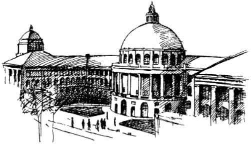
Symbolic verification of cryptographic protocols using Tamarin

Part 1 : Introduction

David Basin
ETH Zurich



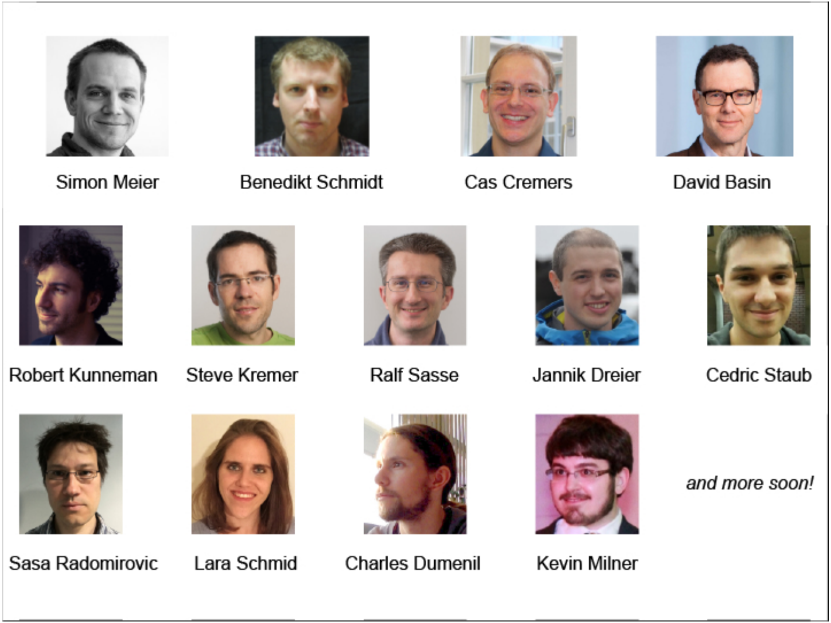
Summer School on Verification Technology,
Systems & Applications
Nancy France
August 2018

Organization of the course

- Two lectures covering:
  - Background on security protocols
  - Modeling and reasoning
  - Aspects of a state-of-the-art model checker (Tamarin)
- Lectures hopefully interactive: ask questions!
- Main resource: Tamarin website: `https://tamarin-prover.github.io/`
  - Tool source, ready for download
  - 100+ page manual
  - Examples, case studies
  - Algorithms, theses, papers, ...

What is described here is a team effort

Tamarin Contributors



Simon Meier

Benedikt Schmidt

Cas Cremers

David Basin

Robert Kunneman

Steve Kremer

Ralf Sasse

Jannik Dreier

Cedric Staub

Sasa Radomirovic

Lara Schmid

Charles Dumenil

Kevin Milner

and more soon!

Contents

module	content
1	Introduction to Security Protocols, Protocol Specification
2	Term Rewriting; Protocol Syntax and Semantics
3	Security Properties and Algorithmic Verification
4	Advanced Features [time permitting]

Tamarin?!

Tamarin?!

The **Tamarin** prover is a *security protocol verification tool* that supports both falsification and unbounded *verification* in the *symbolic model*.

Outline

1. Motivation

2. Building a key establishment protocol

3. Formalizing Security Protocols: An Example

4. Protocol attacks

5. Outlook

Plan

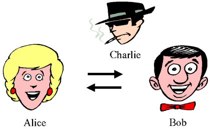
1. Motivation
2. Building a key establishment protocol
3. Formalizing Security Protocols: An Example
4. Protocol attacks
5. Outlook

Motivation

- Many good cryptographic primitives:
  RSA, DSA, ElGamal, AES, SHA3 . . .
- How can we construct secure distributed applications with them?
  - E-commerce
  - E-banking
  - E-voting
  - Mobile communication
  - Digital contract signing
- Even if cryptography is hard to break, this is not a trivial task.

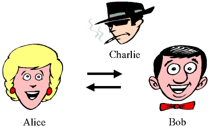
Motivation

Example: Securing an e-banking application



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

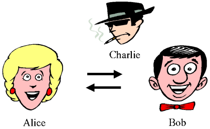
Example: Securing an e-banking application



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

- **How does B know the message originated from A?**
- **How does B know A just said it?**
- Confidentiality, integrity, accountability, non-repudiation, ...?

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?**
- Confidentiality, integrity, accountability, non-repudiation, ...?

Solutions involve protocols like IPsec, Kerberos, SSH, SSL/TLS, Signal, PGP, ...

Let's consider underlying ideas and some example protocols.

What is a protocol?

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

Plan

1. Motivation

2. Building a key establishment protocol

3. Formalizing Security Protocols: An Example

4. Protocol attacks

5. Outlook

Building a key establishment protocol

- Let's try to design a cryptographic protocol from first principles.
- We choose a common scenario:
  - A **set of users**, any 2 of whom may wish to establish a new **session key** for subsequent secure communications.

    Note: Users are not necessarily honest! (More later)
  - There is an **honest server**.

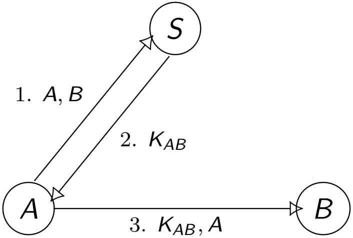
    Note: Often called "trusted server", but trust $\neq$ honesty! We assume that an honest server never cheats and never gives out user secrets.
- We thus design a protocol with three **roles**: initiator role A, responder role B, and server role S.

Preliminaries

- In a concrete execution of a protocol, the roles are **played by agents** a.k.a. principals: a, b, c, s, i (intruder), ...
- We use i as the name of an agent whose long-term keys are known to the adversary. This fact isn't known by other (honest) agents in the model.
- **Security goals of the protocol:**
  - Key secrecy: At the end of the protocol, the session key K_{AB} is known to A and B, and possibly S, but to no other parties.
  - Key freshness: A and B know that K_{AB} is freshly generated.
- Formalization questions (that we will consider later):
  - How do we formalize the protocol steps and goals?
  - How about "knowledge", "secrecy", "freshness"?

