Verification of security protocols: from confidentiality to privacy

Stéphanie Delaune

LSV, CNRS & ENS Cachan, France

Tuesday, August 25th, 2015

ENS Cachan

- 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

SECSI team

Security of Information Systems

- 4 permanents: David Baelde, H. Comon-Lundh, S. Delaune, et J. Goubault-Larrecq.

- 1 engineer + 1 postdoc
- 3 phd students

Cryptographic protocols everywhere !

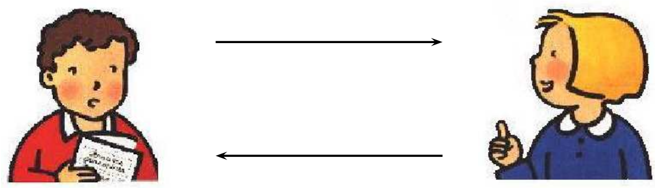
Goal: they aim at securing communications over public/insecure networks

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.
- ...

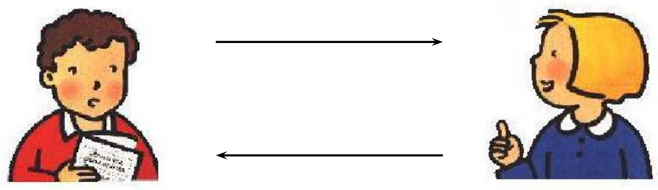
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



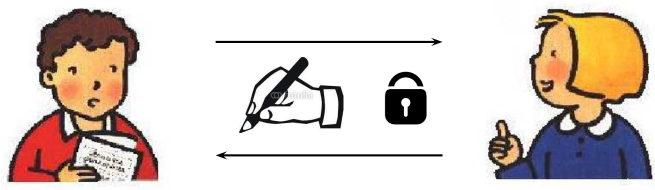
How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages



How does a cryptographic protocol work (or not)?

Protocol: small programs explaining how to exchange messages

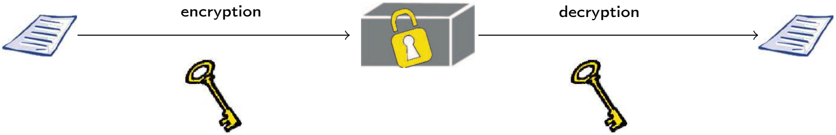


Cryptographic: make use of cryptographic primitives

Examples: symmetric encryption, asymmetric encryption, signature, hashes, . . .

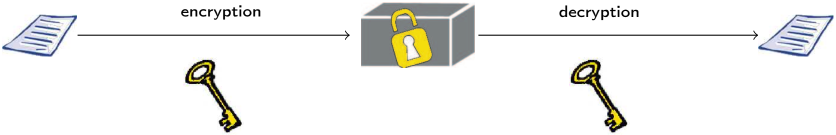
What is a symmetric encryption scheme?

Symmetric encryption

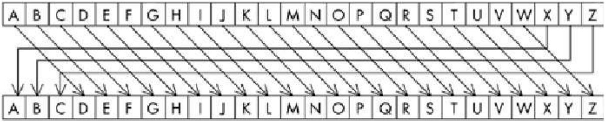


What is a symmetric encryption scheme?

Symmetric encryption



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



Today: DES (1977), AES (2000)

A famous example

Enigma machine (1918-1945)

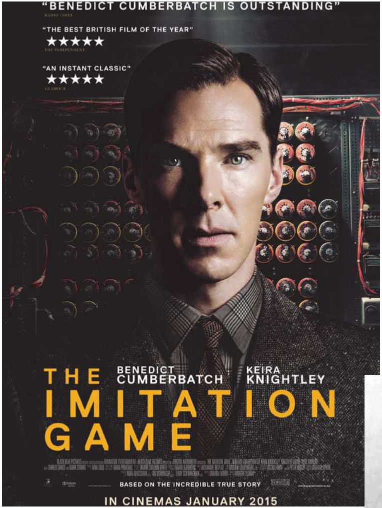
- electro-mechanical rotor cipher machines used by the German to encrypt during Wold War II
- permutations and substitutions

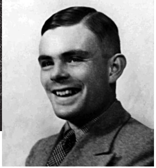
A bit of history

- 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.

