Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Tuesday, August 25th, 2015
12 academic departments: mathematics, computer science, chemistry, social sciences, . . .

13 research laboratories

Laboratoire Spécification & Vérification
Research at LSV

Verification of critical software and systems

Goal: develop the mathematical and algorithmic foundations to the development of tools for automatically proving correctness and detecting flaws.

Applications: computerized systems, databases, security protocols

LSV in figures
- Founded in 1997
- Around 25 permanents + 15 PhD students
- 5 research teams
Security of Information Systems


- 1 engineer + 1 postdoc

- 3 phd students
Cryptographic protocols everywhere!

**Goal:** they aim at securing communications over public/insecure networks
Some security properties

- **Secrecy**: May an intruder learn some secret message between two honest participants?

- **Authentication**: Is the agent Alice really talking to Bob?

- **Anonymity**: Is an attacker able to learn something about the identity of the participants who are communicating?

- **Non-repudiation**: Alice sends a message to Bob. Alice cannot later deny having sent this message. Bob cannot deny having received the message.

- ...
How does a cryptographic protocol work (or not)?

**Protocol:** small programs explaining how to exchange messages
Protocol: small programs explaining how to exchange messages
How does a cryptographic protocol work (or not)?

**Protocol:** small programs explaining how to exchange messages

**Cryptographic:** make use of cryptographic primitives

**Examples:** symmetric encryption, asymmetric encryption, signature, hashes, ...
What is a symmetric encryption scheme?

Symmetric encryption

Encryption -> Lock -> Decryption
What is a symmetric encryption scheme?

Symmetric encryption

Example: This might be as simple as shifting each letter by a number of places in the alphabet (e.g. Caesar cipher)

A famous example

**Enigma machine (1918-1945)**
- electro-mechanical rotor cipher machines used by the German to encrypt during World War II
- permutations and substitutions

**A bit of history**
- **1918**: invention of the Enigma machine
- **1940**: Battle of the Atlantic during which Alan Turing’s Bombe was used to test Enigma settings.

→ Everything about the breaking of the Enigma cipher systems remained secret until the mid-1970s.
What is an asymmetric encryption scheme?

Asymmetric encryption

encryption

public key

decryption

private key
What is an asymmetric encryption scheme?

Asymmetric encryption

Examples:

- **1976**: first system published by W. Diffie, and M. Hellman,

→ their security relies on well-known mathematical problems (e.g. factorizing large numbers, computing discrete logarithms)

Today: those systems are still in use
What is a signature scheme?

Example:

The RSA cryptosystem (in fact, most public key cryptosystems) can be used as a signature scheme.
How does a cryptographic protocol work (or not)?

Example: A simplified version of the Denning-Sacco protocol (1981)

\[
\begin{align*}
A &\rightarrow B : \text{aenc}\left(\text{sign}(k, \text{priv}(A)), \text{pub}(B)\right) \\
B &\rightarrow A : \text{senc}(s, k)
\end{align*}
\]

What about secrecy of \(s\)?
How does a cryptographic protocol work (or not)?

Example: A simplified version of the Denning-Sacco protocol (1981)

\[ A \rightarrow B : \text{aenc(\text{sign}(k, \text{priv}(A)), \text{pub}(B))} \]
\[ B \rightarrow A : \text{senc}(s, k) \]

What about secrecy of \( s \)?

Consider a scenario where \( A \) starts a session with \( C \) who is dishonest.

1. \( A \rightarrow C : \text{aenc(\text{sign}(k, \text{priv}(A)), \text{pub}(C))} \)
   \( C \) knows the key \( k \)
How does a cryptographic protocol work (or not)?

Example: A simplified version of the Denning-Sacco protocol (1981)

\[
\begin{align*}
A & \rightarrow B : \text{aenc}\left(\text{sign}(k, \text{priv}(A)), \text{pub}(B)\right) \\
B & \rightarrow A : \text{senc}(s, k)
\end{align*}
\]

What about secrecy of \( s \)?

Consider a scenario where \( A \) starts a session with \( C \) who is dishonest.

1. \( A \rightarrow C : \text{aenc}\left(\text{sign}(k, \text{priv}(A)), \text{pub}(C)\right) \)

   \( C \) knows the key \( k \)

2. \( C(A) \rightarrow B : \text{aenc}\left(\text{sign}(k, \text{priv}(A)), \text{pub}(B)\right) \)

3. \( B \rightarrow A : \text{senc}(s, k) \) \( \text{Attack!} \)
Exercise

We propose to fix the Denning-Sacco protocol as follows:

**Version 1**

\[
\begin{align*}
A \rightarrow B & : \ \text{aenc}(\langle A, B, \text{sign}(k, \text{priv}(A)) \rangle, \text{pub}(B)) \\
B \rightarrow A & : \ \text{senc}(s, k)
\end{align*}
\]

**Version 2**

\[
\begin{align*}
A \rightarrow B & : \ \text{aenc}(\text{sign}(\langle A, B, k \rangle, \text{priv}(A)) \rangle, \text{pub}(B)) \\
B \rightarrow A & : \ \text{senc}(s, k)
\end{align*}
\]

Which version would you prefer to use?
We propose to fix the Denning-Sacco protocol as follows:

**Version 1**

\[
\begin{align*}
A \rightarrow B & : \ \text{aenc}(\langle A, B, \text{sign}(k, \text{priv}(A)) \rangle, \text{pub}(B)) \\
B \rightarrow A & : \ \text{senc}(s, k)
\end{align*}
\]

**Version 2**

\[
\begin{align*}
A \rightarrow B & : \ \text{aenc}(\text{sign}(\langle A, B, k \rangle, \text{priv}(A)) \rangle, \text{pub}(B)) \\
B \rightarrow A & : \ \text{senc}(s, k)
\end{align*}
\]

Which version would you prefer to use? Version 2

→ Version 1 is still vulnerable to the aforementioned attack.
What about protocols used in real life?

- Mobile phone
- Passport
- Credit card
- Digital signature device
- E-Voting website
- HTTPS protocol

S. Delaune (LSV)
Credit Card payment protocol

Serge Humpich case - “Yescard“ (1997)
Credit Card payment protocol

Serge Humpich case - "Yescard" (1997)

Step 1: A **logical flaw** in the protocol allows one to copy a card and to use it without knowing the PIN code.

→ not a real problem, there is still a bank account to withdraw
Credit Card payment protocol

Serge Humpich case - “Yescard” (1997)

Step 1: A logical flaw in the protocol allows one to copy a card and to use it without knowing the PIN code.

→ not a real problem, there is still a bank account to withdraw

Step 2: breaking encryption via factorisation of the following (96 digits) number: 213598703592091008239502270499962879705109534182
6417406442524165008583957746445088405009430865999

→ now, the number that is used is made of 232 digits
Lots of bugs and attacks, with fixes every month

**FREAK attack discovered by Baraghavan et al (Feb. 2015)**

1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

→ 'export-grade' were introduced under the pressure of US governments agencies to ensure that they would be able to decrypt all foreign encrypted communication.
Does the protocol satisfy a security property?
Does the protocol satisfy a security property?

Two main tasks

1. Modelling cryptographic protocols and their security properties
2. Designing verification algorithms
Modelling messages
and
Deciding knowledge
(in a simple setting)
Symbolic model

Various models \( (e.g. \ [Dolev \ \& \ \text{Yao, 81}] \) having some common features
Various models (e.g. [Dolev & Yao, 81]) having some common features

Messages

They are abstracted by terms.
Symbolic model

Various models (e.g. [Dolev & Yao, 81]) having some common features

Messages

They are abstracted by terms.

The attacker
Various models (e.g. [Dolev & Yao, 81]) having some common features.

Messages

They are abstracted by terms.

The attacker

- may read every message sent on the network,
- may intercept and send new messages according to its deduction capabilities.
- only symbolic manipulations on terms.
Messages as terms

→ It is important to have a tight modelling of messages
Messages as terms

It is important to have a tight modelling of messages.

