Automated Validation of Security Protocols (AVASP)

Jorge Cuellar   Sebastian Mödersheim   Luca Viganò

Plan of the day

09:00 – 10:00 Introduction to security protocols (LV)

10:00 – 12:30 Internet protocols (JC)

14:00 – 15:00 Formal methods for security protocol analysis (LV)

15:00 – 17:30 Methods for automated protocol analysis (SM)

Road map

- Introduction to security protocols.
  - ▶ Motivation.
  - ▶ A brief introduction to cryptography.
  - ▶ Basic notions.
  - ▶ An example: Needham-Schroeder Public Key Protocol.
  - ▶ Protocol attacks.
- Formal methods for security protocol analysis.

N.B.: this is a survey, and as such is partial and incomplete.

Road map

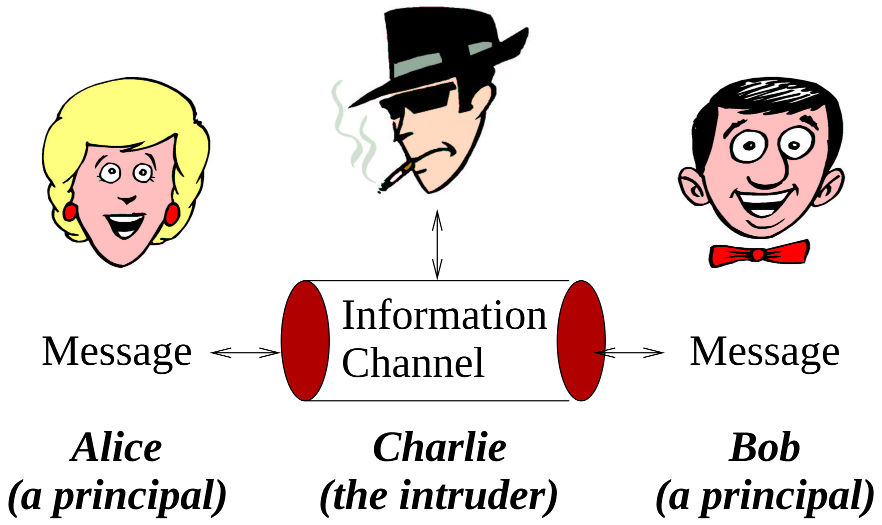
☞ **Introduction to security protocols.**

- ▶ Motivation.
- ▶ A brief introduction to cryptography.
- ▶ Basic notions.
- ▶ An example: Needham-Schroeder Public Key Protocol.
- ▶ Protocol attacks.

- Formal methods for security protocol analysis.

N.B.: this is a survey, and as such is partial and incomplete.

Motivation



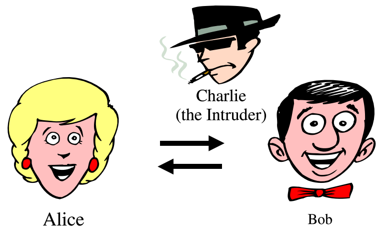
- How do we turn untrustworthy channels into trustworthy ones?

  Confidentiality: Transmitted information remains secret.
  Integrity: Information not corrupted (or alterations detected).
  Authentication: Principals know who they are speaking to.

- Cryptography is the enabling technology.

Motivation — Examples

- **Example:** Securing an e-banking application.



$A \rightarrow B$: "Send \$10,000 to account X"
$B \rightarrow A$: "I'll transfer it now"

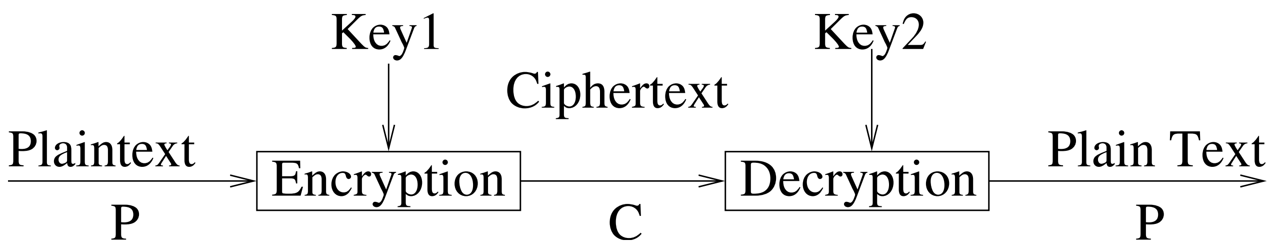
  - How does B know the message originated from A?
  - How does B know A just said it?

- **Other examples:**
  - Constructing secure channels in wireless networks.
  - A micro-payment scheme for E-Commerce.
  - An access control system for area-wide ski-lifts.

- How can one build distributed algorithms for doing this?

  Solutions involve protocols like IPSEC, KERBEROS, SSH, SSL, SET, PGP... which exploit cryptographic algorithms.

General cryptographic schema



where $E_{\mathrm{Key1}}(\mathrm{P}) = \mathrm{C}, \;\; D_{\mathrm{Key2}}(\mathrm{C}) = \mathrm{P}$

- Symmetric algorithms.
  - Key1 $=$ Key2, or are easily derived from each other.
- Asymmetric or public key algorithms.
  - Different keys, which cannot be derived from each other.
  - Public key can be published without compromising private key.
- Encryption and decryption should be easy, if keys are known.
- Security depends on secrecy of the key, not the algorithm.

Encryption/decryption

- $\mathcal{A}$, the alphabet, is a finite set.
- $\mathcal{M} \subseteq \mathcal{A}^*$ is the message space. $M \in \mathcal{M}$ is a plaintext (message).
- $\mathcal{C}$ is the ciphertext space, whose alphabet may differ from $\mathcal{M}$.
- $\mathcal{K}$ denotes the key space of keys.
- Each $e \in \mathcal{K}$ determines a bijective function from $\mathcal{M}$ to $\mathcal{C}$, denoted by E_e. E_e is the encryption function (or transformation).
- For each $d \in \mathcal{K}$, D_d denotes a bijection from $\mathcal{C}$ to $\mathcal{M}$.
  D_d is the decryption function.
- Applying E_e (or D_d) is called encryption (or decryption).

Encryption/decryption (cont.)

- An encryption scheme (or cipher) consists of a set $\{E_e : e \in \mathcal{K}\}$ and a corresponding set $\{D_d : d \in \mathcal{K}\}$ with the property that for each $e \in \mathcal{K}$ there is a unique $d \in \mathcal{K}$ such that $D_d = E_e^{-1}$; i.e.,

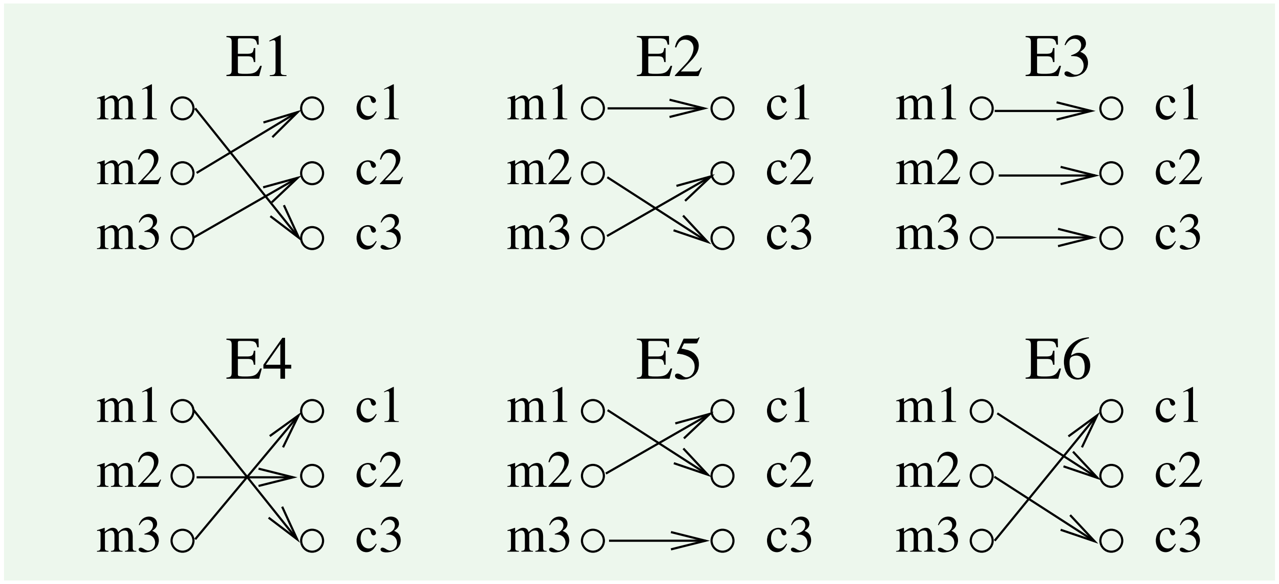
$$D_d(E_e(m)) = m \text{ for all } m \in \mathcal{M}\,.$$

- The keys e and d above form a key pair, sometimes denoted by (e, d). They can be identical (i.e., the symmetric key).

- To construct an encryption scheme requires fixing a message space $\mathcal{M}$, a ciphertext space $\mathcal{C}$, and a key space $\mathcal{K}$, as well as encryption transformations $\{E_e : e \in \mathcal{K}\}$ and corresponding decryption transformations $\{D_d : d \in \mathcal{K}\}$.

An example

Let $\mathcal{M} = \{m_1, m_2, m_3\}$ and $\mathcal{C} = \{c_1, c_2, c_3\}$. There are 3! $= 6$ bijections from $\mathcal{M}$ to $\mathcal{C}$. The key space $\mathcal{K} = \{1, 2, 3, 4, 5, 6\}$ specifies these transformations.



Suppose Alice and Bob agree on the transformation E_1. To encrypt m_1, Alice computes $E_1(m_1) = c_3$. Bob decrypts c_3 by reversing the arrows on the diagram for E_1 and observing that c_3 points to m_1.

Symmetric key encryption

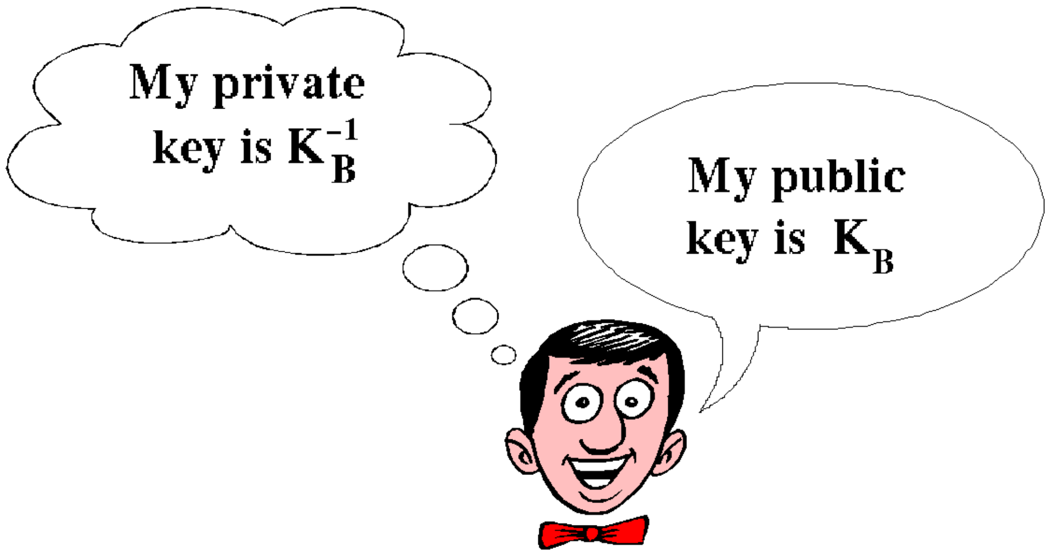
- Consider an encryption scheme $\{E_e : e \in \mathcal{K}\}$ and $\{D_d : d \in \mathcal{K}\}$. The scheme is symmetric-key if for each associated pair (e, d) it is computationally "easy" to determine d knowing only e and to determine e from d. In practice $e = d$.
- Other terms: single-key, one-key, private-key, and conventional encryption.
- A block cipher is an encryption scheme that breaks up the plaintext message into strings (blocks) of a fixed length t and encrypts one block at a time.
- A stream cipher is one where the block-length is 1.

Background: one-way functions

- A function $f : X \rightarrow Y$ is a one-way function, if f is "easy" to compute for all $x \in X$, but f^{-1} is "hard" to compute.
- **Example:** Problem of modular cube roots.
  - Select primes $p = 48611$ and $q = 53993$.
  - Let $n = pq = 2624653723$ and $X = \{1, 2, \ldots, n-1\}$.
  - Define $f : X \rightarrow N$ by $f(x) = x^3$ mod n.
  - Example: $f(2489991) = 1981394214$. Computing f is easy.
  - Inverting f is hard: find x which is cubed and yields remainder!
- A trapdoor one-way function is a one-way function $f : X \rightarrow Y$ where, given extra information (the trapdoor information) it is feasible to find, for $y \in Im(f)$, an $x \in X$ where $f(x) = y$.
- **Example:** Computing modular cube roots (above) is easy when p and q are known (basic number theory).

Public-key cryptography

- Public key cryptography is based on two keys: e and d.
  - Schema designed so that given a pair (E_e, D_d), knowing E_e it is infeasible, given $c \in \mathcal{C}$ to find an $m \in \mathcal{M}$ where $E_e(m) = c$. This implies it is infeasible to determine d from e.
  - E_e constitutes a trap-door one-way function with trapdoor d.
- Public key as e can be public information:



Definitions

- 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**.

Messages

- Message constructors are:

  Names: A, B or *Alice*, *Bob*, ...

  Keys: K and **inverse keys** K^{-1} (for signing, not decryption)
    Encryption: $\{M\}_K$. Encryption with A's public key: $\{M\}_{K_A}$.
    Signing: $\{M\}_{K^{-1}}$. Signing with A's private key: $\{M\}_{K_A^{-1}}$.
    Symmetric keys: $\{M\}_{K_{AB}}$.

  Nonces: NA, $N1$, ... fresh data items used for challenge/response.
    N.B.: sometimes subscripts are used, e.g. N_A, but it does not mean that principals can find out that N_A is related to (e.g. was generated by) A.

  Timestamps: T. Denote time, e.g. used for key expiration.

  Message concatenation: $\{M_1, M_2\}$.

- Example: $\{A, T_1, K_{AB}\}_{K_B}$.

Communication

- Fundamental event is communication.

$$A \rightarrow B : \{A, T_1, K_{AB}\}_{K_B}$$

- A and B are roles.

  A role is a procedure specified for each party in a protocol and that can be instantiated by any principal playing in the role.

  - A, B, ... are protocol variables corresponding to roles.
  - Their values (e.g. $Alice$, Bob, a, b,...) are principals.

- Communication is asynchronous (depending on semantic model).
- Sender/receiver names "$A \rightarrow B$" are not part of the message.
- Protocol specifies actions of principals.
  Equivalently: it defines a set of event sequences (traces/states).

Road map

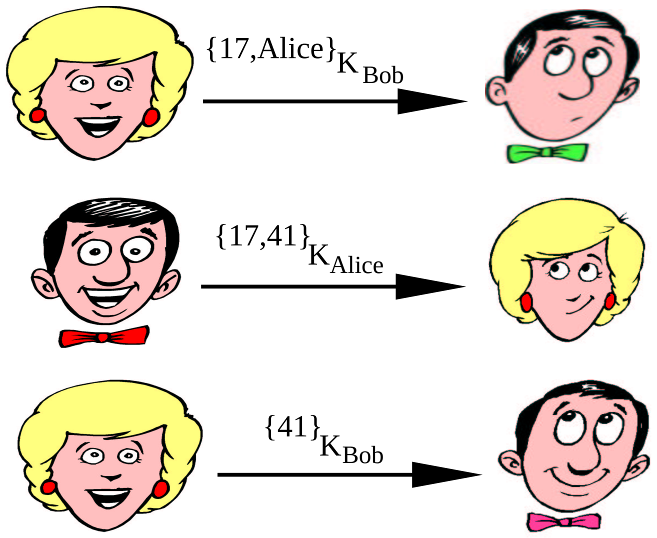
- Introduction to security protocols.
  - ▶ Motivation.
  - ▶ A brief introduction to cryptography.
  - ▶ Basic notions.
  - ☞ **An example: Needham-Schroeder Public Key Protocol.**
  - ▶ Protocol attacks.
- Formal methods for security protocol analysis.