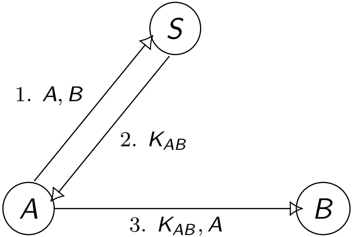
First attempt

A 3 message protocol



1. A contacts S by sending the identities of the 2 parties who are going to share the session key.
2. S sends the key K_{AB} to A.
3. A forwards K_{AB} to B.

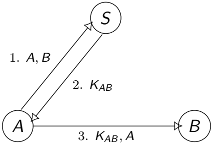
Alice&Bob notation



$$\begin{array}{lll} 1. & A \rightarrow S: & A,\ B \\ 2. & S \rightarrow A: & K_{AB} \\ 3. & A \rightarrow B: & K_{AB},\ A \end{array}$$

Note: sender-receiver pairs "$A \rightarrow B$" in Alice&Bob notation are not part of the communicated message.

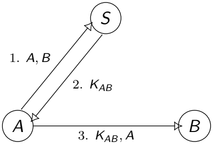
Protocol specification issues and conventions



$$\begin{array}{lll} 1. & A \to S: & A,\ B \\ 2. & S \to A: & K_{AB} \\ 3. & A \to B: & K_{AB},\ A \end{array}$$

- K_{AB} contains no information about A and B.
  It simply names the bit-string representing the session key.
- What if improperly formatted message or no message is received?
  - Only messages exchanged in a successful protocol run specified.
- No specification of internal actions of principals.
  E.g., "create fresh K_{AB}", "store K_{AB} as a key for A and B".
- Roles "know" what protocol run received messages are part of.

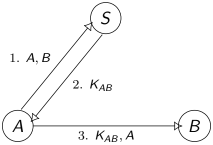
Security issues



$$\begin{array}{lll} 1. & A \to S: & A,\ B \\ 2. & S \to A: & K_{AB} \\ 3. & A \to B: & K_{AB},\ A \end{array}$$

- First problem with this protocol?

Security issues



$$
\begin{array}{lll}
1. & A \to S: & A, B \\
2. & S \to A: & K_{AB} \\
3. & A \to B: & K_{AB}, A
\end{array}
$$

- First problem with this protocol? **Secrecy.**
  The session key K_{AB} must be transported to A and B, but readable by *no other parties*.
- A realistic assumption in typical communication systems such as the Internet and corporate networks:

Assumption (Threat Assumption 1)

The adversary is able to eavesdrop on all sent messages.

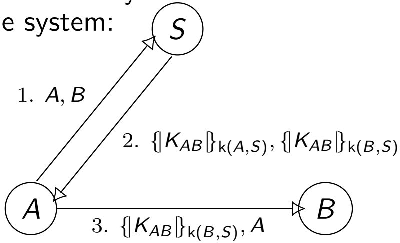
⇒ Use cryptography.

Second Attempt

Use cryptography

- Assume that S initially shares a secret key $\mathsf{k}(U, S)$ with each user U of the system:
  - $\mathsf{k}(A, S)$ with A,
  - $\mathsf{k}(B, S)$ with B,

  and encrypt message 2.



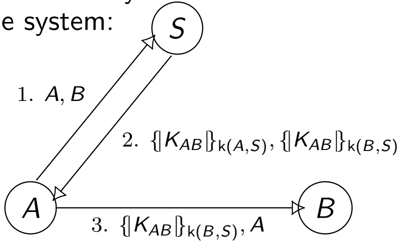
S
1. A, B
2. $\{\!|K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}$
A
B
3. $\{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}, A$

Second Attempt

Use cryptography

- Assume that S initially shares a secret key $\mathsf{k}(U, S)$ with each user U of the system:
    - $\mathsf{k}(A, S)$ with A,
    - $\mathsf{k}(B, S)$ with B,

  and encrypt message 2.
- Problems with protocol?
    - Eavesdropping?



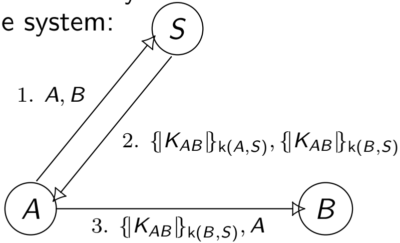
S
1. A, B
2. $\{\!|K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}$
A
B
3. $\{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}, A$

Second Attempt

Use cryptography

- Assume that S initially shares a secret key $\mathsf{k}(U, S)$ with each user U of the system:
  - $\mathsf{k}(A, S)$ with A,
  - $\mathsf{k}(B, S)$ with B,

  and encrypt message 2.
- Problems with protocol?
  - Eavesdropping? – No.



S
1. A, B
2. $\{\!|K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}$
A
B
3. $\{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}, A$

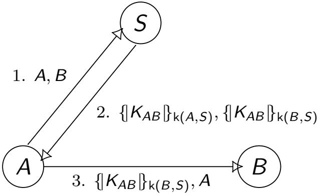
Assumption (Perfect Cryptography)

Encrypted messages may be read only by recipients who have the required decryption key.

Assuming that cryptography is perfect allows us to abstract away from the details of cryptographic algorithms.

Second Attempt

- Problem: information about who else has K_{AB} is unprotected, i.e. the principal's names are not bound to K_{AB}.


S
1. A, B
2. $\{\!|K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}$
A
B
3. $\{\!|K_{AB}|\!\}_{\mathsf{k}(B,S)}, A$

- Adversary may not only eavesdrop on sent messages, but also capture and modify them.

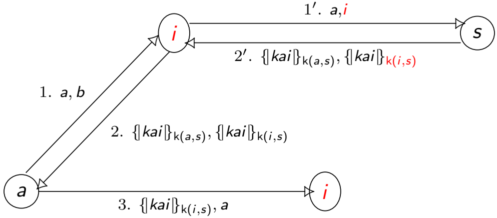
Assumption (Threat Assumption 2)

The adversary is able to intercept messages on the network and send to anybody (under any sender name) modified or new messages based on any information available.

- In other words: the adversary has complete control of the network channel(s) over which protocol messages flow.
  The adversary has complete control over the network.
- In contrast to ordinary communication protocols, we assume the worst-case network adversary.
  - Although there may be only a few messages involved in a legitimate protocol session, there are infinitely many variations where the adversary can participate.
  - These variations involve an unbounded number of messages and each must satisfy the protocol's security requirements.