Terms

They are built over a signature $\mathcal{F}$, and an infinite set of names $\mathcal{N}$.

\[
t ::= n \quad \text{name } n \in \mathcal{N} \\
| f(t_1, \ldots, t_k) \quad \text{application of symbol } f \in \mathcal{F}
\]

- Names are used to model atomic data
  \[\rightarrow \text{e.g. keys, nonces, agent names, } \ldots\]
- Function symbols are used to model cryptographic primitives
  \[\rightarrow \text{e.g. encryption, signature, } \ldots\]
A typical signature

Standard primitives

\[ \mathcal{F} = \{ \text{senc}, \text{aenc}, \text{sk}, \text{sign}, \langle \rangle \} \]
A typical signature

Standard primitives

\[ \mathcal{F} = \{ \text{senc}, \text{aenc}, \text{sk}, \text{sign}, \langle \rangle \} \]

Going back to the Denning Sacco protocol

\[
\begin{align*}
    A \rightarrow B & : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \\
    B \rightarrow A & : \text{senc}(s, k)
\end{align*}
\]

These messages can be modelled as follows:

1. \text{aenc}(\text{sign}(k, \text{sk}(a)), b);
2. \text{senc}(s, k)
Capabilities of the attacker

Symbolic manipulation on terms
He may build new messages following deduction rules

Pairing

\[
\begin{array}{ccc}
\langle x, y \rangle & \frac{\langle x, y \rangle}{x} & \frac{\langle x, y \rangle}{y} \\
\langle x, y \rangle & x & y \\
\end{array}
\]

Symmetric encryption

\[
\begin{array}{ccc}
\langle x, y \rangle & \frac{senc(x, y)}{x} & \frac{senc(x, y)}{y} \\
\langle x, y \rangle & x & y \\
\end{array}
\]

Asymmetric encryption

\[
\begin{array}{ccc}
\langle x, y \rangle & \frac{aenc(x, y)}{x} & \frac{aenc(x, y)}{sk(y)} \\
\langle x, y \rangle & x & sk(y) \\
\end{array}
\]

Signature

\[
\begin{array}{ccc}
\langle x, y \rangle & \frac{sign(x, sk(y))}{x} & \frac{sign(x, sk(y))}{x} \\
\langle x, y \rangle & x & sign(x, sk(y)) \\
\end{array}
\]
Deduction relation $T \vdash u$

We say that $u$ is **deducible from** $T$ if there exists a proof tree such that:

1. each leaf is labeled by $v$ with $v \in T$;
2. for each node labeled by $v_0$ and having $n$ sons labeled by $v_1, \ldots, v_n$, there exists a deduction rule $R$ such that
   \[
   \frac{v_1 \ldots v_n}{v_0}
   \]
   is an instance of $R$
3. the root is labeled by $u$. 
We say that \( u \) is **deducible from** \( T \) if there exists a proof tree such that:

1. each leaf is labeled by \( v \) with \( v \in T \);
2. for each node labeled by \( v_0 \) and having \( n \) sons labeled by \( v_1, \ldots, v_n \), there exists a deduction rule \( R \) such that
   \[
   \begin{array}{c}
   v_1 \quad \cdots \quad v_n \\
   \hline \\
   v_0
   \end{array}
   \]
   is an instance of \( R \)
3. the root is labeled by \( u \).

**Exercise - Going back to the Denning Sacco protocol**

Let \( T = \{a, b, c, \text{sk}(c), \text{aenc}(\text{sign}(k, \text{sk}(a)), c), \text{senc}(s, k)\} \).

Is \( s \) deducible from \( T \)?
Exercise - Going back to the Denning Sacco protocol

Let $T = \{a, b, c, sk(c), aenc(sign(k, sk(a)), c), senc(s, k)\}$.
Is $s$ deducible from $T$?
Exercise - Going back to the Denning Sacco protocol

Let \( T = \{ a, b, c, sk(c), aenc(sign(k, sk(a)), c), senc(s, k) \} \).

Is \( s \) deducible from \( T \)?

**Answer:** Of course, Yes!

\[
\begin{align*}
\text{senc}(s, k) & \quad \text{sk}(c) \\
\text{aenc}(\text{sign}(k, \text{sk}(a)), c) & \quad \text{sk}(c) \\
\text{sign}(k, \text{sk}(a)) & \quad k \\
\hline
s & \quad \text{s}
\end{align*}
\]
1. $A \rightarrow C : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(C))$

2. $C(A) \rightarrow B : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(B))$

3. $B \rightarrow A : \text{senc}(s, k)$

Attack!

Exercise (continued)

Let $T_0 = \{a, b, c, \text{sk}(c), \text{aenc}(\text{sign}(k, \text{sk}(a)), c)\}$.

Is $\text{aenc}(\text{sign}(k, \text{sk}(a)), b)$ deducible from $T_0$?
Denning Sacco protocol

1. $A \to C : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(C))$

2. $C(A) \to B : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(B))$

3. $B \to A : \text{senc}(s, k)$ \hspace{1cm} \text{Attack !}

Exercise (continued)

Let $T_0 = \{a, b, c, \text{sk}(c), \text{aenc}(\text{sign}(k, \text{sk}(a)), c)\}$. Is $\text{aenc}(\text{sign}(k, \text{sk}(a)), b)$ deducible from $T_0$?

Answer: Of course, Yes !

\[
\begin{align*}
\text{aenc}(\text{sign}(k, \text{sk}(a)), c) & \quad \text{sk}(c) \\
\text{sign}(k, \text{sk}(a)) & \quad b \\
\hline \\
\text{aenc}(\text{sign}(k, \text{sk}(a)), b) & \\
\end{align*}
\]
Deciding deduction (in this simple setting)

The deduction problem

**Input:** a finite set of terms $T$ (the knowledge of the attacker) and a term $u$ (the secret),

**Output:** Is $u$ deducible from $T$?
Deciding deduction (in this simple setting)

**The deduction problem**

**Input:** a finite set of terms $T$ (the knowledge of the attacker) and a term $u$ (the secret),

**Output:** Is $u$ deducible from $T$?

**Proposition**

The deduction problem is decidable in PTIME.
Deciding deduction (in this simple setting)

The deduction problem

**Input:** a finite set of terms \( T \) (the knowledge of the attacker) and a term \( u \) (the secret),

**Output:** Is \( u \) deducible from \( T \)?

Proposition

The deduction problem is decidable in PTIME.

Algorithm

1. **Saturation** of \( T \) with terms in \( St(T \cup \{u\}) \) that are deducible in one step;
2. if \( u \) is in the saturated set then return Yes else return No.
Soundness  If the algorithm returns \textbf{Yes} then $u$ is indeed deducible from $T$. \hspace{200px} \rightarrow \hspace{10px} \text{easy to prove}
Soundness, completeness, and termination

**Soundness**  If the algorithm returns *Yes* then $u$ is indeed deducible from $T$.  $\rightarrow$ easy to prove

**Termination**  The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.  $\rightarrow$ easy to prove for the deduction rules under study
Soundness If the algorithm returns Yes then $u$ is indeed deducible from $T$.  

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.  

Completeness If the term $u$ is deducible from $T$, then the algorithm returns Yes. Otherwise, it returns No.  

Locality lemma  
Let $T$ and $u$ be such that $T \vdash u$. There exists a prooftree witnessing this fact for which all the nodes are labeled by some $v$ with $v \in St(T \cup \{u\})$.  

S. Delaune (LSV)  
Verification of security protocols  
25th August 2015  
28 / 60
Proof sketch

Locality lemma

Let $T$ and $u$ be such that $T \vdash u$. There exists a tree witnessing this fact for which all the nodes are labeled by some $v$ with $v \in St(T \cup \{u\})$.

Let $P$ be a proof tree witnessing the fact that $T \vdash u$ having a minimal size (number of nodes). We show by induction on $P$ that:

- if $P$ ends with root labeled by $v$ then $P$ only contains terms in $St(T \cup \{v\})$.
Proof sketch

Locality lemma

Let $T$ and $u$ be such that $T \vdash u$. There exists a tree witnessing this fact for which all the nodes are labeled by some $v$ with $v \in St(T \cup \{u\})$.