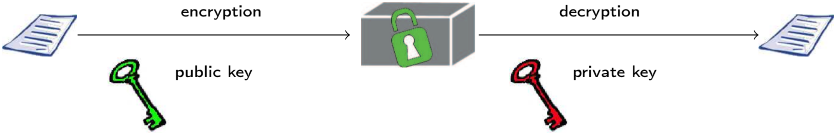
Advertisement





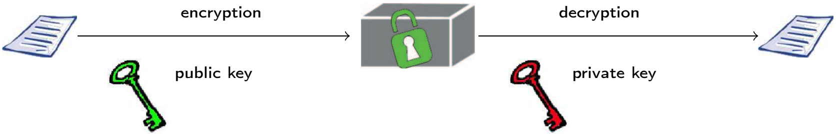
What is an asymmetric encryption scheme?

Asymmetric encryption



What is an asymmetric encryption scheme?

Asymmetric encryption



Examples:

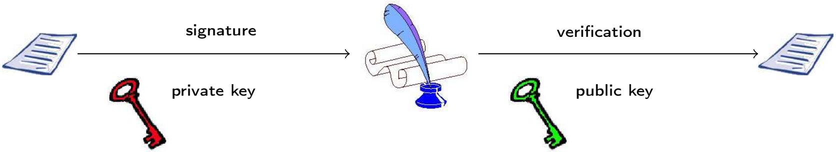
- 1976: first system published by W. Diffie, and M. Hellman,
- 1977: RSA system published by R. Rivest, A. Shamir, and L. Adleman.

$\longrightarrow$ 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?

Signature



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)

$$\begin{array}{rcl} A \rightarrow B & : & \mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B)) \\ B \rightarrow A & : & \mathrm{senc}(s, k) \end{array}$$

What about secrecy of s ?

How does a cryptographic protocol work (or not)?

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

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

What about secrecy of s ?

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

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

C knows the key k

How does a cryptographic protocol work (or not)?

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

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

What about secrecy of s ?

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

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

C knows the key k

2. $C(A) \to B : \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B))$
3. $B \to A : \mathsf{senc}(s, k)$ Attack !

Exercise

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

Version 1

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\langle A, B, \mathrm{sign}(k, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Version 2

$$\begin{array}{lcl} A \to B & : & \mathrm{aenc}(\mathrm{sign}(\langle A, B, k\rangle, \mathrm{priv}(A))\rangle, \mathrm{pub}(B)) \\ B \to A & : & \mathrm{senc}(s, k) \end{array}$$

Which version would you prefer to use?

Exercise

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

Version 1

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\langle A, B, \mathsf{sign}(k, \mathsf{priv}(A))\rangle, \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Version 2

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(\langle A, B, k\rangle, \mathsf{priv}(A))\rangle, \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Which version would you prefer to use? Version 2

$\longrightarrow$ Version 1 is still vulnerable to the aforementioned attack.

What about protocols used in real life ?

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.

$\longrightarrow$ 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
641740644252416500858395774644508840500943086599

⟶ now, the number that is used is made of 232 digits

HTTPS connections

Lots of bugs and attacks, with fixes every month

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

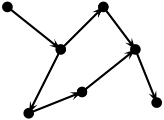
1. a logical flaw that allows a man in the middle attacker to downgrade connections from 'strong' RSA to 'export-grade' RSA;
2. breaking encryption via factorisation of such a key can be easily done.

$\longrightarrow$ 'export-grade' were introduced under the pressure of US governments agencies to ensure that they would be able to decrypt all foreign encrypted communication.

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

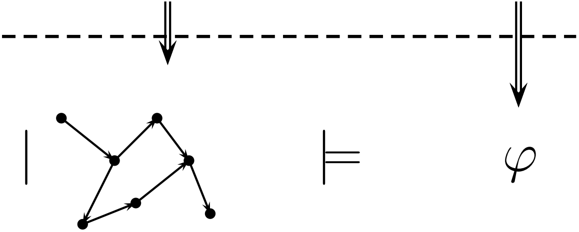
Modelling

$|$  $\models \varphi$

This talk: formal methods for protocol verification

Does the protocol *satisfy* a security property?

Modelling



Two main tasks

1. Modelling cryptographic protocols and their security properties
2. Designing verification algorithms

Modelling messages

and

Deciding knowledge

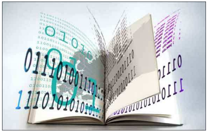
(in a simple setting)

Symbolic model

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

Symbolic model

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

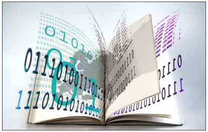


Messages

They are abstracted by terms.

Symbolic model

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



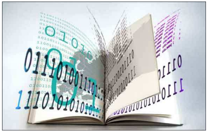
Messages

They are abstracted by terms.

