and convincing...

# An Authentication Protocol

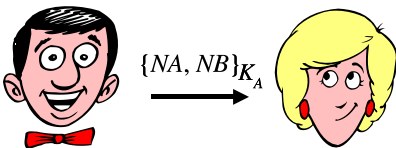
## The Needham-Schroeder Public Key Protocol (NSPK):

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

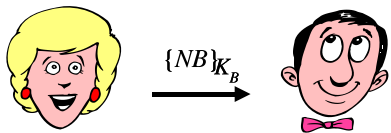
Goal: mutual authentication. Translation:



“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.”



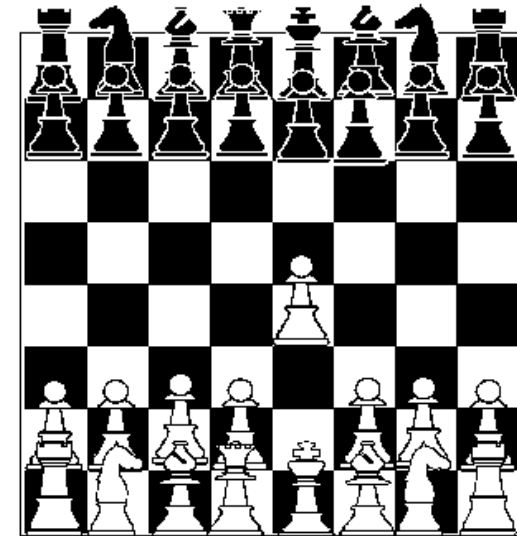
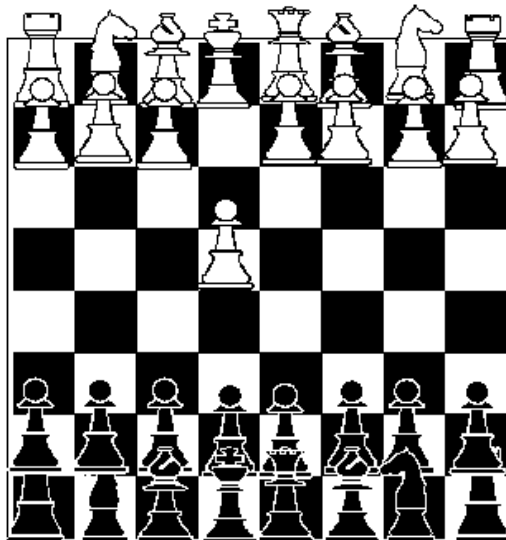
“You sent me  $NB$ . Since only Alice can read this and I sent it back, you must be Alice.”

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

# How to at Least Tie Against a Chess Grandmaster



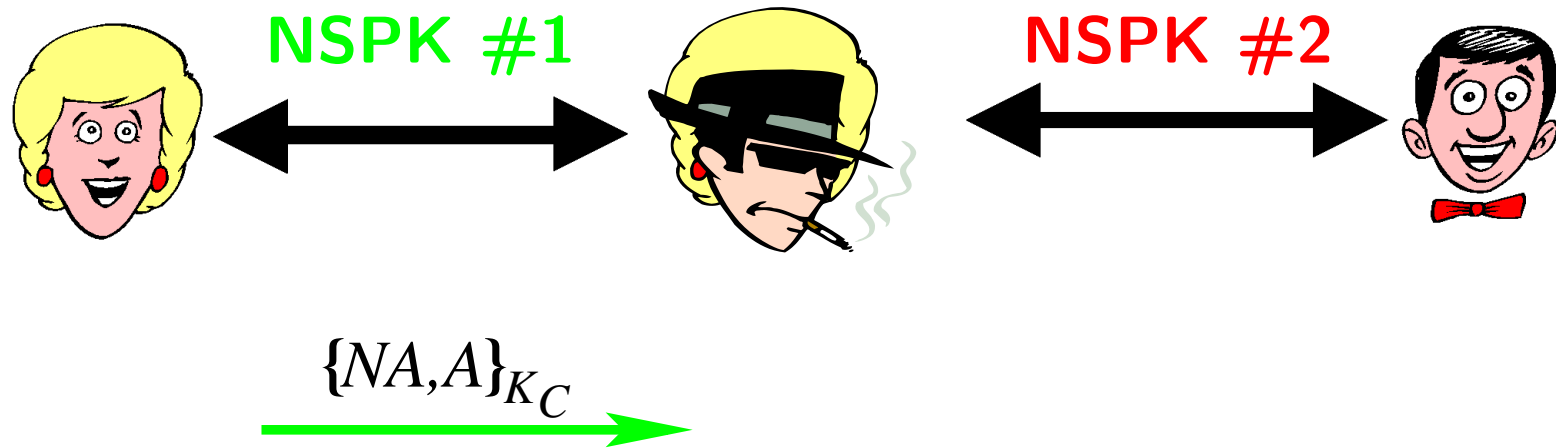
# Man-in-the-Middle Attack

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{K_B} \\ B \rightarrow A &: \{NA, NB\}_{K_A} \\ A \rightarrow B &: \{NB\}_{K_B} \end{aligned}$$



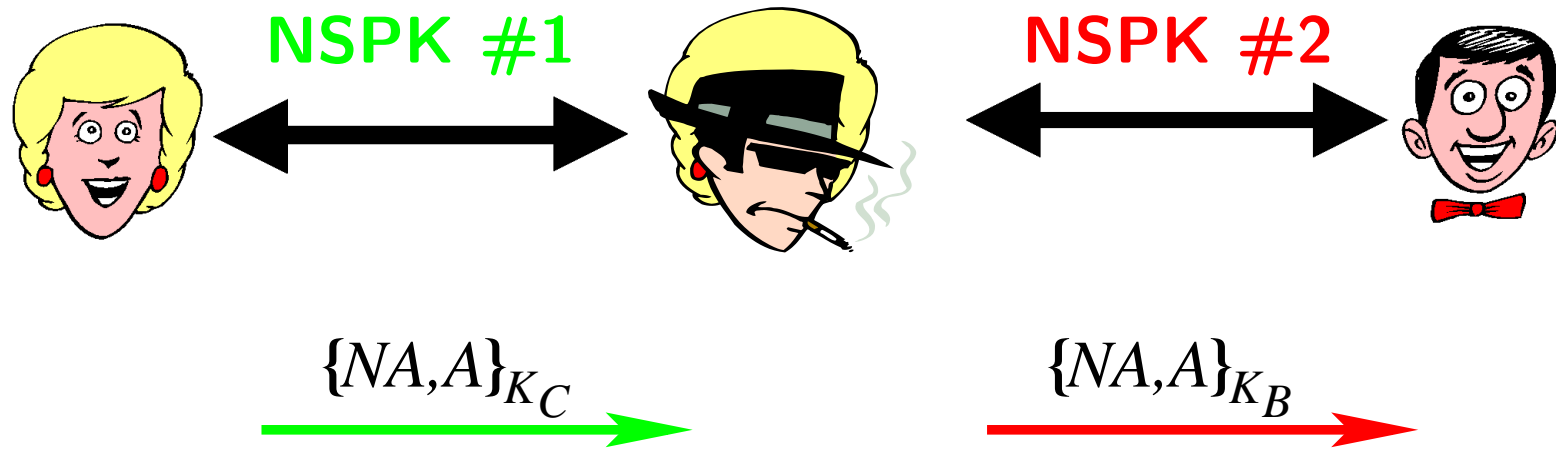
# Man-in-the-Middle Attack

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



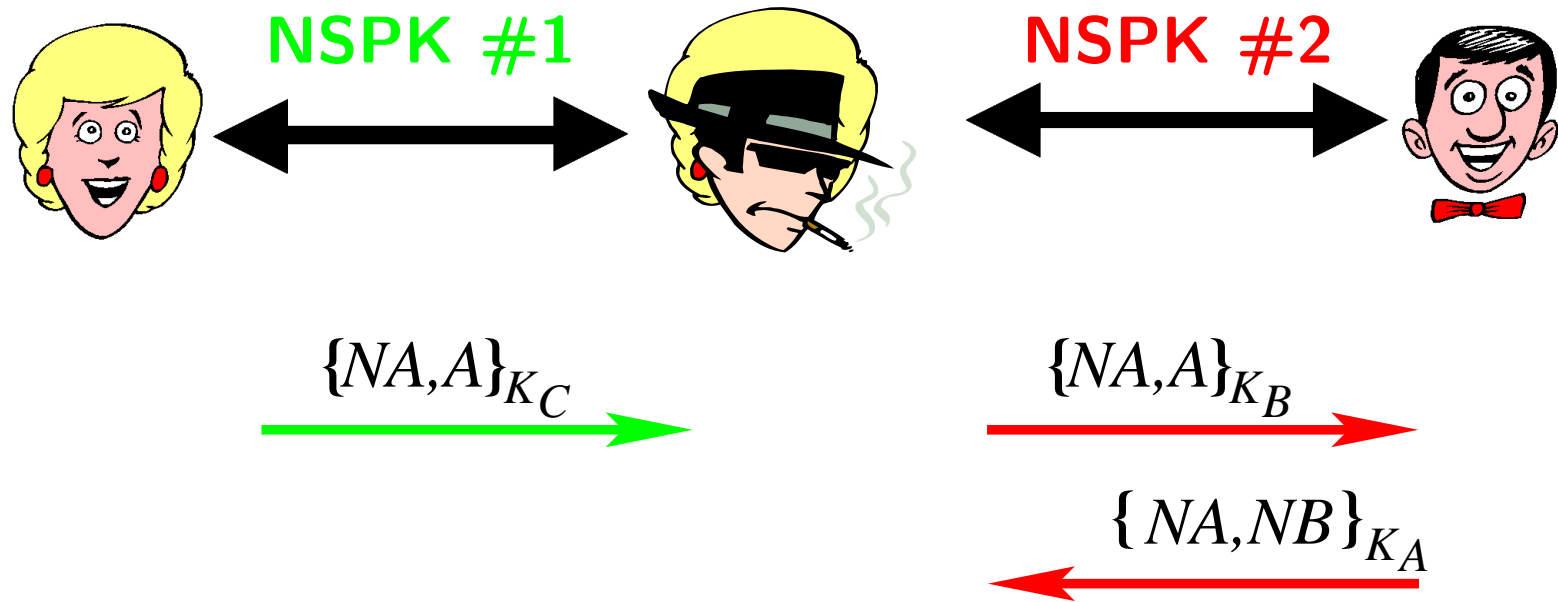
# Man-in-the-Middle Attack

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{K_B} \\ B \rightarrow A &: \{NA, NB\}_{K_A} \\ A \rightarrow B &: \{NB\}_{K_B} \end{aligned}$$