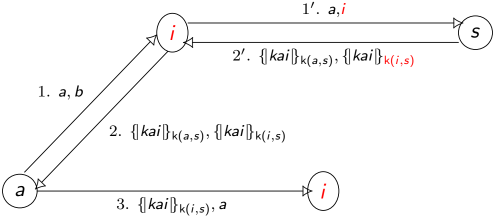
Binding Attack

There are several binding attacks on the second protocol attempt, e.g.:


i
s
a
i
$1'.\ a, i$
$2'.\ \{\!|kai|\!\}_{\mathsf{k}(a,s)}, \{\!|kai|\!\}_{\mathsf{k}(i,s)}$
$1.\ a, b$
$2.\ \{\!|kai|\!\}_{\mathsf{k}(a,s)}, \{\!|kai|\!\}_{\mathsf{k}(i,s)}$
$3.\ \{\!|kai|\!\}_{\mathsf{k}(i,s)}, a$

- Modify message from a to s so that s generates a key kai for a and i and encrypts it with key $\mathsf{k}(i, s)$ known by adversary.
- Since a cannot distinguish between encrypted messages meant for other agents, she will not detect this.
- a will believe the protocol was successfully completed with b. However, the adversary knows kai and can masquerade as b and also learn all information that a sends intended for b.

Binding Attack



$1'.\ a, i$
s
i
$2'.\ \{|kai|\}_{k(a,s)}, \{|kai|\}_{k(i,s)}$
$1.\ a, b$
$2.\ \{|kai|\}_{k(a,s)}, \{|kai|\}_{k(i,s)}$
a
i
$3.\ \{|kai|\}_{k(i,s)}, a$

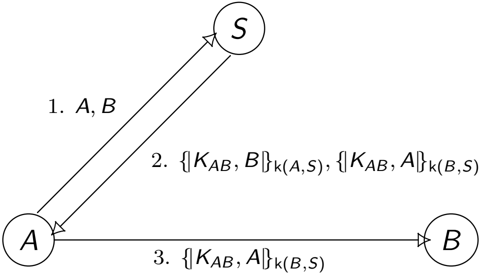
- Note: This attack will only succeed if i is a legitimate system user known to s, which is a realistic assumption:

Assumption (Threat Assumption 3)

The adversary may be a legitimate protocol participant (an insider), or an external party (an outsider), or a combination of both.

Third Attempt

- To overcome the last attack, the names of the principals who should share K_{AB} must be bound cryptographically to the key.



S
1. A, B
2. $\{\!|K_{AB}, B|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}$
A
B
3. $\{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}$

- Improvement: An adversary cannot attack the protocol by eavesdropping or modifying the messages sent between honest parties (so now the previous two attacks fail).
- However, the protocol still does not provide security in normal operating conditions.

Third Attempt

Security Issues

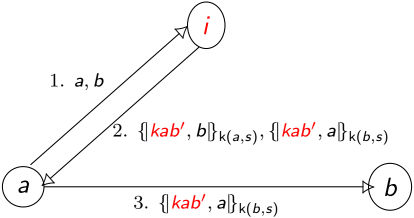
- A whole class of attacks becomes possible when old keys (or other security-relevant data) can be **replayed** in subsequent sessions. (Replay is possible by Threat Assumptions 1 & 2.)
- Additional problem: difference in quality between long-term keys and keys K_{AB} generated for each protocol session.

Assumption (Threat Assumption 4)

The adversary is able to obtain the value of the session key K_{AB} used in any "sufficiently old" previous run of the protocol.

- Reasons for using session keys (as opposed to long-term keys):
  - Key distribution problem: $\mathcal{O}(n^2)$ keys needed for n principals.
  - Encrypted messages are vulnerable to attack (by cryptanalysis).
  - Communications in different sessions should be separated. In particular, it should not be possible to replay messages from previous sessions.

Replay Attack and Session Key Compromise



i
1. a, b
2. $\{|kab', b|\}_{k(a,s)}, \{|kab', a|\}_{k(b,s)}$
a
3. $\{|kab', a|\}_{k(b,s)}$
b

- *i* masquerades as *s* and replays kab', an old key used by *a* and *b* in a previous session.
    - By Threat Assumptions 1 & 2, the adversary can be expected to know and replay the encrypted messages in which kab' was transported to *a* and *b*.
- After protocol run, adversary can decrypt, modify, or insert messages encrypted with kab' (no confidentiality or integrity).
    - By Threat Assumption 4, the adversary can be expected to know kab'.

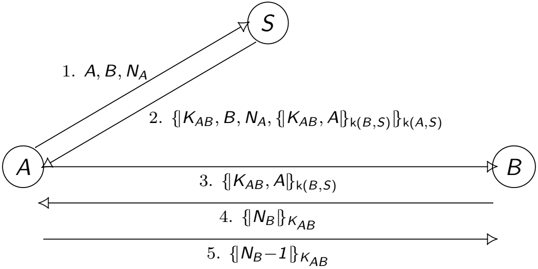
Thwarting the Replay Attack

- The replay attack can still be regarded as successful even if the adversary has not obtained the value of kab':
  - Adversary gets a and b to accept an old session key!
  - The intruder i can therefore replay messages protected by kab' sent in the previous session.
- Of course: *provided that a and b don't check the key!*
  - But "Principals don't think", they just follow the protocol.
- Various techniques may be used to guard against replay of session key, such as incorporating **challenge-response**

Definition

A **nonce** ("a number used only once") is a random value generated by one principal and returned to that principal to show that a message is newly generated.

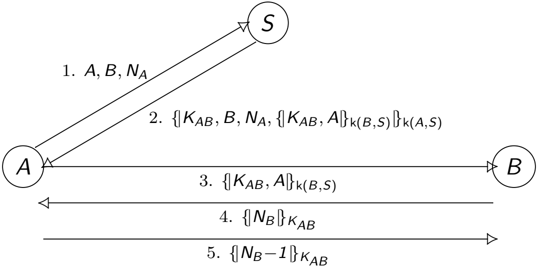
Fourth Attempt



S
1. A, B, N_A
2. $\{\!|K_{AB}, B, N_A, \{\!|K_{AB}, A|\!\}_{k(B,S)}|\!\}_{k(A,S)}$
A
B
3. $\{\!|K_{AB}, A|\!\}_{k(B,S)}$
4. $\{\!|N_B|\!\}_{K_{AB}}$
5. $\{\!|N_B - 1|\!\}_{K_{AB}}$

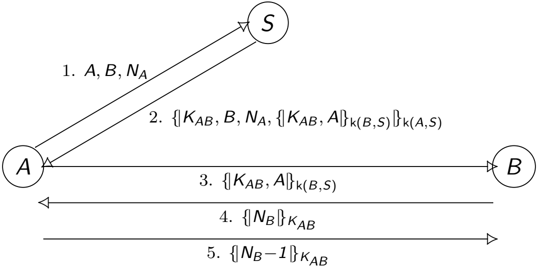
- A sends her nonce N_A to S with the request for a new key.
    - Note: N_A just names a number;
      nothing in N_A identifies who created it.
- If this same value is received with the session key, then A can deduce that the key has not been replayed.
  This reasoning is valid if session key and nonce are bound together cryptographically so that only S can form such a message.