An authentication protocol

The Needham-Schroeder Public Key protocol (NSPK):

$$\begin{array}{lll} 1. & A \rightarrow B: & \{NA, A\}_{K_B} \\ 2. & B \rightarrow A: & \{NA, NB\}_{K_A} \\ 3. & A \rightarrow B: & \{NB\}_{K_B} \end{array}$$

Here is an instance (a protocol run):



How the protocol is executed

$$\begin{array}{llll} 1. & A \rightarrow B: & \{NA, A\}_{K_B} \\ 2. & B \rightarrow A: & \{NA, NB\}_{K_A} \\ 3. & A \rightarrow B: & \{NB\}_{K_B} \end{array}$$

Each principal executes a "protocol automaton", e.g. Alice in role A.

State s_1:

- Generate nonce $NAlice$, pair it with name, and encrypt with K_{Bob}.
- Send $\{NAlice, Alice\}_{K_{Bob}}$ to Bob.
- Goto state s_2.

State s_2:

- Receive message C and decrypt it: $M = D_{K^{-1}_{Alice}}(C)$.
- If M is not of the form $\{NAlice, X\}$ for some nonce X, then goto reject state else goto state s_3.

State s_3: ...

State reject: terminate with failure.

N.B.: principals can be engaged in multiple runs (= multiple automata).

Assumptions and goals

Assumptions (for principals): Implicit (or explicit) prerequisites.

- Principals know their private keys and public keys of others.
- Principals can generate nonces.

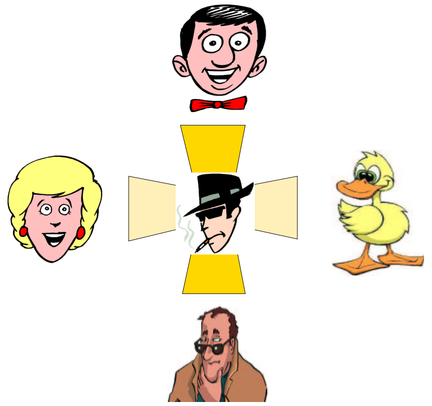
Goals: What the protocol should achieve. E.g.

- Authenticate messages, binding them to their originator.
- Ensure timeliness of messages (recent, fresh, ...).
- Guarantee secrecy of certain items (e.g. generated keys).

Theses:

- A protocol without clear goals (and assumptions) is useless.
- A protocol without a proof of correctness is probably wrong.

Assumptions: intruder



How do we model the intruder (attacker, adversary, spy, penetrator, saboteur, enemy, ... Mallory, Malice, Charlie,...)? Possibilities:

- He knows the protocol but cannot break cryptography. (Standard: perfect cryptography.)
- He is passive but overhears all communications.
- He is active and can intercept and generate messages.

  "Transfer \$20 to Bob $\leadsto$ "Transfer \$10,000 to Charlie"
- He might even be one of the principals running the protocol!

A friend's just an enemy in disguise. You can't trust nobody.

(Charles Dickens, *Oliver Twist*)

Standard intruder model

- The intruder is active. Namely:
  - He can intercept and read all messages.
  - He can decompose messages into their parts.
    But cryptography is secure: decryption requires inverse keys.
  - He can build new messages with the different constructors.
  - He can send messages at any time.
- Sometimes called the Dolev-Yao intruder model.
- Strongest possible assumptions about the intruder

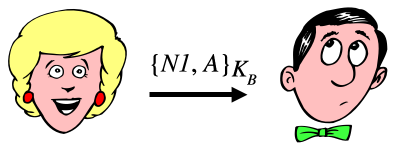
$$\Longrightarrow$$

correct protocols function in the largest range of environments.

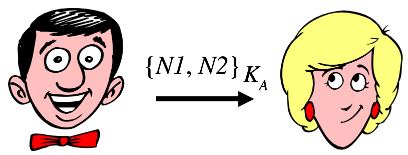
Problem with protocols

$$\begin{array}{lll} 1. & A \rightarrow B : & \{NA, A\}_{K_B} \\ 2. & B \rightarrow A : & \{NA, NB\}_{K_A} \\ 3. & A \rightarrow B : & \{NB\}_{K_B} \end{array}$$

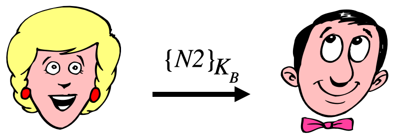
- Goal: mutual (entity) authentication.
- Principals can be involved in multiple runs.
  Goal should hold in all interleaved protocol runs.
- Correctness argument (informal):



"This is Alice and I have chosen a nonce $N1$."



"Here is your nonce $N1$. Since I could read it,
I must be Bob. I also have a challenge $N2$ for you."

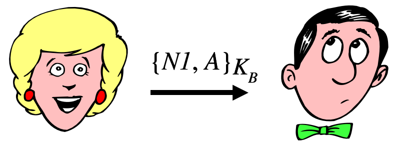


"You sent me $N2$. Since only Alice can read this
and I sent it back, you must be Alice."

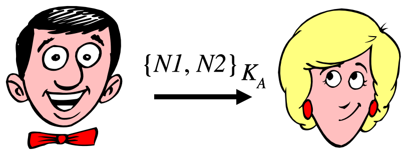
Problem with protocols

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

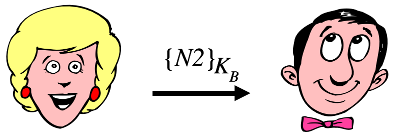
- Goal: mutual (entity) authentication.
- Principals can be involved in multiple runs.
  Goal should hold in all interleaved protocol runs.
- Correctness argument (informal):



"This is Alice and I have chosen a nonce $N1$."



"Here is your nonce $N1$. Since I could read it, I must be Bob. I also have a challenge $N2$ for you."



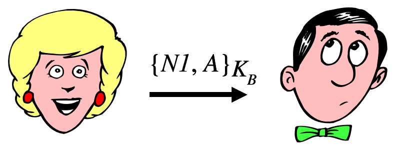
"You sent me $N2$. Since only Alice can read this and I sent it back, you must be Alice."

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

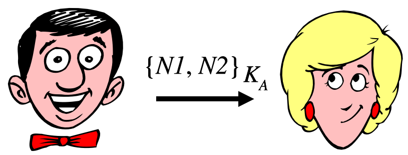
Problem with protocols

$$\begin{array}{lll} 1. & A \rightarrow B: & \{NA, A\}_{K_B} \\ 2. & B \rightarrow A: & \{NA, NB\}_{K_A} \\ 3. & A \rightarrow B: & \{NB\}_{K_B} \end{array}$$

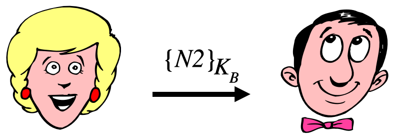
- Goal: mutual (entity) authentication.
- Principals can be involved in multiple runs.
  Goal should hold in all interleaved protocol runs.
- Correctness argument (informal):



"This is Alice and I have chosen a nonce $N1$."



"Here is your nonce $N1$. Since I could read it,
I must be Bob. I also have a challenge $N2$ for you."



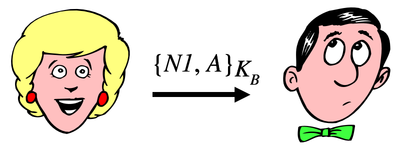
"You sent me $N2$. Since only Alice can read this
and I sent it back, you must be Alice."

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

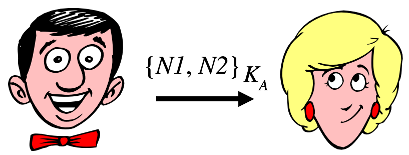
Problem with protocols

$$\begin{array}{lll} 1. & A \rightarrow B: & \{NA, A\}_{K_B} \\ 2. & B \rightarrow A: & \{NA, NB\}_{K_A} \\ 3. & A \rightarrow B: & \{NB\}_{K_B} \end{array}$$

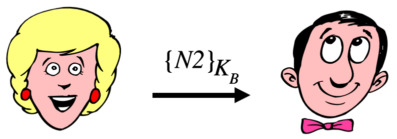
- Goal: mutual (entity) authentication.
- Principals can be involved in multiple runs.
  Goal should hold in all interleaved protocol runs.
- Correctness argument (informal):



"This is Alice and I have chosen a nonce $N1$."



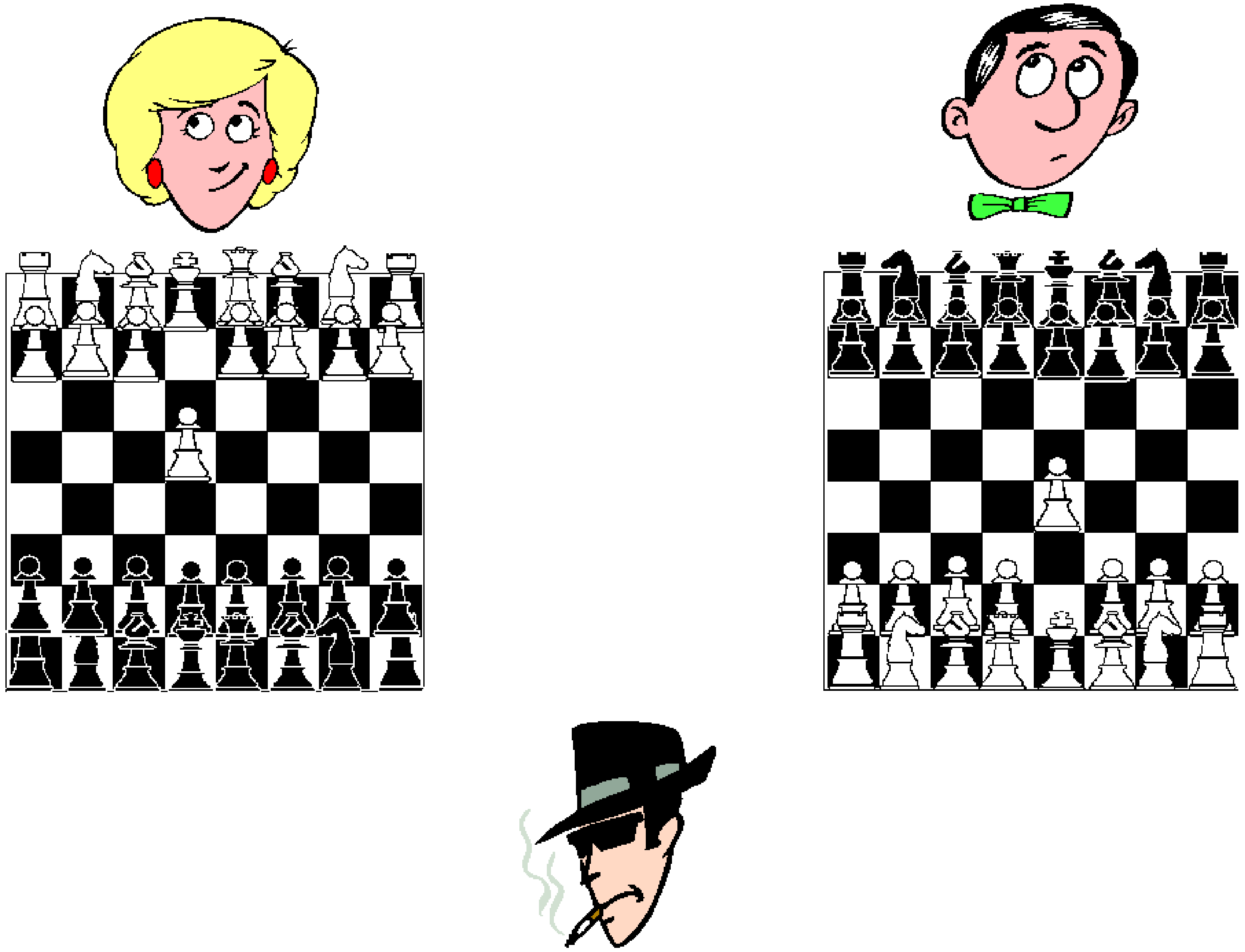
"Here is your nonce $N1$. Since I could read it,
I must be Bob. I also have a challenge $N2$ for you."



"You sent me $N2$. Since only Alice can read this
and I sent it back, you must be Alice."

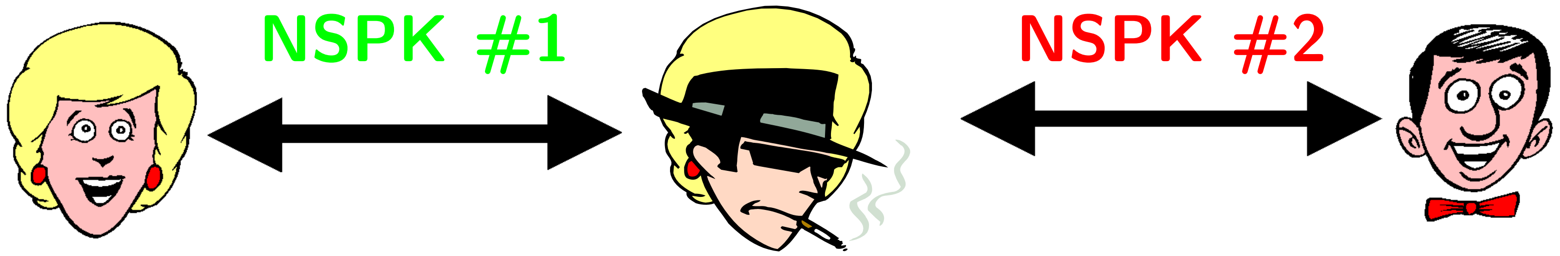
NSPK proposed in 1970s and used for decades, until...
Protocols are typically small and convincing... and wrong!