# Man-in-the-Middle Attack

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

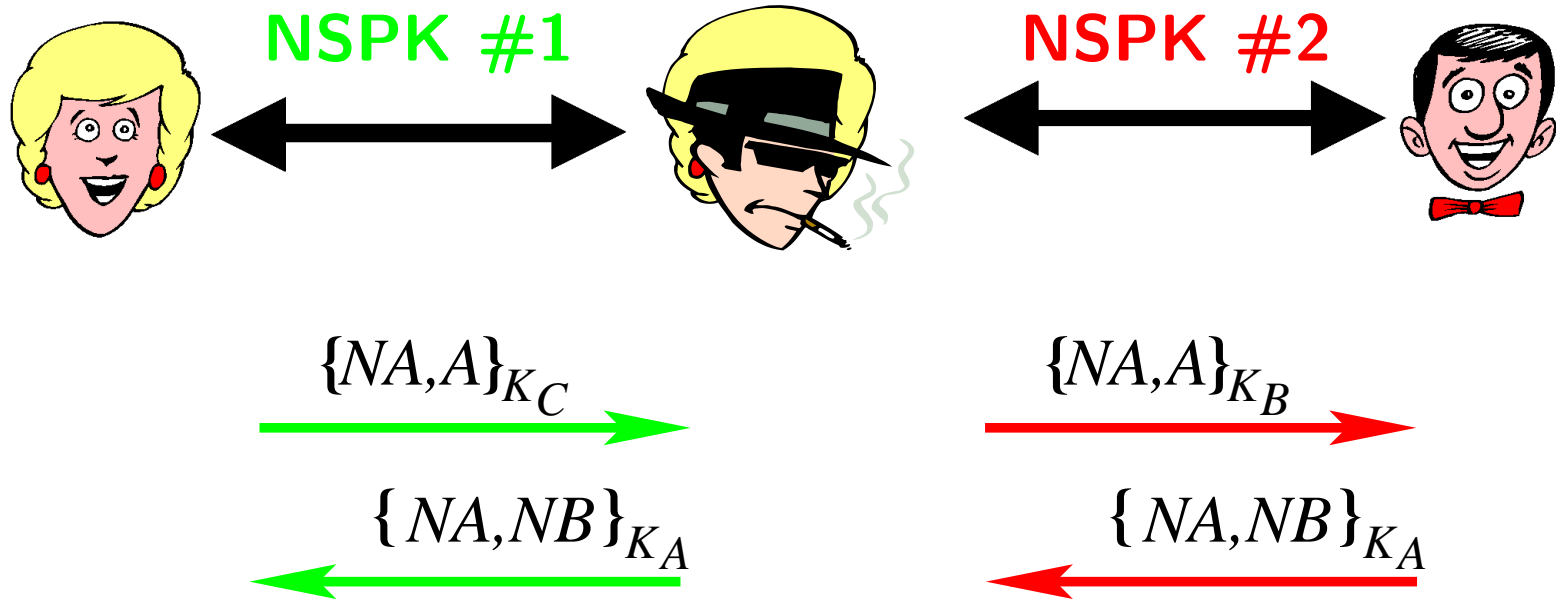


# Man-in-the-Middle Attack

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

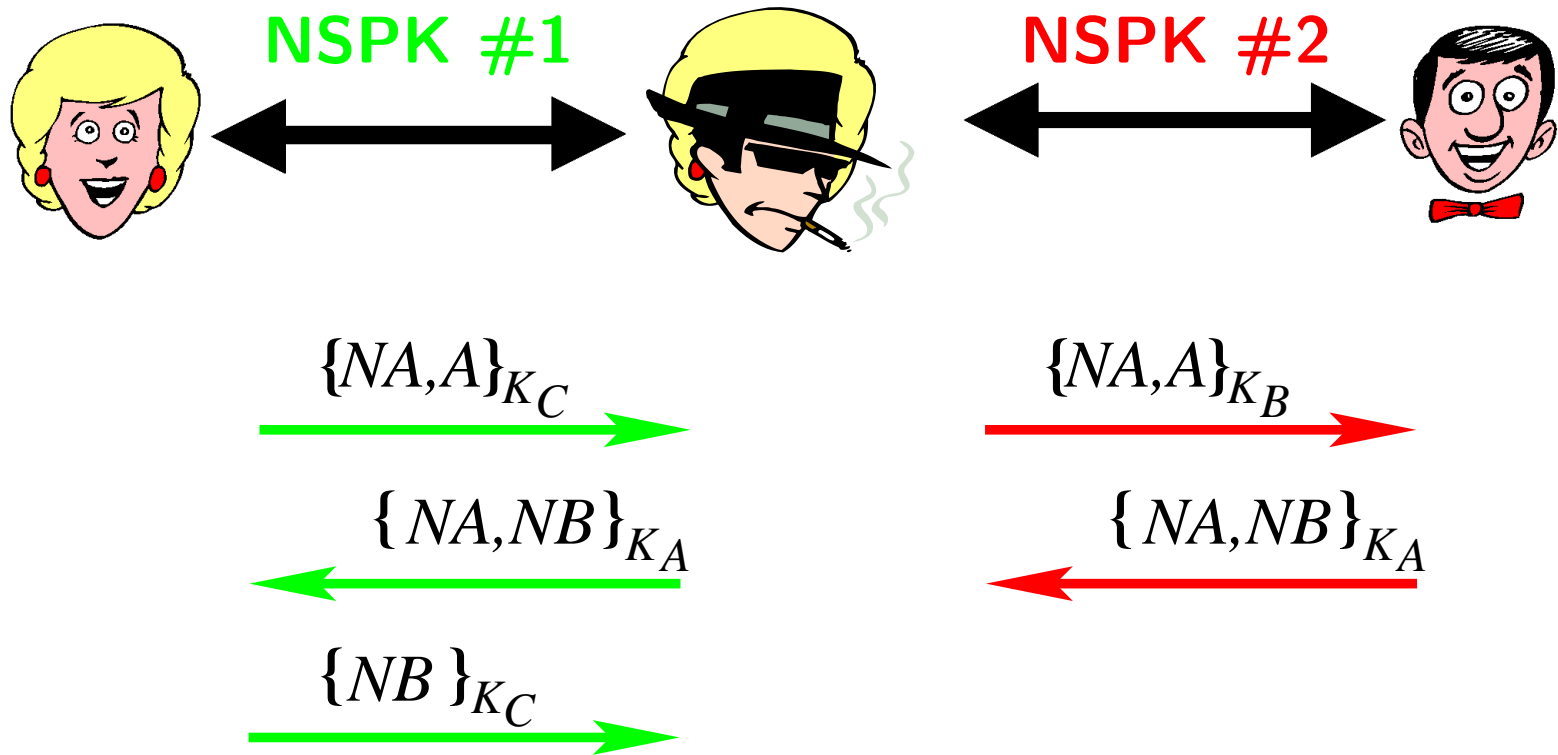
$$B \rightarrow A : \{NA, NB\}_{K_A}$$

$$A \rightarrow B : \{NB\}_{K_B}$$



# Man-in-the-Middle Attack

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

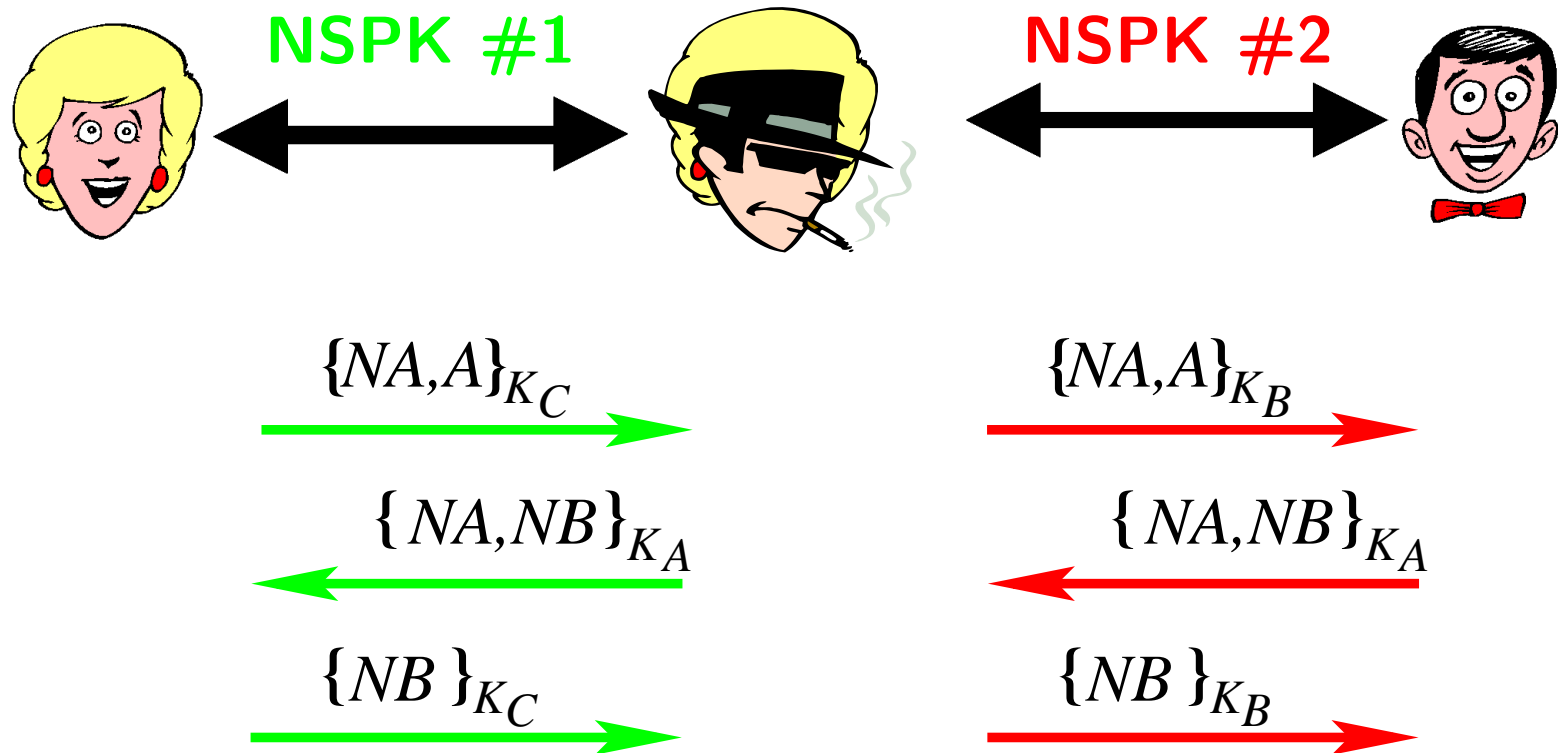


# Man-in-the-Middle Attack

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$



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

- Is the improved version now correct?



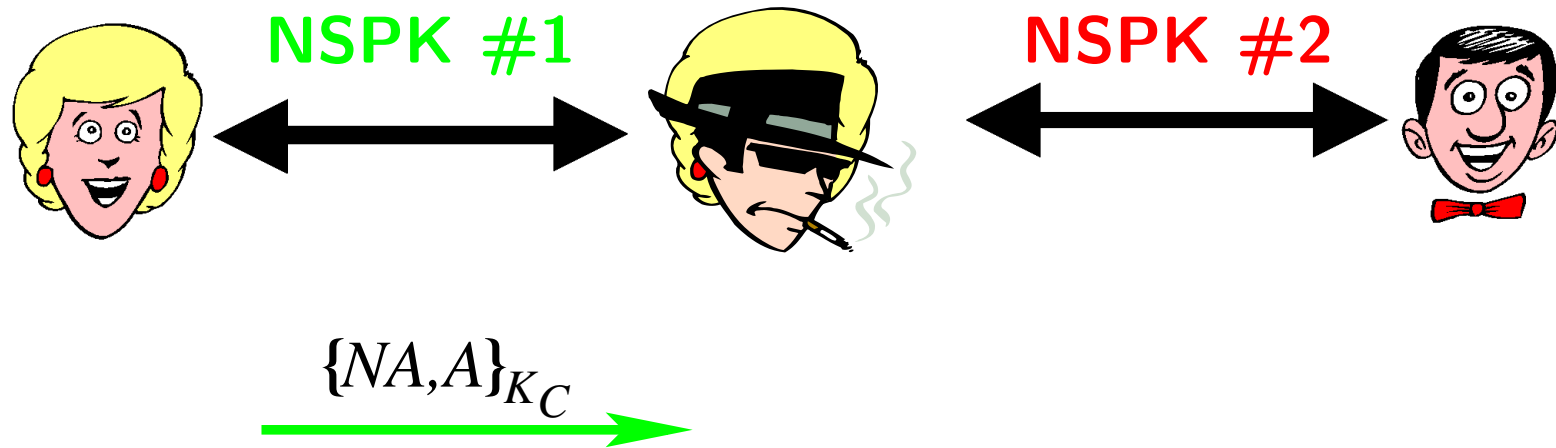
# The NSL Protocol

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{K_B} \\ B \rightarrow A &: \{NA, NB, B\}_{K_A} \\ A \rightarrow B &: \{NB\}_{K_B} \end{aligned}$$