Fourth Attempt


S
1. A, B, N_A
2. $\{\!|K_{AB}, B, N_A, \{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}|\!\}_{\mathsf{k}(A,S)}$
A
B
3. $\{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}$
4. $\{\!|N_B|\!\}_{K_{AB}}$
5. $\{\!|N_B - 1|\!\}_{K_{AB}}$

- If the encrypted key for B is included in the encrypted part of A's message, then A gains assurance that it is fresh.
- It is tempting to believe that A may pass this assurance on to B in an extra handshake:
  - B generates nonce N_B and sends it to A protected by K_{AB}.
  - A uses K_{AB} to send a reply to B, transformed (-1) to avoid replay of message 4.

The NSCK Protocol



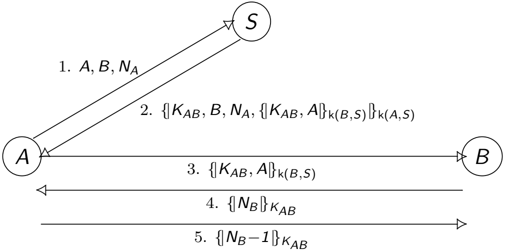
S
A
B
1. A, B, N_A
2. $\{\!| K_{AB}, B, N_A, \{\!| K_{AB}, A |\!\}_{\mathsf{k}(B,S)} |\!\}_{\mathsf{k}(A,S)}$
3. $\{\!| K_{AB}, A |\!\}_{\mathsf{k}(B,S)}$
4. $\{\!| N_B |\!\}_{K_{AB}}$
5. $\{\!| N_B - 1 |\!\}_{K_{AB}}$

- This is a well-known security protocol:
  Needham Schroeder with Conventional Keys
- Published by Needham and Schroeder in 1978,
- Basis for a whole class of related protocols.

Looks pretty good! Right?

The NSCK Protocol

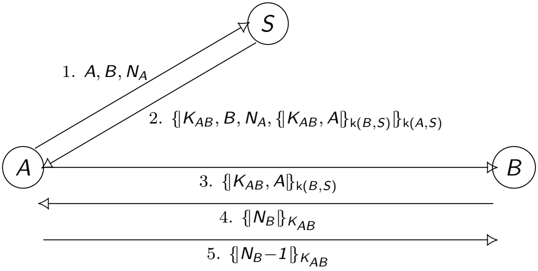
Security Issue



S
A
B
1. A, B, N_A
2. $\{\!|K_{AB}, B, N_A, \{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}|\!\}_{\mathsf{k}(A,S)}$
3. $\{\!|K_{AB}, A|\!\}_{\mathsf{k}(B,S)}$
4. $\{\!|N_B|\!\}_{K_{AB}}$
5. $\{\!|N_B - 1|\!\}_{K_{AB}}$

- Unfortunately, this protocol is vulnerable to a (now) well-known attack due to Denning and Sacco.
    - Problem: assumption that only A can form correct reply to message 4 from B.
- Since the adversary can be expected to know the value of old session keys, this assumption is unrealistic.

Attack on NSCK



1. A, B, N_A

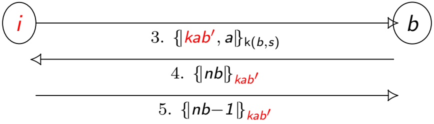
2. $\{|K_{AB}, B, N_A, \{|K_{AB}, A|\}_{\mathsf{k}(B,S)}|\}_{\mathsf{k}(A,S)}$

3. $\{|K_{AB}, A|\}_{\mathsf{k}(B,S)}$

4. $\{|N_B|\}_{K_{AB}}$

5. $\{|N_B - 1|\}_{K_{AB}}$

Adversary masquerades as *a* and convinces *b* to use old key: kab':



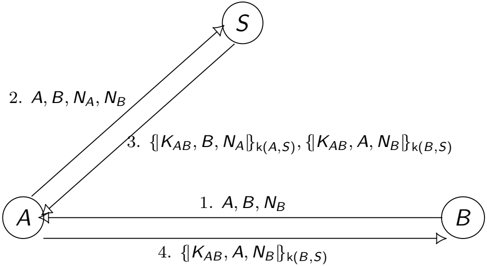
3. $\{|kab', a|\}_{\mathsf{k}(b,s)}$

4. $\{|nb|\}_{kab'}$

5. $\{|nb - 1|\}_{kab'}$

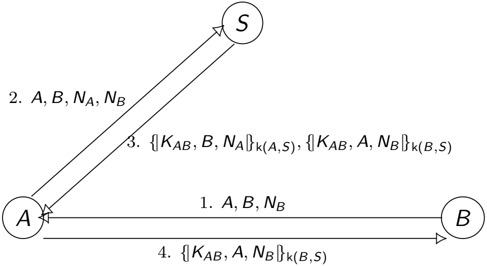
Fifth and Final Attempt

Different approach: remove the assumption that it is inconvenient for both B and A to send their challenges to S.


S
2. A, B, N_A, N_B
3. $\{|K_{AB}, B, N_A|\}_{\mathsf{k}(A,S)}, \{|K_{AB}, A, N_B|\}_{\mathsf{k}(B,S)}$
1. A, B, N_B
A
B
4. $\{|K_{AB}, A, N_B|\}_{\mathsf{k}(B,S)}$

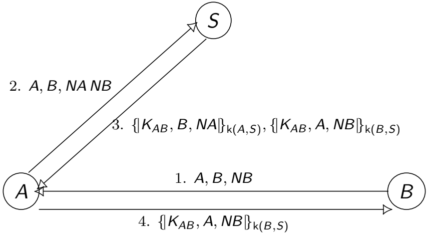
- The protocol is now initiated by B who sends his nonce N_B first to A.
- A adds her nonce N_A and sends both to S, who now sends K_{AB} in separate messages for A and B, which can be verified as fresh by the respective recipients.

