Reducing the number of agents in equivalence properties.

Antoine Dallon

July 8th, 2015

Introduction

- Security properties should hold for an unbounded number of agents using the protocol.
- We often forget about it.
- I will show that with some hypothesis, it is sufficient to verify a protocol with a small number of agents.

Introduction

- Security properties should hold for an unbounded number of agents using the protocol.
- We often forget about it.
- I will show that with some hypothesis, it is sufficient to verify a protocol with a small number of agents.

Introduction

- Security properties should hold for an unbounded number of agents using the protocol.
- We often forget about it.
- I will show that with some hypothesis, it is sufficient to verify a protocol with a small number of agents.

Goal

Theorem (Informal)

There exists a class C of protocols (containing at least some interesting examples) such that for every $P, Q \in C$, if there exists an attack against $P \approx Q$, then there is an attack against $P \approx Q$ involving a small finite and calculable number of agents.

Table of contents

Table of contents

Generalities

- Bounding the number of agents implies controlling the keys.
- So we assume that there are private unary function symbols pk, sk, *hon*, *dis*.
- pk represents the public key of an agent, sk the private key.
- *hon* and *dis* are predicates that represent the fact that an agent is honest or dishonest.
- We can also add a binary function symbol shk that will represent shared keys.

Generalities

- Bounding the number of agents implies controlling the keys.
- So we assume that there are private unary function symbols pk, sk, *hon*, *dis*.
- pk represents the public key of an agent, sk the private key.
- *hon* and *dis* are predicates that represent the fact that an agent is honest or dishonest.
- We can also add a binary function symbol shk that will represent shared keys.

Generalities

- Bounding the number of agents implies controlling the keys.
- So we assume that there are private unary function symbols pk, sk, *hon*, *dis*.
- pk represents the public key of an agent, sk the private key.
- *hon* and *dis* are predicates that represent the fact that an agent is honest or dishonest.
- We can also add a binary function symbol shk that will represent shared keys.

Generalities

- Bounding the number of agents implies controlling the keys.
- So we assume that there are private unary function symbols pk, sk, *hon*, *dis*.
- pk represents the public key of an agent, sk the private key.
- *hon* and *dis* are predicates that represent the fact that an agent is honest or dishonest.
- We can also add a binary function symbol shk that will represent shared keys.

Generalities

- Bounding the number of agents implies controlling the keys.
- So we assume that there are private unary function symbols pk, sk, *hon*, *dis*.
- pk represents the public key of an agent, sk the private key.
- *hon* and *dis* are predicates that represent the fact that an agent is honest or dishonest.
- We can also add a binary function symbol shk that will represent shared keys.

Header

$$P_{Header} =$$
$$!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_H, c).\,\mathsf{new}\ ag_H\ .\,\mathsf{out}(c, < ag_H, \mathsf{pk}(ag_H), \mathsf{hon}(ag_H) >)$$
$$|!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_D, c).\,\mathsf{new}\ ag_D\ .\,\mathsf{out}(c, < ag_D, \mathsf{pk}(ag_D), \mathsf{dis}(ag_D), \mathsf{sk}(ag_D) >$$
$$|K$$

where K, defined as follows, is used to do key distribution :

$$K =!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_K, c).\,\mathsf{in}(\mathsf{dis}(x), \mathsf{pk}(y)).\,\mathsf{out}(\mathsf{shk}(x, y))$$

Example : Needham-Schroeder-Lowe

The protocol

1. $A \rightarrow B : \{N_a, A\}_{\mathrm{pk}(B)}$
2. $B \rightarrow A : \{N_a, N_b, B\}_{\mathrm{pk}(A)}$
3. $A \rightarrow B : \{N_b\}_{\mathrm{pk}(B)}$
4. $P : B \rightarrow ? : \{m_1\}_{N_b}$ or $Q : B \rightarrow ? : \{m_1\}_k$ where k is a fresh key.