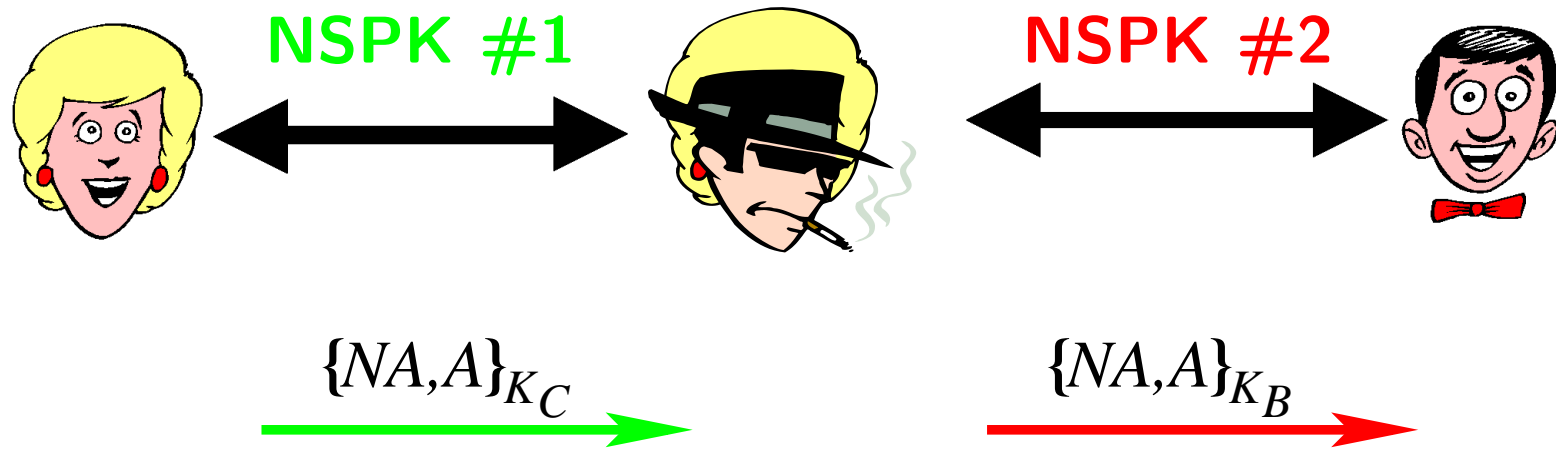
# The NSL Protocol

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{K_B} \\ B \rightarrow A &: \{NA, NB, B\}_{K_A} \\ A \rightarrow B &: \{NB\}_{K_B} \end{aligned}$$



# The NSL Protocol

$$\begin{aligned} A \rightarrow B &: \{NA, A\}_{K_B} \\ B \rightarrow A &: \{NA, NB, B\}_{K_A} \\ A \rightarrow B &: \{NB\}_{K_B} \end{aligned}$$

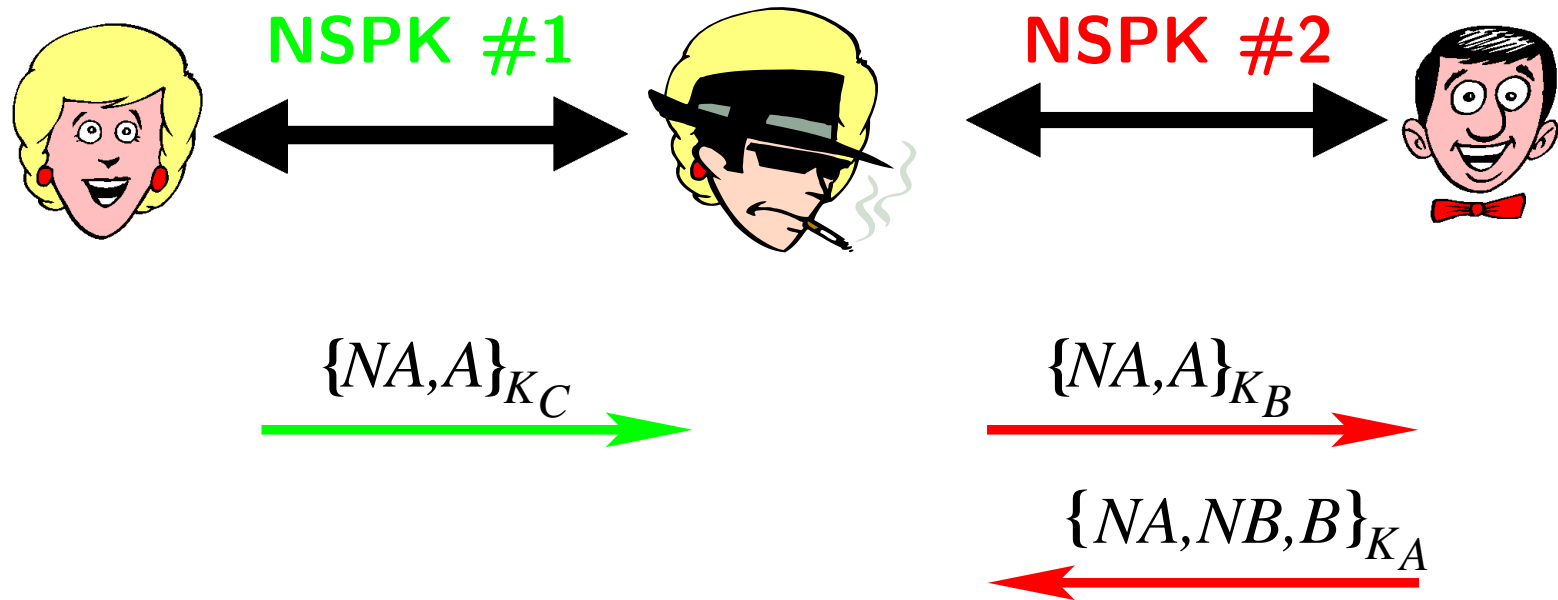


# The NSL Protocol

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

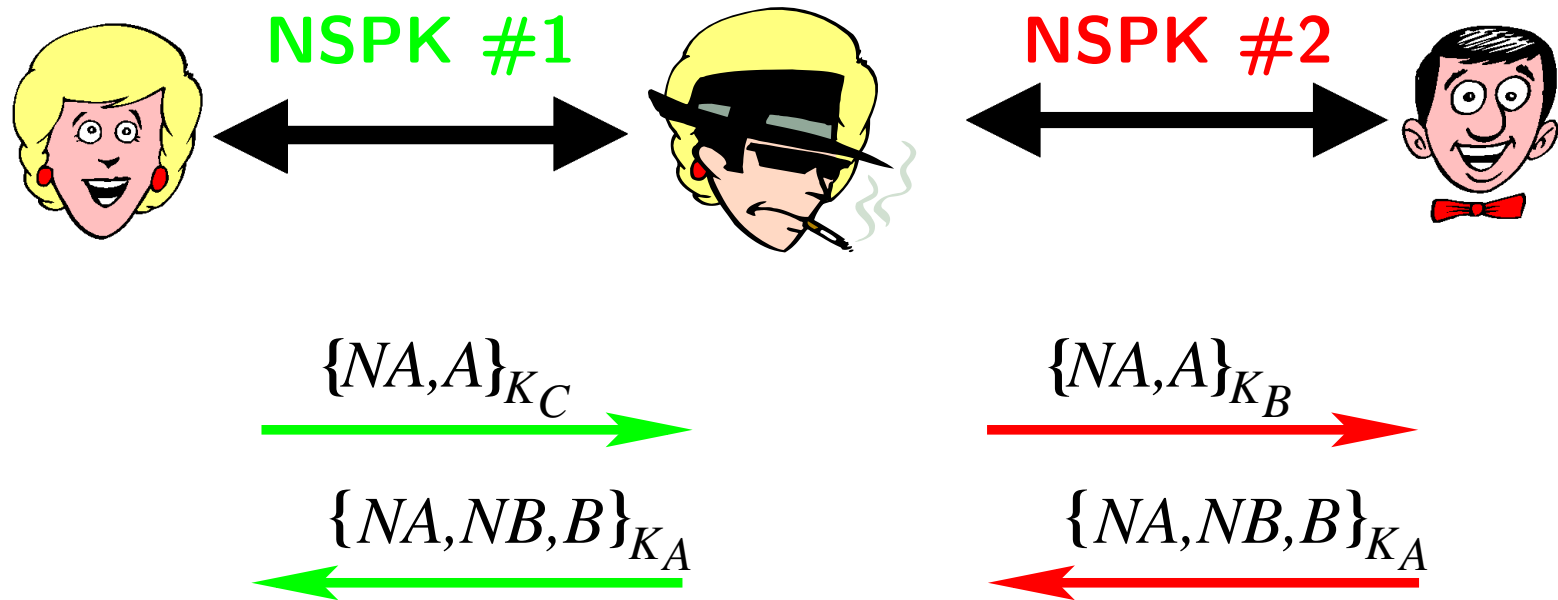


# The NSL Protocol

$$A \rightarrow B : \{NA, A\}_{K_B} \quad 11$$

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

$$A \rightarrow B : \{NB\}_{K_B}$$



**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.**

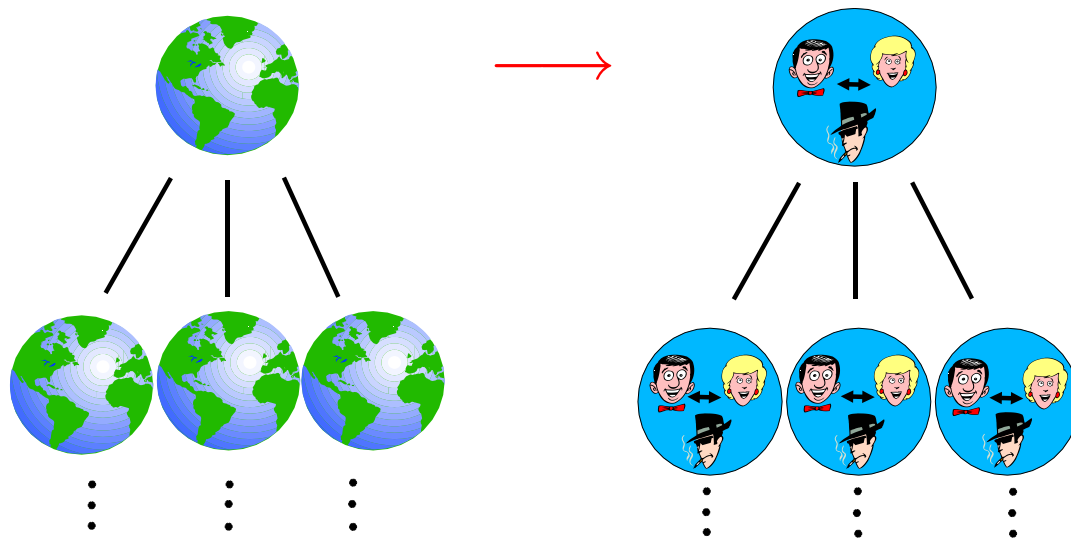
- OFMC (& the AVISPA Tool) in more detail.
- Conclusions.

# Formal Modeling and Analysis of Protocols

Standard protocol notation is not formal!

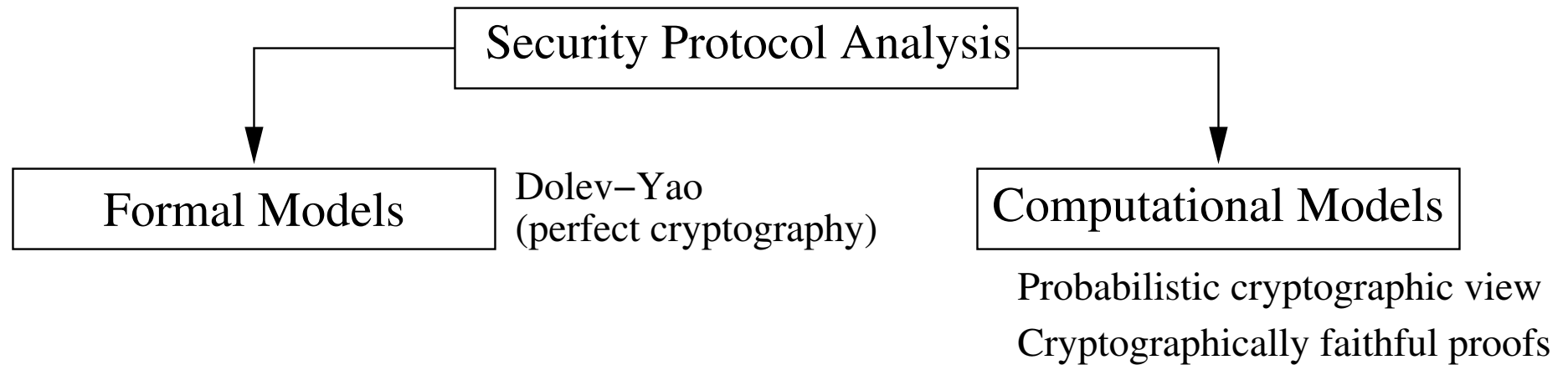
**Goal:** formally model protocols and their properties and provide a mathematically sound means for reasoning about these models.