We first split the deduction rules into two categories:

1. composition rules: encryption, signature, and pairing
2. decomposition rules: decryption, projections, . . .

Let $P$ be a proof tree witnessing the fact that $T \vdash u$ having a minimal size (number of nodes). We show by induction on $P$ that:

- if $P$ ends with root labeled by $v$ then $P$ only contains terms in $St(T \cup \{v\})$;
- if $P$ ends with a decomposition rule then $P$ only contains terms in $St(T)$.

→ this is left as an exercise
Consider the following set of deduction rules:

\[
\begin{array}{c}
  \frac{\text{sign}(x, \text{sk}(y))}{\text{sk}(y)} \\
  \frac{\text{sign}(x, \text{sk}(y))}{\text{vk}(y)} \\
  \frac{\text{sign}(x, \text{sk}(y))}{x} \\
  \frac{\text{sign}(x, \text{sk}(y))}{\text{vk}(y)}
\end{array}
\]

1. Give an example showing that these deduction rules are not local.
2. Extend the notion of subterms to restore the locality property, and show that the deduction problem is decidable.
Exercise

Consider the following set of deduction rules:

\[
\begin{array}{c}
\frac{x \ sk(y)}{\text{sign}(x, sk(y))} \\
\frac{\text{sign}(x, sk(y)) \ vk(y)}{x} \\
\frac{y}{\text{vk}(y)}
\end{array}
\]

1. Give an example showing that these deduction rules are not local.
2. Extend the notion of subterms to restore the locality property, and show that the deduction problem is decidable.

Solution

1. Let \( T = \{ \text{sign}(s, sk(a)); a \} \) and \( u = s \).
2. \( St^+(T) = St(T) \cup \{ \text{vk}(u) \mid sk(u) \in \text{vk}(u) \in St(T) \} \).

\( \rightarrow \) the locality proof is left as an exercise.
Consider the following set of deduction rules:

\[
\frac{x \quad y}{\langle x, y \rangle} \quad \frac{\langle x, y \rangle}{x} \quad \frac{\langle x, y \rangle}{y} \quad \frac{x \quad y}{\text{senc}(x, y)} \quad \frac{\text{senc}(x, y) \quad y}{x}
\]

In order to decide whether a term \( u \) is deducible from a set of terms \( T \), we propose the following algorithm:

1. Starting from \( T \), apply as much as possible the decryption and the projection rules. This leads to a set of terms called Decomposition(\( T \)).
2. Check whether \( u \) can be obtained by applying the composition rules on top of terms in Decomposition(\( T \)).
3. In case of success, the algorithm returns Yes. Otherwise, it returns No.

Questions
What about termination, soundness, and completeness?
Modelling messages
and
Deciding knowledge
(in a richer setting)
More cryptographic primitives

We may want to consider a richer term algebra and rely on an equation theory $E$ to take into account the properties of the primitives

Exclusive or operator:

\[
(x \oplus y) \oplus z = x \oplus (y \oplus z) \\
\quad x \oplus y = y \oplus x \\
\quad x \oplus 0 = x \\
\quad x \oplus x = 0
\]
More cryptographic primitives

We may want to consider a richer term algebra and rely on an **equational theory** $E$ to take into account the properties of the primitives.

**Exclusive or operator:**

\[
(x \oplus y) \oplus z = x \oplus (y \oplus z) \quad x \oplus x = 0 \\
 x \oplus y = y \oplus x \quad x \oplus 0 = x
\]

**Blind signature** (used in evoting protocol)

\[
\text{check}(\text{sign}(x, y), \text{vk}(y)) = x \\
\text{unblind}(\text{blind}(y, y), y) = x \\
\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) = \text{sign}(x, z)
\]
More cryptographic primitives

We may want to consider a richer term algebra and rely on an equational theory $E$ to take into account the properties of the primitives

Exclusive or operator:

\[(x \oplus y) \oplus z = x \oplus (y \oplus z) \quad x \oplus x = 0\]
\[x \oplus y = y \oplus x \quad x \oplus 0 = x\]

Blind signature (used in evoting protocol)

\[\text{check}(\text{sign}(x, y), \text{vk}(y)) = x\]
\[\text{unblind}(\text{blind}(y, y), y) = x\]
\[\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) = \text{sign}(x, z)\]

Homomorphic encryption:

\[\text{sdec}(\text{senc}(x, y), y) = x\]
\[\text{enc}(\langle x, y \rangle, z) = \langle \text{enc}(x, z), \text{enc}(y, z) \rangle\]
\[\text{dec}(\langle x, y \rangle, z) = \langle \text{dec}(x, z), \text{dec}(y, z) \rangle\]
\[\text{proj}_1(\langle x, y \rangle) = x\]
\[\text{proj}_2(\langle x, y \rangle) = y\]
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

What function symbols and equations do we need to model this protocol?
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \ aenc(\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \ senc(s, k) \]

What function symbols and equations do we need to model this protocol?

1. symmetric encryption: \( \text{senc}(\cdot, \cdot), \text{sdec}(\cdot, \cdot) \)
   \[ \rightarrow \text{sdec}(\text{senc}(x, y), y) = x \]
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \text{aenc} (\text{sign} (k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc} (s, k) \]

What function symbols and equations do we need to model this protocol?

1. **Symmetric encryption**: \( \text{senc} (\cdot, \cdot), \text{sdec} (\cdot, \cdot) \)
   \[ \rightarrow \text{sdec} (\text{senc} (x, y), y) = x \]

2. **Asymmetric encryption**: \( \text{aenc} (\cdot, \cdot), \text{adec} (\cdot, \cdot), \text{pk}(\cdot) \)
   \[ \rightarrow \text{adec} (\text{aenc} (x, \text{pk}(y)), y) = x \]
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \ aenc(\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \ senc(s, k) \]

What function symbols and equations do we need to model this protocol?

1. **Symmetric encryption**: \( senc(\cdot, \cdot), sdec(\cdot, \cdot) \)
   
   \[ \rightarrow sdec(senc(x, y), y) = x \]

2. **Asymmetric encryption**: \( aenc(\cdot, \cdot), \text{adec}(\cdot, \cdot), \text{pk}(\cdot) \)
   
   \[ \rightarrow \text{adec}(aenc(x, \text{pk}(y)), y) = x \]

3. **Signature**: \( \text{sign}(\cdot, \cdot), \text{check}(\cdot, \cdot) \)
   
   \[ \rightarrow \text{check}(\text{sign}(x, y), \text{pk}(y)) = x \]
Deduction in this more general setting

Deduction rules are as follows:

\[
\frac{u_1 \cdots u_k}{f(u_1, \ldots, u_k)} \quad f \in \mathcal{F} \\
\frac{u}{u'} \quad u \equiv_E u'
\]
Deduction in this more general setting

Deduction rules are as follows:

\[
\begin{align*}
\frac{u_1 \cdots u_k}{f(u_1, \ldots, u_k)} \quad & f \in \mathcal{F} \\
\frac{u}{u'} \quad & u \equiv_E u'
\end{align*}
\]

Example: Let \( E := \text{sdec}(\text{senc}(x, y), y) = x \) and \( T = \{ \text{senc}(\text{secret}, k), k \} \). We have that \( T \vdash \text{secret} \).

\[
\begin{align*}
\frac{\text{senc}(\text{secret}, k) \quad k}{\text{sdec}(\text{senc}(\text{secret}, k), k)} \quad & \text{sdec} \in \mathcal{F} \\
\frac{\text{sdec}(\text{senc}(\text{secret}, k), k) \quad \text{secret}}{\text{sdec}(\text{senc}(x, y), y) = x}
\end{align*}
\]
The deduction problem: is $u$ deducible from $\phi$?

We consider a signature $\mathcal{F}$ and an equational theory $E$.

| Input  | A sequence $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$ of terms and a term $u$ |
| Output | Is $u$ deducible from $\phi$? |
The deduction problem: is $u$ deducible from $\phi$?

We consider a signature $\mathcal{F}$ and an equational theory $E$.

The deduction problem

**Input** A sequence $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$ of terms and a term $u$

**Output** Is $u$ deducible from $\phi$?

**Characterization of deduction**

$T \vdash u$ if, and only if, there exists a term $R$ such that $R\phi \equiv_E u$.