The attacker

Symbolic model

⟶ 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

$\longrightarrow$ It is important to have a tight modelling of messages

Messages as terms

$\longrightarrow$ 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 $\mathcal{N}$.

$$\begin{array}{lcll} t & ::= & n & \text{name } n \in \mathcal{N} \\ & \mid & \mathsf{f}(t_1, \ldots, t_k) & \text{application of symbol } \mathsf{f} \in \mathcal{F} \end{array}$$

- Names are used to model atomic data
  $\longrightarrow$ *e.g.* keys, nonces, agent names, ...
- Function symbols are used to model cryptographic primitives
  $\longrightarrow$ *e.g.* encryption, signature, ...

A typical signature

Standard primitives

$$\mathcal{F} = \{\mathsf{senc},\ \mathsf{aenc},\ \mathsf{sk},\ \mathsf{sign},\ \langle\ \rangle\}$$

A typical signature

Standard primitives

$$\mathcal{F} = \{\mathsf{senc},\ \mathsf{aenc},\ \mathsf{sk},\ \mathsf{sign},\ \langle\ \rangle\}$$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

These messages can be modelled as follows:

1. $\mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(a)), b)$;
2. $\mathsf{senc}(s, k)$

Symbolic manipulation on terms

He may build new messages following deduction rules

Pairing

$$\frac{x \quad y}{\langle x, y\rangle} \qquad \frac{\langle x, y\rangle}{x} \qquad \frac{\langle x, y\rangle}{y}$$

Symmetric encryption

$$\frac{x \quad y}{\mathrm{senc}(x, y)} \qquad \frac{\mathrm{senc}(x, y) \quad y}{x}$$

Asymmetric encryption

$$\frac{x \quad y}{\mathrm{aenc}(x, y)} \qquad \frac{\mathrm{aenc}(x, y) \quad \mathrm{sk}(y)}{x}$$

Signature

$$\frac{x \quad \mathrm{sk}(y)}{\mathrm{sign}(x, \mathrm{sk}(y))} \qquad \frac{\mathrm{sign}(x, \mathrm{sk}(y))}{x}$$

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

$$\frac{v_1 \quad \ldots \quad v_n}{v_0} \quad \text{is an instance of R}$$

3. the root is labeled by u.

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

$$\frac{v_1 \quad \ldots \quad v_n}{v_0} \quad \text{is an instance of R}$$

3. the root is labeled by u.

Exercise - Going back to the Denning Sacco protocol

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

Is s deducible from T ?

Exercise

Exercise - Going back to the Denning Sacco protocol

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

Is s deducible from T ?

Exercise

Exercise - Going back to the Denning Sacco protocol

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

Is s deducible from T ?

Answer: Of course, Yes !

$$\dfrac{\mathsf{senc}(s,k) \qquad \dfrac{\dfrac{\mathsf{aenc}(\mathsf{sign}(k,\mathsf{sk}(a)),c) \quad \mathsf{sk}(c)}{\mathsf{sign}(k,\mathsf{sk}(a))}}{k}}{s}$$

Denning Sacco protocol

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

   2. $C(A) \to B$: $\mathrm{aenc}(\mathrm{sign}(k, \mathrm{priv}(A)), \mathrm{pub}(B))$
   3. $B \to A$: $\mathrm{senc}(s, k)$ Attack !

Exercise (continued)

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

Is $\mathrm{aenc}(\mathrm{sign}(k, \mathrm{sk}(a)), b)$ deducible from T_0?

Denning Sacco protocol

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

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

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

Exercise (continued)

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

Is $\mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(a)), b)$ deducible from T_0?

Answer: Of course, Yes !

$$\dfrac{\dfrac{\mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(a)), c) \quad \mathsf{sk}(c)}{\mathsf{sign}(k, \mathsf{sk}(a))} \quad b}{\mathsf{aenc}(\mathsf{sign}(k, \mathsf{sk}(a)), b)}$$

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, completeness, and termination

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

Soundness, completeness, and termination

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

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.
$\longrightarrow$ easy to prove for the deduction rules under study

Soundness, completeness, and termination

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

Termination The set of subterms is finite and polynomial, and one-step deducibility can be checked in polynomial time.
$\longrightarrow$ easy to prove for the deduction rules under study

Completeness If the term u is deducible from T, then the algorithm returns Yes. Otherwise, it returns No.
$\longrightarrow$ this relies on a locality property

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 $v \in St(T \cup \{u\})$.

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 $St(T)$.

$\longrightarrow$ this is left as an exercise

Exercise

Consider the following set of deduction rules:

$$\frac{x \quad \mathsf{sk}(y)}{\mathsf{sign}(x, \mathsf{sk}(y))} \qquad \frac{\mathsf{sign}(x, sk(y)) \quad \mathsf{vk}(y)}{x} \qquad \frac{y}{\mathsf{vk}(y)}$$

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 de deduction problem is decidable.