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



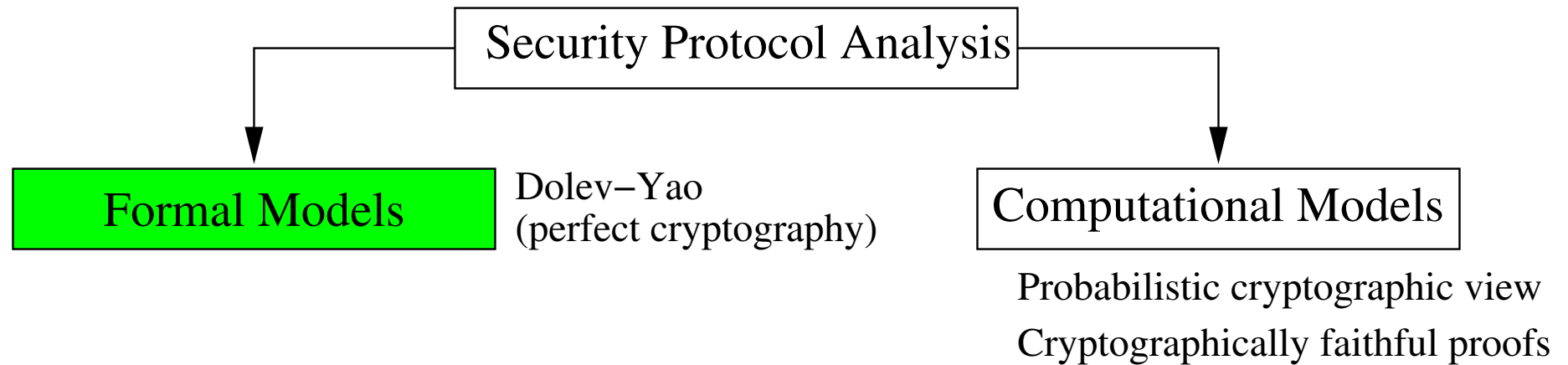
**Analysis:** with formal methods based on mathematics and logic.

# Formal Security Protocol Analysis



Also: semi-formal (engineering) approaches.

# 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.

- Crypt algorithms 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.

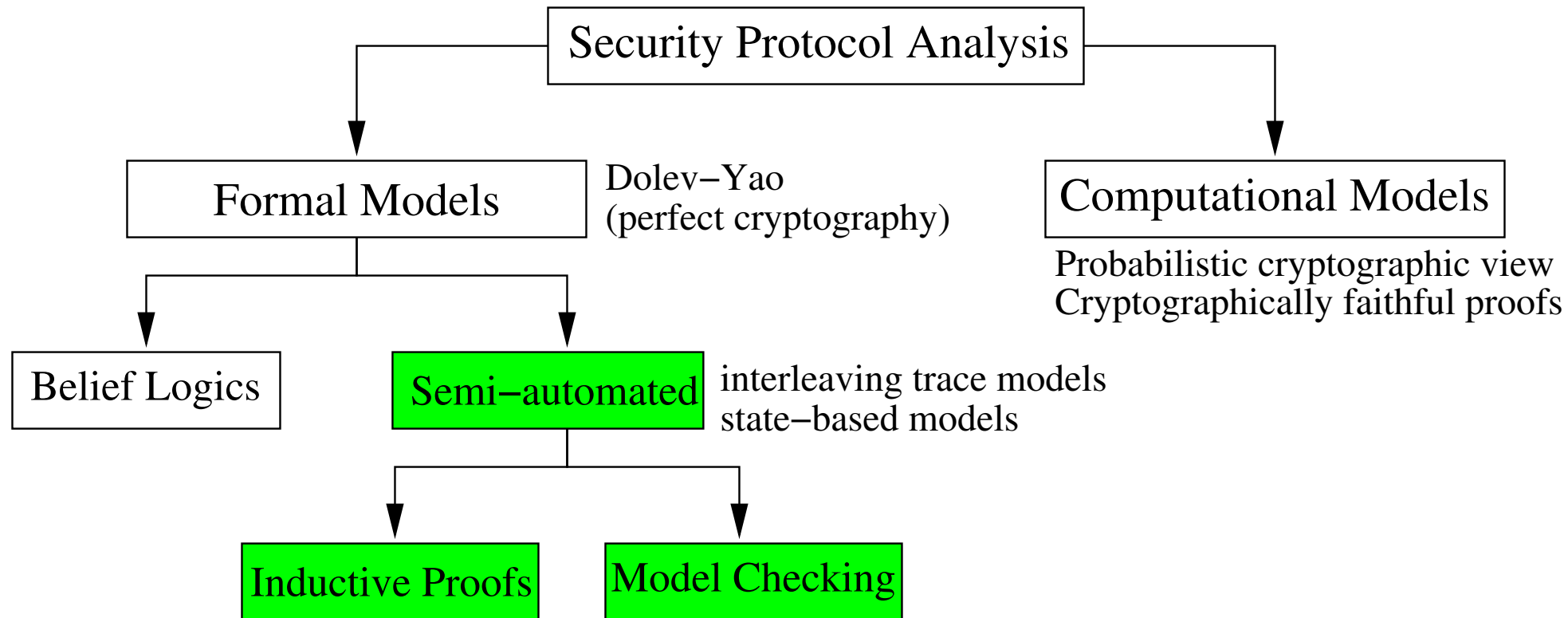
*A friend's just an enemy in disguise. You can't trust nobody.* (C. Dickens, *Oliver Twist*)

- 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.

$$A \rightarrow B : M_1$$

$$B \rightarrow A : M_2$$

⋮

# Interleaving Trace Models

- Modeling idea: model possible communication events.

$$A \rightarrow B : M_1$$

$$C \rightarrow D : P_1$$

$$B \rightarrow A : M_2$$

$$C \rightarrow D : P_2$$

⋮

# Interleaving Trace Models

- Modeling idea: model possible communication events.

$$\begin{array}{ccc}
 A \rightarrow B : M_1 & A \rightarrow C : P_1 & A \rightarrow B : M_1 \\
 C \rightarrow D : P_1 & A \rightarrow B : M_1 & B \rightarrow A : M_2 \\
 i \rightarrow A : M_2 & \text{or } B \rightarrow A : M_2 & \text{or } C \rightarrow D : P_1 \quad \dots \\
 C \rightarrow D : P_2 & C \rightarrow A : P_2 & i \rightarrow C : P_1 \\
 \vdots & \vdots & \vdots
 \end{array}$$

- 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 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:

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

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

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

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

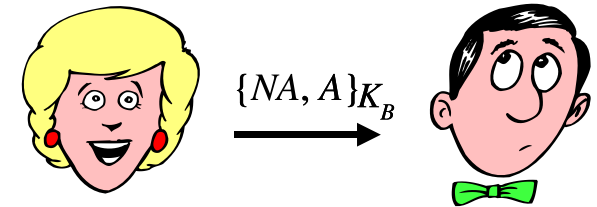
$$\frac{\{\!|m_2|\!\}_{m_1^{-1}} \in \mathcal{DY}(M) \quad m_1 \in \mathcal{DY}(M)}{m_2 \in \mathcal{DY}(M)} A_{\text{crypt}}^{-1}$$

# 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”

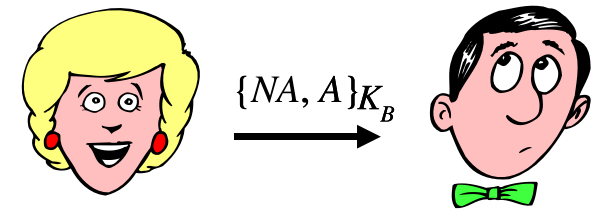
- Several **semi-automated tools** have been developed to analyze protocols under the **perfect cryptography assumption**, **but** (in most cases) they are limited to small and medium-scale protocols.
  - ▶ For example, **Clark/Jacob protocol library**:  
NSPK, NSSK, Otway-Rees, Yahalom,  
Woo-Lam, Denning-Sacco, ...



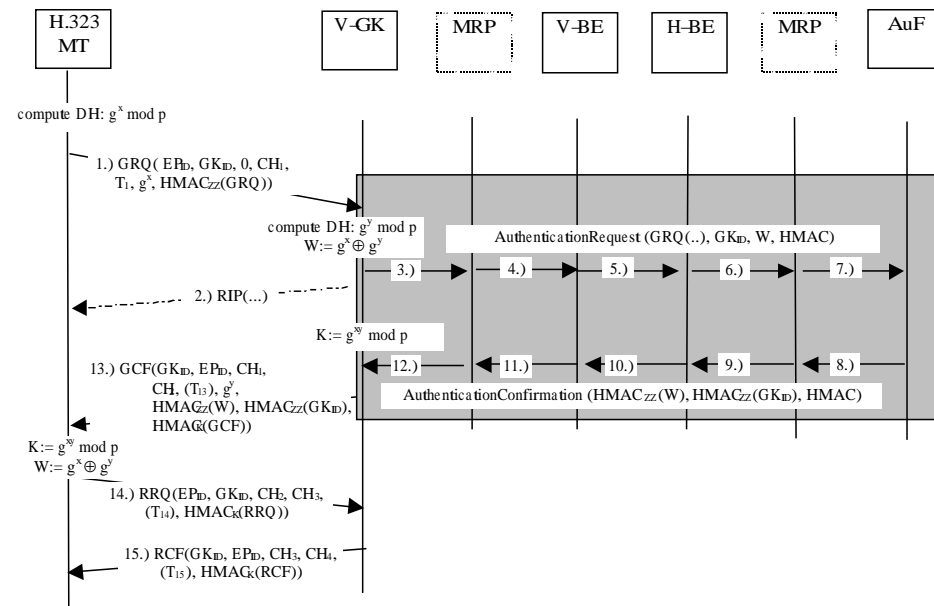
# The State of the Art... “Yesterday”

- 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](http://www.avispa-project.org)

The screenshot displays the AVISPA Web Tool interface in a Mozilla browser window. The main content area is titled "Protocol" and shows the following details:

```

% PROTOCOL: H. 530: Symmetric security procedures
%           for H. 323 mobility in H. 510
% PURPOSE: Establish an authenticated (Diffie-Hellman)
%           shared-key between a mobile terminal (MT) and a visited
%           gate-keeper (VGK) who don't know each other, but who know
%           an authentication facility (AuF) in MT's home domain.
% REFERENCE: \url(http://www.itu.int/rec/recommendation.asp?typ
% ALICE_BOB:
% 1. MT -> VGK : M1.F(ZZ,M1)
% 2. VGK -> AuF : M2.F(ZZ_VA,M2)
% 3. AuF -> VGK : M3.F(ZZ_VA,M3)
% 4. VGK -> MT : M4.F(exp(exp(G,X),Y),M4)
% 5. MT -> VGK : M5.F(exp(exp(G,X),Y),M5)
% 6. VGK -> MT : M6.F(exp(exp(G,X),Y),M6)
-----
% M1 = MT.VGK.NIL.CH1.exp(G,X)
% M2 = M1.F(ZZ,M1).VGK.exp(G,X).XOR.exp(G,Y)
% M3 = VGK.MT.F(ZZ.VGK).F(ZZ.exp(G,X).XOR.exp(G,Y))
    
```

Below the protocol details is a "Tools" section with a tree structure:

```

Tools
├── HLP5L
│   ├── HLP5L2IF
│   └── F
│       ├── OFMC
│       ├── ATSE
│       ├── SATMC
│       └── TR45P
    
```