$\longrightarrow$ such a term $R$ is a **recipe** of the term $u$. 
The deduction problem: is $u$ deducible from $\phi$?

We consider a signature $\mathcal{F}$ and an equational theory $E$.

The deduction problem

Input  A sequence $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$ of terms and a term $u$

Output Is $u$ deducible from $\phi$?

Characterization of deduction

$T \vdash u$ if, and only if, there exists a term $R$ such that $R\phi \equiv_E u$.

$\rightarrow$ such a term $R$ is a recipe of the term $u$.

Example: Let $\phi = \{w_1 \triangleright pk(ska); w_2 \triangleright pk(skb); w_3 \triangleright skc; w_4 \triangleright aenc(sign(k, ska), pk(skc)); w_5 \triangleright senc(s, k)\}$.

We have that:
The deduction problem: is \( u \) deducible from \( \phi \)?

We consider a signature \( \mathcal{F} \) and an equational theory \( E \).

**The deduction problem**

**Input** A sequence \( \phi = \{ w_1 \triangleright v_1, \ldots, w_n \triangleright v_n \} \) of terms and a term \( u \)

**Output** Is \( u \) deducible from \( \phi \)?

**Characterization of deduction**

\( T \vdash u \) if, and only if, there exists a term \( R \) such that \( R\phi =_E u \).

\( \rightarrow \) such a term \( R \) is a recipe of the term \( u \).

**Example:** Let \( \phi = \{ w_1 \triangleright pk(ska); \ w_2 \triangleright pk(skb); \ w_3 \triangleright skc; \ w_4 \triangleright aenc(\text{sign}(k, ska), pk(skc)); \ w_5 \triangleright senc(s, k) \} \).

We have that:

- \( k \) is deducible from \( \phi \) using \( R_1 = \text{check}(\text{adec}(w_4, w_3), w_1) \),
The deduction problem: is $u$ deducible from $\phi$?

We consider a signature $\mathcal{F}$ and an equational theory $E$.

**The deduction problem**

**Input** A sequence $\phi = \{w_1 \triangleright v_1, \ldots, w_n \triangleright v_n\}$ of terms and a term $u$

**Output** Is $u$ deducible from $\phi$?

**Characterization of deduction**

$T \vdash u$ if, and only if, there exists a term $R$ such that $R\phi \equiv_E u$.

$\longrightarrow$ such a term $R$ is a **recipe** of the term $u$.

**Example:** Let $\phi = \{w_1 \triangleright pk(ska); w_2 \triangleright pk(skb); w_3 \triangleright skc; w_4 \triangleright aenc(sign(k, ska), pk(skb)); w_5 \triangleright senc(s, k)\}$.

We have that:

- $k$ is deducible from $\phi$ using $R_1 = \text{check}(\text{adec}(w_4, w_3), w_1)$,
- $s$ is deducible from $\phi$ using $R_2 = \text{sdec}(w_5, R_1)$. 

Proposition

The deduction problem is decidable for the equational theory modelling the DS protocol (and actually any subterm convergent equational theory).

Algorithm:

1. saturation of $\phi$ with its deducible subterm; we get $\phi^+$
2. does there exist a recipe $R$ such that $R\phi^+ = s$ (syntaxic equality)
Deduction problem in this richer setting

**Proposition**

The **deduction problem** is decidable for the equational theory modelling the DS protocol (and actually any subterm convergent equational theory).

**Algorithm:**

1. **saturation** of $\phi$ with its deducible subterm; we get $\phi^+$
2. does there exist a recipe $R$ such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

- $\phi = \{ w_1 \triangleright pk(ska); \ w_2 \triangleright pk(skb); \ w_3 \triangleright skc; \ w_4 \triangleright aenc(sign(k, ska), pk(skc)); \ w_5 \triangleright senc(s, k) \}$. 
Proposition

The deduction problem is decidable for the equational theory modelling the DS protocol (and actually any subterm convergent equational theory).

Algorithm:

1. saturation of $\phi$ with its deducible subterm; we get $\phi^+$
2. does there exist a recipe $R$ such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

$\phi = \{ w_1 \triangleright pk(ska); w_2 \triangleright pk(skb); w_3 \triangleright skc; w_4 \triangleright aenc(sign(k, ska), pk(skc)); w_5 \triangleright senc(s, k) \}$. 
Deduction problem in this richer setting

**Proposition**

The **deduction problem** is decidable for the equational theory modelling the DS protocol (and actually any subterm convergent equational theory).

**Algorithm:**

1. **saturation** of $\phi$ with its deducible **subterm**; we get $\phi^+$
2. does there exist a recipe $R$ such that $R\phi^+ = s$ (syntaxic equality)

Going back to the previous example:

- $\phi = \{ w_1 \triangleright pk(ska); w_2 \triangleright pk(skb); w_3 \triangleright skc; w_4 \triangleright aenc(sign(k, ska), pk(skb)); w_5 \triangleright senc(s, k) \}.$
- $\phi^+ = \phi \cup \{ w_6 \triangleright sign(k, ska); w_7 \triangleright pk(skc); w_8 \triangleright k; w_9 \triangleright s \}.$
Some other equational theories

Blind signature

\[
\begin{align*}
\text{check}(\text{sign}(x, y), \text{vk}(y)) &= x \\
\text{unblind}(\text{blind}(y, y), y) &= x \\
\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)
\end{align*}
\]

Decidability can be shown in a similar fashion extending the notion of subterm.

\[\text{sign}(m, k) \text{ will be considered as a subterm of } \text{sign}(\text{blind}(m, r), k)\]
Some other equational theories

Blind signature

\[
\begin{align*}
\text{check}(\text{sign}(x, y), \text{vk}(y)) &= x \\
\text{unblind}(\text{blind}(y, y), y) &= x \\
\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)
\end{align*}
\]

Decidability can be shown in a similar fashion extending the notion of subterm.

\[\text{sign}(m, k) \text{ will be considered as a subterm of } \text{sign}(\text{blind}(m, r), k)\]

Exclusive or

\[
\begin{align*}
(x \oplus y) \oplus z &= x \oplus (y \oplus z) \\
x \oplus x &= 0 \\
x \oplus y &= y \oplus x \\
x \oplus 0 &= x
\end{align*}
\]

The deduction problem can be reduced to the problem of solving systems of linear equations over \(\mathbb{Z}/2\mathbb{Z}\).
Deduction is not always sufficient

The intruder knows the values yes and no!

The real question

Is the intruder able to tell whether Alice sends yes or no?
### The static equivalence problem

**Input** Two frames $\phi$ and $\psi$

\[
\phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \quad \psi = \{ w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell \}
\]

**Output** Can the attacker distinguish the two frames, *i.e.* does there exist a test $R_1 \equiv R_2$ such that:

\[
R_1\phi =_E R_2\phi \text{ but } R_1\psi \neq_E R_2\psi \text{ (or the converse).}
\]
Static equivalence

The static equivalence problem

**Input**  Two frames $\phi$ and $\psi$

\[
\phi = \{ w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell \} \quad \psi = \{ w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell \}
\]

**Output**  Can the attacker distinguish the two frames, i.e. does there exist a test $R_1 \not\equiv R_2$ such that:

\[
R_1 \phi =_E R_2 \phi \text{ but } R_1 \psi \neq_E R_2 \psi \text{ (or the converse).}
\]

**Example:** Consider the frames:

- $\phi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc(\text{yes}, \text{pk}(\text{sks}))} \}$; and
- $\psi = \{ w_1 \triangleright \text{pk}(\text{sks}); \ w_2 \triangleright \text{aenc(\text{no}, \text{pk}(\text{sks}))} \}$.