Exercise

Consider the following set of deduction rules:

$$\frac{x \quad \mathsf{sk}(y)}{\mathsf{sign}(x, \mathsf{sk}(y))} \qquad \frac{\mathsf{sign}(x, sk(y)) \quad \mathsf{vk}(y)}{x} \qquad \frac{y}{\mathsf{vk}(y)}$$

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 de deduction problem is decidable.

Solution

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

$\longrightarrow$ the locality proof is left as an exercise

Exercise

Consider the following set of deduction rules:

$$\frac{x \quad y}{\langle x, y\rangle} \qquad \frac{\langle x, y\rangle}{x} \qquad \frac{\langle x, y\rangle}{y} \qquad \frac{x \quad y}{\mathsf{senc}(x, y)} \qquad \frac{\mathsf{senc}(x, y) \quad y}{x}$$

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 completness?

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 equational theory E to take into account the properties of the primitives

Exclusive or operator:

$$\begin{array}{rcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) \\ x \oplus y & = & y \oplus x \end{array} \qquad \begin{array}{rcl} x \oplus x & = & 0 \\ x \oplus 0 & = & x \end{array}$$

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:

$$
\begin{array}{rclcrcl}
(x \oplus y) \oplus z & = & x \oplus (y \oplus z) & \qquad & x \oplus x & = & 0 \\
x \oplus y & = & y \oplus x & & x \oplus 0 & = & x
\end{array}
$$

Blind signature (used in evoting protocol)

$$
\begin{array}{rcl}
\mathsf{check}(\mathsf{sign}(x, y), \mathsf{vk}(y)) & = & x \\
\mathsf{unblind}(\mathsf{blind}(y, y), y) & = & x \\
\mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) & = & \mathsf{sign}(x, z)
\end{array}
$$

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:

$$\begin{array}{rclcrcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) & \qquad & x \oplus x & = & 0 \\ x \oplus y & = & y \oplus x & & x \oplus 0 & = & x \end{array}$$

Blind signature (used in evoting protocol)

$$\begin{array}{rcl} \mathsf{check}(\mathsf{sign}(x, y), \mathsf{vk}(y)) & = & x \\ \mathsf{unblind}(\mathsf{blind}(y, y), y) & = & x \\ \mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x, y), z), y) & = & \mathsf{sign}(x, z) \end{array}$$

Homomorphic encryption:

$$\begin{array}{rcl} \mathsf{enc}(\langle x, y\rangle, z) & = & \langle \mathsf{enc}(x, z), \mathsf{enc}(y, z)\rangle \\ \mathsf{dec}(\langle x, y\rangle, z) & = & \langle \mathsf{dec}(x, z), \mathsf{dec}(y, z)\rangle \end{array} \qquad \begin{array}{rcl} \mathsf{sdec}(\mathsf{senc}(x, y), y) & = & x \\ \mathsf{proj}_1(\langle x, y\rangle) & = & x \\ \mathsf{proj}_2(\langle x, y\rangle) & = & y \end{array}$$

Going back to the Denning Sacco protocol

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B))$$
$$B \to A \quad : \quad \mathsf{senc}(s, k)$$

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

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

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

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

Going back to the Denning Sacco protocol

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B))$$
$$B \to A \quad : \quad \mathsf{senc}(s, k)$$

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

1. symmetric encryption: $\mathsf{senc}(\cdot, \cdot)$, $\mathsf{sdec}(\cdot, \cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

2. asymmetric encryption: $\mathsf{aenc}(\cdot, \cdot)$, $\mathsf{adec}(\cdot, \cdot)$, $\mathsf{pk}(\cdot)$

   $\longrightarrow \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

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

1. symmetric encryption: $\mathsf{senc}(\cdot,\cdot)$, $\mathsf{sdec}(\cdot,\cdot)$

   $\longrightarrow \mathsf{sdec}(\mathsf{senc}(x, y), y) = x$

2. asymmetric encryption: $\mathsf{aenc}(\cdot,\cdot)$, $\mathsf{adec}(\cdot,\cdot)$, $\mathsf{pk}(\cdot)$

   $\longrightarrow \mathsf{adec}(\mathsf{aenc}(x, \mathsf{pk}(y)), y) = x$

3. signature: $\mathsf{sign}(\cdot,\cdot)$, $\mathsf{check}(\cdot,\cdot)$

   $\longrightarrow \mathsf{check}(\mathsf{sign}(x, y), \mathsf{pk}(y)) = x$

Deduction in this more general setting

Deduction rules are as follows:

$$\frac{u_1 \quad \cdots \quad u_k}{\mathsf{f}(u_1, \ldots, u_k)} \quad \mathsf{f} \in \mathcal{F} \qquad\qquad \frac{u}{u'} \quad u =_{\mathsf{E}} u'$$