How to at least tie against a grandmaster in chess



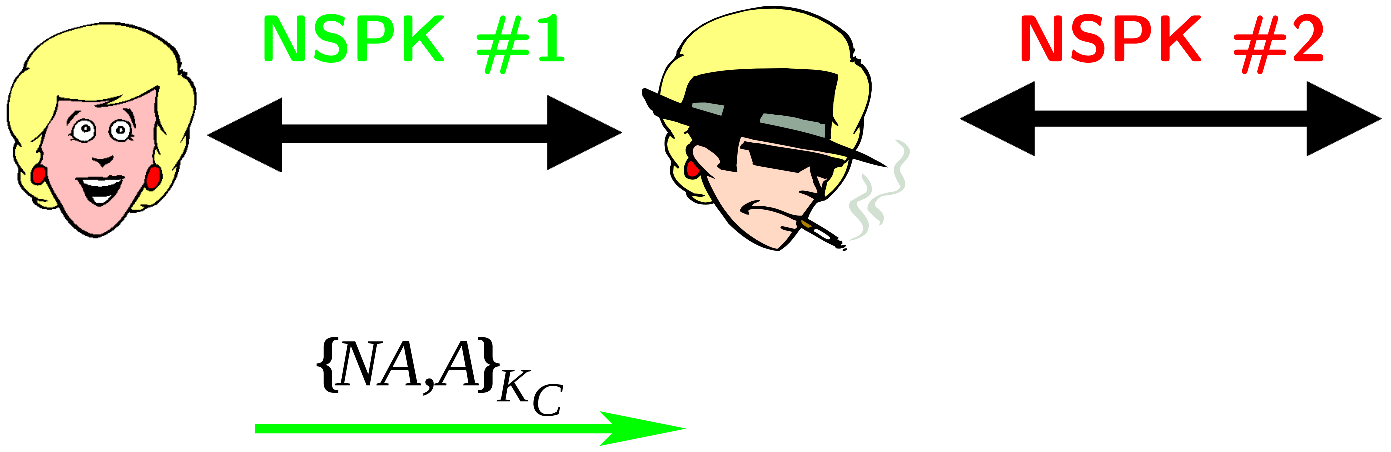
Man-in-the-middle attack

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



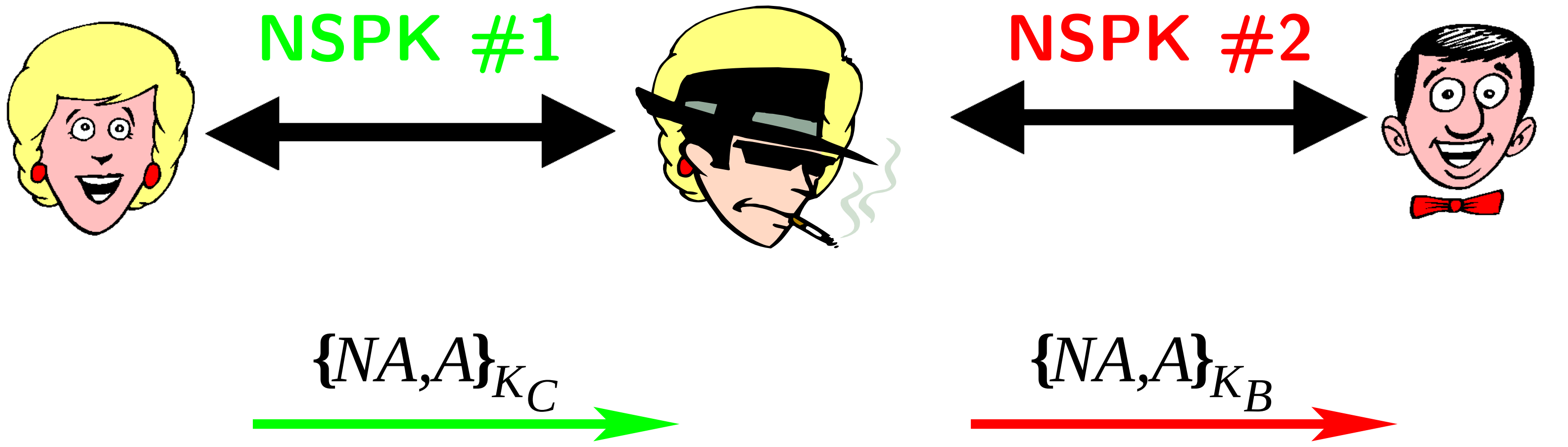
Man-in-the-middle attack

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



Man-in-the-middle attack

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

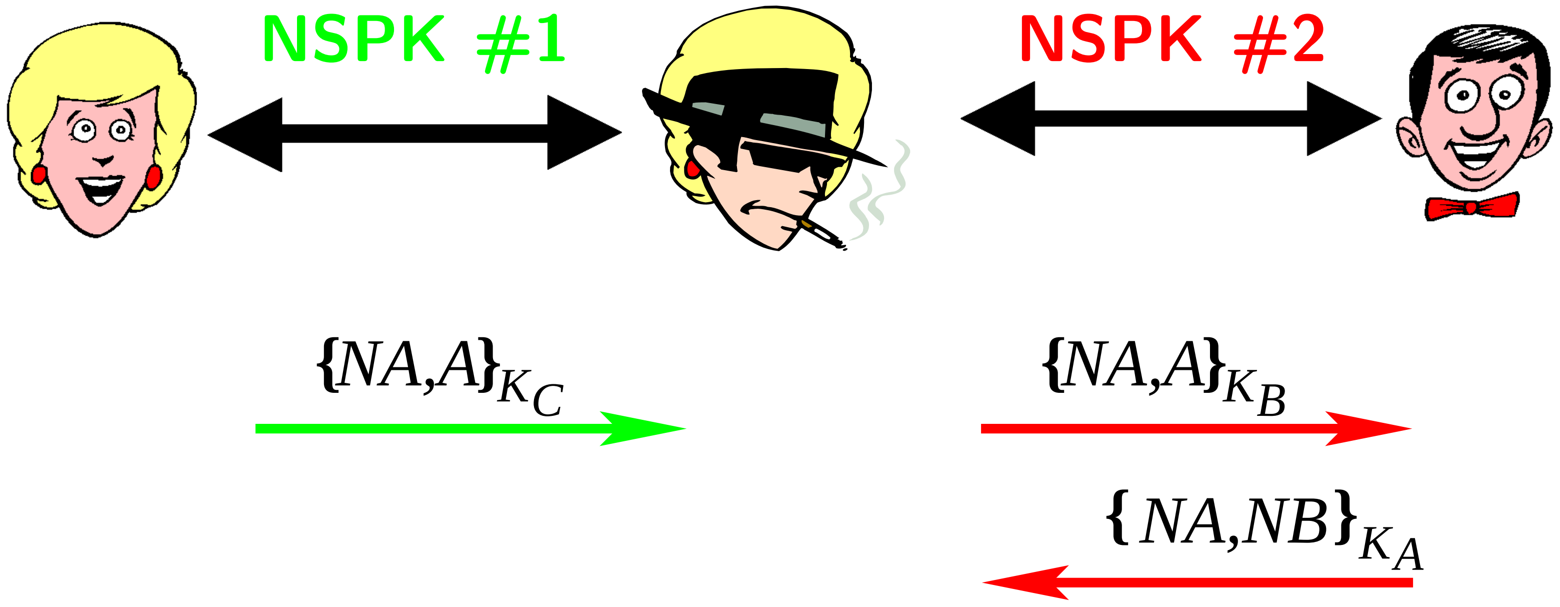


Man-in-the-middle attack

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

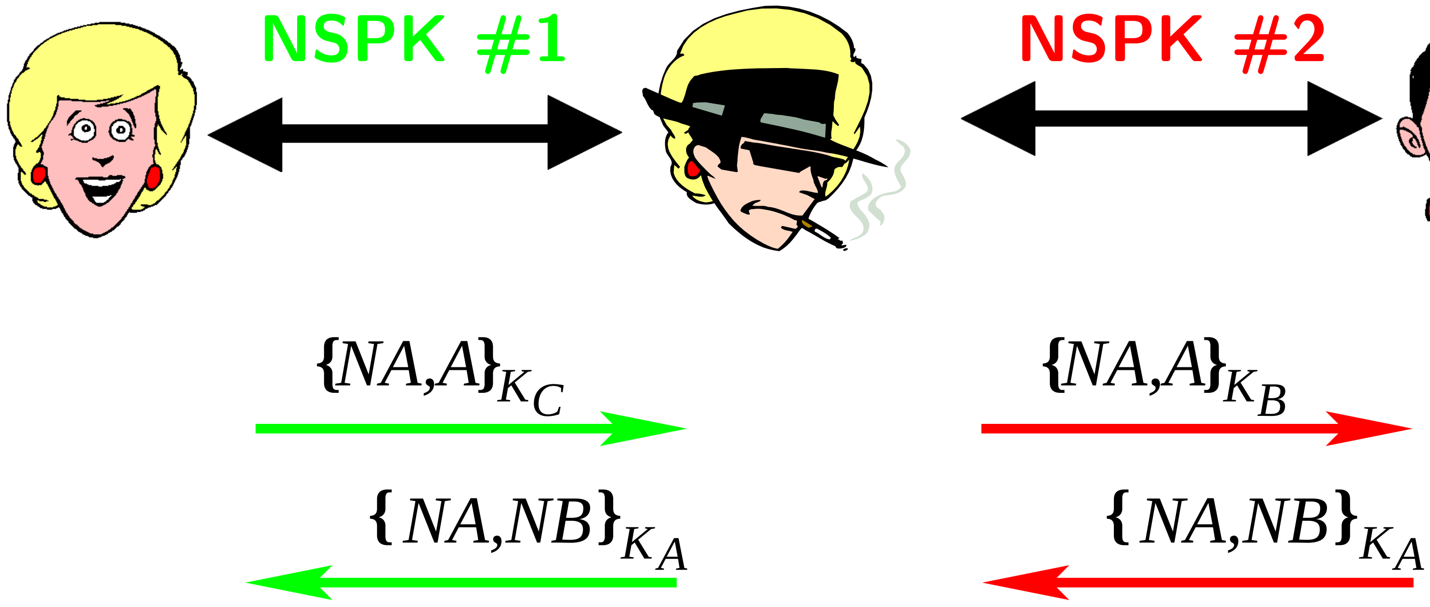
$B \rightarrow A : \{NA, NB\}_{K_A}$

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



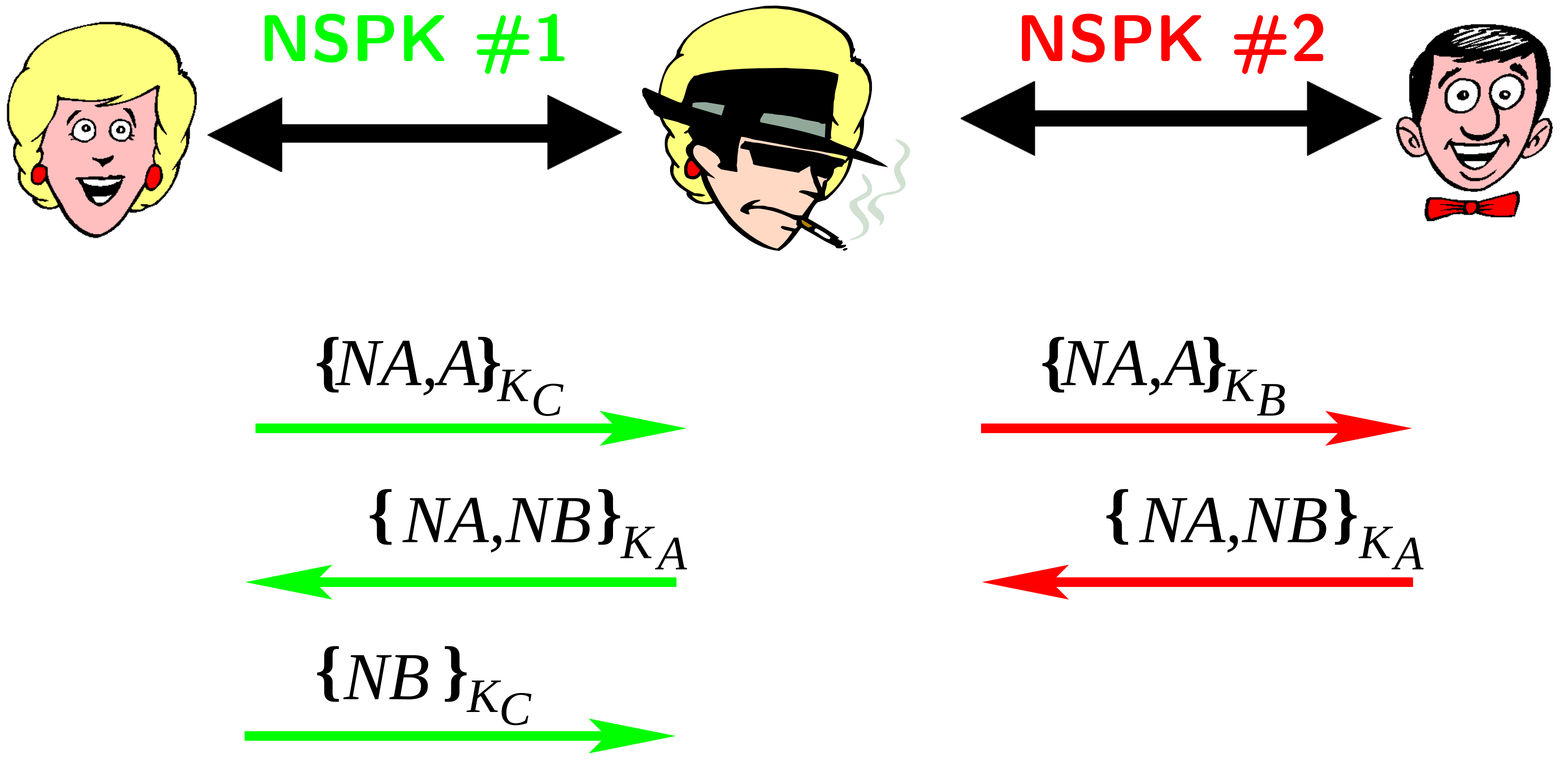
Man-in-the-middle attack

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



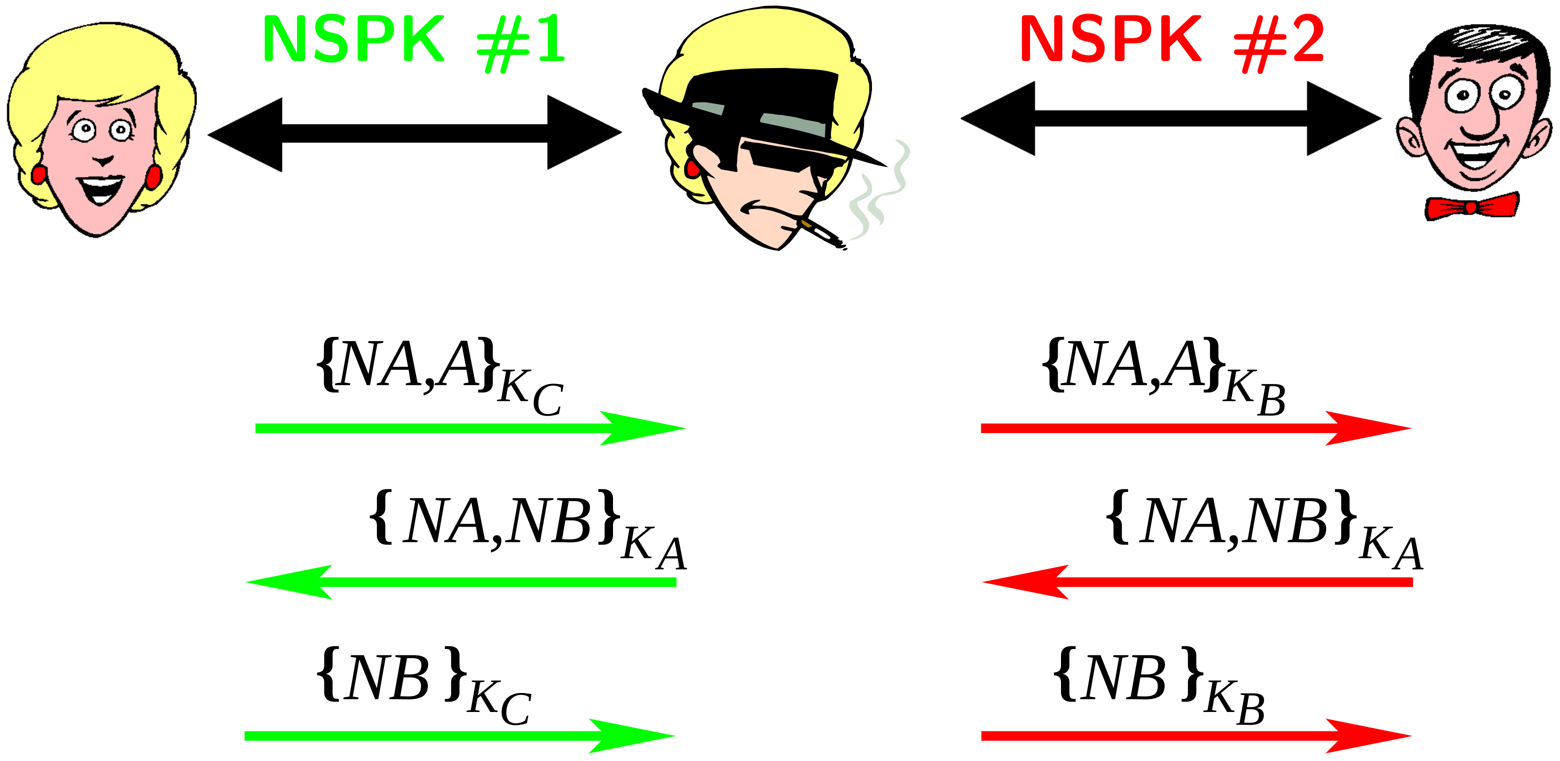
Man-in-the-middle attack

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



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

What went wrong?

- Problem in step 2.

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

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

- Is the improved version now correct?

  Against this and other kinds of attacks?

Road map

- Introduction to security protocols.
  - ▶ Motivation.
  - ▶ A brief introduction to cryptography.
  - ▶ Basic notions.
  - ▶ An example: Needham-Schroeder Public Key Protocol.
  - ☞ **Protocol attacks.**
- Formal methods for security protocol analysis.

Examples of kinds of attack

- Man-in-the-middle (or parallel sessions) attack: pass messages through to another session $A \leftrightarrow \mathsf{i} \leftrightarrow B$.
- Replay (or freshness) attack: record and later re-introduce a message or part.
- Masquerading attack: pretend to be another principal, e.g.
  - ▶ i forges source address (e.g. present in network protocols), or
  - ▶ i convinces other principals that A's public key is K_{i}.
- Reflection attack: send transmitted information back to originator.
- Oracle attack: take advantage of normal protocol responses as encryption and decryption "services".
- Type flaw (confusion) attack: substitute a different type of message field (e.g. a key vs. a name).