They are *not* in static equivalence: $\text{aenc(\text{yes}, w_1)} \not\equiv w_2$. 
Exercise

Consider the equational theories:

- $E_{\text{senc}}$ defined by $s\text{dec}(s\text{enc}(x, y), y) = x$, and
- $E_{\text{cipher}}$ which extends $E_{\text{senc}}$ by the equation $s\text{enc}(s\text{dec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent? Whenever applicable give the distinguishing test.

\[
\begin{align*}
\{ w_1 \triangleright \text{yes} \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{no} \} \\
\{ w_1 \triangleright \text{senc(\text{yes}, k)} \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc(\text{no}, k)} \} \\
\{ w_1 \triangleright \text{senc(n, k)}, w_2 \triangleright k \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc(n, k)}, w_2 \triangleright k' \} \\
\{ w_1 \triangleright \text{senc(n, k)}, w_2 \triangleright k \} & \sim_{E_{\text{cipher}}} \{ w_1 \triangleright \text{senc(n, k)}, w_2 \triangleright k' \}
\end{align*}
\]
Exercise

Consider the equational theories:

- $E_{senc}$ defined by $sdec(senc(x, y), y) = x$, and
- $E_{cipher}$ which extends $E_{senc}$ by the equation $senc(sdec(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent? Whenever applicable give the distinguishing test.

\[
\begin{align*}
\{ w_1 \triangleright yes \} & \not\sim_{E_{senc}} \{ w_1 \triangleright no \} \quad \checkmark \\
\{ w_1 \triangleright senc(yes, k) \} & \not\sim_{E_{senc}} \{ w_1 \triangleright senc(no, k) \} \\
\{ w_1 \triangleright senc(n, k), w_2 \triangleright k \} & \not\sim_{E_{senc}} \{ w_1 \triangleright senc(n, k), w_2 \triangleright k' \} \\
\{ w_1 \triangleright senc(n, k), w_2 \triangleright k \} & \not\sim_{E_{cipher}} \{ w_1 \triangleright senc(n, k), w_2 \triangleright k' \}
\end{align*}
\]
Consider the equational theories:

- $E_{\text{senc}}$ defined by $\text{sdec}(\text{senc}(x, y), y) = x$, and
- $E_{\text{cipher}}$ which extends $E_{\text{senc}}$ by the equation $\text{senc}(\text{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent? Whenever applicable give the distinguishing test.

- $\{ w_1 \triangleright \text{yes} \} \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{no} \}$
- $\{ w_1 \triangleright \text{senc}(\text{yes}, k) \} \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc}(\text{no}, k) \}$
- $\{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k \} \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k' \}$
- $\{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k \} \sim_{E_{\text{cipher}}} \{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k' \}$
Consider the equational theories:

- $E_{\text{senc}}$ defined by $\text{sdec}(\text{senc}(x, y), y) = x$, and
- $E_{\text{cipher}}$ which extends $E_{\text{senc}}$ by the equation $\text{senc}(\text{sdec}(x, y), y) = x$.

Questions

Which of the following pairs of frames are statically equivalent? Whenever applicable give the distinguishing test.

\[
\begin{align*}
\{ w_1 \triangleright \text{yes} \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{no} \} & \times \\
\{ w_1 \triangleright \text{senc}(\text{yes}, k) \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc}(\text{no}, k) \} & \checkmark \\
\{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k \} & \sim_{E_{\text{senc}}} \{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k' \} & \times \\
\{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k \} & \sim_{E_{\text{cipher}}} \{ w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k' \} & \\
\end{align*}
\]
Consider the equational theories:

- \( E_{\text{senc}} \) defined by \( \text{sdec}(\text{senc}(x, y), y) = x \), and
- \( E_{\text{cipher}} \) which extends \( E_{\text{senc}} \) by the equation \( \text{senc}(\text{sdec}(x, y), y) = x \).

**Questions**

Which of the following pairs of frames are statically equivalent? Whenever applicable give the distinguishing test.

<table>
<thead>
<tr>
<th>Frame 1</th>
<th>Frame 2</th>
<th>Static Equivalent</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ ( w_1 \triangleright \text{yes} ) }</td>
<td>( \sim_{E_{\text{senc}}} ) { ( w_1 \triangleright \text{no} ) }</td>
<td>X</td>
</tr>
<tr>
<td>{ ( w_1 \triangleright \text{senc}(\text{yes}, k) ) }</td>
<td>( \sim_{E_{\text{senc}}} ) { ( w_1 \triangleright \text{senc}(\text{no}, k) ) }</td>
<td>✓</td>
</tr>
<tr>
<td>{ ( w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k ) }</td>
<td>( \sim_{E_{\text{senc}}} ) { ( w_1 \triangleright \text{senc}(n, k), \ w_2 \triangleright k' ) }</td>
<td>X</td>
</tr>
<tr>
<td>{ ( w_1 \triangleright \text{senc}(n, k), w_2 \triangleright k ) }</td>
<td>( \sim_{E_{\text{cipher}}} ) { ( w_1 \triangleright \text{senc}(n, k), \ w_2 \triangleright k' ) }</td>
<td>✓</td>
</tr>
</tbody>
</table>
Static equivalence

Proposition

The static equivalence problem is decidable in PTIME for the theory modelling the DS protocol (and actually any subterm convergent equational theory).
Static equivalence

Proposition

The static equivalence problem is decidable in PTIME for the theory modelling the DS protocol (and actually any subterm convergent equational theory).

Algorithm:

1. **saturation** of $\phi/\psi$ with their deducible subterms $\phi^+/\psi^+$

2. does there exist a test $R_1 \equiv R_2$ such that $R_1 \phi^+ = R_2 \phi^+$ whereas $R_1 \psi^+ \neq R_2 \psi^+$ (again syntaxic equality) ?

   → actually, we only need to consider **small tests**
Consider the frames:

- \( \phi = \{ w_1 \triangleright aenc(\langle yes, r_1 \rangle, pk(sks)); w_2 \triangleright sks \} \); and
- \( \psi = \{ w_1 \triangleright aenc(\langle no, r_2 \rangle, pk(sks)); w_2 \triangleright sks \} \).

They are **not** in static equivalence: \( \text{proj}_1(\text{adec}(w_1, w_2)) \not\equiv yes \).
Example

Consider the frames:

- \( \phi = \{ w_1 \triangleright aenc(\langle yes, r_1 \rangle, pk(sks)); w_2 \triangleright sks \}; \) and
- \( \psi = \{ w_1 \triangleright aenc(\langle no, r_2 \rangle, pk(sks)); w_2 \triangleright sks \}. \)

They are not in static equivalence: \( \text{proj}_1(\text{adec}(w_1, w_2)) = \text{yes}. \)

Applying the algorithm on these frames, we get:

- \( \phi^+ = \phi \uplus \{ \}
- \( \psi^+ = \psi \uplus \{ \}

, and

.
Example

Consider the frames:

- $\phi = \{ w_1 \triangleright aenc(\langle yes, r_1 \rangle, pk(sks)); w_2 \triangleright sks \}$; and
- $\psi = \{ w_1 \triangleright aenc(\langle no, r_2 \rangle, pk(sks)); w_2 \triangleright sks \}$.

They are not in static equivalence: $\text{proj}_1(\text{adec}(w_1, w_2)) \neq yes$.

Applying the algorithm on these frames, we get:

- $\phi^+ = \phi \uplus \{ w_3 \triangleright \langle yes, r_1 \rangle \}$, and
- $\psi^+ = \psi \uplus \{ w_3 \triangleright \langle no, r_2 \rangle \}$. 
Consider the frames:

- \( \phi = \{ w_1 \triangleright \text{aenc}(\langle \text{yes}, r_1 \rangle, \text{pk}(\text{sks})); \ w_2 \triangleright \text{sks} \} \); and
- \( \psi = \{ w_1 \triangleright \text{aenc}(\langle \text{no}, r_2 \rangle, \text{pk}(\text{sks})); \ w_2 \triangleright \text{sks} \} \).

They are not in static equivalence: \( \text{proj}_1(\text{adec}(w_1, w_2)) \neq \text{yes} \).