Fifth and Final Attempt


S
2. A, B, N_A, N_B
3. $\{|K_{AB}, B, N_A|\}_{\mathsf{k}(A,S)}, \{|K_{AB}, A, N_B|\}_{\mathsf{k}(B,S)}$
1. A, B, N_B
A
B
4. $\{|K_{AB}, A, N_B|\}_{\mathsf{k}(B,S)}$

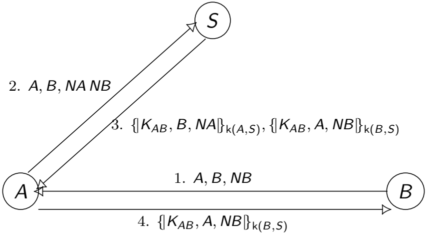
- It may seem that we have achieved more than the previous protocol using fewer messages, but in fact...
  - In NSCK, A could verify that B has actually received the key. This **key confirmation** property was achieved due to B's use of the key in message 4, assuming that $\{|N_B|\}_{K_{AB}}$ cannot be formed without knowledge of K_{AB}.
  - In above protocol, neither A nor B can deduce at the end of a successful run that its partner actually received K_{AB}. **(Is this a problem?)**

So is our Protocol Secure?


S
2. $A, B, NA\, NB$
3. $\{\!|K_{AB}, B, NA|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}, A, NB|\!\}_{\mathsf{k}(B,S)}$
1. A, B, NB
A
B
4. $\{\!|K_{AB}, A, NB|\!\}_{\mathsf{k}(B,S)}$

- This protocol avoids all the attacks shown so far.
  Under assumption that cryptographic algorithms used work correctly and the server S acts correctly.
- So is it correct?

So is our Protocol Secure?


S
2. $A, B, NA\,NB$
3. $\{\!|K_{AB}, B, NA|\!\}_{\mathsf{k}(A,S)}, \{\!|K_{AB}, A, NB|\!\}_{\mathsf{k}(B,S)}$
1. A, B, NB
A
B
4. $\{\!|K_{AB}, A, NB|\!\}_{\mathsf{k}(B,S)}$

- This protocol avoids all the attacks shown so far.
  Under assumption that cryptographic algorithms used work correctly and the server S acts correctly.
- So is it correct?
  It would be rash to make such a claim before giving a precise meaning to all of this.
- The security of a protocol must always be considered relative to its goals and assumptions.

  We must formalize (and prove) protocol security!

Exercise

Can you improve on these protocols using Diffie-Hellman?
In what sense is it an improvement?

Summary: adversary, attacks, and defenses

The **adversary** must be expected to

(TA 1) eavesdrop on messages, but cannot break cryptography,

(TA 2) completely control the network, i.e.,

- immediately intercept, modify, and fake messages,
- compose/decompose messages with the available keys,

(TA 3) participate in the protocol (as insider or outsider), and

(TA 4) be able to obtain old session keys.

TA 1-3: worst-case assumption of network adversary (after Dolev-Yao)

Attacks and defenses:

- Eavesdropping: encrypt session keys using long-term keys
- Binding attack: cryptographically bind names to session keys
- Replay attack: use challenge-response based on nonces

Plan

1. Motivation
2. Building a key establishment protocol
3. Formalizing Security Protocols: An Example
4. Protocol attacks
5. Outlook

An authentication protocol

The Needham-Schroeder Public Key protocol (NSPK, 1978):

$$\begin{array}{lll} 1. & A \to B: & \{NA, A\}_{\mathsf{pk}(B)} \\ 2. & B \to A: & \{NA, NB\}_{\mathsf{pk}(A)} \\ 3. & A \to B: & \{NB\}_{\mathsf{pk}(B)} \end{array}$$

Security goal: mutual authentication of A and B.

How the protocol is executed

1. $A \rightarrow B: \quad \{NA, A\}_{\mathrm{pk}(B)}$
2. $B \rightarrow A: \quad \{NA, NB\}_{\mathrm{pk}(A)}$
3. $A \rightarrow B: \quad \{NB\}_{\mathrm{pk}(B)}$

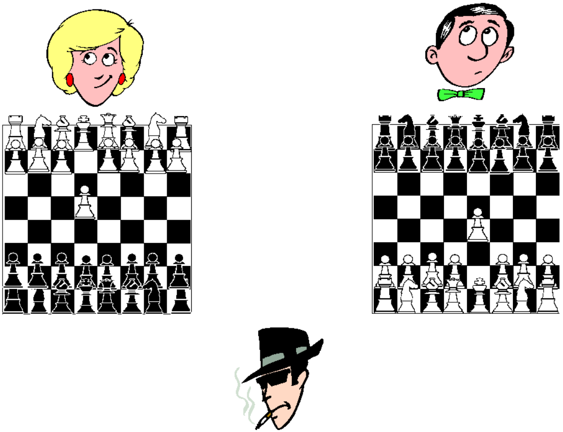
Role A:

1. Construct and send message 1.
   - Generate nonce NA, concatenate it with name A, and encrypt with $\mathrm{pk}(B)$.
   - Send $\{NA, A\}_{\mathrm{pk}(B)}$ to B.
2. Receive a message M and check that it is message 2.
   Q: how to do this when running multiple sessions (or protocols) in parallel?
   - Decrypt M with $\mathrm{sk}(A)$, call it M'. If decryption fails, reject M.
     Q: how to detect wrong decryption?
     Q: what to do about rejected messages?
   - Split the message into two nonces NA' and NB. If not possible, reject M.
     Q: how to check this?
   - Check that $NA' = NA$; if not, reject M.
3. Construct and send message 3.
   - Encrypt NB with $\mathrm{pk}(B)$
   - Send $\{NB\}_{\mathrm{pk}(B)}$ to B.

Informal correctness

1. $A \to B : \{NA, A\}_{\mathsf{pk}(B)}$ “This is Alice and I have chosen a nonce *NA*.”
2. $B \to A : \{NA, NB\}_{\mathsf{pk}(A)}$ “Here is your nonce *NA*. Since I could read it, I must be Bob. I also have a challenge *NB* for you.”
3. $A \to B : \{NB\}_{\mathsf{pk}(B)}$ “You sent me *NB*. Since only Alice can read this and I sent it back, I must be Alice.”