On the right side, an Emacs window displays the HLP5L role definition:

```

Mode
** Emacs: H.530.hlp5l
File Edit View Cmds Tools Options Buffers AVISPA Help

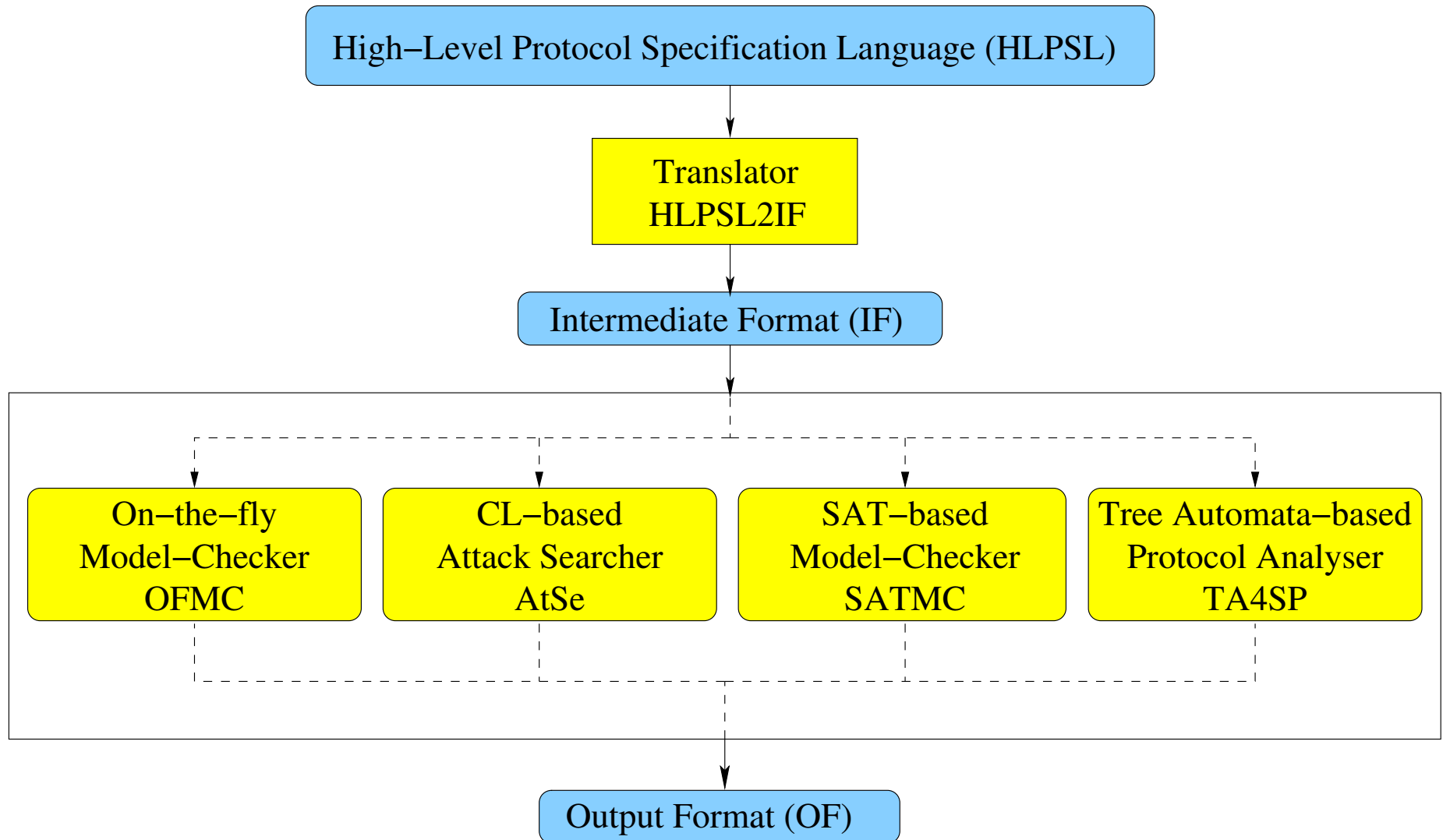
Role VisitedGateKeeper (
  MT,VGK,AuF : agent,
  SND,RCV   : channel(dy),
  F         : function,
  ZZ_VA    : symmetric_key,
  NIL,G    : text)
played by VGK def=
local
  State      : nat,
  GK,Key,Key1,FM1,FM2,FM3,M2 : message,
  Y,CH2,CH4  : text (fresh),
  CH1,CH3    : text
init State = 0
transition
1. State = 0 /\ RCV(MT.VGK.NIL.CH1'.GX'.FM1') =|>
  State' = 1 /\ Key' = exp(GX',Y')
  /\ M2' = MT.VGK.NIL.CH1'.GX'.FM1'.VGK.xor(GX',exp(G,Y'))
  /\ SND(M2'.F(ZZ_VA.M2'))
  /\ witness(VGK.MT,key',Key')
2. State = 1 /\ RCV(VGK.MT.FM2'.FM3'.F(ZZ_VA.VGK.MT.FM2'.FM3')) =|>
  State' = 2 /\ SND(VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'.
  F(Key.VGK.MT.CH1.CH2'.exp(G,Y).FM3'.FM2'))
  /\ RCV(MT.VGK.CH2.CH3'.F(Key.MT.VGK.CH2.CH3')) =|>
  State' = 3 /\ SND(VGK.MT.CH3'.CH4'.F(Key.VGK.MT.CH3'.CH4'))
  /\ request(VGK.MT,key1,Key)
  /\ secret(Key,MT)

end role
IS08-----XEmacs: H.530.hlp5l (HLP5L -- AVISPA Font)----L1--C0--All--
    
```

At the bottom, an "msc ATTACK TRACE" diagram shows the interaction between three agents: Agent i, Agent (a.3), and Agent (a.7). The trace includes the following messages:

- Agent i sends `start` to Agent (a.3).
- Agent i sends `a.b.nil.CH1(1).exp(g.X(1)).f(zz.a.auf.a.b.nil.CH1(1).exp(g.X(1)))` to Agent (a.3).
- Agent (a.3) sends `b.a.nil.x99.g.x92` to Agent (a.7).
- Agent (a.7) sends `b.a.nil.x99.g.x92.a.g` to Agent (a.3).
- Agent i sends `a.b.nil.CH1(1).exp(g.X(1)).f(zz.a.auf.a.b.nil.CH1(1).exp(g.X(1)))` to Agent (a.3).
- Agent (a.3) sends `a.b.x99.CH2(3).exp(g.Y(2)).CH1(1).exp(g.X(1)).nil.f(exp(g.Y(2)).a.b.x99.CH2(3).exp(g.Y(2)).CH1(1).exp(g.X(1)).nil)` to Agent (a.7).
- Agent (a.7) sends `b.a.CH2(3).x102.f(exp(g.Y(2)).b.a.CH2(3).x102)` to Agent (a.3).
- Agent (a.3) sends `a.b.x102.CH4(4).f(exp(g.Y(2)).a.b.x102.CH4(4))` to Agent (a.7).

# 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.

**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.

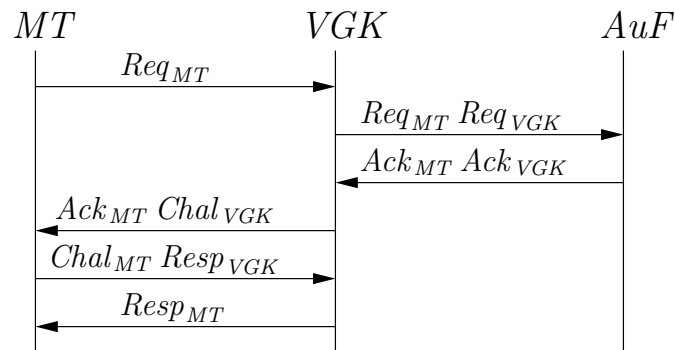
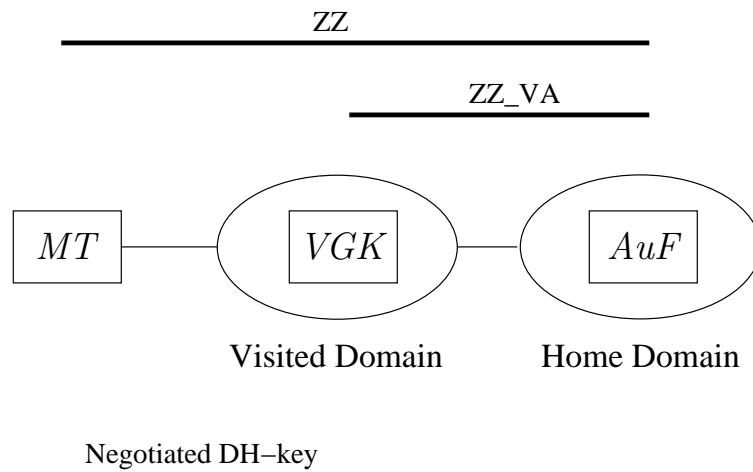
# The AVISPA Tool and the AVISPA Library: Results <sup>30</sup>

- 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.

# A Simple Example: the H.530 Protocol



1. MT  $\rightarrow$  VGK : MT, VGK, NIL, CH1,  $G^{\wedge}DHX$ ,  
 $F(ZZ, MT, VGK, NIL, CH1, G^{\wedge}DHX)$
2. VGK  $\rightarrow$  AuF : MT, VGK, NIL, CH1,  $G^{\wedge}DHX$ ,  
 $F(ZZ, MT, VGK, NIL, CH1, G^{\wedge}DHX)$ ,  
 VGK,  $G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY$ ,  
 $F(ZZ\_VA, MT, VGK, NIL, CH1, G^{\wedge}DHX)$ ,  
 $F(ZZ, MT, VGK, NIL, CH1, G^{\wedge}DHX)$ ,  
 VGK,  $G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY$
3. AuF  $\rightarrow$  VGK : VGK, MT,  $F(ZZ, VGK)$ ,  
 $F(ZZ, G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY)$ ,  
 $F(ZZ\_VA, VGK, MT, F(ZZ, VGK))$ ,  
 $F(ZZ, G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY)$
4. VGK  $\rightarrow$  MT : VGK, MT, CH1, CH2,  $G^{\wedge}DHY$ ,  
 $F(ZZ, G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY)$ ,  
 $F(ZZ, VGK)$ ,  
 $F((G^{\wedge}DHX)^{\wedge}DHY, VGK, MT, CH1, CH2, G^{\wedge}DHY)$ ,  
 $F(ZZ, G^{\wedge}DHX \text{ XOR } G^{\wedge}DHY), F(ZZ, VGK)$
5. MT  $\rightarrow$  VGK : MT, VGK, CH2, CH3,  
 $F((G^{\wedge}DHX)^{\wedge}DHY, MT, VGK, CH2, CH3)$
6. VGK  $\rightarrow$  MT : VGK, MT, CH3, CH4,  
 $F((G^{\wedge}DHX)^{\wedge}DHY, VGK, MT, CH3, CH4)$

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](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 HPSL/IF (e.g. translator from HPSL 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

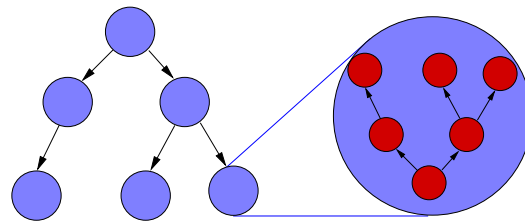
- Challenging as general problem is **undecidable**.
- Several **sources of infinity** in protocol analysis:
  - ▶ Unbounded **number of possible intruder messages** (unbounded **message depth**).
  - ▶ Unbounded **number of sessions** or protocol steps (and **agents**).
- Possible approaches:
  - ▶ **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

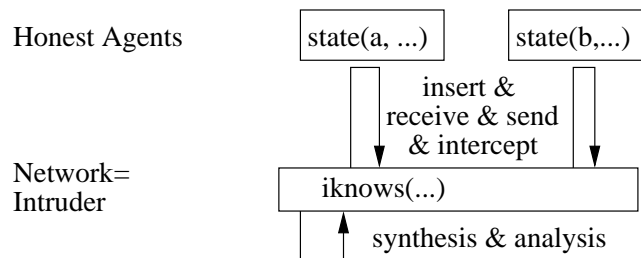
- Challenging as general problem is **undecidable**.
- Several **sources of infinity** in protocol analysis:
  - ▶ Unbounded **number of possible intruder messages** (unbounded **message depth**).
  - ▶ Unbounded **number of sessions** or protocol steps (and **agents**).
- **OFMC**: an On-the-Fly Model-Checker 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 36