Applying the algorithm on these frames, we get:

- \( \phi^+ = \phi \uplus \{ w_3 \triangleright \langle \text{yes}, r_1 \rangle; \ w_4 \triangleright \text{yes} \} \), and
- \( \psi^+ = \psi \uplus \{ w_3 \triangleright \langle \text{no}, r_2 \rangle; \ w_4 \triangleright \text{no} \} \).
Example

Consider the frames:

- $\phi = \{ w_1 \triangleright aenc(\langle yes, r_1 \rangle, pk(sks)); w_2 \triangleright sks \};$ and
- $\psi = \{ w_1 \triangleright aenc(\langle no, r_2 \rangle, pk(sks)); w_2 \triangleright sks \}.$

They are not in static equivalence: $\text{proj}_1(\text{adec}(w_1, w_2)) \not= yes.$

Applying the algorithm on these frames, we get:

- $\phi^+ = \phi \cup \{ w_3 \triangleright \langle yes, r_1 \rangle; w_4 \triangleright yes; w_5 \triangleright r_1 \},$ and
- $\psi^+ = \psi \cup \{ w_3 \triangleright \langle no, r_2 \rangle; w_4 \triangleright no; w_5 \triangleright r_2 \}.$
Example

Consider the frames:

- $\phi = \{w_1 \triangleright \text{aenc}(\langle \text{yes}, r_1 \rangle, \text{pk}(sks)); w_2 \triangleright sks\};$ and
- $\psi = \{w_1 \triangleright \text{aenc}(\langle \text{no}, r_2 \rangle, \text{pk}(sks)); w_2 \triangleright sks\}.$

They are not in static equivalence: $\text{proj}_1(\text{adec}(w_1, w_2)) \not\equiv \text{yes}.$

Applying the algorithm on these frames, we get:

- $\phi^+ = \phi \uplus \{w_3 \triangleright \langle \text{yes}, r_1 \rangle; w_4 \triangleright \text{yes}; w_5 \triangleright r_1\},$ and
- $\psi^+ = \psi \uplus \{w_3 \triangleright \langle \text{no}, r_2 \rangle; w_4 \triangleright \text{no}; w_5 \triangleright r_2\}.$

$\longrightarrow$ Conclusion: $\phi^+$ and $\psi^+$ are not in static equivalence: $w_4 \not\equiv \text{yes}.$
Some other equational theories

Blind signature

\[
\begin{align*}
\text{check}(\text{sign}(x, y), \text{vk}(y)) &= x \\
\text{unblind}(\text{blind}(x, y), y) &= x \\
\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)
\end{align*}
\]

This can be done in a similar fashion extending a bit the notion of subterm \( \rightarrow \) again \( \text{sign}(m, k) \) will be considered as a subterm of \( \text{sign}(\text{blind}(m, r), k) \).
Some other equational theories

Blind signature

\[
\begin{align*}
\text{check}(\text{sign}(x, y), \text{vk}(y)) &= x \\
\text{unblind}(\text{blind}(x, y), y) &= x \\
\text{unblindsign}(\text{sign}(\text{blind}(x, y), z), y) &= \text{sign}(x, z)
\end{align*}
\]

This can be done in a similar fashion extending a bit the notion of subterm again $\text{sign}(m, k)$ will be considered as a subterm of $\text{sign}(\text{blind}(m, r), k)$.

Exclusive or

\[
\begin{align*}
(x \oplus y) \oplus z &= x \oplus (y \oplus z) \\
x \oplus y &= y \oplus x \\
x \oplus x &= 0 \\
x \oplus 0 &= x
\end{align*}
\]

The static equivalence problem can be reduced in PTIME to the problem of deciding whether two systems of linear equations have the same set of solutions overs $\mathbb{Z}/2\mathbb{Z}$. 
## Existing decidability/complexity results and tools

<table>
<thead>
<tr>
<th>Theory E</th>
<th>Deduction</th>
<th>Static Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>subterm convergent</td>
<td>PTIME</td>
<td>decidable</td>
</tr>
<tr>
<td>blind sign., addition,</td>
<td></td>
<td></td>
</tr>
<tr>
<td>homo. encryption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ACU</td>
<td>NP-complete</td>
<td>PTIME</td>
</tr>
<tr>
<td>Exclusive Or</td>
<td>PTIME</td>
<td>PTIME</td>
</tr>
<tr>
<td>Abelian Group</td>
<td>PTIME</td>
<td></td>
</tr>
<tr>
<td>ACUNh/AGh</td>
<td>PTIME</td>
<td>decidable</td>
</tr>
</tbody>
</table>

- A combination result for disjoint theories [Cortier & D., JAR’12]

- Automatic tools for checking static equivalence: YAPA M. Baudet (2006); KISS S. Ciobaca (2010); and FAST B. Conchinha (2011)
Modelling protocols
and
security properties
Protocols as processes

**Applied pi calculus** [Abadi & Fournet, 01]

basic programming language with constructs for *concurrency* and *communication*

→ based on the $\pi$-calculus [Milner *et al.*, 92], and in some ways similar to the *spi-calculus* [Abadi & Gordon, 98]
Protocols as processes

Applied pi calculus [Abadi & Fournet, 01]

basic programming language with constructs for concurrency and communication

based on the $\pi$-calculus [Milner et al., 92], and in some ways similar to the spi-calculus [Abadi & Gordon, 98]

Some advantages:

- allows us to model cryptographic primitives
- both reachability and equivalence-based specification of properties
Syntax: \[ P, Q \] := 0 \quad \text{null process}
\begin{align*}
\text{in}(c, x).P & \quad \text{input} \\
\text{out}(c, u).P & \quad \text{output} \\
\text{if } u = v \text{ then } P \text{ else } Q & \quad \text{conditional} \\
P \mid Q & \quad \text{parallel composition} \\
!P & \quad \text{replication} \\
\text{new } n.P & \quad \text{fresh name generation}
\end{align*}
Protocols as processes - syntax and semantics

**Syntax**:  
\[ P, Q ::= \quad 0 \quad \text{null process} \]
\[ \text{in}(c, x).P \quad \text{input} \]
\[ \text{out}(c, u).P \quad \text{output} \]
\[ \text{if } u = v \text{ then } P \text{ else } Q \quad \text{conditional} \]
\[ P | Q \quad \text{parallel composition} \]
\[ !P \quad \text{replication} \]
\[ \text{new } n.P \quad \text{fresh name generation} \]

**Semantics** \( \rightarrow \):  
\[ \text{Comm} \quad \text{out}(c, M).P \mid \text{in}(c, x).Q \rightarrow P \mid Q\{M/x\} \]
\[ \text{Then} \quad \text{if } M = N \text{ then } P \text{ else } Q \rightarrow P \text{ when } M \equiv_{E} N \]
\[ \text{Else} \quad \text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \text{ when } M \not\equiv_{E} N \]

closed by structural equivalence (\( \equiv \)) and application of evaluation contexts.
Going back to Denning Sacco protocol

\[ A \rightarrow B : \text{aenc(sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]
Going back to Denning Sacco protocol

\[ A \rightarrow B : \text{aenc(sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

Alice and Bob as processes:

\[ P_A(\text{sk}_a, \text{pk}_b) = \text{new } k. \text{out}(c, \text{aenc(sign}(k, \text{sk}_a), \text{pk}_b)). \]
\[ \text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in...} \]
Going back to Denning Sacco protocol

\[ A \rightarrow B : \text{aenc(sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

Alice and Bob as processes:

\[
P_A(s_{k_a}, p_{k_b}) = \text{new } k. \text{out}(c, \text{aenc(sign}(k, s_{k_a}), p_{k_b})).
\]
\[
\quad \text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in...}
\]
\[
P_B(s_{k_b}, p_{k_a}) = \text{in}(c, x_b). \text{let } y_b = \text{check(adec}(x_b, s_{k_b}), p_{k_a}) \text{ in}
\]
\[
\quad \text{new } s. \text{out}(c, \text{senc}(s, y_b))
\]
Going back to Denning Sacco protocol

\[
A \rightarrow B : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(B))
\]
\[
B \rightarrow A : \text{senc}(s, k)
\]

Alice and Bob as processes:

\[
P_A(\text{sk}_a, \text{pk}_b) = \text{new } k. \text{out}(c, \text{aenc}(\text{sign}(k, \text{sk}_a), \text{pk}_b)).
\]
\[
\text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in} ...
\]
\[
P_B(\text{sk}_b, \text{pk}_a) = \text{in}(c, x_b). \text{let } y_b = \text{check}(\text{adec}(x_b, \text{sk}_b), \text{pk}_a) \text{ in}
\]
\[
\text{new } s. \text{out}(c, \text{senc}(s, y_b))
\]