Example : Needham-Schroeder-Lowe

The role of A

$$P_A = Q_A =$$

$$!\,\mathsf{new}\ c.\ \mathsf{out}(c_A, c).$$

$$\mathsf{in}(c, < \mathsf{pk}(x_a), \mathsf{pk}(x_b) >).$$

$$\mathsf{new}\ n_a.$$

$$\mathsf{out}(c, \mathsf{aenc}(< n_a, x_a >, \mathsf{pk}(x_b))).$$

$$\mathsf{in}(c, \mathsf{aenc}(< n_a, x_{nb}, x_b >, \mathsf{pk}(x_a))).$$

$$\mathsf{out}(c, \mathsf{aenc}(x_{nb}, \mathsf{pk}(x_b)))$$

Example : Needham-Schroeder-Lowe

The honest role of B - P side

$$P_B^H =$$

$$!\,\mathsf{new}\ c.\ \mathsf{out}(c_{BH}, c).$$

$$\mathsf{in}(c, < \mathsf{hon}(z_a), \mathsf{hon}(z_b) >).$$

$$\mathsf{in}(c, \mathsf{aenc}(< z_{na}, z_a >, \mathsf{pk}(z_b))).$$

$$\mathsf{new}\ n_b.$$

$$\mathsf{out}(c, \mathsf{aenc}(< z_{na}, n_b, z_b >, \mathsf{pk}(z_a))).$$

$$\mathsf{in}(c, \mathsf{aenc}(n_b, \mathsf{pk}(z_b))).$$

$$\mathsf{out}(c, \mathsf{senc}(m_1, n_b))$$

Example : Needham-Schroeder-Lowe

The honest role of B - P side

$$P_B^H =$$

$$!\,\mathsf{new}\ c.\ \mathsf{out}(c_{BH}, c).$$

$$\mathsf{in}(c, < \mathsf{hon}(z_a), \mathsf{hon}(z_b) >).$$

$$\mathsf{in}(c, \mathsf{aenc}(< z_{na}, z_a >, \mathsf{pk}(z_b))).$$

$$\mathsf{new}\ n_b.$$

$$\mathsf{out}(c, \mathsf{aenc}(< z_{na}, n_b, z_b >, \mathsf{pk}(z_a))).$$

$$\mathsf{in}(c, \mathsf{aenc}(n_b, \mathsf{pk}(z_b))).$$

$$\mathsf{out}(c, \mathsf{senc}(m_1, n_b))$$

Example : Needham-Schroeder-Lowe

The honest role of B - Q side

$$
\begin{aligned}
&Q_B^H = \\
&!\,\mathsf{new}\ c.\,\mathsf{out}(c_{BH}, c). \\
&\mathsf{in}(c, < \mathsf{hon}(z_a), \mathsf{hon}(z_b) >). \\
&\mathsf{in}(c, \mathsf{aenc}(< z_{na}, z_a >, \mathsf{pk}(z_b))). \\
&\mathsf{new}\ n_b. \\
&\mathsf{out}(c, \mathsf{aenc}(< z_{na}, n_b, z_b >, \mathsf{pk}(z_a))). \\
&\mathsf{in}(c, \mathsf{aenc}(z_{nb}, \mathsf{pk}(z_b))). \\
&\mathsf{new}\ k.\,\mathsf{out}(c, \mathsf{senc}(m_2, k))
\end{aligned}
$$

Example : Needham-Schroeder-Lowe

The honest role of B - Q side

$$Q_B^H =$$

$$!\,\mathsf{new}\ c.\,\mathsf{out}(c_{BH}, c).$$

$$\mathsf{in}(c, < \mathsf{hon}(z_a), \mathsf{hon}(z_b) >).$$

$$\mathsf{in}(c, \mathsf{aenc}(< z_{na}, z_a >, \mathsf{pk}(z_b))).$$

$$\mathsf{new}\ n_b.$$

$$\mathsf{out}(c, \mathsf{aenc}(< z_{na}, n_b, z_b >, \mathsf{pk}(z_a))).$$

$$\mathsf{in}(c, \mathsf{aenc}(z_{nb}, \mathsf{pk}(z_b))).$$

$$\mathsf{new}\ k.\,\mathsf{out}(c, \mathsf{senc}(m_2, k))$$

Example : Needham-Schroeder-Lowe

The dishonest role of B

$$P_B^D = Q_B^D =$$

$$!\,\mathsf{new}\ c.\ \mathsf{out}(c_{BD}, c).$$

$$\mathsf{in}(c, < \mathsf{pk}(y_a), \mathsf{pk}(y_b) >).$$

$$\mathsf{in}(c, \mathsf{aenc}(< y_{na}, y_a >, \mathsf{pk}(y_b))).$$

$$\mathsf{new}\ n_b.$$

$$\mathsf{out}(c, \mathsf{aenc}(< y_{na}, n_b, y_b >, \mathsf{pk}(y_a))).$$

$$\mathsf{in}(c, \mathsf{aenc}(y_{nb}, \mathsf{pk}(y_b)))$$

We get :

$$P = P_{Header} | P_A | P_B^D | P_B^H \text{ and } Q = P_{Header} | Q_A | Q_B^D | Q_B^H$$