Deduction in this more general setting

Deduction rules are as follows:

$$\frac{u_1 \quad \cdots \quad u_k}{\mathsf{f}(u_1, \ldots, u_k)} \quad \mathsf{f} \in \mathcal{F} \qquad\qquad \frac{u}{u'} \quad u =_{\mathsf{E}} u'$$

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

$$\dfrac{\dfrac{\mathsf{senc}(\mathit{secret}, k) \qquad k}{\mathsf{sdec}(\mathsf{senc}(\mathit{secret}, k), k)}\ \mathsf{sdec} \in \mathcal{F}}{\mathit{secret}}\ \mathsf{sdec}(\mathsf{senc}(\mathsf{x}, \mathsf{y}), \mathsf{y}) = \mathsf{x}$$

The deduction problem: is u deducible from ϕ?

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 ϕ ?

The deduction problem: is u deducible from ϕ?

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 ϕ ?

Characterization of deduction

$T \vdash u$ if, and only if, there exists a term R such that $R\phi =_{\mathsf{E}} u$.

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

The deduction problem: is u deducible from ϕ?

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 ϕ ?

Characterization of deduction

$T \vdash u$ if, and only if, there exists a term R such that $R\phi =_{\mathsf{E}} u$.
$\longrightarrow$ such a term R is a recipe of the term u.

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

We have that:

The deduction problem: is u deducible from ϕ?

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 ϕ ?

Characterization of deduction

$T \vdash u$ if, and only if, there exists a term R such that $R\phi =_\mathsf{E} u$.
$\longrightarrow$ such a term R is a recipe of the term u.

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

We have that:

- k is deducible from ϕ using $R_1 = \mathsf{check}(\mathsf{adec}(w_4, w_3), w_1)$,

The deduction problem: is u deducible from ϕ?

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 ϕ ?

Characterization of deduction

$T \vdash u$ if, and only if, there exists a term R such that $R\phi =_{\mathsf{E}} u$.

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

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

We have that:

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

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

Going back to the previous example:

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

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

Going back to the previous example:

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

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

Going back to the previous example:

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

Some other equational theories

Blind signature

$$
\begin{aligned}
\mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) &= x \\
\mathsf{unblind}(\mathsf{blind}(y,y),y) &= x \\
\mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) &= \mathsf{sign}(x,z)
\end{aligned}
$$

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

$\longrightarrow$ $\mathsf{sign}(m,k)$ will be considered as a subterm of $\mathsf{sign}(\mathsf{blind}(m,r),k)$

Some other equational theories

Blind signature

$$\begin{array}{rcl} \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) & = & x \\ \mathsf{unblind}(\mathsf{blind}(y,y),y) & = & x \\ \mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) & = & \mathsf{sign}(x,z) \end{array}$$

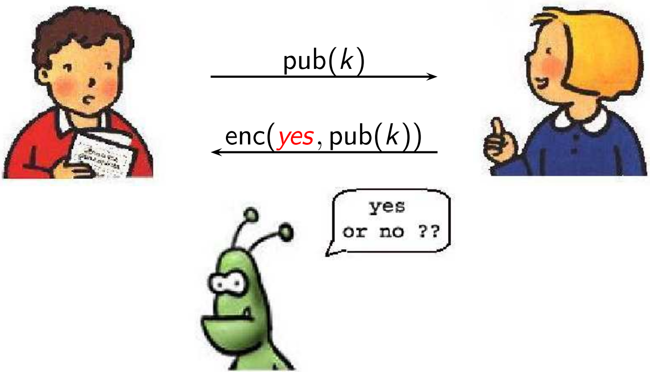
Decidability can be shown in a similar fashion extending the notion of subterm.
$\longrightarrow$ $\mathsf{sign}(m,k)$ will be considered as a subterm of $\mathsf{sign}(\mathsf{blind}(m,r),k)$

Exclusive or

$$\begin{array}{rclcrcl} (x \oplus y) \oplus z & = & x \oplus (y \oplus z) & \quad & x \oplus x & = & 0 \\ x \oplus y & = & y \oplus x & & x \oplus 0 & = & x \end{array}$$

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?