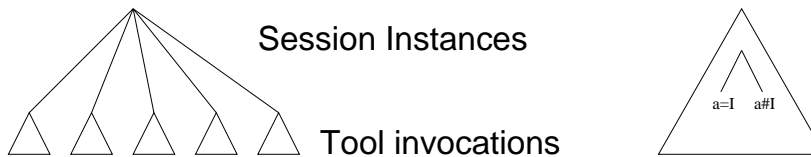


- The Lazy Intruder

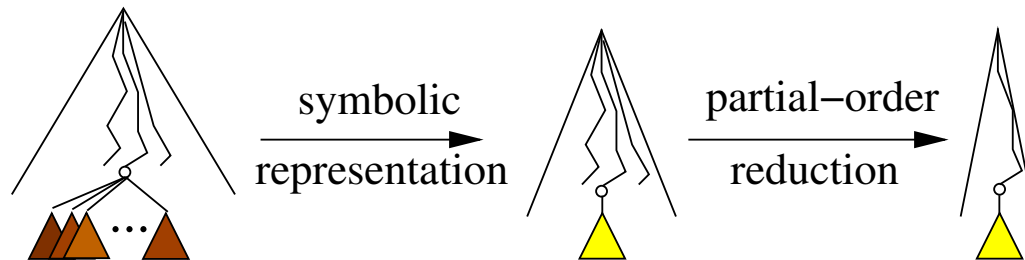


- Compressions

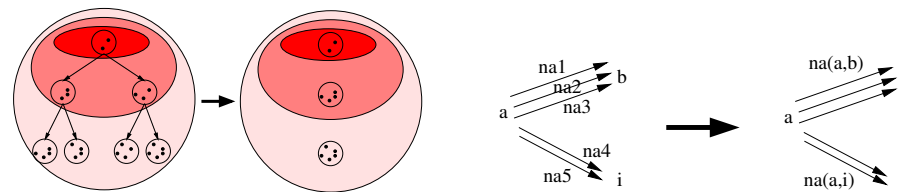
- Symbolic Sessions



- 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.  $A \rightarrow B : M, A, B, \{ \llbracket NA, M, A, B \rrbracket \}_{K_{AS}}$

# The Lazy Intruder: Idea

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

## The Lazy Intruder: Idea

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

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

## The Lazy Intruder: Idea

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

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

Idea: postpone this decision.

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

*IK*: current Intruder Knowledge

## The Lazy Intruder: Idea

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

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

Idea: postpone this decision.

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

*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:

$$\mathit{from}(T, IK)$$

- $\llbracket \mathit{from}(T, IK) \rrbracket = \{ \sigma \mid \text{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma)) \}$

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

- Semantics hence relates *from*-constraints to the Dolev-Yao model.

# The Lazy Intruder: Formally

- Constraints of the lazy intruder:

$$from(T, IK)$$

- $\llbracket from(T, IK) \rrbracket = \{\sigma \mid \text{ground}(T\sigma \cup IK\sigma) \wedge (T\sigma \subseteq \mathcal{DY}(IK\sigma))\}$

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

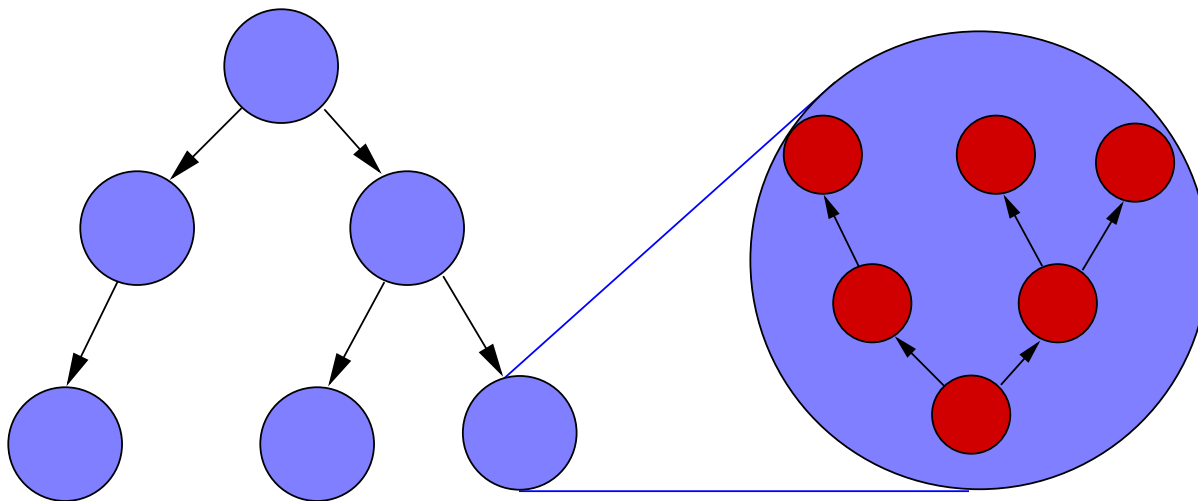
- Semantics hence relates *from*-constraints to the Dolev-Yao model.
- **Theorem.** Satisfiability of (well-formed) *from*-constraints is decidable.
- A restriction on the depth of messages is not necessary.
- Non-atomic keys can easily be handled.

# Integration: Symbolic Transition System

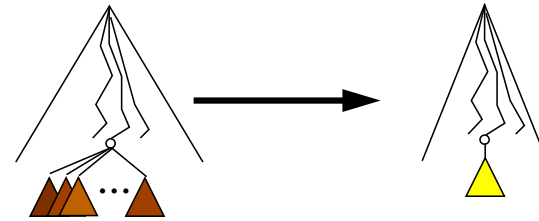
- **Symbolic state** = term with variables + constraint set
- $\llbracket (t, C) \rrbracket = \{t\sigma \mid \sigma \in \llbracket C \rrbracket\}$  (a set of ground states).
- Two layers of search:

**Layer 1:** search in the symbolic state space

**Layer 2:** constraint reduction



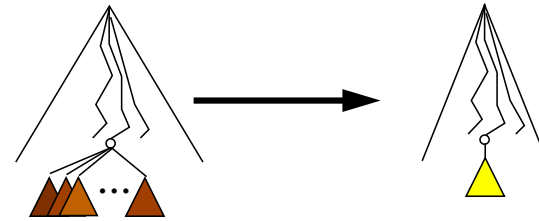
# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$\begin{aligned}
 A \rightarrow B &: \{NA, A\}_{K_B} \\
 B \rightarrow A &: \{NA, NB\}_{K_A} \\
 A \rightarrow B &: \{NB\}_{K_B}
 \end{aligned}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

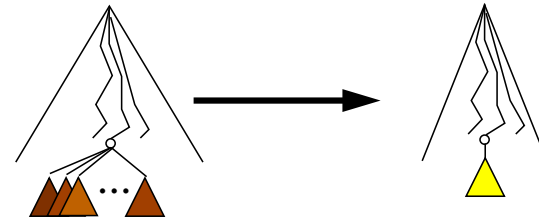
$$a \rightarrow I : \{na, a\}_{K_I}$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$a \rightarrow I : \{na, a\}_{K_I}$$

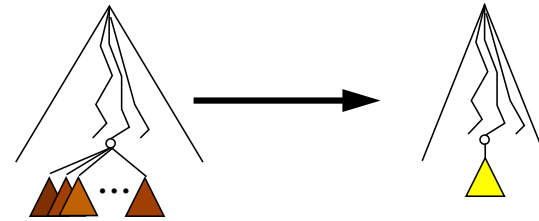
$$I \rightarrow b : X_1$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$a \rightarrow I : \{na, a\}_{K_I}$$

$$I \rightarrow b : X_1$$

$$b \rightarrow I : \{X_2, nb\}_{K_{X_3}}$$

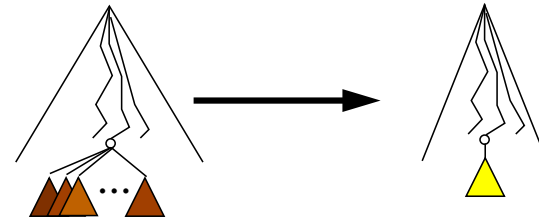
$$X_1 = \{X_2, X_3\}_{K_b}$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$a \rightarrow I : \{na, a\}_{K_I}$$

$$I \rightarrow b : \{X_2, X_3\}_{K_b}$$

$$b \rightarrow I : \{X_2, nb\}_{K_{X_3}}$$

$$I \rightarrow a : \{X_2, nb\}_{K_{X_3}}$$

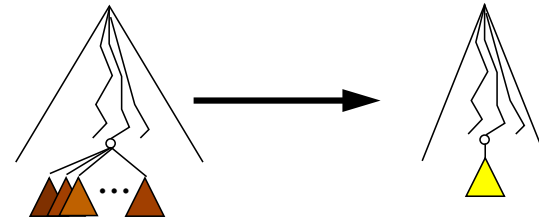
$$X_1 = \{X_2, X_3\}_{K_b}$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$a \rightarrow I : \{na, a\}_{K_I}$$

$$I \rightarrow b : \{X_2, X_3\}_{K_b}$$

$$b \rightarrow I : \{X_2, nb\}_{K_{X_3}}$$

$$I \rightarrow a : \{X_2, nb\}_{K_{X_3}}$$

$$a \rightarrow I : \{nb\}_{K_I}$$

$$X_1 = \{na, a\}_{K_b}$$

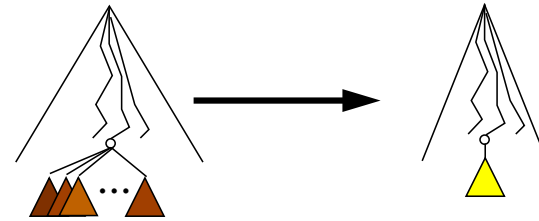
$$X_2 = na, X_3 = a$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# NSPK and the Lazy Intruder



We allow messages to contain **variables** and employ **unification**.

$$a \rightarrow I : \{na, a\}_{K_I}$$

$$I \rightarrow b : \{na, a\}_{K_b}$$

$$b \rightarrow I : \{na, nb\}_{K_a}$$

$$I \rightarrow a : \{na, nb\}_{K_a}$$

$$a \rightarrow I : \{nb\}_{K_I}$$

$$I \rightarrow b : \{nb\}_{K_b}$$

$$X_1 = \{na, a\}_{K_b}$$

$$X_2 = na, X_3 = a$$

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

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

$$A \rightarrow B : \{NB\}_{K_B}$$

# Road Map

- Introduction.
- Security protocols.
- Formal Protocol Analysis.
- 👉 **OFMC (& the AVISPA Tool) in more detail.**
  - ▶ Lazy Intruder.
  - ▶ **Constraint Differentiation.**
- Conclusions.

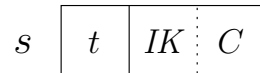
# Two Key Challenges and their Solutions

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.