How not to lose against a grandmaster in chess



Man-in-the-middle attack

NSPK (1978):

$$\begin{array}{lll} 1. & A \rightarrow B : & \{NA, A\}_{\mathsf{pk}(B)} \\ 2. & B \rightarrow A : & \{NA, NB\}_{\mathsf{pk}(A)} \\ 3. & A \rightarrow B : & \{NB\}_{\mathsf{pk}(B)} \end{array}$$

Attack (Lowe 1996):

$$\begin{array}{lllllll} 1. & a \rightarrow i : & \{na, a\}_{\mathsf{pk}(i)} & & & & \\ & & & & 1.' & i(a) \rightarrow b : & \{na, a\}_{\mathsf{pk}(b)} \\ & & & & 2.' & b \rightarrow i(a) : & \{na, nb\}_{\mathsf{pk}(a)} \\ 2. & \color{red}{i \rightarrow a :} & \color{red}{\{na, nb\}_{\mathsf{pk}(a)}} & & & & \\ 3. & a \rightarrow i : & \{nb\}_{\mathsf{pk}(i)} & & & & \\ & & & & 3.' & i(a) \rightarrow b : & \{nb\}_{\mathsf{pk}(b)} \end{array}$$

Property that "*b* authenticates *a*" is violated:
b believes to be talking to *a*, where he is in fact talking to *i*.

What went wrong?

- Problem in step 2.

$$B \rightarrow A : \{NA, NB\}_{\mathsf{pk}(A)}$$

  This message does not say where it comes from!
  (*A* cannot check her assumption on partner)

- Agent *B* should also give his name: $\{NA, NB, B\}_{\mathsf{pk}(A)}$.
  Known as Lowe's Fix.
- Is the improved version now correct?

Plan

1. Motivation
2. Building a key establishment protocol
3. Formalizing Security Protocols: An Example
4. Protocol attacks
5. Outlook

Types of protocol attacks

- **Man-in-the-middle attack**: $a \leftrightarrow i \leftrightarrow b$.
- **Replay** (or **freshness**) **attack**: reuse parts of previous messages.
- **Masquerading attack**: pretend to be another principal.
- **Reflection attack**: send transmitted information back to originator
- **Oracle attack**: take advantage of normal protocol responses as encryption and decryption "services".
- **Binding attack**: using messages in a different context/for a different purpose than originally intended.
- **Type flaw attack**: substitute a different type of message field.

Note that these attack types are not formally defined and there may be overlaps between them.

Diffie-Hellman: man-in-the-middle attack

- Textbook Diffie-Hellman can be attacked:

$$\begin{array}{llllll}
1. & a \to i(b): & \exp(g,x) & & & \\
 & & & 1.' & i(a) \to b: & \exp(g,z) \\
 & & & 2.' & b \to i(a): & \exp(g,y) \\
2. & i(b) \to a: & \exp(g,z) & & &
\end{array}$$

- a believes to share key $\exp(\exp(g,z),x)$ with b.
- b believes to share key $\exp(\exp(g,z),y)$ with a.
- The adversary knows both keys.

Diffie-Hellman: man-in-the-middle attack

- A "half" man-in-the-middle attack is possible, too:

$$\begin{array}{ll} 1. & a \to i(b): \quad \exp(g, x) \\ 2. & i(b) \to a: \quad \exp(g, z) \end{array}$$

- Countermeasure: authenticate the half-keys, e.g., with digital signatures:

$$\begin{array}{ll} 1. & A \to B: \quad \{\exp(g, X)\}_{\mathsf{sk}(A)} \\ 2. & B \to A: \quad \{\exp(g, Y)\}_{\mathsf{sk}(B)} \end{array}$$

- Many protocols are based on (authenticated) Diffie-Hellman, which is not a bad idea!

Example of a reflection attack

This challenge-response authentication protocol

$$\begin{array}{lll} \text{M1.} & A \rightarrow B: & \{\!|NA|\!\}_{\mathsf{k}(A,B)} \\ \text{M2.} & B \rightarrow A: & \{\!|NA+1|\!\}_{\mathsf{k}(A,B)} \end{array}$$

admits a reflection attack (with ‘oracle’):

Example of a reflection attack

This challenge-response authentication protocol

$$\begin{array}{lll} \text{M1.} & A \to B : & \{\!|NA|\!\}_{\mathsf{k}(A,B)} \\ \text{M2.} & B \to A : & \{\!|NA+1|\!\}_{\mathsf{k}(A,B)} \end{array}$$

admits a reflection attack (with 'oracle'):

$$\begin{array}{llll} \text{M1.1.} & a \to i(b) : \{\!|na|\!\}_{\mathsf{k}(a,b)} & & \\ & & \text{M2.1.} & i(b) \to a : \{\!|na|\!\}_{\mathsf{k}(a,b)} \\ & & \text{M2.2.} & a \to i(b) : \{\!|na+1|\!\}_{\mathsf{k}(a,b)} \\ \text{M1.2.} & i(b) \to a : \{\!|na+1|\!\}_{\mathsf{k}(a,b)} & & \end{array}$$

a works on behalf of the adversary: *a* acts as an 'oracle', since she provides the correct answer to her own question.

a believes (at least) that *b* is operational, whereas *b* may no longer exist.

Fix: add *A*'s name to message M1 or use different keys for each direction (i.e., $\mathsf{k}(a,b) \neq \mathsf{k}(b,a)$).

Example of a reflection attack

Reflection attack (with 'oracle'):

M1.1. $a \to i(b) : \{\!|na|\!\}_{\mathsf{k}(a,b)}$

M2.1. $i(b) \to a : \{\!|na|\!\}_{\mathsf{k}(a,b)}$

M2.2. $a \to i(b) : \{\!|na + 1|\!\}_{\mathsf{k}(a,b)}$

M1.2. $i(b) \to a : \{\!|na + 1|\!\}_{\mathsf{k}(a,b)}$

This attack requires that *a* executes several protocol runs in parallel.

Assumption (Threat Assumption 5)

The adversary may start any number of parallel protocol runs between any principals including different runs involving the same principals and with principals taking the same or different protocol roles.