Type flaw attacks

- A message consists of a sequence of submessages.

  Examples: a principal's name, a nonce, a key, ...

- Messages sent as bit strings. No type information.

$$1011\ 0110\ 0010\ 1110\ \ 0011\ 0111\ 1010\ 0000$$

- Type flaw is when $A \rightarrow B :\ M$ and B accepts M as valid but parses it differently. I.e., B interprets the bits differently than A.

- Let's consider some examples.

The Otway-Rees protocol

Server-based protocol providing authenticated key distribution (with key authentication and key freshness) but without entity authentication or key confirmation.

$$\begin{array}{lll}
\mathrm{M1.} & A \rightarrow B: & I, A, B, \{NA, I, A, B\}_{K_{AS}} \\
\mathrm{M2.} & B \rightarrow S: & I, A, B, \{NA, I, A, B\}_{K_{AS}}, \{NB, I, A, B\}_{K_{BS}} \\
\mathrm{M3.} & S \rightarrow B: & I, \{NA, K\}_{K_{AS}}, \{NB, K\}_{K_{BS}} \\
\mathrm{M4.} & B \rightarrow A: & I, \{NA, K\}_{K_{AS}}
\end{array}$$

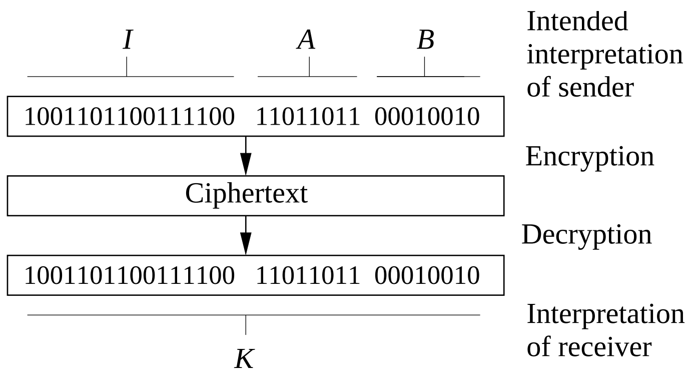
where server keys already known and I is protocol run identifier (e.g. an integer).

Why should(n't) it have the above properties?

Type flaw attack on the Otway-Rees protocol

$$\begin{array}{lll}
\text{M1.} & A \rightarrow B: & I, A, B, \{NA, I, A, B\}_{K_{AS}} \\
\text{M2.} & B \rightarrow S: & I, A, B, \{NA, I, A, B\}_{K_{AS}}, \{NB, I, A, B\}_{K_{BS}} \\
\text{M3.} & S \rightarrow B: & I, \{NA, K\}_{K_{AS}}, \{NB, K\}_{K_{BS}} \\
\text{M4.} & B \rightarrow A: & I, \{NA, K\}_{K_{AS}}
\end{array}$$

- Suppose $|\{I, A, B\}| = |\{K\}|$, e.g. I is 32 bits, A and B are 16 bits, and K is 64 bits.



- Attack 1 (Reflection/type-flaw): The intruder i replays parts of message 1 as message 4 (omitting steps 2 and 3).

$$\begin{array}{lll}
\text{M1.} & A \rightarrow \mathsf{i}(B): & I, A, B, \{NA, I, A, B\}_{K_{AS}} \\
\text{M4.} & \mathsf{i}(B) \rightarrow A: & I, \{NA, I, A, B\}_{K_{AS}}
\end{array}$$

- A sees NA and wrongly accepts $\{I, A, B\}$ as the session key.

Type flaw attack on the Otway-Rees protocol (cont.)

$$\begin{array}{lll}
\text{M1.} & A \rightarrow B: & I, A, B, \{NA, I, A, B\}_{K_{AS}} \\
\text{M2.} & B \rightarrow S: & I, A, B, \{NA, I, A, B\}_{K_{AS}}, \{NB, I, A, B\}_{K_{BS}} \\
\text{M3.} & S \rightarrow B: & I, \{NA, K\}_{K_{AS}}, \{NB, K\}_{K_{BS}} \\
\text{M4.} & B \rightarrow A: & I, \{NA, K\}_{K_{AS}}
\end{array}$$

Attack 2: intruder i can play the role of S in M2 and M3 by reflecting the encrypted components of M2 back to B. Namely:

$$\begin{array}{lll}
\text{M1.} & A \rightarrow B: & I, A, B, \{NA, I, A, B\}_{K_{AS}} \\
\text{M2.} & B \rightarrow \mathsf{i}(S): & I, A, B, \{NA, I, A, B\}_{K_{AS}}, \{NB, I, A, B\}_{K_{BS}} \\
\text{M3.} & \mathsf{i}(S) \rightarrow B: & I, \{NA, I, A, B\}_{K_{AS}}, \{NB, I, A, B\}_{K_{BS}} \\
\text{M4.} & B \rightarrow A: & I, \{NA, I, A, B\}_{K_{AS}}
\end{array}$$

$\Rightarrow$ A and B accept wrong key and i can decrypt their subsequent communication! So key authentication (and secrecy) fails!

Example of parallel sessions attack

The one-way authentication protocol

$$\begin{array}{lll} \text{M1.} & A \to B: & \{NA\}_{K_{AB}} \\ \text{M2.} & B \to A: & \{NA+1\}_{K_{AB}} \end{array}$$

admits a parallel sessions attack (with 'oracle'):

$$\begin{array}{llll} \text{M1.1.} & A \to \mathsf{i}(B) : \{NA\}_{K_{AB}} & & \\ & & \text{M2.1.} & \mathsf{i}(B) \to A : \{NA\}_{K_{AB}} \\ & & \text{M2.2.} & A \to \mathsf{i}(B) : \{NA+1\}_{K_{AB}} \\ \text{M1.2.} & \mathsf{i}(B) \to A : \{NA+1\}_{K_{AB}} & & \end{array}$$

A is forced to do work on behalf of the intruder: A acts as an 'oracle' against herself, because she provides the correct answer to her own question.

At very least, A believes then that B is operational, while B may no longer exist.

Fix: add A's name to message M1.

$$\text{M1.} \quad A \to B : \{NA, A\}_{K_{AB}}$$

Prudent engineering of security protocols

- Principles proposed by Abadi and Needham (1994, 1995):
  - ▶ Every message should say what it means.
  - ▶ Specify clearly conditions for a message to be acted on.
  - ▶ Mention names explicitly if they are essential to the meaning.
  - ▶ Be clear as to why encryption is being done: confidentiality, message authentication, binding of messages, ...
    e.g. $\{X, Y\}_{K^{-1}}$ versus $\{X\}_{K^{-1}}, \{Y\}_{K^{-1}}$
  - ▶ Be clear on what properties you are assuming.
  - ▶ Beware of clock variations (for timestamps).
  - ▶ etc.

- Good advice, but
  - ▶ Is the protocol guaranteed to be secure then?
  - ▶ Is it optimal and/or minimal then?
  - ▶ Have you considered all types of attacks?
  - ▶ etc.

Summary

- Security protocols can achieve properties that cryptographic primitives alone cannot offer, e.g. authentication, secrecy, ...
- The examples shown are 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 to security protocols.

☞ **Formal methods for security protocol analysis.**

  ▶ Dolev-Yao intruder and ping-pong protocols.
  ▶ (Un-)decidability.
  ▶ Logics of belief.
  ▶ (Semi-)automated methods.
  ▶ Early and recent methods/tools.
  ▶ Computational models.
  ▶ Conclusions.

Formal Methods for Security Protocols

Protocols can be understood as mathematical objects.

Formal methods based on mathematics and logic should be used to model, analyze, and construct them.

Doing so can substantially improve protocol security.

What are formal models?

- 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.
- Examples: propositional logic, first-order logic.
- A model (or construction) is formal when it is specified with a formal language.

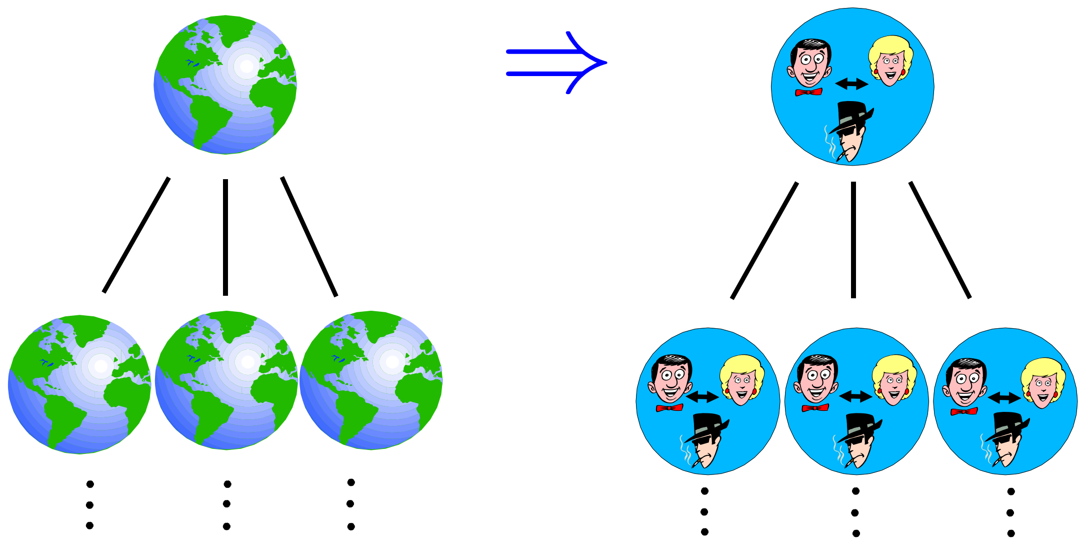
$$\forall x. bird(x) \rightarrow flies(x) \qquad bird(tweety)$$

- Standard protocol notation is not formal.

Formal modeling and analysis of protocols

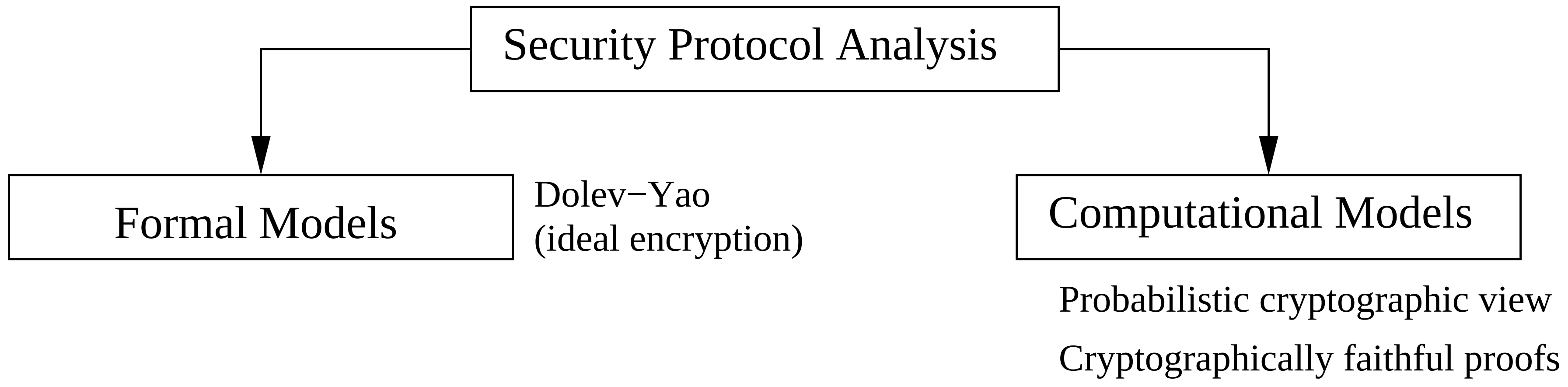
Goal: formally model protocols and their properties and provide a mathematically sound means for reasoning about these models.

Basis: suitable abstraction of protocols and information flow.

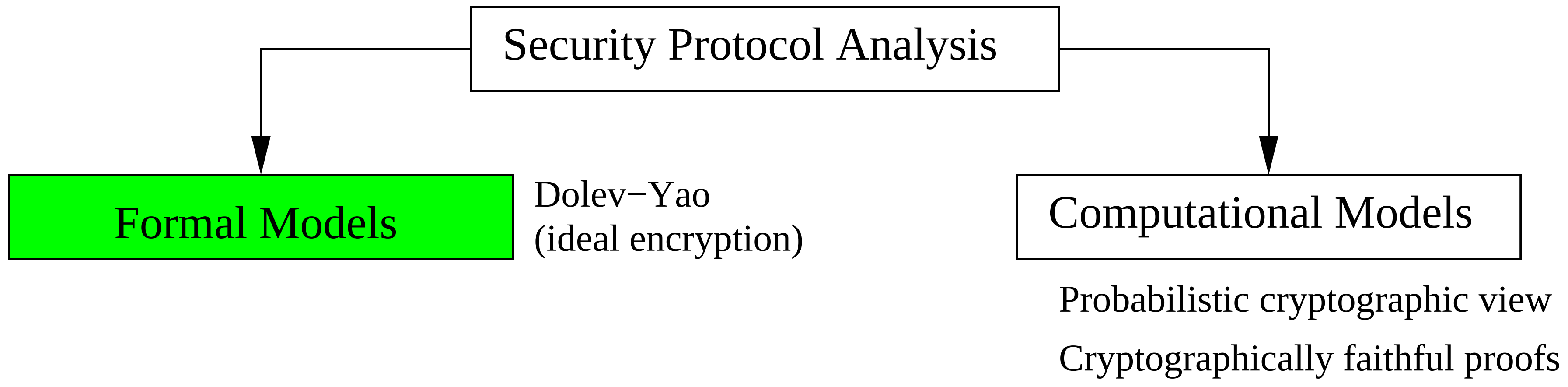


Analysis: with formal methods based on mathematics and logic.

Formal security protocol analysis


Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs

Formal methods for security protocol analysis


Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs

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!

Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- Consider a public key system in which
  - every user X has an encryption function E_X and a decryption function D_X such that $E_X(D_X(M)) = D_X(E_X(M)) = M$,
  - cryptography is perfect (all D_X private, decryption only with key, ...).
- Introduce the Dolev-Yao intruder:
  - He can obtain any message passing through the network.
  - He is a legitimate user of the network, and thus in particular can initiate a conversation with any other user.
  - He can be a receiver to any user A.

  - *He can decompose messages into their parts and can build new messages with the different constructors.*
  - *He can send messages at any time.*

With some *modifications*, this is the most commonly used intruder model today for formal protocol analysis.

Model by Dolev, Yao, Even and 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).
  - ▶ An intruder who can
    - ∗ read all traffic,
    - ∗ modify, delete and create traffic,
    - ∗ perform cryptographic operations available to legitimate users of the system,
    - ∗ is in league with a subset of "corrupt" principals.
  - ▶ An arbitrary number of principals.
  - ▶ Protocol executions may be interleaved (concurrent).

With some modifications, this is the most commonly used model today for formal methods analysis of security protocols.

Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- Give a formal model for the analysis of cascade protocols:
  - ▶ Messages built only by (concatenation and) the encryption-decryption operations (several layers possible), e.g.

$$\begin{aligned} A \rightarrow B: &\quad A, B, E_B(M) \\ B \rightarrow A: &\quad B, A, E_A(M) \end{aligned}$$

  - ▶ Secure if and only if (i.e. transmitted plaintext is secret iff)
    1. the messages transmitted between A and B always contain some layers of encryption functions E_A or E_B,
    2. in generating a reply message, each X never applies D_X without also applying E_X.

⇒ A simple characterization of security and an efficient algorithm for deciding whether a given cascade protocol is secure.

Cascade protocol example

- Cascade protocol secure (i.e. transmitted plaintext is secret) iff

  1. the messages transmitted between A and B always contain some layers of encryption functions E_A or E_B,
  2. in generating a reply message, each X never applies D_X without also applying E_X.