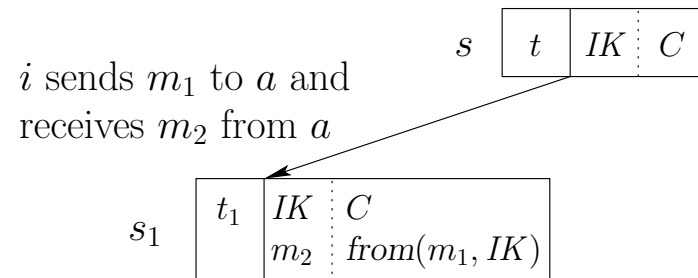
# Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



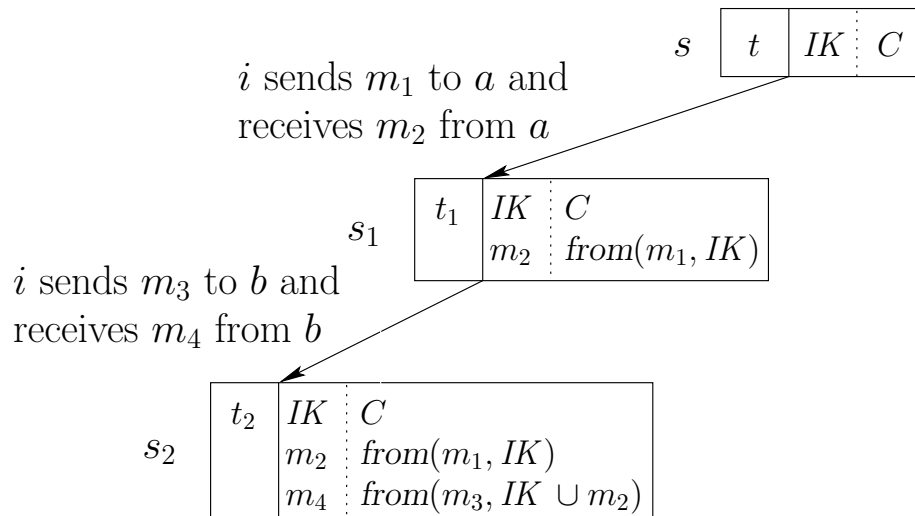
## Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



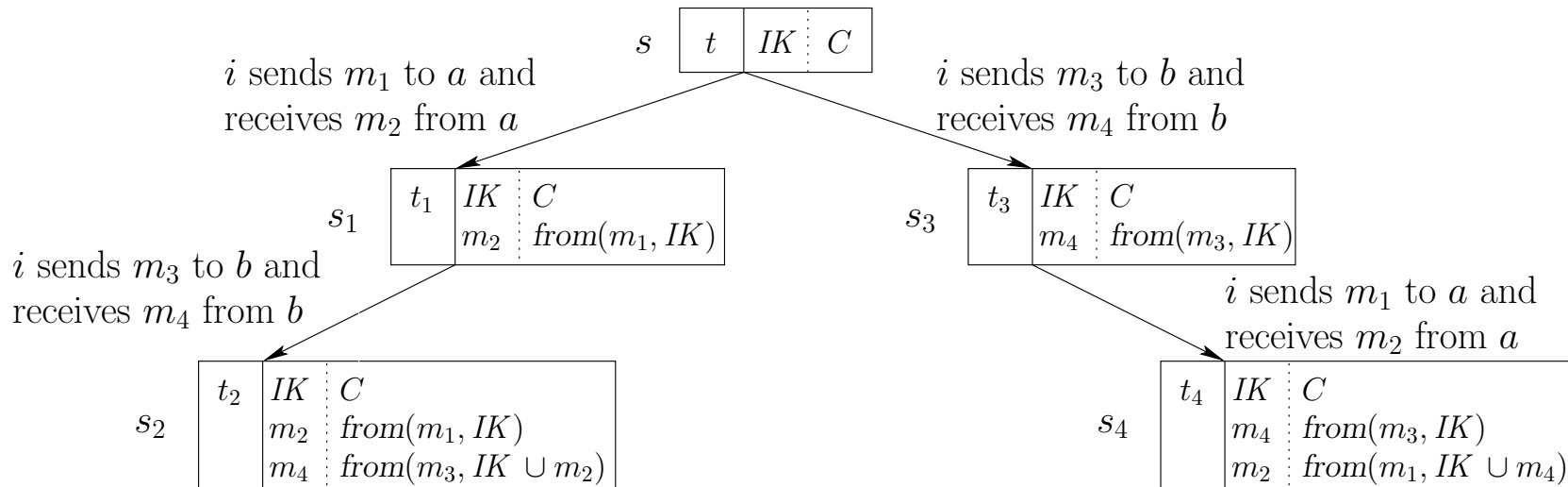
# Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



# Constraint Differentiation: Idea

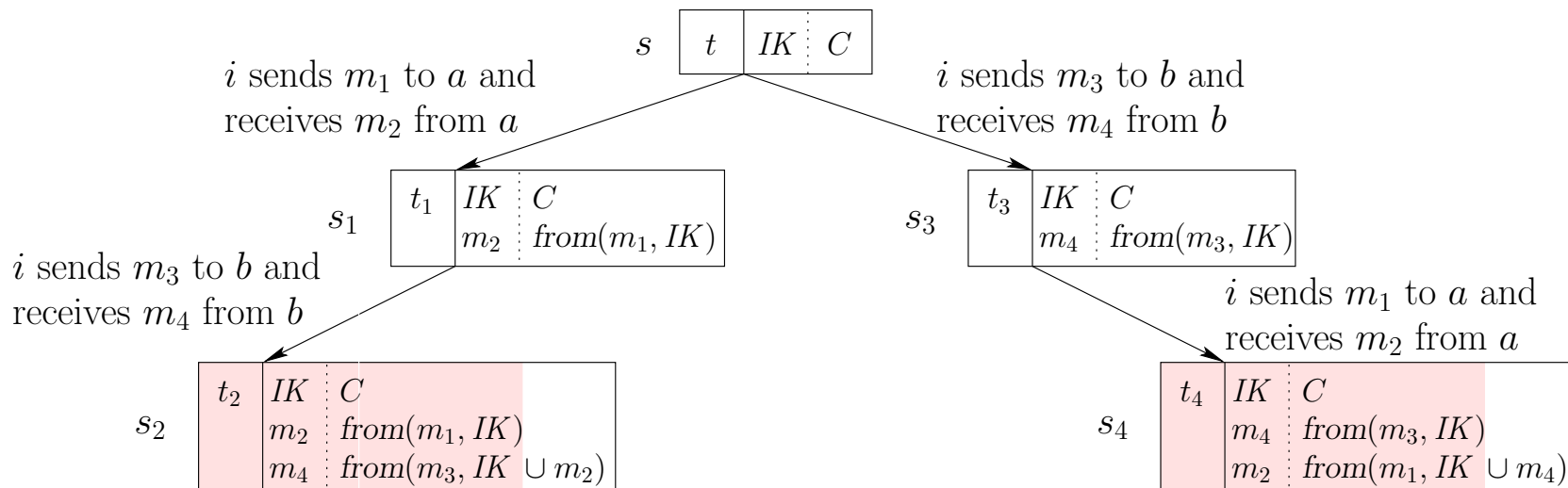
Typical situation: 2 independent actions executable in either order:



(where  $t_2 = t_4$ )

# Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:

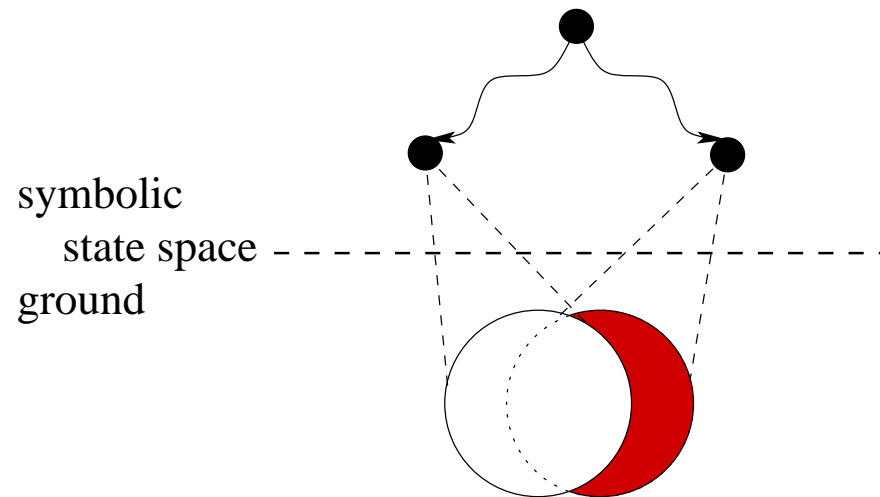


(where  $t_2 = t_4$ )

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

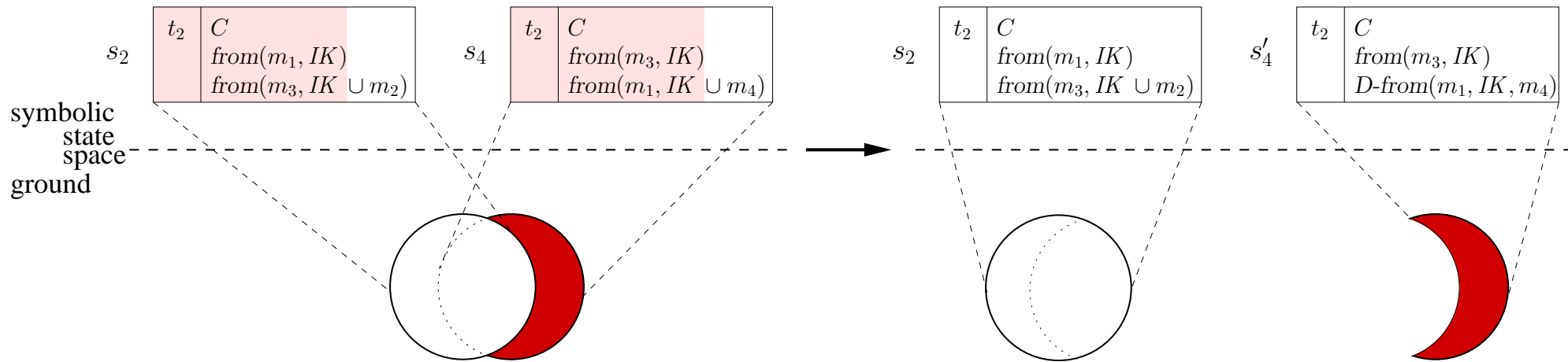
## Constraint Differentiation: Idea

Typical situation: 2 independent actions executable in either order:



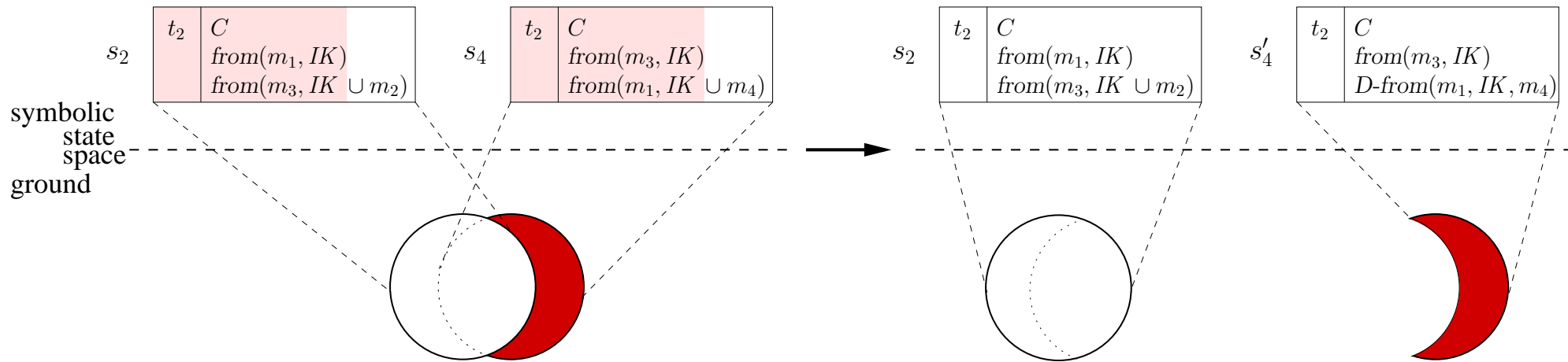
Idea: exploit redundancies in the symbolic states, i.e. reduction exploits overlapping of the sets of ground states.

# Constraint Differentiation (1)



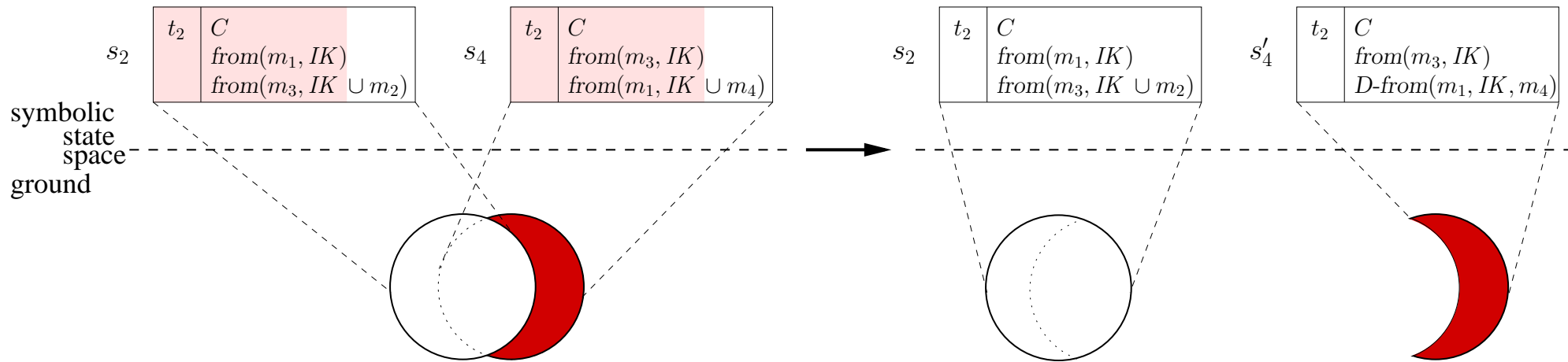
- New kind of constraints:  $D\text{-from}(T, IK, NIK)$ .
- Intuition:
  - ▶ Intruder has just learned some **new intruder knowledge**  $NIK$ .