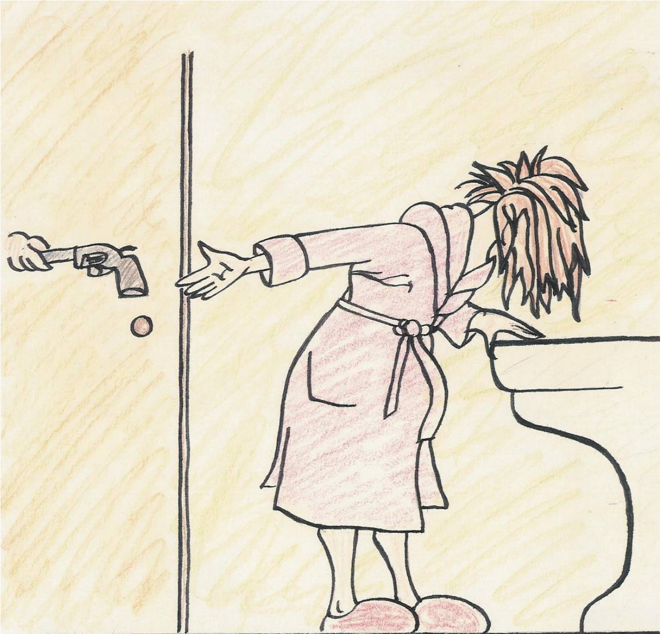
Hence, our formal protocol model will allow for an unbounded number of protocol runs of arbitrary roles and with arbitrary participating principals.

Type flaw attacks

- A message consists of a sequence of submessages.
  Examples: a principal's name, a nonce, a key, ...
- Real messages are bit strings without 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 a couple examples.



"HONEY... HAND ME THE HAIR DRYER."

The Otway-Rees protocol

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

$$\begin{array}{lll} \text{M1.} & A \rightarrow B: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)} \\ \text{M2.} & B \rightarrow S: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, M, A, B|\!\}_{\mathsf{k}(B,S)} \\ \text{M3.} & S \rightarrow B: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, K_{AB}|\!\}_{\mathsf{k}(B,S)} \\ \text{M4.} & B \rightarrow A: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)} \end{array}$$

where server keys already known and M is a session id (e.g., an integer).

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

Type flaw attack on the Otway-Rees protocol

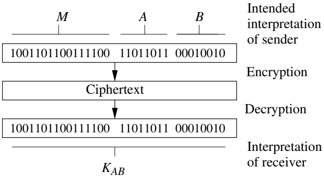
M1. $A \to B: \quad M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)}$

M2. $B \to S: \quad M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, M, A, B|\!\}_{\mathsf{k}(B,S)}$

M3. $S \to B: \quad M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, K_{AB}|\!\}_{\mathsf{k}(B,S)}$

M4. $B \to A: \quad M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}$

- Suppose $|M, A, B| = |K_{AB}|$, e.g., M is 32 bits, A and B are 16 bits, and K_{AB} is 64 bits.



M A B

Intended interpretation of sender

1001101100111100 11011011 00010010

Encryption

Ciphertext

Decryption

1001101100111100 11011011 00010010

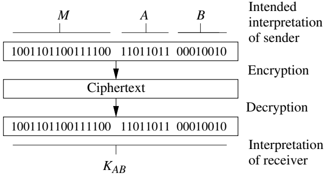
Interpretation of receiver

K_{AB}

Type flaw attack on the Otway-Rees protocol

$$\begin{array}{lll}
\text{M1.} & A \to B: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)} \\
\text{M2.} & B \to S: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, M, A, B|\!\}_{\mathsf{k}(B,S)} \\
\text{M3.} & S \to B: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, K_{AB}|\!\}_{\mathsf{k}(B,S)} \\
\text{M4.} & B \to A: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}
\end{array}$$

- Suppose $|M, A, B| = |K_{AB}|$, e.g., M is 32 bits, A and B are 16 bits, and K_{AB} is 64 bits.



1001101100111100 11011011 00010010

Ciphertext

1001101100111100 11011011 00010010

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

$$\begin{array}{lll}
\text{M1.} & a \to i(b): & m, a, b, \{\!|na, m, a, b|\!\}_{\mathsf{k}(a,s)} \\
\text{M4.} & i(b) \to a: & m, \{\!|na, \underbrace{m, a, b}_{\text{mistaken as } kab} |\!\}_{\mathsf{k}(a,s)}
\end{array}$$

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

$$
\begin{array}{lll}
\text{M1.} & A \to B : & M, A, B, \{\!| NA, M, A, B |\!\}_{\mathsf{k}(A,S)} \\
\text{M2.} & B \to S : & M, A, B, \{\!| NA, M, A, B |\!\}_{\mathsf{k}(A,S)}, \{\!| NB, M, A, B |\!\}_{\mathsf{k}(B,S)} \\
\text{M3.} & S \to B : & M, \{\!| NA, K_{AB} |\!\}_{\mathsf{k}(A,S)}, \{\!| NB, K_{AB} |\!\}_{\mathsf{k}(B,S)} \\
\text{M4.} & B \to A : & M, \{\!| NA, K_{AB} |\!\}_{\mathsf{k}(A,S)}
\end{array}
$$

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

$$
\begin{array}{lll}
\text{M1.} & a \to b : & m, a, b, \{\!| na, m, a, b |\!\}_{\mathsf{k}(a,s)} \\
\text{M2.} & b \to i(s) : & m, a, b, \{\!| na, m, a, b |\!\}_{\mathsf{k}(a,s)}, \{\!| nb, m, a, b |\!\}_{\mathsf{k}(b,s)} \\
\text{M3.} & i(s) \to b : & m, \{\!| na, m, a, b |\!\}_{\mathsf{k}(a,s)}, \{\!| nb, m, a, b |\!\}_{\mathsf{k}(b,s)} \\
\text{M4.} & b \to a : & m, \{\!| na, m, a, b |\!\}_{\mathsf{k}(a,s)}
\end{array}
$$

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

$$\begin{array}{lll}
\text{M1.} & A \to B: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)} \\
\text{M2.} & B \to S: & M, A, B, \{\!|NA, M, A, B|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, M, A, B|\!\}_{\mathsf{k}(B,S)} \\
\text{M3.} & S \to B: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}, \{\!|NB, K_{AB}|\!\}_{\mathsf{k}(B,S)} \\
\text{M4.} & B \to A: & M, \{\!|NA, K_{AB}|\!\}_{\mathsf{k}(A,S)}
\end{array}$$