- An example:

$$\begin{array}{lll} A \rightarrow B: & A, B, E_B(M) & \\ B \rightarrow A: & B, A, E_A(M) & \text{generated as } m \rightarrow E_A(D_B(m)) \end{array}$$

Insecure because B applies D_B without E_B. Attack:

$$\begin{array}{lll} A \rightarrow \mathsf{i}(B): & A, B, E_B(M) & A \text{ creates secret } M \\ \mathsf{i} \rightarrow B: & \mathsf{i}, B, E_B(M) & \mathsf{i} \text{ intercepts and forwards to } B \\ B \rightarrow \mathsf{i}: & B, \mathsf{i}, E_{\mathsf{i}}(M) & \mathsf{i} \text{ can apply } D_{\mathsf{i}} \text{ to get } M \end{array}$$

Danny Dolev & Andrew C. Yao

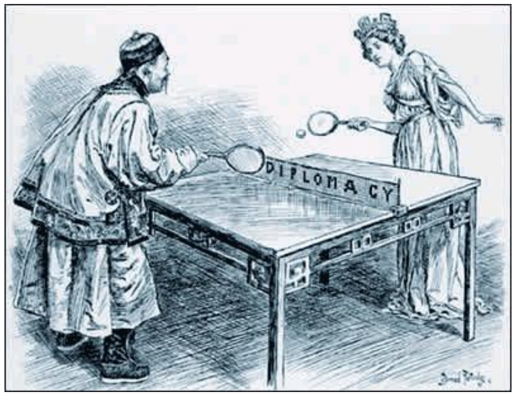
On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

- A formal model for the analysis of name-stamp protocols:
  - Users may append, delete, and check names encrypted together with the plaintext (can also contain layers of encryptions), e.g.

$$\begin{aligned} A \rightarrow B: &\quad A, B, E_B(M, A) \\ B \rightarrow A: &\quad B, A, E_A(M, B) \end{aligned}$$

- Ping-pong protocols: action of a principal on receiving a message is to apply some sequence of operations to it and send it back out.

  Message sent back and forth like a ping-pong ball (hence the name).



Danny Dolev & Andrew C. Yao

On the Security of Public Key Protocols (IEEE Trans. Inf. Th., 1983)

Danny Dolev, Shimon Even and Richard M. Karp

On the Security of Ping-Pong Protocols (Info and Control, 1982)

- Dolev-Yao and Dolev, Even and Karp (and later others) found PTIME algorithms for determining whether a ping-pong protocol would reveal secrets to an intruder.
  - Related to problem of finding intersection of 2 formal languages.
  - Freshness issues (e.g. replay) were not considered.
- Relaxing the restrictions on the protocols even slightly makes the security problem undecidable.

(Un-)Decidability

- General problem of protocol security is undecidable.
- Post Correspondence Problem:
  - ▶ Given an indexed finite set of pairs of words (strings) (U_i, V_i), is there a sequence of indices $i_1, \ldots, i_n$ so that $U_{i_1} \ldots U_{i_n} = V_{i_1} \ldots V_{i_n}$?
  - ▶ Example:

	U_i	V_i
1	a	ab
2	b	ca
3	ca	a
4	abc	c

Solution: 1, 2, 3, 1, 4 $\Rightarrow$ a b ca a abc $=$ ab ca a ab c

No solution: 1 $\Rightarrow$ a $\neq$ b

(Un-)Decidability: PCP as a protocol (construction)

Reduce word problem to protocol security (message secrecy).
(Even & Goldreich 1983; Heintze & Tygar, 1994)

- Protocol with $n+2$ parties.
  K is a shared secret key that the intruder does not know.
- Compromises secret M if solution exists.

  Initiator

  $send$ $\{$empty,empty$\}_K$

  Role for pair i (honest principal appends pair (U_i, V_I))

  $receive\ \{X, Y\}_K$

  $send\ \{XU_i, YV_i\}_K$

  Compromiser

  $receive\ \{X, X\}_K$ if $X \neq$ empty

  $send\ M$
- Requires unbounded message length.

(Un-)Decidability

Decidability of the protocol security problem can be shown for certain restrictions:

	Without nonces	With Nonces
No bounds	Undecidable	Undecidable
Infinite # sessions, bounded msgs	DEXPTIME-complete	Undecidable
Finite # sessions, unbounded msgs (also with choice points)	NP-complete	NP-complete

Various bounds by

- Rusinowitch & Turuani.
- Durgin, Lincoln, Mitchell, Scedrov.
- Millen & Shmatikov.
- ...

Restricted classes of protocols

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

- Bounded number of sessions.
  - 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.
- Bounded number of principals.

  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.
- Each role has a bounded number of steps.

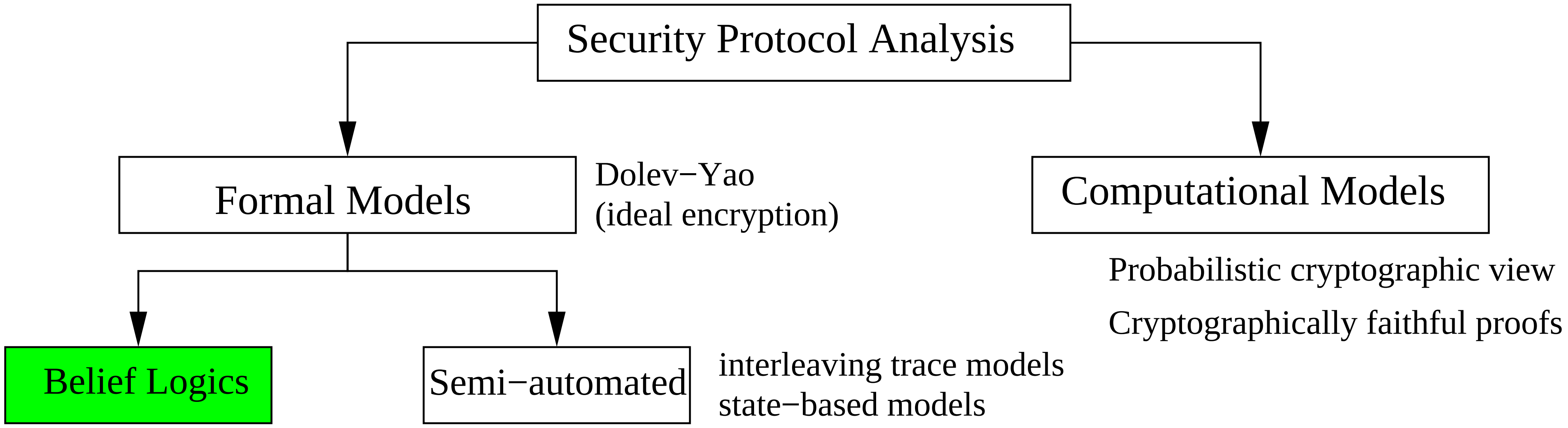
Restricted classes of protocols (cont.)

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

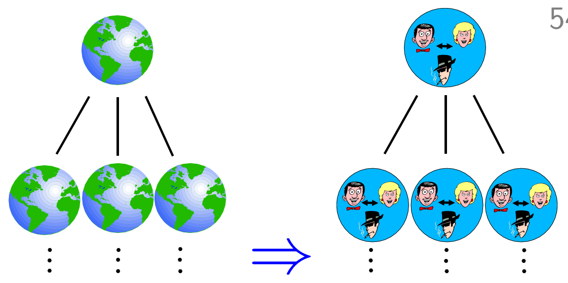
- Bounded message size.
  - Fixed number of fields in a message.
  - Fixed set of message constants.
  - Fixed depth restriction.
  - Allow nonces (but only “create new nonce” and $=?$).
- Everything constant, except for number of roles and number of new nonces.

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

Logics of belief


Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs
Belief Logics
Semi−automated
interleaving trace models
state−based models

BAN and other authentication logics



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.

Formal analysis: BAN (by Burrows, Abadi and Needham).

- A logic for analyzing the evolution of the beliefs of the agents as a consequence of communication in a particular protocol run.
- Simple and powerful but limited (and problematic).
- Extensions: partial solutions and other security properties (GNY, SVO, Simple logic, ...).
  - Logics of belief: view of user (to prove authentication).
  - Logics of knowledge: view of intruder (to prove secrecy).

BAN Predicates

P **bel** X ($P \mid\!\equiv X$)	P may act as though X is true.
P **sees** X ($P \triangleleft X$)	P has received (and can read and repeat, possibly after some decryption) a message containing X.
P **said** X ($P \mid\!\sim X$)	At some time, P sent a message including X. N.B.: it is not known when message was sent, but it is known that P believed X then.
P **controls** X ($P \mid\!\Rightarrow X$)	P has jurisdiction over X and should be trusted on this matter (e.g. server **controls** keys).
secret(X, P, Q) ($P \overset{X}{\rightleftharpoons} Q$)	X is secret known only to P and Q (& S). P and Q may use X to prove their identities to one another.

BAN Predicates (cont.)

$\mathbf{fresh}(X)$ $(\#(X))$	Formula X has not been sent in a message before the current run of the protocol.
$\mathbf{pubk}(K_P, P)$ $(\overset{K}{\longmapsto} P)$	P has public key K_P. Private key K_P^{-1} known only to P (and perhaps trusted server S).
$\mathbf{shk}(K, P, Q)$ $(P \overset{K}{\longleftrightarrow} Q)$	P and Q may use shared key K to communicate. K is a good key: known only to P and Q (and perhaps trusted server S).
$\{X\}_K$	X encrypted under key K. Abbreviates $\{X\}_K$ *from* P.
$\langle X\rangle_Y$	X combined (concatenated) with Y

BAN deduction system: Inference rules

Rules express how beliefs evolve as a result of communication.

- Message-meaning rules: interpretation of messages (how to derive beliefs about origin of message).

  Encrypted messages:

$$\frac{P \textbf{ bel shk}(K, Q, P) \quad P \textbf{ sees } \{X\}_K}{P \textbf{ bel } Q \textbf{ said } X} \text{sk (from } R \neq P)$$

$$\frac{P \textbf{ bel pubk}(K, Q) \quad P \textbf{ sees } \{X\}_{K^{-1}}}{P \textbf{ bel } Q \textbf{ said } X} \text{pk}$$

  Messages with secrets:

$$\frac{P \textbf{ bel secret}(Y, Q, P) \quad P \textbf{ sees } \langle X \rangle_Y}{P \textbf{ bel } Q \textbf{ said } X} \text{secret } (\langle X \rangle_Y \text{ not by } P)$$

BAN deduction system: Inference rules (cont.)

- Nonce-verification rule: check that message is recent ($\Rightarrow$ sender still believes in it).

$$\frac{P\ \mathbf{bel\ fresh}(X) \qquad P\ \mathbf{bel}\ (Q\ \mathbf{said}\ X)}{P\ \mathbf{bel}\ (Q\ \mathbf{bel}\ X)}\,\mathrm{nonce}\ (X\ \mathrm{cleartext}\ ,\ X \neq \{Y\}_K)$$

- Jurisdiction rule:

$$\frac{P\ \mathbf{bel}\ (Q\ \mathbf{controls}\ X) \quad P\ \mathbf{bel}\ (Q\ \mathbf{bel}\ X)}{P\ \mathbf{bel}\ X}\,\mathrm{jurisdiction}$$

N.B.: quantifiers are implicit, which may cause ambiguities, e.g.

$P\ \mathbf{bel}\ \forall K.\,(S\ \mathbf{controls}\ (Q\ \mathbf{controls\ shk}(K, P, Q)))$ vs.

$P\ \mathbf{bel}\ (S\ \mathbf{controls}\ \forall K.\,(Q\ \mathbf{controls\ shk}(K, P, Q)))$

- Freshness rule: if a part of a formula is fresh, then the whole formula is fresh.

$$\frac{P\ \mathbf{bel\ fresh}(X)}{P\ \mathbf{bel\ fresh}(X, Y)}\,\mathrm{fresh}$$

BAN deduction system: Inference rules (cont.)

- Formulas and their components:

$$\frac{P \textbf{ sees } (X,Y)}{P \textbf{ sees } X} \qquad \frac{P \textbf{ sees } \langle X\rangle_Y}{P \textbf{ sees } X} \qquad \frac{P \textbf{ bel shk}(K,Q,P) \quad P \textbf{ sees } \{X\}_K}{P \textbf{ sees } X}$$

$$\frac{P \textbf{ bel pubk}(K,P) \quad P \textbf{ sees } \{X\}_K}{P \textbf{ sees } X} \qquad \frac{P \textbf{ bel pubk}(K,Q) \quad P \textbf{ sees } \{X\}_{K^{-1}}}{P \textbf{ sees } X}$$

where $\{X\}_K \equiv \{X\}_K$ from R, and $R \neq P$.

- Other rules:

$$\frac{P \textbf{ bel } (X,Y)}{P \textbf{ bel } X} \qquad \frac{P \textbf{ bel } X \quad P \textbf{ bel } Y}{P \textbf{ bel } (X,Y)} \qquad \frac{P \textbf{ bel } Q \textbf{ bel } (X,Y)}{P \textbf{ bel } Q \textbf{ bel } X}$$

$$\frac{P \textbf{ bel } Q \textbf{ said } (X,Y)}{P \textbf{ bel } Q \textbf{ said } X}$$

- etc.

BAN: Analysis of protocols in 4 steps

1. Idealized protocol: message $\rightsquigarrow$ formula.

2. Assumptions about initial state state, e.g., which agents are trusted $(S\ \mathbf{controls}\ \mathbf{shk}(K, P, Q))$.

3. Assertions about the state of the system after each message:

$$\begin{array}{ll} \text{After message} & P \to Q : Y \\ \text{it holds} & Q\ \mathbf{sees}\ Y\,. \end{array}$$

4. Rules are applied to derive beliefs of agents, e.g.

$$\begin{array}{ll} & A\ \mathbf{bel}\ \mathbf{pubk}(K_B, B) \text{ and } B\ \mathbf{bel}\ \mathbf{pubk}(K_A, A) \\ \text{or also:} & A\ \mathbf{bel}\ (B\ \mathbf{bel}\ \mathbf{pubk}(K_A, A)) \\ \text{or weaker:} & A\ \mathbf{bel}\ (B\ \mathbf{bel}\ X) \text{ for some } X \quad \text{or...} \end{array}$$

BAN-analysis: Main ideas

- Main idea: see if the protocol goals can be derived using the inference rules from the formulas representing the initial assumptions and the protocol steps.

- Failure to reach the required goals can indicate the need
  - ▶ to change details of the protocol, or
  - ▶ for further assumptions.

- Alternatively: the analysis can identify places where
  - ▶ the assumptions are unnecessarily strong, or
  - ▶ the protocol is over-engineered (e.g. an unnecessary encryption of a term).

The full NSPK

$$\begin{array}{lll}
\text{M1.} & A \rightarrow S: & A, B \\
\text{M2.} & S \rightarrow A: & \{K_B, B\}_{K_S^{-1}} \\
\text{M3.} & A \rightarrow B: & \{NA, A\}_{K_B} \\
\text{M4.} & B \rightarrow S: & B, A \\
\text{M5.} & S \rightarrow B: & \{K_A, A\}_{K_S^{-1}} \\
\text{M6.} & B \rightarrow A: & \{NA, NB\}_{K_A} \\
\text{M7.} & A \rightarrow B: & \{NB\}_{K_B}
\end{array}$$

- Initially: A and B hold K_S.
- Aim 1: A and B obtain each other's public keys (M 1, 2, 4, 5).
- Aim 2: A and B use keys to communicate secret nonces NA and NB (M 3, 6, 7).
- 'Later': A and B use NA and NB to sign further messages.

Protocol is flawed: no guarantee that public keys K_A and K_B are fresh $\Rightarrow$ replay attack!

BAN-analysis of full NSPK: Idealization

M1.	$A \to S:$	A, B	$\rightsquigarrow$	omitted (no logical content)
M2.	$S \to A:$	$\{K_B, B\}_{K_S^{-1}}$	$\rightsquigarrow$	$S \to A:\ \{\mathbf{pubk}(K_B, B)\}_{K_S^{-1}}$
M3.	$A \to B:$	$\{NA, A\}_{K_B}$	$\rightsquigarrow$	$A \to B:\ \{NA\}_{K_B}$
M4.	$B \to S:$	B, A	$\rightsquigarrow$	omitted (no logical content)
M5.	$S \to B:$	$\{K_A, A\}_{K_S^{-1}}$	$\rightsquigarrow$	$S \to B:\ \{\mathbf{pubk}(K_A, A)\}_{K_S^{-1}}$
M6.	$B \to A:$	$\{NA, NB\}_{K_A}$	$\rightsquigarrow$	$B \to A:\ \{\langle \mathbf{secret}(NB, A, B)\rangle_{NA}\}_{K_A}$
M7.	$A \to B:$	$\{NB\}_{K_B}$	$\rightsquigarrow$	$A \to B:\ \{\langle \mathbf{secret}(NA, A, B),\ B\ \mathbf{bel\ secret}(NB, A, B)\rangle_{NB}\}_{K_B}$

Remarks:

M3.: used to give NA to B.
NA not known to B, and thus not used to prove identity of A.