Static equivalence

The static equivalence problem

Input Two frames ϕ and ψ

$$\phi = \{w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell\} \qquad \psi = \{w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell\}$$

Ouput Can the attacker distinguish the two frames, *i.e.* does there exist a test $R_1 \stackrel{?}{=} R_2$ 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 ϕ and ψ

$$\phi = \{w_1 \triangleright u_1, \ldots, w_\ell \triangleright u_\ell\} \qquad \psi = \{w_1 \triangleright v_1, \ldots, w_\ell \triangleright v_\ell\}$$

Ouput Can the attacker distinguish the two frames, *i.e.* does there exist a test $R_1 \stackrel{?}{=} R_2$ 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:

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

They are not in static equivalence: $\mathsf{aenc}(yes, w_1) \stackrel{?}{=} w_2$.

Exercise

Consider the equational theories:

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

Questions

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

$$\begin{array}{rcl}
\{w_1 \rhd \mathsf{yes}\} & \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} & \{w_1 \rhd \mathsf{no}\} \\
\{w_1 \rhd \mathsf{senc}(\mathsf{yes}, k)\} & \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} & \{w_1 \rhd \mathsf{senc}(\mathsf{no}, k)\} \\
\{w_1 \rhd \mathsf{senc}(n, k), w_2 \rhd k\} & \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} & \{w_1 \rhd \mathsf{senc}(n, k), w_2 \rhd k'\} \\
\{w_1 \rhd \mathsf{senc}(n, k), w_2 \rhd k\} & \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} & \{w_1 \rhd \mathsf{senc}(n, k), w_2 \rhd k'\}
\end{array}$$

Exercise

Consider the equational theories:

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

Questions

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

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

Exercise

Consider the equational theories:

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

Questions

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

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \qquad \text{✗}$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \qquad \checkmark$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

Exercise

Consider the equational theories:

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

Questions

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

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \qquad \checkmark$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\}$$

Exercise

Consider the equational theories:

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

Questions

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

$$\{w_1 \triangleright \mathsf{yes}\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{no}\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(\mathsf{yes}, k)\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(\mathsf{no}, k)\} \qquad \checkmark$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{senc}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \qquad X$$

$$\{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k\} \overset{?}{\sim}_{\mathsf{E}_{\mathsf{cipher}}} \{w_1 \triangleright \mathsf{senc}(n, k), w_2 \triangleright k'\} \qquad \checkmark$$

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 ϕ/ψ with their deducible subterms ϕ^+/ψ^+
2. does there exist a test $R_1 \stackrel{?}{=} R_2$ such that $R_1\phi^+ = R_2\phi^+$ whereas $R_1\psi^+ \neq R_2\psi^+$ (again syntaxic equality) ?

   $\longrightarrow$ actually, we only need to consider small tests

Example

Consider the frames:

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

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

Example

Consider the frames:

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

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

Applying the algorithm on these frames, we get:

- $\phi^+ = \phi \uplus \{$, and
- $\psi^+ = \psi \uplus \{$.

Example

Consider the frames:

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

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

Applying the algorithm on these frames, we get:

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

Example

Consider the frames:

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

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

Applying the algorithm on these frames, we get:

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

Example

Consider the frames:

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

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

Applying the algorithm on these frames, we get:

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

Example

Consider the frames:

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

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

Applying the algorithm on these frames, we get:

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

$\longrightarrow$ Conclusion: ϕ^+ and ψ^+ are not in static equivalence: $w_4 \stackrel{?}{=} yes$.

Some other equational theories

Blind signature

$$\begin{aligned} \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) &= x \\ \mathsf{unblind}(\mathsf{blind}(x,y),y) &= x \\ \mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) &= \mathsf{sign}(x,z) \end{aligned}$$

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

Some other equational theories

Blind signature

$$\begin{aligned} \mathsf{check}(\mathsf{sign}(x,y),\mathsf{vk}(y)) &= x \\ \mathsf{unblind}(\mathsf{blind}(x,y),y) &= x \\ \mathsf{unblindsign}(\mathsf{sign}(\mathsf{blind}(x,y),z),y) &= \mathsf{sign}(x,z) \end{aligned}$$

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

Exclusive or

$$\begin{aligned} (x \oplus y) \oplus z &= x \oplus (y \oplus z) \qquad & x \oplus x &= 0 \\ x \oplus y &= y \oplus x \qquad & x \oplus 0 &= x \end{aligned}$$

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

Theory E	Deduction	Static Equivalence
subterm convergent	PTIME	
blind sign., addition, homo. encryption	decidable [Abadi & Cortier, TCS'06]	
ACU	NP-complete	PTIME
Exclusive Or Abelian Group	PTIME	PTIME
ACUNh/AGh	PTIME	decidable
	[D., IPL'05;Cortier & D., JAR'12]	

⟶ 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

$\longrightarrow$ based on the π-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

$\longrightarrow$ based on the π-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