# Constraint Differentiation (1)



- New kind of constraints:  $D-from(T, IK, NIK)$ .
- Intuition:
  - ▶ Intruder has just learned some **new intruder knowledge**  $NIK$ .
  - ▶ All solutions  $\llbracket from(T, IK \cup NIK) \rrbracket$  are **“correct”**

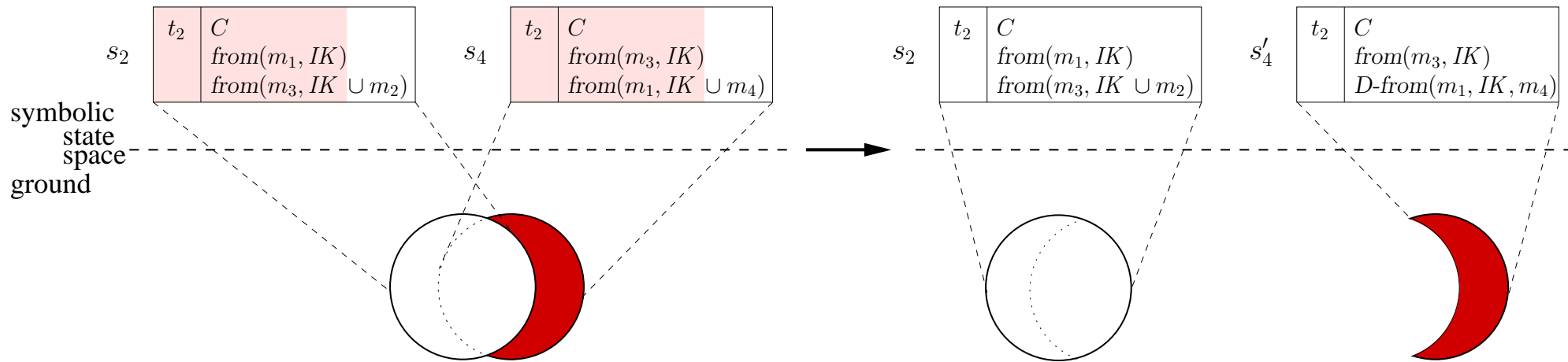
# Constraint Differentiation (1)



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

$$\llbracket D-from(T, IK, NIK) \rrbracket = \llbracket from(T, IK \cup NIK) \rrbracket \setminus \llbracket from(T, IK) \rrbracket.$$

# Constraint Differentiation (2)



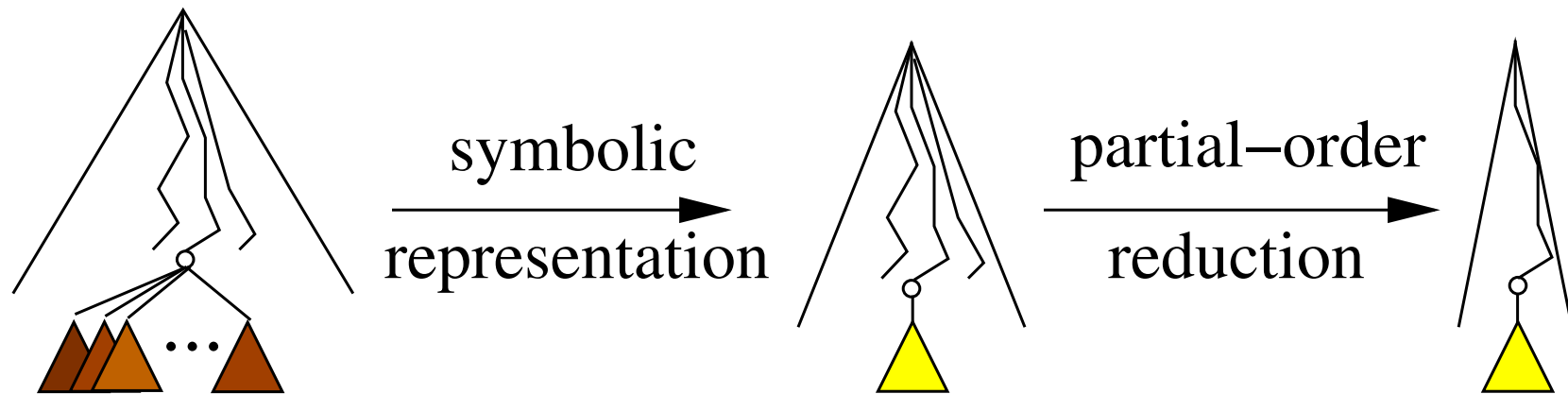
- $\llbracket D\text{-from}(T, IK, NIK) \rrbracket = \llbracket \text{from}(T, IK \cup NIK) \rrbracket \setminus \llbracket \text{from}(T, IK) \rrbracket$
- **Theorem.** *Satisfiability of (well-formed) D-from constraints is decidable.*
- **Theorem.**  $\llbracket s_2 \rrbracket \cup \llbracket s_4 \rrbracket = \llbracket s_2 \rrbracket \cup \llbracket s'_4 \rrbracket$

# Constraint Differentiation: Experimental Results

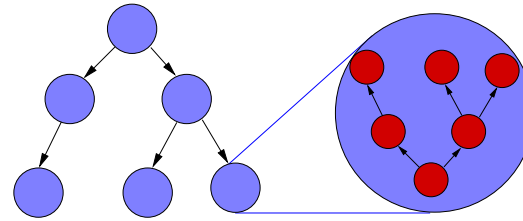
IKE Aggressive Mode Pre-Shared Key without and with CD: the nodes for each ply of the search tree and search time

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	

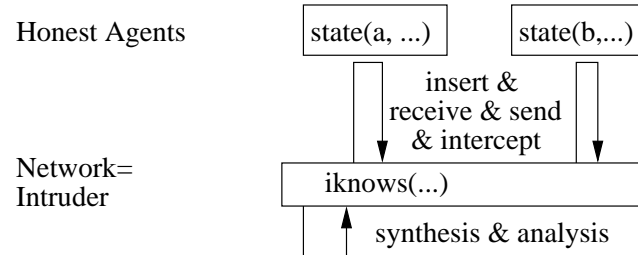
# Lazy Intruder and Constraint Reduction



# Graphical Overview of some Symbolic Reductions 55

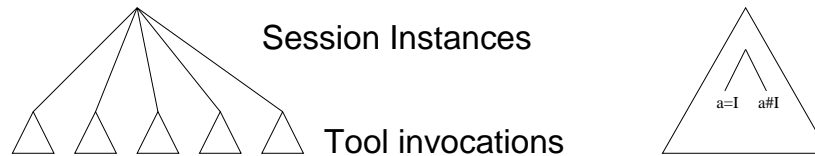


- The Lazy Intruder

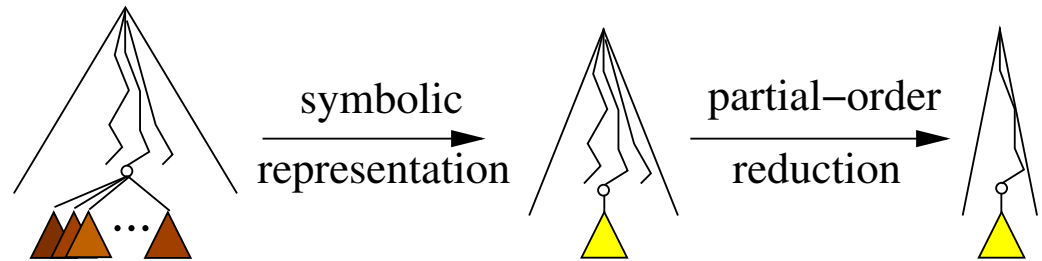


- Compressions

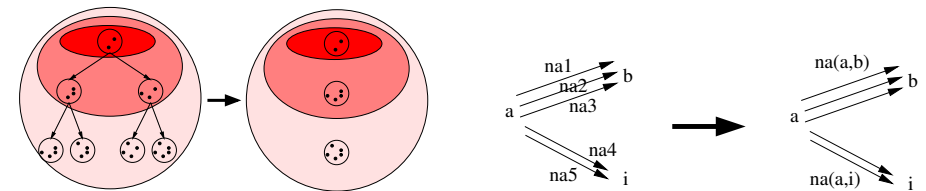
- Symbolic Sessions



- 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

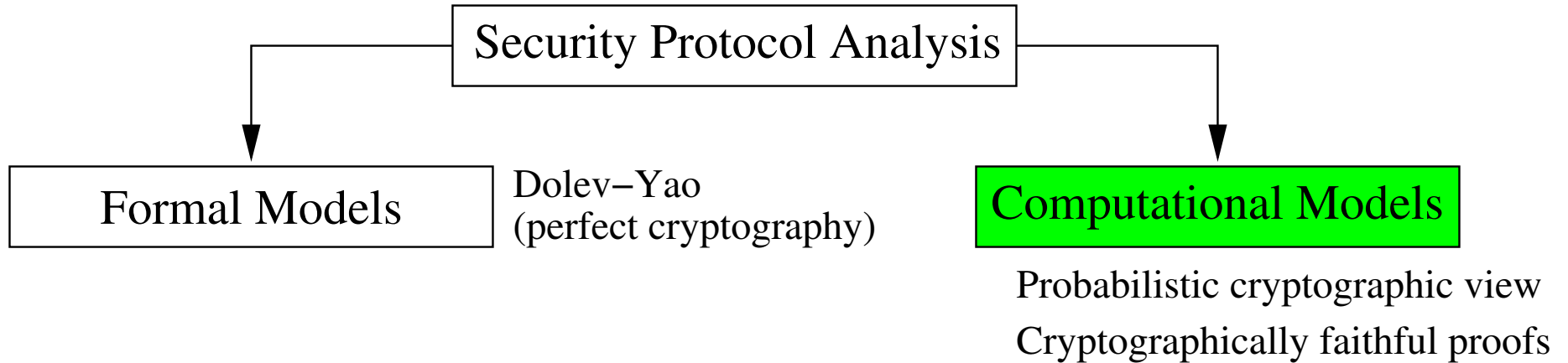
- The AVISPA Tool is a state-of-the-art, integrated environment for the automatic validation of Internet security protocols.  
**AVISPA package** (& web-interface): [www.avispa-project.org](http://www.avispa-project.org)
- Current work:
  - ▶ Extending the AVISPA library with further protocols and properties.
  - ▶ 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 HPSL/IF (e.g. translator from HPSL to Applied Pi Calculus to then apply ProVerif).

- **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.

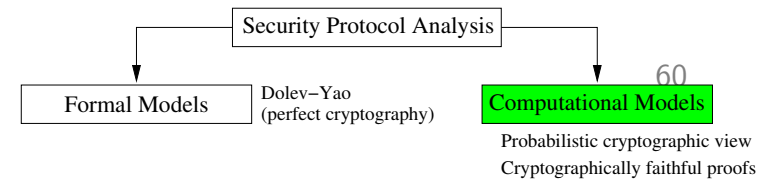
Both analyses have positive results: they demonstrate that **at the abstraction level of each analysis, the protocol is error-free**.

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

# Computational Models



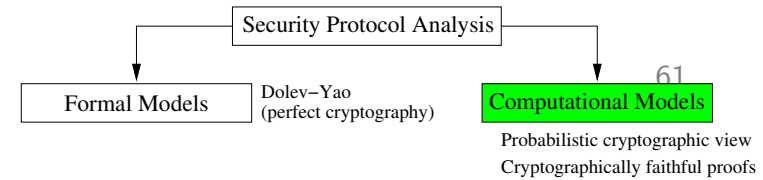
# Computational Models



- 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 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.