M6. & M7.: NA and NB used as secrets.

Alternative to M7.: $A \to B:\ \{\langle \mathbf{secret}(NA, A, B)\rangle_{NB}\}_{K_B}$.

BAN-analysis of NSPK: Initial assumptions

- Public keys:

$$A \textbf{ bel pubk}(K_A, A) \quad A \textbf{ bel pubk}(K_S, S)$$
$$B \textbf{ bel pubk}(K_B, B) \quad B \textbf{ bel pubk}(K_S, S)$$
$$S \textbf{ bel pubk}(K_A, A) \quad S \textbf{ bel pubk}(K_B, B) \quad S \textbf{ bel pubk}(K_S, S)$$

- Trust in the server:

$$A \textbf{ bel } (S \textbf{ controls pubk}(K_B, B))$$
$$B \textbf{ bel } (S \textbf{ controls pubk}(K_A, A))$$

- Belief in own secret

$$A \textbf{ bel fresh}(NA) \quad A \textbf{ bel secret}(NA, A, B)$$
$$B \textbf{ bel fresh}(NB) \quad B \textbf{ bel secret}(NB, A, B)$$

BAN-analysis of NSPK: Derivations of beliefs

Logical analysis yields:

$$\begin{array}{ll} A \textbf{ bel pubk}(K_B, B) & A \textbf{ bel } (B \textbf{ bel secret}(NB, A, B)) \\ B \textbf{ bel pubk}(K_A, A) & B \textbf{ bel } (A \textbf{ bel secret}(NA, A, B)) \\ & B \textbf{ bel } (A \textbf{ bel } (B \textbf{ bel secret}(NB, A, B))) \end{array}$$

but only if we further assume

$$A \textbf{ bel fresh}(\textbf{pubk}(K_B, B)) \quad \text{and} \quad B \textbf{ bel fresh}(\textbf{pubk}(K_A, A))$$

i.e. A must assume that K_B is fresh, and B that K_A is fresh.

Else: replay attack.

$\Rightarrow$ modify the protocol by adding timestamps to $\mathrm{M2}$ and $\mathrm{M5}$:

$$\begin{array}{ll} \text{M2'.} & S \rightarrow A : \; \{T_S, \textbf{pubk}(K_B, B)\}_{K_S^{-1}} \\ \text{M5'.} & S \rightarrow B : \; \{T_S, \textbf{pubk}(K_A, A)\}_{K_S^{-1}} \end{array}$$

BAN-analysis of NSPK: Example derivation

$$\dfrac{\dfrac{}{A\ \textbf{bel}\ (S\ \textbf{controls}\ \textbf{pubk}(K_B,B))}\ \text{ass} \qquad \dfrac{\dfrac{}{A\ \textbf{bel}\ \textbf{fresh}(\textbf{pubk}(K_B,B))}\ \text{ASS} \qquad \dfrac{\dfrac{}{A\ \textbf{bel}\ \textbf{pubk}(K_S,S)}\ \text{ass} \qquad \dfrac{}{A\ \textbf{sees}\ \{\textbf{pubk}(K_B,B)\}_{K_S^{-1}}}\ \text{M2.}}{A\ \textbf{bel}\ (S\ \textbf{said}\ \textbf{pubk}(K_B,B))}\ \text{pk}}{A\ \textbf{bel}\ (S\ \textbf{bel}\ \textbf{pubk}(K_B,B))}\ \text{nonce}}{A\ \textbf{bel}\ \textbf{pubk}(K_B,B)}\ \text{jurisdiction}$$

Start from the goal and apply rules backwards to arrive at the initial assumptions (ass) or to identify further required assumptions (like ASS) in this case.

Limitations and extensions of BAN

BAN is simple and powerful, but is limited and misses many flaws.

- By design: inability to reason about certain events or variations.
  - ▶ properties other than authentication (secrecy, anonymity,...),
  - ▶ intruders,
  - ▶ temporal properties (no explicit time),
  - ▶ attacks resulting from multiple runs of a protocol, ...
- More dramatic limitation: BAN-analysis of the Nesset protocol

$$\begin{aligned} &\text{M1.} \quad A \to B: \ \{NA, K_{AB}\}_{K_A^{-1}} \\ &\text{M2.} \quad B \to A: \ \{NB\}_{K_{AB}} \end{aligned}$$

yields

$$B \ \mathbf{bel} \ \mathbf{shk}(K_{AB}, A, B)$$

but everyone can decipher M1 by using A's public key K_A!

Limitations and extensions of BAN (cont.)

- ► Burrows et al.'s answer: BAN focuses only on authentication of honest principals by design (and does not attempt to detect unauthorized release of secrets).
- ► Problem is not the logic, but idealization: informal $\rightsquigarrow$ formal.
  BAN paper: "*only knowledge of entire protocol can determine the essential logical content of each message*"
- ► This is dangerous and often wrong! Solution: extend BAN.
  - ∗ Robustness: each message can be considered separately.
  - ∗ Define idealization 'formally'.

- Other extensions:
  - ► Add notions of time (e.g. modal operator $\Diamond^-$ for past-time).
  - ► Allow for checking security properties other than authentication.
  - ► etc.

Limitations and extensions of BAN (cont.)

- Another problem: semantics.

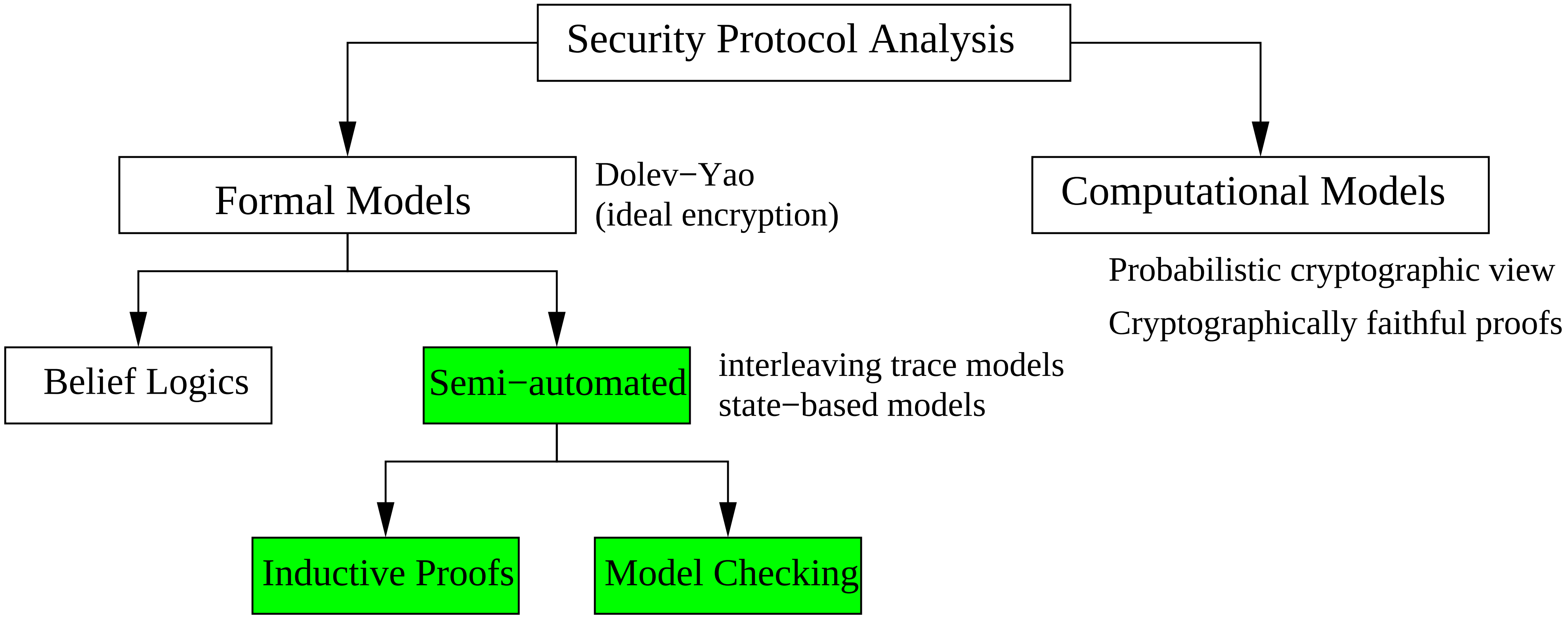
  Can a principal believe something that is false?

- Improved semantics (e.g. possible world semantics) have been provided subsequently, but it can still be quite difficult to interpret the results of a BAN-analysis.

- Summary:
  - ▶ BAN logic is undoubtedly seminal and has been highly influential in the field.

    "*It has plenty of scalps under its belt.*" (R. Needham)
  - ▶ Limitations can be tackled by restrictions/extensions...
  - ▶ ...or: just use BAN for the insights it can give.

Formal methods for security protocol analysis



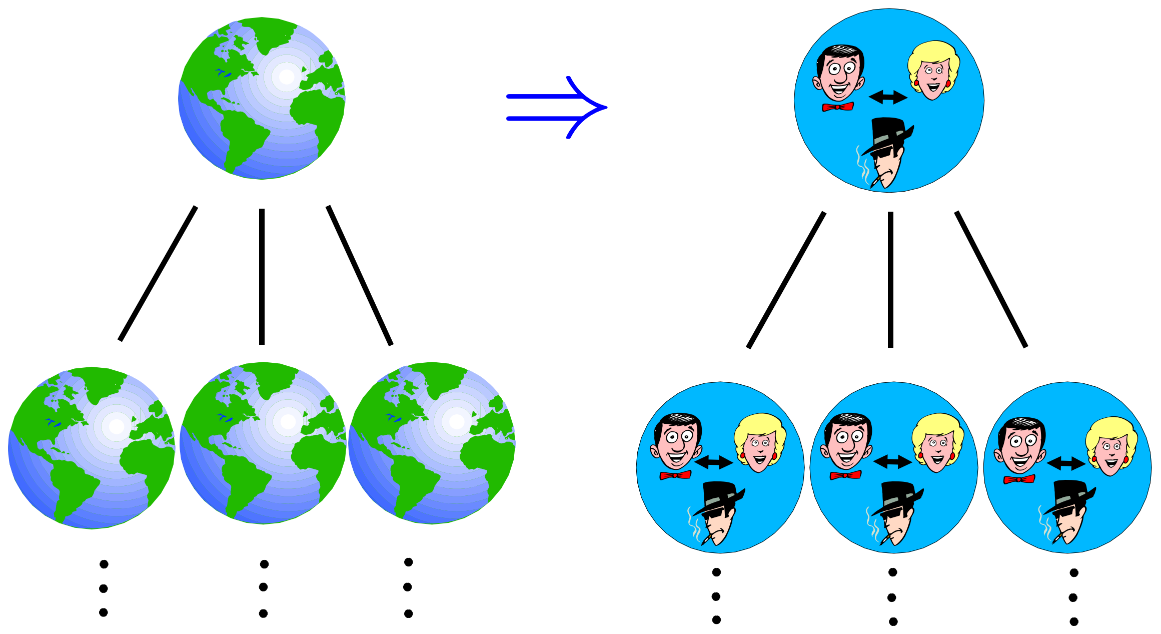
Road map

- Security protocols.
- Formal methods for security protocol analysis.
  - ▶ Dolev-Yao intruder and ping-pong protocols.
  - ▶ (Un-)decidability.
  - ▶ Logics of belief.
  - ☞ **(Semi-)automated methods.**
    - ∗ Interleaving trace models.
    - ∗ Falsification vs. verification of protocols.
  - ▶ Early and recent methods/tools.
  - ▶ Computational models.
  - ▶ Conclusions.

Modeling 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.



Interleaving trace models

- Modeling idea: model possible communication events.

$$A \longrightarrow B : M_1$$

$$B \longrightarrow A : M_2$$

$$\vdots$$

Interleaving trace models

- Modeling idea: model possible communication events.

$$\begin{aligned}A &\to B : M_1\\ C &\to D : P_1\\ B &\to A : M_2\\ C &\to D : P_2\\ &\vdots\end{aligned}$$

Interleaving trace models

- Modeling idea: model possible communication events.

$$\begin{array}{lclcll} A \rightarrow B : M_1 & & A \rightarrow C : P_1 & & A \rightarrow B : M_1 & \\ C \rightarrow D : P_1 & & A \rightarrow B : M_1 & & B \rightarrow A : M_2 & \\ \mathsf{i} \rightarrow A : M_2 & \text{or} & B \rightarrow A : M_2 & \text{or} & C \rightarrow D : P_1 & \ldots \\ C \rightarrow D : P_2 & & C \rightarrow A : P_2 & & \mathsf{i} \rightarrow C : P_1 & \\ \vdots & & \vdots & & \vdots & \end{array}$$

- 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.

Modeling: protocol as an inductively defined set

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

Set P formalizes protocol steps (example: NSPK).

0. $\langle\rangle \in P$

1. $t,\ A \rightarrow B: \{NA, A\}_{K_B} \in P$ if $t \in P$ and $fresh_t(NA)$

2. $t,\ B \rightarrow A: \{NA, NB\}_{K_A} \in P$ if $t \in P$, $fresh_t(NB)$, and $A' \rightarrow B: \{NA, A\}_{K_B} \in t$

3. $t,\ A \rightarrow B: \{NB\}_{K_B} \in P$ if $t \in P$, $A \rightarrow B: \{NA, A\}_{K_B} \in t$ and $B' \rightarrow A: \{NA, NB\}_{K_A} \in t$

4. t, i $\rightarrow B: X \in P$ if $t \in P$ and $X \in synthesize\ (analyze\ (sees\ t))$

Rules 0–3 formalize the protocol steps and rule 4 the intruder model (what messages the intruder can generate from the terms he can analyze from the messages he has seen in the trace (function $sees$).

Predicate $fresh_t$ checks whether a nonce has been used before in a trace t.

Modeling the intruder: Paulson's sets

Given a set M of messages, the set $\mathit{synthesize}(M)$ is the smallest extension of M including agent identifiers and closed under pairing and encryption:

$$\frac{m \in M}{m \in \mathit{synthesize}(M)} \qquad \frac{m_1 \in \mathit{synthesize}(M) \quad m_2 \in \mathit{synthesize}(M)}{m_1, m_2 \in \mathit{synthesize}(M)}$$

$$\frac{m \in \mathit{synthesize}(M) \quad k \in \mathit{synthesize}(M)}{\{m\}_k \in \mathit{synthesize}(M)}$$

The set $\mathit{analyze}(M)$ is the smallest extension of M closed under projection and decryption by keys in $\mathit{analyze}(M)$:

$$\frac{m \in M}{m \in \mathit{analyze}(M)} \qquad \frac{m_1, m_2 \in \mathit{analyze}(M)}{m_i \in \mathit{analyze}(M)} (i \in \{1,2\})$$

$$\frac{\{m\}_k \in \mathit{analyze}(M) \quad k^{-1} \in \mathit{analyze}(M)}{m \in \mathit{analyze}(M)}$$

The set $\mathit{parts}(M)$ is the smallest extension of M obtained by adding the components of compound messages and the bodies of encrypted messages:

$$\frac{m \in M}{m \in \mathit{parts}(M)} \qquad \frac{m_1, m_2 \in \mathit{parts}(M)}{m_i \in \mathit{parts}(M)} (i \in \{1,2\}) \qquad \frac{\{m\}_k \in \mathit{parts}(M)}{m \in \mathit{parts}(M)}$$

Modeling the intruder: Paulson's sets

Some remarks:

- The above rules 0–3 assume a synchronous model, where if A sends a message to B then B really receives it.
  Models where communication is asynchronous can also be considered (but require slightly different rules).
- The above *analyze* rules assume that $(k^{-1})^{-1} = k$. When that is not the case one needs two rules for decryption:

$$\frac{\{m\}_k \in analyze(M) \quad k^{-1} \in analyze(M)}{m \in analyze(M)} \qquad \frac{\{m\}_{k^{-1}} \in analyze(M) \quad k \in analyze(M)}{m \in analyze(M)}$$

  Similarly, one then needs also two rules for synthesizing encrypted messages. One could also have different decryption and encryption rules for asymmetric and symmetric keys.

- Note also that if one wants to consider, e.g., composed keys, then rule 4 must be changed to make sure that the intruder can first synthesize a composed key k (say $k = (k_1, k_2)$), then use k to decrypt some message $\{m\}_k$, and finally use m to compose some new message to be sent.