Protocols as processes - syntax and semantics

Syntax : $P, Q \; := $

$P, Q \; :=$	0	null process
	$\mathsf{in}(c, x).P$	input
	$\mathsf{out}(c, u).P$	output
	if $u = v$ then P else Q	conditional
	$P \mid Q$	parallel composition
	$!P$	replication
	new $n.P$	fresh name generation

Protocols as processes - syntax and semantics

Syntax : $P, Q \quad := \quad$

0	null process
$\text{in}(c, x).P$	input
$\text{out}(c, u).P$	output
$\text{if } u = v \text{ then } P \text{ else } Q$	conditional
$P \mid Q$	parallel composition
$!P$	replication
$\text{new } n.P$	fresh name generation

Semantics $\rightarrow$:

Comm	$\text{out}(c, M).P \mid \text{in}(c, x).Q \rightarrow P \mid Q\{M/x\}$
Then	$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow P \text{ when } M =_{\mathsf{E}} N$
Else	$\text{if } M = N \text{ then } P \text{ else } Q \rightarrow Q \quad \text{when } M \neq_{\mathsf{E}} N$

closed by structural equivalence ($\equiv$) and application of evaluation contexts.

Going back to Denning Sacco protocol

$$
\begin{array}{lcl}
A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\
B \to A & : & \mathsf{senc}(s, k)
\end{array}
$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{rcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\ \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \quad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in} \ldots \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{rcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\, \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \qquad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in}\ldots \end{array}$$

$$\begin{array}{rcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \mathsf{let}\ y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a)\ \mathsf{in} \\ & & \qquad \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \mathsf{new}\ k.\ \mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \qquad \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in}\ldots \end{array}$$

$$\begin{array}{lcl} P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \mathsf{let}\ y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a)\ \mathsf{in} \\ & & \qquad \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$P_{\mathsf{DS}} = \mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big)$$

Going back to Denning Sacco protocol

$$A \to B \quad : \quad \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B))$$
$$B \to A \quad : \quad \mathsf{senc}(s, k)$$

Alice and Bob as processes:

$$P_A(sk_a, pk_b) \;=\; \text{new } k.\,\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\; \mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in} \ldots$$

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

One possible scenario:

$$P_{\mathsf{DS}} \;=\; \text{new } sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big)$$
$$\to \;\; \text{new } sk_a, sk_b, k.\big(\mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in } \ldots \mid \text{let } y_b = k \text{ in new } s.\mathsf{out}(c, \mathsf{senc}(s, y_b))\big)$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$\begin{array}{lcl} P_A(sk_a, pk_b) & = & \text{new } k.\,\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)). \\ & & \quad \mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in}\ldots \\ P_B(sk_b, pk_a) & = & \mathsf{in}(c, x_b).\ \text{let } y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a) \text{ in} \\ & & \quad \text{new } s.\mathsf{out}(c, \mathsf{senc}(s, y_b)) \end{array}$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \text{new } sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big) \\ & \to & \text{new } sk_a, sk_b, k.\big(\mathsf{in}(c, x_a).\ \text{let } y_a = \mathsf{sdec}(x_a, k) \text{ in } \ldots \\ & & \qquad \mid\ \text{let } y_b = k \text{ in new } s.\mathsf{out}(c, \mathsf{senc}(s, y_b))\big) \\ & \to & \text{new } sk_a, sk_b, k, s.\big(\ \text{let } y_a = \mathsf{sdec}(\mathsf{senc}(s, k), k) \text{ in } \ldots \mid 0\big) \end{array}$$

Going back to Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

Alice and Bob as processes:

$$P_A(sk_a, pk_b) \;=\; \mathsf{new}\ k.\,\mathsf{out}(c, \mathsf{aenc}(\mathsf{sign}(k, sk_a), pk_b)).\ \mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in} \ldots$$

$$P_B(sk_b, pk_a) \;=\; \mathsf{in}(c, x_b).\ \mathsf{let}\ y_b = \mathsf{check}(\mathsf{adec}(x_b, sk_b), pk_a)\ \mathsf{in}\ \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b))$$

One possible scenario:

$$\begin{array}{rcl} P_{\mathsf{DS}} & = & \mathsf{new}\ sk_a, sk_b.\big(P_A(sk_a, \mathsf{pk}(sk_b)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big) \\ & \rightarrow & \mathsf{new}\ sk_a, sk_b, k.\big(\mathsf{in}(c, x_a).\ \mathsf{let}\ y_a = \mathsf{sdec}(x_a, k)\ \mathsf{in}\ \ldots \\ & & \qquad \mid\ \mathsf{let}\ y_b = k\ \mathsf{in}\ \mathsf{new}\ s.\mathsf{out}(c, \mathsf{senc}(s, y_b))\big) \\ & \rightarrow & \mathsf{new}\ sk_a, sk_b, k, s.\big(\ \mathsf{let}\ y_a = \mathsf{sdec}(\mathsf{senc}(s, k), k)\ \mathsf{in}\ \ldots \mid 0\big) \end{array}$$

$\longrightarrow$ this simply models a normal execution between two honest participants

Security properties - confidentiality

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 $C[\mathsf{out}(c, s).Q']$ with c public.