One possible scenario:

\[
P_{DS} = \text{new } \text{sk}_a, \text{sk}_b. (P_A(\text{sk}_a, \text{pk}(\text{sk}_b)) \mid P_B(\text{sk}_b, \text{pk}(\text{sk}_a)))
\]
Going back to Denning Sacco protocol

\[ A \to B : \text{aenc(sign(k, priv(A)), pub(B))} \]
\[ B \to A : \text{senc(s, k)} \]

Alice and Bob as processes:

\[
P_A(sk_a, pk_b) = \text{new } k. \text{out}(c, \text{aenc(sign(k, sk_a), pk_b))).
\]
\[
in(c, x_a). \text{let } y_a = \text{sdec(x_a, k)} \text{ in...}
\]

\[
P_B(sk_b, pk_a) = \text{in}(c, x_b). \text{let } y_b = \text{check(adec(x_b, sk_b), pk_a)} \text{ in}
\]
\[
\text{new } s. \text{out}(c, \text{senc(s, y_b)})
\]

One possible scenario:

\[
P_{DS} = \text{new } sk_a, sk_b.(P_A(sk_a, pk(sk_b)) \mid P_B(sk_b, pk(sk_a)))
\]
\[
\rightarrow \text{new } sk_a, sk_b, k. (\text{in}(c, x_a). \text{let } y_a = \text{sdec(x_a, k}) \text{ in ...}
\]
\[
\mid \text{let } y_b = k \text{ in new } s. \text{out}(c, \text{senc(s, y_b)})
\]
Going back to Denning Sacco protocol

\[ A \rightarrow B : \text{aenc}(\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

Alice and Bob as processes:

\[ P_A(\text{sk}_a, \text{pk}_b) = \text{new } k. \text{out}(c, \text{aenc}(\text{sign}(k, \text{sk}_a), \text{pk}_b)) \]
\[ \text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in...} \]
\[ P_B(\text{sk}_b, \text{pk}_a) = \text{in}(c, x_b). \text{let } y_b = \text{check}(\text{adec}(x_b, \text{sk}_b), \text{pk}_a) \text{ in } \]
\[ \text{new } s. \text{out}(c, \text{senc}(s, y_b)) \]

One possible scenario:

\[ P_{DS} = \text{new } \text{sk}_a, \text{sk}_b. (P_A(\text{sk}_a, \text{pk}(\text{sk}_b)) | P_B(\text{sk}_b, \text{pk}(\text{sk}_a))) \]
\[ \rightarrow \text{new } \text{sk}_a, \text{sk}_b, k. (\text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in } \cdots ) \]
\[ | \text{let } y_b = k \text{ in } \text{new } s. \text{out}(c, \text{senc}(s, y_b)) \]
\[ \rightarrow \text{new } \text{sk}_a, \text{sk}_b, k, s. (\text{let } y_a = \text{sdec}(\text{senc}(s, k), k) \text{ in } \cdots | 0) \]
Going back to Denning Sacco protocol

\[ A \rightarrow B : \text{aenc(sign}(k, \text{priv}(A)), \text{pub}(B)) \]

\[ B \rightarrow A : \text{senc}(s, k) \]

Alice and Bob as processes:

\[ P_A(\text{sk}_a, \text{pk}_b) = \text{new } k. \text{out}(c, \text{aenc(sign}(k, \text{sk}_a), \text{pk}_b)). \]
\[ \text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in...} \]

\[ P_B(\text{sk}_b, \text{pk}_a) = \text{in}(c, x_b). \text{let } y_b = \text{check(adec}(x_b, \text{sk}_b), \text{pk}_a) \text{ in} \]
\[ \text{new } s. \text{out}(c, \text{senc}(s, y_b)) \]

One possible scenario:

\[ P_{\text{DS}} = \text{new } \text{sk}_a, \text{sk}_b. (P_A(\text{sk}_a, \text{pk}(\text{sk}_b)) \mid P_B(\text{sk}_b, \text{pk}(\text{sk}_a))) \]
\[ \rightarrow \text{new } \text{sk}_a, \text{sk}_b, k. (\text{in}(c, x_a). \text{let } y_a = \text{sdec}(x_a, k) \text{ in } ... \]
\[ \mid \text{let } y_b = k \text{ in } \text{new } s. \text{out}(c, \text{senc}(s, y_b)) \]
\[ \rightarrow \text{new } \text{sk}_a, \text{sk}_b, k, s. (\text{let } y_a = \text{sdec(senc}(s, k), k) \text{ in } ... \mid 0) \]

\[ \rightarrow \text{this simply models a normal execution between two honest participants} \]
Confidentiality for process $P$ w.r.t. secret $s$

For all processes $A$ such that $A \mid P \rightarrow^* Q$, we have that $Q$ is not of the form $C[\text{out}(c, s).Q']$ with $c$ public.
Confidentiality for process $P$ w.r.t. secret $s$

For all processes $A$ such that $A \mid P \rightarrow^* Q$, we have that $Q$ is not of the form $C[\text{out}(c, s).Q']$ with $c$ public.

Some difficulties:

- we have to consider all the possible executions in presence of an arbitrary adversary (modelled as a process)
- we have to consider realistic initial configurations
  - replications to model an unbounded number of sessions,
  - reveal public keys and private keys to model dishonest agents,
  - $P_A/P_B$ may play with other (and perhaps) dishonest agents, ...
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

The aforementioned attack

1. \[ A \rightarrow C : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(C)) \]
2. \[ C(A) \rightarrow B : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
3. \[ B \rightarrow A : \text{senc}(s, k) \]

The “minimal” initial configuration to retrieve the attack is:

\[
\text{new } sk_a.\text{new } sk_b. (\text{out}(c, \text{pk}(sk_b)) \mid P_A(sk_a, \text{pk}(sk_c)) \mid P_B(sk_b, \text{pk}(sk_a)))
\]
Going back to the Denning Sacco protocol

\[ A \rightarrow B : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \]
\[ B \rightarrow A : \text{senc}(s, k) \]

The aforementioned attack

1. \( A \rightarrow C : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(C)) \)
2. \( C(A) \rightarrow B : \text{aenc} (\text{sign}(k, \text{priv}(A)), \text{pub}(B)) \)
3. \( B \rightarrow A : \text{senc}(s, k) \)

The “minimal” initial configuration to retrieve the attack is:

\[ \text{new } sk_a.\text{new } sk_b. (\text{out}(c, \text{pk}(sk_b)) | P_A(sk_a, \text{pk}(sk_c)) | P_B(sk_b, \text{pk}(sk_a))) \]

Exercise: Exhibit the process \( A \) (the behaviour of the attacker) that witnesses the aforementioned attack.
This can be expressed as a correspondence property:

if B finishes a session, thinking he has talked to A then A has also finished a session, thinking she has talked to B (+ possibly agreement on some values).

Enriched syntax for processes:

\[ P, Q := \begin{align*}
& 0 \quad \text{null process} \\
& \text{in}(c, x).P \quad \text{input} \\
& \ldots \\
& \text{event } p(u_1, \ldots, u_n).P \quad \text{event}
\end{align*} \]

Authentication properties with agreement on some values:

\[ \forall x. \text{EndB}(a, b, x) \Rightarrow \text{EndA}(a, b, x) \]
State of the art in a nutshell

confidentiality for an unbounded number of sessions

- **undecidable** in general  
  [Even & Goldreich, 83; Durgin et al, 99]

- some **decidability results** for some restricted fragment, e.g. one variable per protocol’s rule  
  [Comon & Cortier, 03]

- **ProVerif**: A tool that does not correspond to any decidability result but works well in practice.  
  [Blanchet, 01]
State of the art in a nutshell

confidentiality for a bounded number of sessions

- a decidability result (NP-complete)
  [Rusinowitch & Turuani, 01; Millen & Shmatikov, 01]
- result extended to deal with various cryptographic primitives.