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*:

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

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

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

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

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

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.

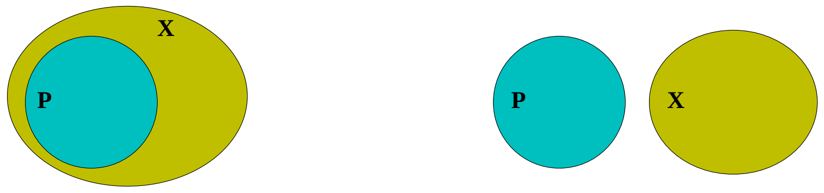
$$\begin{array}{lcll} AauthenticatesB(t) & \equiv & \textbf{if} & A \to B : \{NA, A\}_{K_B} \in t \ \ \textbf{and} \\ & & & B' \to A : \{NA, NB\}_{K_A} \in t \\ & & \textbf{then} & B \to A : \{NA, NB\}_{K_A} \in t \\ \mathrm{i}attacksA(t) & \equiv & \neg AauthenticatesB(t) & \end{array}$$

  Secrecy: Intruder cannot discover NB, e.g.

  For all $t \in P$, if $B \to A : \{NA, NB\}_{K_A} \in t$ 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.

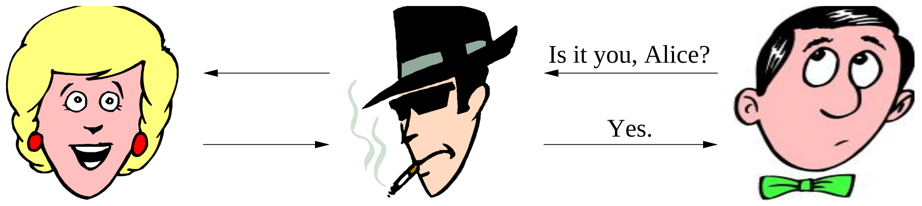


- Every proposition is either true or false.

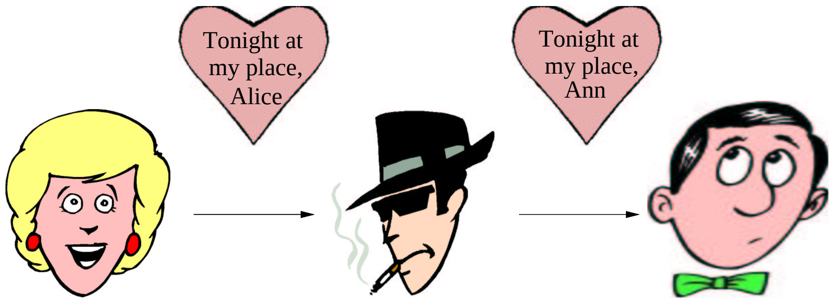
  How do we determine which holds?

About authentication

- Different forms of entity (or user) authentication.



- Contrast to message (or data-origin) authentication.



  - ▶ Observe that message authentication implies message integrity.
  - ▶ Message Authentication Codes (MACs) and digital signatures are two main techniques for establishing message authentication.

A hierarchy of authentication specifications (Lowe)

- Other goals... many can be re-phrased in terms of secrecy and authentication (as reachability problems).

- Safety temporal properties ranging from one-way to mutual, weak to strong, authentication, e.g. for a protocol initiator A:
  - ► Aliveness: a protocol guarantees to A aliveness of a another principal B if, whenever A completes a run of the protocol, apparently with responder B, then B has previously been running the protocol.
  - ► Weak agreement: a protocol guarantees to A weak agreement with another principal B if, whenever A completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A.

A hierarchy of authentication specifications (Lowe)

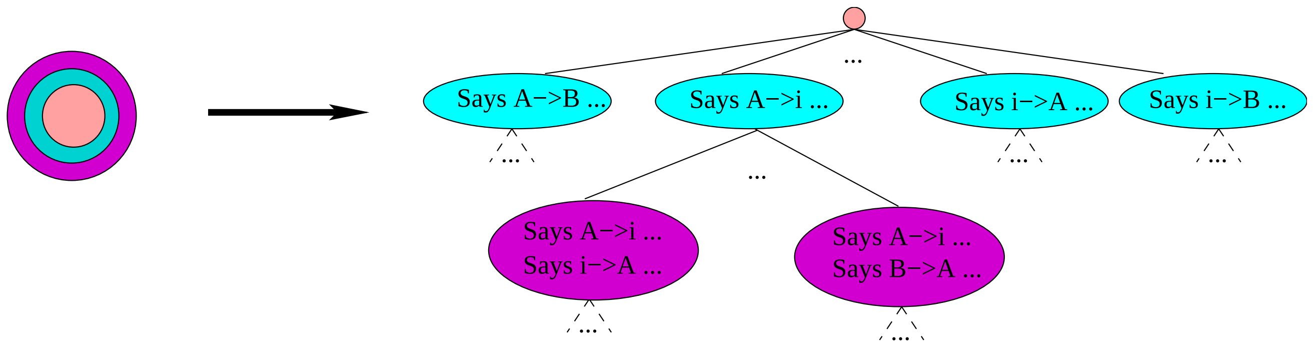
- Non-injective agreement: a protocol guarantees to A non-injective agreement with a responder B on a set of data ds if, whenever A completes a run of the protocol, apparently with responder B, then B has previously been running the protocol, apparently with A, and B was running the protocol apparently with A, and B was acting as a responder in his run, and the two principals agreed on the values corresponding to all the variables in ds.
- Agreement: non-injective agreement where each such run of A corresponds to a unique run of B.
- Also issues of recentness: insist that B's run was recent (within t time units).

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.

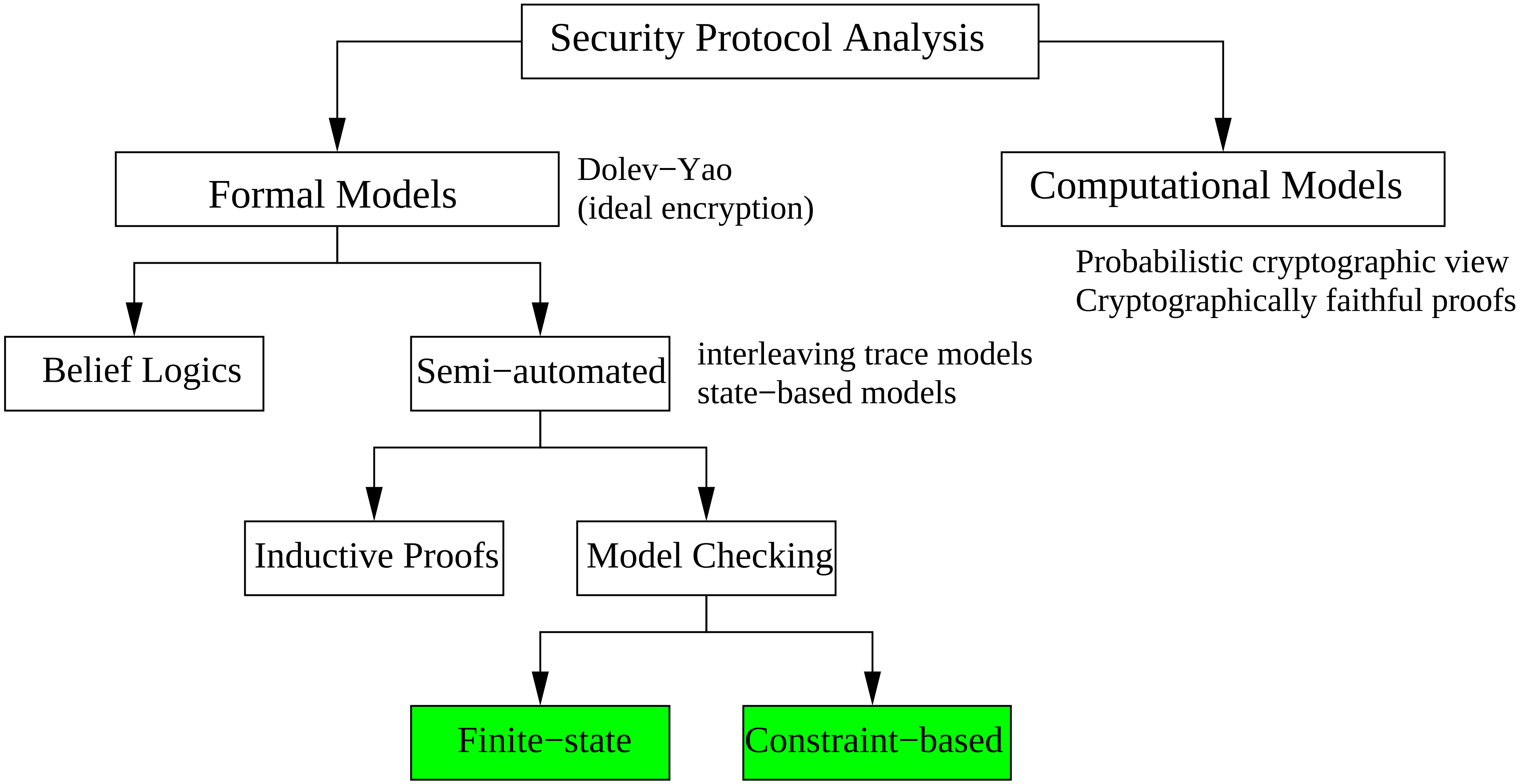


- Properties correspond to a subset of nodes, e.g., i$attacks A(t)$.
- State enumeration can be used to find attack in the infinite tree.
- But naive search is hopeless! Challenges:

  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 (cf. the restrictions we saw above).
  - ▶ Problem can be partially solved by infinite-state model checking (e.g. based on symbolic methods and abstractions).

Road map

- Security protocols.
- Formal methods for security protocol analysis.
  - ▶ Dolev-Yao intruder and ping-pong protocols.
  - ▶ (Un-)decidability.
  - ▶ Logics of belief.
  - ▶ (Semi-)automated methods.
    - ∗ Interleaving trace models.
    - ∗ Falsification vs. verification of protocols.
  - ☞ **Early and recent methods/tools.**
  - ▶ Computational models.
  - ▶ Conclusions.

Early methods/tools

- Interrogator (Millen, earliest version 1984).
  - ▶ Backward state search for attack from goal state pattern.
  - ▶ No guarantee of completeness or termination.
- NRL Protocol Analyzer (Meadows, earliest version 1989).
  - ▶ Automated Dolev-Yao intruder.
  - ▶ Used symbolic representation of states, and supported use of lemmas to reduce infinite state space to a finite one.
  - ▶ In earliest versions, lemmas proved by hand, later offered automated support (inductive proofs).
  - ▶ No guarantee of completeness or termination.
  - ▶ Can sometimes prove a protocol correct.

Also: some tool support for BAN and other authentication logics (decidable and completely automated, in some cases).

Recent methods/tools: inductive proofs

- Approach: like proofs of program correctness.
  - Induction to prove "loop invariant".
- General-purpose specification/verification system support:
  - Kemmerer, using Ina Jo and ITP (1987). [the first]
  - Paulson, using Isabelle (from mid 90s). [the new wave]
  - Dutertre and Schneider, using PVS.
  - Bolignano, using Coq.
  - Cohen, using TAPS.
- Also manually: CSP (Schneider), Strands (Thayer et al.).
- Facts that can be proved:
  - Secret keys are never lost.
  - Nonces uniquely identify their message of origin.
  - Nonces stay secret (under certain conditions).

Recent ...: model-checkers and symbolic methods

- Man-in-the-middle attack on NSPK: found by Lowe, using process calculus CSP (for specification) and model-checker FDR (for analysis).
  - Several model-checkers applied to protocols, both general purpose and special-purpose: CSP/FDR, Murphi, SMV, Maude, ...
  - Mostly finite-state, but also work on infinite-state checking.
- Athena
  - Special-purpose model-checker (Song, Berezin, Perrig, 2001–).
  - Uses strand-space model (free algebra, atomic keys).
  - Usually terminates, no fixed bound.
- Model-checkers and provers based on symbolic constraint solving (Millen & Shmatikov, Boreale & Buscemi, Corin & Etalle, CL-atse, OFMC,...).

Approaches, methods & tools I haven't told you about

- Modeling:
  - Specification languages (CAPSL/CIL, Casper, HLPSL/IF).
  - Typed vs. untyped models.
  - Compound vs. atomic keys.
  - Free algebra vs. algebraic properties of cryptographic operators.

$$\{M_1\}_{K_1} = \{M_2\}_{K_2} \;\Rightarrow\; M_1 = M_2 \text{ and } K_1 = K_2$$

vs.

$$(M_1 \oplus M_1) \oplus M_2 = M_2 \qquad\qquad (g^x)^y = (g^y)^x$$

  - Intruder models (e.g. guessing).
  - Open-ended (stream) protocols.
  - Protocol composition.

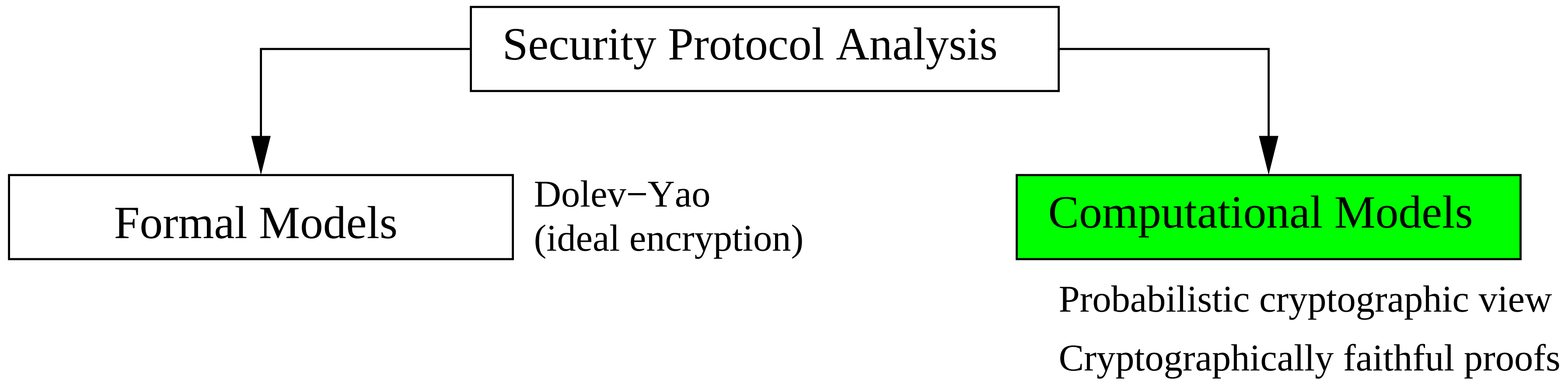
Approaches, methods & tools I haven't told you about

- Methods and tools:
  - Process-algebras (e.g. CSP, CCS, Spi-calculus).
  - Strand-spaces.
    - Graph-theoretic approach easy to visualize.
    - Can be used also as a basis for comparison between models.
  - Multi-set rewriting.
  - Type-checking.
  - Logic programming and planning (SATMC).
  - Abstractions for infinite-state model-checking/verification.
- Analysis of Internet Protocols $\Rightarrow$ AVISPA Tool.

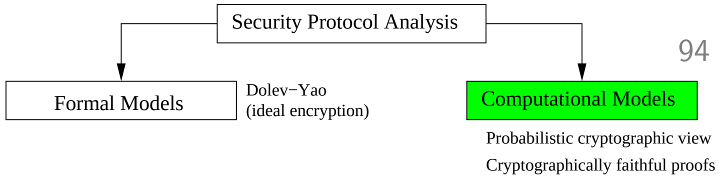
Road map

- Security protocols.
- Formal methods for security protocol analysis.
  - ▶ Dolev-Yao intruder and ping-pong protocols.
  - ▶ (Un-)decidability.
  - ▶ Logics of belief.
  - ▶ (Semi-)automated methods.
    - ∗ Interleaving trace models.
    - ∗ Falsification vs. verification of protocols.
  - ▶ Early and recent methods/tools.
  - ☞ **Computational models.**
  - ▶ Conclusions.