Security properties - confidentiality

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 $C[\mathrm{out}(c, s).Q']$ 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
  $\longrightarrow$ replications to model an unbounded number of sessions,
  $\longrightarrow$ reveal public keys and private keys to model dishonest agents,
  $\longrightarrow$ P_A/P_B may play with other (and perhaps) dishonest agents, ...

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \rightarrow B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \rightarrow A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

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

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

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a.\mathsf{new}\ sk_b.\big(\mathsf{out}(c, \mathsf{pk}(sk_b)) \mid P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a))\big)$$

Going back to the Denning Sacco protocol

$$\begin{array}{lcl} A \to B & : & \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ B \to A & : & \mathsf{senc}(s, k) \end{array}$$

The aforementioned attack

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

$$\begin{array}{ll} 2. & C(A) \to B : \mathsf{aenc}(\mathsf{sign}(k, \mathsf{priv}(A)), \mathsf{pub}(B)) \\ 3. & \quad B \to A : \mathsf{senc}(s, k) \end{array}$$

The "minimal" initial configuration to retrieve the attack is:

$$\mathsf{new}\ sk_a.\mathsf{new}\ sk_b.(\mathsf{out}(c, \mathsf{pk}(sk_b)) \mid P_A(sk_a, \mathsf{pk}(sk_c)) \mid P_B(sk_b, \mathsf{pk}(sk_a)))$$

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	$:=$	0	null process
		$\mathrm{in}(c, x).P$	input
		$\ldots$	
		$\text{event } p(u_1, \ldots, u_n).P$	event

Authentication properties with agreement on some values:

$$\forall x.\mathrm{EndB}(a, b, x) \Rightarrow \mathrm{EndA}(a, b, x)$$

State of the art in a nutshell

confidentiality for an unbounded number of sessions

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

  More details

- 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]

  More details

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.

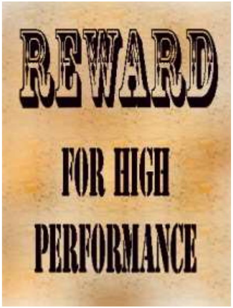
$\longrightarrow$ various automatic tools, e.g. AVISPA platform [Armando *et al.*, 05]

More details about this tomorrow !

Challenge

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

$$
\begin{aligned}
A \rightarrow B &: \quad \{A, N_a\}_{\mathsf{pub}(B)} \\
B \rightarrow A &: \quad \{N_a, N_b\}_{\mathsf{pub}(A)} \\
A \rightarrow B &: \quad \{N_b\}_{\mathsf{pub}(B)}
\end{aligned}
$$



To help you:
`http://www.lsv.ens-cachan.fr/~delaune/VTSA/proverif.pdf`

Questions ?

See you tomorrow !

Undecidability

Post Correspondence Problem

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

Output Does there exist $k \geq 1$, and $1 \leq i_1, \dots, i_k \leq n$ such that

$$u_{i_1} \dots . u_{i_k} = v_{i_1} \dots v_{i_k}$$

Post Correspondence Problem

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

Output Does there exist $k \geq 1$, and $1 \leq i_1, \dots, i_k \leq n$ such that

$$u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$$

Example:

u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
aba	*bbb*	*aab*	*bb*	*a*	*aaa*	*abab*	*babba*

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).

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:

u_1	u_2	u_3	u_4	v_1	v_2	v_3	v_4
aba	bbb	aab	bb	a	aaa	$abab$	$babba$

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.

Undecidability proof

Reduction from PCP

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

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

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

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

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

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

Building words

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

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

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

Building words

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

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

Back

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

$\longrightarrow$ still, ProVerif works well in practice.

Protocol	Result	ms
Needham-Schroeder shared key	Attack	52
Needham-Schroeder shared key corrected	Secure	109
Denning-Sacco	Attack	6
Denning-Sacco corrected	Secure	7
Otway-Rees	Secure	10
Otway-Rees, variant of Paulson98	Attack	12
Yahalom	Secure	10
Simpler Yahalom	Secure	11
Main mode of Skeme	Secure	23

Pentium III, 1 GHz.

Back