→ various automatic tools, e.g. AVISPA platform [Armando et al., 05]
  More details about this tomorrow!
Challenge

Would you be able to find the attack on the well-known Needham-Schroeder protocol?

\[
\begin{align*}
A \rightarrow B & : \ \{A, N_a\}_{pub(B)} \\
B \rightarrow A & : \ \{N_a, N_b\}_{pub(A)} \\
A \rightarrow B & : \ \{N_b\}_{pub(B)}
\end{align*}
\]

To help you:
Questions?

See you tomorrow!
### Post Correspondence Problem

**Input** A sequence of tiles \((u_0, v_0)(u_1, v_1)\ldots(u_n, v_n)\) with \(u_i, v_i \in \{a, b\}^*\).

**Output** Does there exist \(k \geq 1\), and \(1 \leq i_1, \ldots, i_k \leq n\) such that

\[
u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}\]
Undecidability

Post Correspondence Problem

Input  A sequence of tiles \((u_0, v_0) (u_1, v_1) \ldots (u_n, v_n)\) with 
\(u_i, v_i \in \{a, b\}^*\).

Output  Does there exist \(k \geq 1\), and \(1 \leq i_1, \ldots, i_k \leq n\) such that 
\(u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}\)

Example:

\[
\begin{array}{cccccccc}
  u_1 & u_2 & u_3 & u_4 & v_1 & v_2 & v_3 & v_4 \\
  aba & bbb & aab & bb & a & aaa & abab & babba \\
\end{array}
\]

A solution is 1431. Indeed, we have that:

\[
u_1.u_4.u_3.u_1 = aba.bb.aab.aba = a.babba.abab.a = v_1.v_4.v_3.v_1\]

No solution if we remove the tile \((u_4, v_4)\).
Post Correspondence Problem

Input  A sequence of tiles \((u_0, v_0) (u_1, v_1) \ldots (u_n, v_n)\) with 
\(u_i, v_i \in \{a, b\}^*\).

Output  Does there exist \(k \geq 1\), and \(1 \leq i_1, \ldots, i_k \leq n\) such that 
\(u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}\)

Example:

\[
\begin{array}{cccccccc}
  u_1 & u_2 & u_3 & u_4 & v_1 & v_2 & v_3 & v_4 \\
  aba & bbb & aab & bb & a & aaa & abab & babba \\
\end{array}
\]

A solution is 1431. Indeed, we have that:

\[
u_1.u_4.u_3.u_1 = aba.bb.aab.aba = a.babba.abab.a = v_1.v_4.v_3.v_1
\]

No solution if we remove the tile \((u_4, v_4)\).

Proposition:  The PCP is undecidable.
Reduction from PCP

We built a protocol that admits an attack (s is revealed) if, and only if, PCP has a solution.
### Reduction from PCP

We built a protocol that admits an attack \((s \text{ is revealed})\) if, and only if, PCP has a solution.

We encode words and concatenation using pairs

- **babba** is encoded as \(\langle\langle\langle b, a\rangle, b\rangle, b\rangle, a\rangle\),
- \(x \cdot (babba)\) is encoded as \(\langle\langle\langle\langle x, b\rangle, a\rangle, b\rangle, b\rangle, a\rangle\)
Undecidability proof

**Reduction from PCP**

We built a protocol that admits an attack \((s\) is revealed) if, and only if, PCP has a solution.

We encode words and concatenation using pairs

- \(babba\) is encoded as \(\langle\langle\langle b, a\rangle, b\rangle, b\rangle, a\rangle\),
- \(x \cdot (babba)\) is encoded as \(\langle\langle\langle x, b\rangle, a\rangle, b\rangle, b\rangle, a\rangle\)

**Initialisation:** \(\text{out}(\text{senc}(\langle u_1, v_1 \rangle, k)) \ldots \text{out}(\text{senc}(\langle u_n, v_n \rangle, k))\)
Undecidability proof

**Reduction from PCP**

We built a protocol that admits an attack ($s$ is revealed) if, and only if, PCP has a solution.

We encode words and concatenation using pairs

- $babba$ is encoded as $⟨⟨⟨⟨ b, a ⟩, b ⟩, b ⟩, a ⟩$.
- $x \cdot (babba)$ is encoded as $⟨⟨⟨⟨ ⟨x, b⟩, a⟩, b⟩, b⟩, a⟩$

**Initialisation:** $\text{out}(\text{senc}(⟨u_1, v_1⟩, k)) \ldots \text{out}(\text{senc}(⟨u_n, v_n⟩, k))$

**Building words**

- $! \ \text{in}(\text{senc}(⟨x, y⟩, k)) \cdot \text{out}(\text{senc}(⟨x \cdot u_1, y \cdot v_1⟩, k))$
- $\ldots$
- $! \ \text{in}(\text{senc}(⟨x, y⟩, k)) \cdot \text{out}(\text{senc}(⟨x \cdot u_1, y \cdot v_1⟩, k))$
Undecidability proof

Reduction from PCP

We built a protocol that admits an attack (\(s\) is revealed) if, and only if, PCP has a solution.

We encode words and concatenation using pairs

- \(babba\) is encoded as \(\langle\langle\langle b, a, b, b, a \rangle\rangle\),
- \(x \cdot (babba)\) is encoded as \(\langle\langle\langle x, b, a, b, b, a \rangle\rangle\)

Initialisation: \(\text{out}(\text{senc}(\langle u_1, v_1 \rangle, k)) \ldots \text{out}(\text{senc}(\langle u_n, v_n \rangle, k))\)

Building words

- \(!\ \text{in}(\text{senc}(\langle x, y \rangle, k))\).\text{out}(\text{senc}(\langle x \cdot u_1, y \cdot v_1 \rangle, k))\)
- \(\ldots\)
- \(!\ \text{in}(\text{senc}(\langle x, y \rangle, k))\).\text{out}(\text{senc}(\langle x \cdot u_1, y \cdot v_1 \rangle, k))\)

Revealing the secret \(s\): \(\text{in}(\text{senc}(\langle z, z \rangle, k))\).\text{out}(s)\)
ProVerif

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

- Online demo available at: http://proverif.rocq.inria.fr/
- Sources available on Bruno Blanchet’s webpage

Advantages

- fully automatic, and quite efficient
- A rich process algebra: replication, else branches, ...
- Handles many cryptographic primitives
- Proves various security properties: secrecy, correspondences, equivalences
ProVerif

ProVerif is a verifier for cryptographic protocols that may prove that a protocol is secure or exhibit attacks.

- Online demo available at: http://proverif.rocq.inria.fr/
- Sources available on Bruno Blanchet’s webpage

**Advantages**

- fully automatic, and quite efficient
- A rich process algebra: replication, else branches, ...
- Handles many cryptographic primitives
- Proves various security properties: secrecy, correspondences, equivalences

**No miracle**

Termination is not guaranteed and sometimes the tool is not able to conclude.
Experimental results

still, ProVerif works well in practice.

<table>
<thead>
<tr>
<th>Protocol</th>
<th>Result</th>
<th>ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Needham-Schroeder shared key</td>
<td>Attack</td>
<td>52</td>
</tr>
<tr>
<td>Needham-Schroeder shared key corrected</td>
<td>Secure</td>
<td>109</td>
</tr>
<tr>
<td>Denning-Sacco</td>
<td>Attack</td>
<td>6</td>
</tr>
<tr>
<td>Denning-Sacco corrected</td>
<td>Secure</td>
<td>7</td>
</tr>
<tr>
<td>Otway-Rees</td>
<td>Secure</td>
<td>10</td>
</tr>
<tr>
<td>Otway-Rees, variant of Paulson98</td>
<td>Attack</td>
<td>12</td>
</tr>
<tr>
<td>Yahalom</td>
<td>Secure</td>
<td>10</td>
</tr>
<tr>
<td>Simpler Yahalom</td>
<td>Secure</td>
<td>11</td>
</tr>
<tr>
<td>Main mode of Skeme</td>
<td>Secure</td>
<td>23</td>
</tr>
</tbody>
</table>

Pentium III, 1 GHz.