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

$$\begin{array}{lll}
\text{M1.} & a \to b: & m, a, b, \{\!|na, m, a, b|\!\}_{\mathsf{k}(a,s)} \\
\text{M2.} & b \to i(s): & m, a, b, \{\!|na, m, a, b|\!\}_{\mathsf{k}(a,s)}, \{\!|nb, m, a, b|\!\}_{\mathsf{k}(b,s)} \\
\text{M3.} & i(s) \to b: & m, \{\!|na, m, a, b|\!\}_{\mathsf{k}(a,s)}, \{\!|nb, m, a, b|\!\}_{\mathsf{k}(b,s)} \\
\text{M4.} & b \to a: & m, \{\!|na, m, a, b|\!\}_{\mathsf{k}(a,s)}
\end{array}$$

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

Prudent engineering of security protocols

- Principles proposed by Abadi and Needham (1994, 1995):
  - Every message should say what it means.
  - Specify clear 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., $\{X, Y\}_{\mathsf{sk}(K)}$ versus $\{X\}_{\mathsf{sk}(K)}, \{Y\}_{\mathsf{sk}(K)}$
  - Be clear on what properties you are assuming.
  - Beware of clock variations (for timestamps).
  - ... and more ...

- 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?

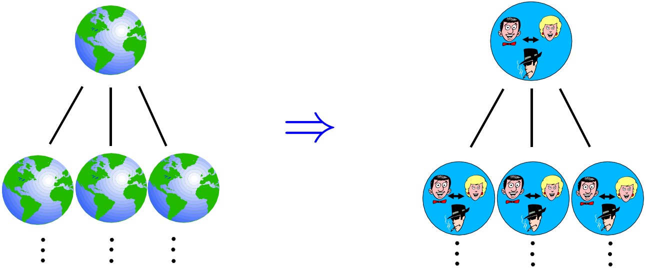
Plan

1. Motivation
2. Building a key establishment protocol
3. Formalizing Security Protocols: An Example
4. Protocol attacks
5. Outlook

Formal modeling and analysis of protocols

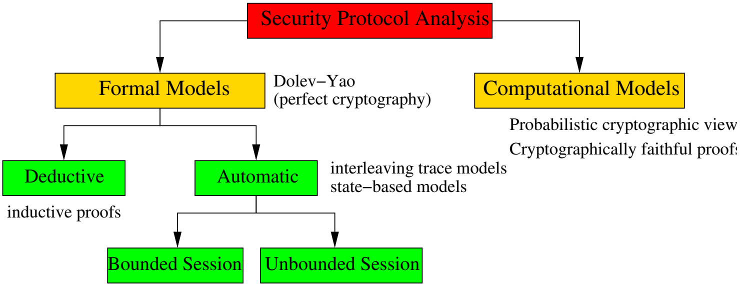
Goal: formally model protocols and their properties and provide a mathematically sound means for reasoning about these models.

Basis: suitable abstraction of protocols:

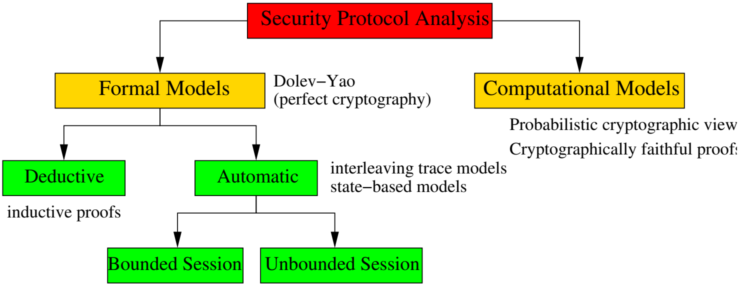


Analysis: with formal methods based on mathematics and logic.

Formal analysis of security protocols



Formal analysis of security protocols



Security Protocol Analysis

Formal Models

Dolev−Yao
(perfect cryptography)

Computational Models

Probabilistic cryptographic view

Cryptographically faithful proofs

Deductive

inductive proofs

Automatic

interleaving trace models
state−based models

Bounded Session

Unbounded Session

Formal modeling of security protocols

Security protocol models

- Preliminaries: **Term rewriting**
- **Syntax**: Alice&Bob ($A \rightarrow B\colon M$) notation, Tamarin: multiset rewrite rules
- **Dolev-Yao adversary**: message derivation

Security properties

- **Semantics**: transition system, event traces
- **Security Goals**: Secrecy, authentication, privacy, . . .

Syntax and semantics of security protocols

Syntax

- Alice&Bob: protocol is a sequence of A&B events:
  $A \rightarrow B : M$.
- Tamarin: multiset rewrite rules (more fine-grained).

Semantics

- Semantics of rewrite rule specification is a transition system.
- Trace: records past events (send, receive, events, ...)
- Send rule: thread passes message to adversary (i.e., network)
- Receive rule: thread obtains an adversary-derivable message

Security properties

- Predicates on the traces for authentication, secrecy, ...
- Equivalence properties: privacy, anonymity, ...

Why is security protocol analysis difficult?

Infinite state space

- Messages: adversary can produce messages of arbitrary size
- Sessions: unbounded number of parallel sessions
- Nonces: unbounded number of nonces (if sessions unbounded)

Undecidability

- Secrecy problem for security protocols is undecidable (Even & Goldreich, 1983)
- even if the number of nonces or the message size is bounded

Approaches that work well in practice

- Symbolic analysis methods: avoid state enumeration
- Sophisticated search strategies to avoid non-termination
- Abstraction techniques: over-approximate reachable states

Summary

- Security protocols can achieve properties that cryptographic primitives alone can't offer, e.g., authentication, freshness, ...
- A protocol without explicit goals and assumptions is useless.
- Even three-liners show how difficult the art of correct design is.
- Protocol without a proof of its properties is probably wrong.
- Formal modeling and analysis of protocols is useful.
  Formal analysis is non-trivial (even assuming perfect cryptography).

Bibliography

- David Basin and Cas Cremers and Catherine Meadows, *Model Checking Security Protocols*, Handbook of Model Checking, 2018.
- Martín Abadi and Roger Needham. *Prudent Engineering Practice for Cryptographic Protocols*. IEEE Transactions on Software Engineering, 22(1):2-15, 1996.
- Ross Anderson and Roger Needham. *Programming Satan's Computer.* In Computer Science Today, vol. 1000 of LNCS, p. 426-440. Springer, 1995.
- John Clark and Jeremy Jacob. *A survey of authentication protocol literature*, 1997.
  `http://www.cs.york.ac.uk/~jac/PublishedPapers/reviewV1_1997.pdf`