Computational Models

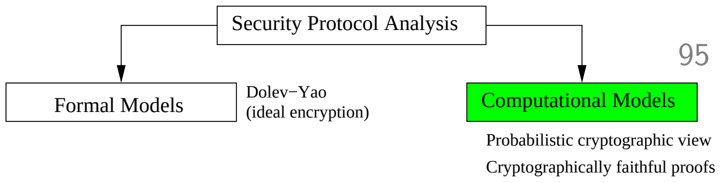

Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs

Computational Models


Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs

- The formal methods and cryptography communities have both developed formal techniques for protocol analysis.
- However, they have quite different points of view.
  - Cryptographers apply complexity and probability theory to reduce protocol security to security of 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.

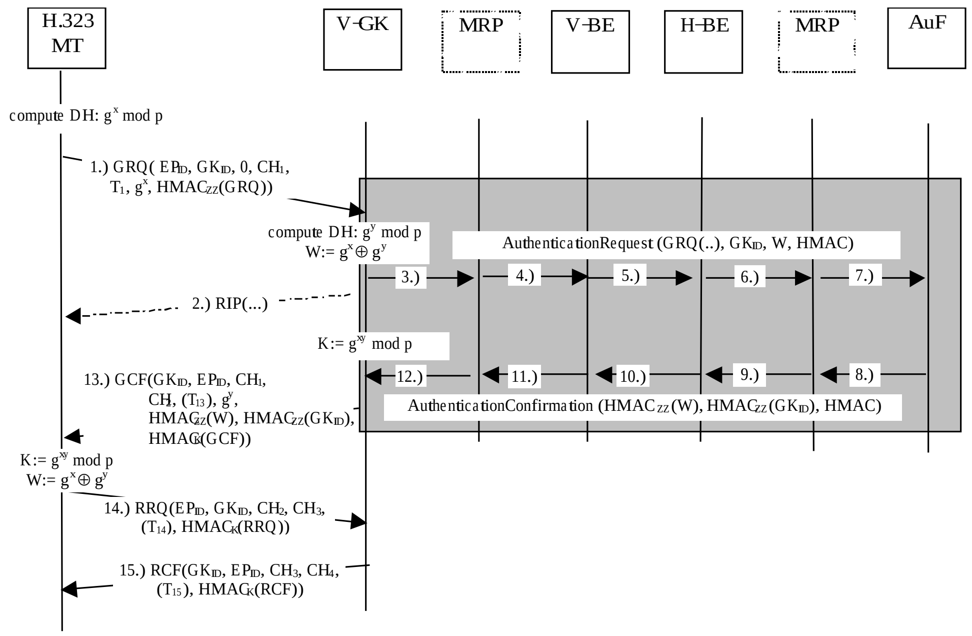
  There are protocols that are secure in the DY model, but become insecure if implemented with some provably secure crypto-primitives.


Security Protocol Analysis
Formal Models
Dolev−Yao
(ideal encryption)
Computational Models
Probabilistic cryptographic view
Cryptographically faithful proofs

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, Mitchell, Rogaway, Scedrov, ...
- 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.

The future: security sensitive protocols


H.323 MT
V-GK
MRP
V-BE
H-BE
MRP
AuF
compute DH: g^x mod p
1.) GRQ(EP_{ID}, GK_{ID}, 0, CH_1, T_1, g^x, $HMAC_{ZZ}$(GRQ))
compute DH: g^y mod p
W:= $g^x \oplus g^y$
AuthenticationRequest (GRQ(..), GK_{ID}, W, HMAC)
3.)
4.)
5.)
6.)
7.)
2.) RIP(...)
K:= g^{xy} mod p
12.)
11.)
10.)
9.)
8.)
13.) GCF(GK_{ID}, EP_{ID}, CH_1, CH_2, (T_{13}), g^y, $HMAC_{ZZ}$(W), $HMAC_{ZZ}$(GK_{ID}), $HMAC_K$(GCF))
AuthenticationConfirmation ($HMAC_{ZZ}$(W), $HMAC_{ZZ}$(GK_{ID}), HMAC)
K:= g^{xy} mod p
W:= $g^x \oplus g^y$
14.) RRQ(EP_{ID}, GK_{ID}, CH_2, CH_3, (T_{14}), $HMAC_K$(RRQ))
15.) RCF(GK_{ID}, EP_{ID}, CH_3, CH_4, (T_{15}), $HMAC_K$(RCF))

Examples:

Mobility: Mobile IP, mobile QoS (hand-off), mobile multi-media.
Session control: SIP, H323.
Authentication: Kerberos variants, AAA, PANA, http-digest.
Key agreement: Son-of-IKE.
End-to-End and Peer-to-Peer scenarios: SIP, SOAP, geopriv.

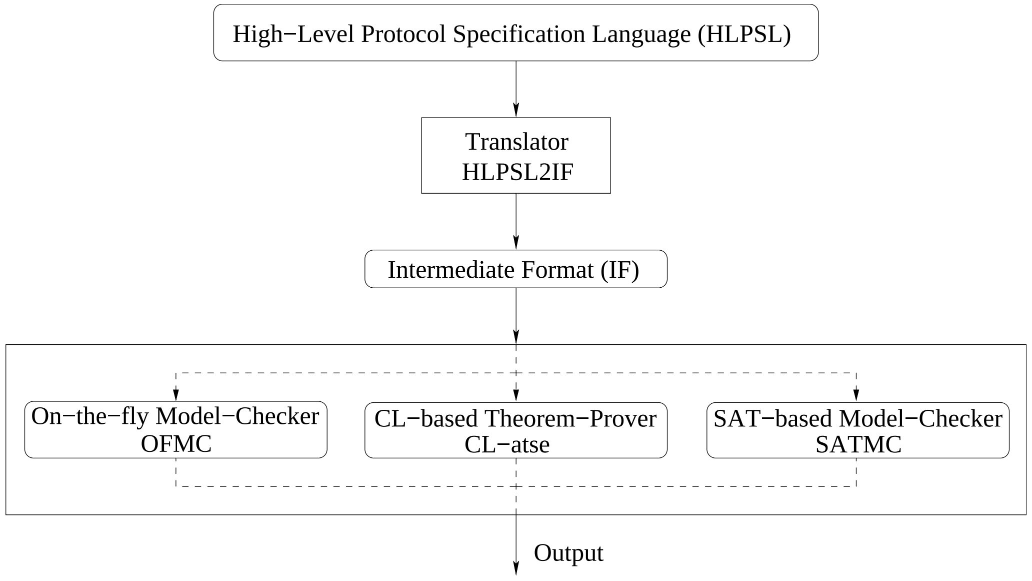
Conclusions

- It is possible and desirable to apply rigorous scientific methods to construct and analyze secure systems.

  $\Longrightarrow$ Requires work in foundations, tool support, and applications.

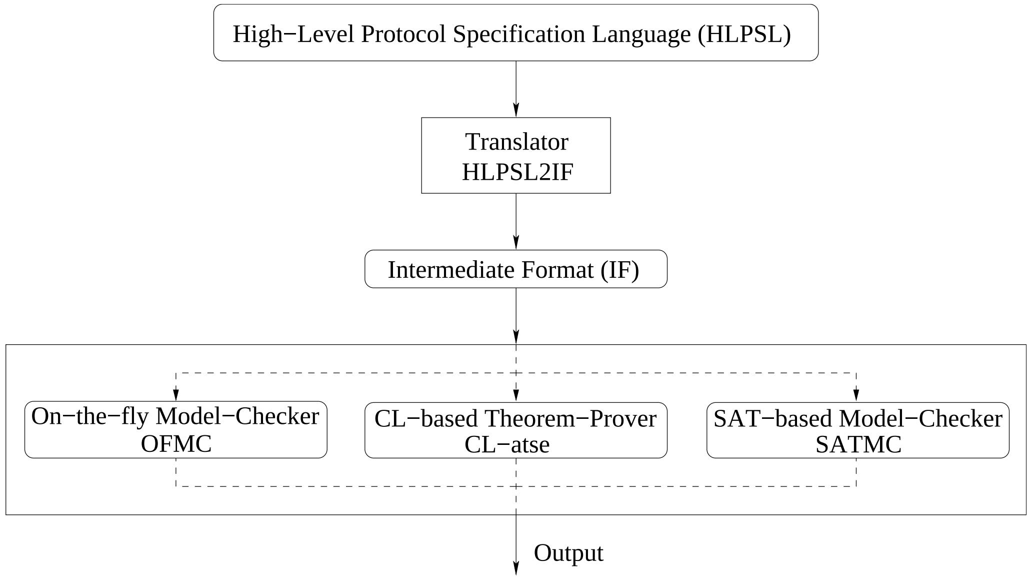
- Automated Validation of Internet Security Protocols and Applications (`www.avispa-project.org`).
  - ▶ AVISPA: a state-of-the-art tool for protocol analysis.
  - ▶ Both falsification and (bounded) verification of security protocols.

The AVISPA Tool www.avispa-project.org


High−Level Protocol Specification Language (HLPSL)
Translator HLPSL2IF
Intermediate Format (IF)
On−the−fly Model−Checker OFMC
CL−based Theorem−Prover CL−atse
SAT−based Model−Checker SATMC
Output

- Specification Languages:
  - ▶ **HLPSL**: a High-Level Protocol Specification Language.
    - ∗ Supports expressive, modular, high-level specifications of security sensitive protocols and properties.
    - ∗ Suitable for protocol designers.
  - ▶ **IF** (Intermediate Format): a tool-independent, low-level protocol specification language suitable for analysis.
  - ▶ **HLPSL2IF**: a translator from HLPSL into IF.

The AVISPA Tool www.avispa-project.org


High−Level Protocol Specification Language (HLPSL)
Translator HLPSL2IF
Intermediate Format (IF)
On−the−fly Model−Checker OFMC
CL−based Theorem−Prover CL−atse
SAT−based Model−Checker SATMC
Output

- Development of automatic analysis techniques to be integrated into robust, state-of-the-art tools.

  CL-atse, a protocol analyzer based on Constraint Logic developed by INRIA,
  SATMC, a SAT-based model-checker developed by UNIGE, and
  OFMC, an on-the-fly model-checker developed by ETHZ.

Bibliography

- Wenbo Mao. *Modern Cryptography*. Paerson, 2004.
- Peter Ryan, Steve Schneider, Michael Goldsmith, Gawin Lowe, Bill Roscoe. *Modelling and Analysis of Security Protocols*. Addison-Wesley, 2000.
- John Clark and Jeremy Jacob: A survey of authentication protocol literature, 1997. `http://www.cs.york.ac.uk/~jac/`
- Catherine Meadows: Formal Methods for Cryptographic Protocol Analysis. Emerging Issues and Trends. *IEEE Journal on Selected Areas in Communication*, 21(1):44–54, 2003.
- Michael Burrows, Martín Abadi, and Roger Needham. A logic of authentication. *ACM Transactions in Computer Systems*, 8(1):18–36, 1990.
- Martín Abadi and Roger Needham: Prudent Engineering Practice for Cryptographic Protocols. IEEE Transactions on Software Engineering, 22(1):2-15, 1996.
- David Basin, Sebastian Möderhseim, Luca Viganò. *An on-the-fly model-checker for security protocol analysis*. To appear in IJIS, 2004. See the papers available on our webpages, as well as on `http://www.avispa-project.org`.

Bibliography

- Various papers by Gawin Lowe.
- Bruce Schneier. *Applied Cryptography*. John Wiley & Sons, New York, 1996.
- Dieter Gollmann. *Computer Security*. Wiley, 2000.
- Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press, 1996.
  Available online at `http://cacr.math.uwaterloo.ca/hac/`
- Williams Stallings. *Cryptography and Network Security*. Prentice Hall, 2003.
- Matt Bishop. *Computer Security (Art and Science)*. Pearson, 2003.
- Kaufman, Perlman, Speciner. *Network Security: Private Communication in a Public World*, Prentice Hall, 2002.

Theorem proving

- In a nutshell:
  - System behavior is defined by a set of formulas.
  - Derivation of formulas is defined by axioms and/or inference rules.
  - Desired behavior or properties of the system being analyzed are specified as a set of theorems to be proved.
  - A proof of a theorem is carried out by using premises and applying axioms, rules, or already proved theorems to reach the desired consequences.

Theorem proving (cont.)

- Proof process can often be mechanized
  - E.g. rewriting a formula to some normal form.
- Mechanical prover may produce impractically large proofs.
  - A theorem proving approach is nevertheless capable of dealing with systems whose behavioral description cannot be represented by a finite structure (e.g. the system has an infinite state space).
  - E.g. induction-based proof of an integer-based mathematical statement.
- However, proofs often require the involvement of human ingenuity: semi-automated proofs.

Theorem proving (cont.)

- A theorem proving approach aims to demonstrate some desired property of a system (e.g. correctness), rather than to find errors in a system.
  - This is because usually an undesirable property cannot be formulated into a theorem.
  - Nevertheless, failure to demonstrate a desired property by a theorem proving system may often result in some insightful ideas leading to a revelation of a hidden error.
  - E.g. BAN and other authentication logics.
- Security protocols are extremely error-prone systems.
  - Correctness of a security protocol can be proved using an interactive (semi-automated) theorem proving approach.
  - This often requires restrictions and/or simplifications, and only semi-automation.

Model-checking

- In a nutshell:
  - System behavior is modeled as a (finite) state system.
  - Properties of the system can be expressed by some state satisfaction relations.
  - Analysis of behavior of the system by state space exploration to check whether certain properties will or will not be satisfied.
- In general:
  - Safety of a system: check that certain undesirable properties never occur.
  - Liveness of a system: check that certain desirable properties do eventually occur.

Model checking of security protocols focuses on safety properties.

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.

BAN deduction system: Inference rules

Rules express how beliefs evolve as a result of communication.

- Message-meaning rules: interpretation of messages (how to derive beliefs about origin of message).

  Encrypted messages:

$$\frac{P \textbf{ bel shk}(K,Q,P) \quad P \textbf{ sees } \{X\}_K}{P \textbf{ bel } Q \textbf{ said } X} \text{sk (from } R \neq P)$$

$$\frac{P \textbf{ bel pubk}(K,Q) \quad P \textbf{ sees } \{X\}_{K^{-1}}}{P \textbf{ bel } Q \textbf{ said } X} \text{pk}$$

  Messages with secrets:

$$\frac{P \textbf{ bel secret}(Y,Q,P) \quad P \textbf{ sees } \langle X\rangle_Y}{P \textbf{ bel } Q \textbf{ said } X} \text{secret } (\langle X\rangle_Y \text{ not by } P)$$

BAN deduction system: Inference rules (cont.)

- Nonce-verification rule: check that message is recent ($\Rightarrow$ sender still believes in it).

$$\frac{P \textbf{ bel fresh}(X) \quad P \textbf{ bel } (Q \textbf{ said } X)}{P \textbf{ bel } (Q \textbf{ bel } X)} \text{nonce } (X \text{ cleartext }, X \neq \{Y\}_K)$$

- Jurisdiction rule:

$$\frac{P \textbf{ bel } (Q \textbf{ controls } X) \quad P \textbf{ bel } (Q \textbf{ bel } X)}{P \textbf{ bel } X} \text{jurisdiction}$$

N.B.: quantifiers are implicit, which may cause ambiguities, e.g.

$P \textbf{ bel } \forall K.\,(S \textbf{ controls } (Q \textbf{ controls shk}(K, P, Q)))$ vs.

$P \textbf{ bel } (S \textbf{ controls } \forall K.\,(Q \textbf{ controls shk}(K, P, Q)))$

- Freshness rule: if a part of a formula is fresh, then the whole formula is fresh.

$$\frac{P \textbf{ bel fresh}(X)}{P \textbf{ bel fresh}(X, Y)} \text{fresh}$$

The full NSPK

$$
\begin{array}{lll}
\text{M1.} & A \rightarrow S : & A, B \\
\text{M2.} & S \rightarrow A : & \{K_B, B\}_{K_S^{-1}} \\
\text{M3.} & A \rightarrow B : & \{NA, A\}_{K_B} \\
\text{M4.} & B \rightarrow S : & B, A \\
\text{M5.} & S \rightarrow B : & \{K_A, A\}_{K_S^{-1}} \\
\text{M6.} & B \rightarrow A : & \{NA, NB\}_{K_A} \\
\text{M7.} & A \rightarrow B : & \{NB\}_{K_B}
\end{array}
$$

- Initially: A and B hold K_S.
- Aim 1: A and B obtain each other's public keys (M 1, 2, 4, 5).
- Aim 2: A and B use keys to communicate secret nonces NA and NB (M 3, 6, 7).
- 'Later': A and B use NA and NB to sign further messages.

Protocol is flawed: no guarantee that public keys K_A and K_B are fresh $\Rightarrow$ replay attack!