Example : Needham-Schroeder-Lowe

The dishonest role of B

$$
\begin{aligned}
& P_B^D = Q_B^D = \\
& !\,\mathsf{new}\ c.\ \mathsf{out}(c_{BD}, c). \\
& \mathsf{in}(c, < \mathsf{pk}(y_a), \mathsf{pk}(y_b) >). \\
& \mathsf{in}(c, \mathsf{aenc}(< y_{na}, y_a >, \mathsf{pk}(y_b))). \\
& \mathsf{new}\ n_b. \\
& \mathsf{out}(c, \mathsf{aenc}(< y_{na}, n_b, y_b >, \mathsf{pk}(y_a))). \\
& \mathsf{in}(c, \mathsf{aenc}(y_{nb}, \mathsf{pk}(y_b)))
\end{aligned}
$$

We get :

$$P = P_{Header} | P_A | P_B^D | P_B^H \text{ and } Q = P_{Header} | Q_A | Q_B^D | Q_B^H$$

Rémy's Transformation

The transformation

Let P be a simple protocol. Let N be a set of nonces.

- We denote by $\overline{P}^N$ the protocol P where we have removed every instruction new n with $n \in N$.
- We denote by $B(c)$ the process occuring in P on channel c, and by $B^\star(c^\star)$ the process obtained from $B(c)$ renaming each new n into new $n^\star$.

Definition

$$\overline{P}^{N,c} = \overline{P}^N | B^\star_{c^\star}$$

Rémy's Transformation

The transformation

Let P be a simple protocol. Let N be a set of nonces.

- We denote by $\overline{P}^N$ the protocol P where we have removed every instruction new n with $n \in N$.
- We denote by $B(c)$ the process occuring in P on channel c, and by $B^\star(c^\star)$ the process obtained from $B(c)$ renaming each new n into new $n^\star$.

Definition

$$\overline{P}^{N,c} = \overline{P}^N | B^\star_{c^\star}$$

Rémy's Transformation

The transformation

Let P be a simple protocol. Let N be a set of nonces.

- We denote by $\overline{P}^N$ the protocol P where we have removed every instruction new n with $n \in N$.
- We denote by $B(c)$ the process occuring in P on channel c, and by $B^\star(c^\star)$ the process obtained from $B(c)$ renaming each new n into new $n^\star$.

Definition

$$\overline{P}^{N,c} = \overline{P}^N | B^\star_{c^\star}$$

Rémy's Transformation

The transformation

Let P be a simple protocol. Let N be a set of nonces.

- We denote by $\overline{P}^N$ the protocol P where we have removed every instruction new n with $n \in N$.
- We denote by $B(c)$ the process occuring in P on channel c, and by $B^\star(c^\star)$ the process obtained from $B(c)$ renaming each new n into new $n^\star$.

Definition

$$\overline{P}^{N,c} = \overline{P}^N | B^\star_{c^\star}$$

Rémy's Transformation

Example : the header

$P_{Header} =$

$!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_H, c).\,\mathsf{new}\ ag_H\ .\,\mathsf{out}(c, < ag_H, \mathsf{pk}(ag_H), \mathsf{hon}(ag_H) >)$

$|!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_D, c).\,\mathsf{new}\ ag_D\ .\,\mathsf{out}(c, < ag_D, \mathsf{pk}(ag_D), \mathsf{dis}(ag_D), \mathsf{sk}(ag_D) >$

$|K$

We take $N = \{ag_H, ag_D\}$ and $c = c_H$.

Rémy's Transformation

Example : the header

$P_{Header} =$

$!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_H, c).\,\mathsf{new}\ ag_H\ .\,\mathsf{out}(c, < ag_H, \mathsf{pk}(ag_H), \mathsf{hon}(ag_H) >)$

$|!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_D, c).\,\mathsf{new}\ ag_D\ .\,\mathsf{out}(c, < ag_D, \mathsf{pk}(ag_D), \mathsf{dis}(ag_D), \mathsf{sk}(ag_D) >$

$|K$

We take $N = \{ag_H, ag_D\}$ and $c = c_H$.

Rémy's Transformation

Example

Example

$$\overline{P}^{N,c_H}_{Header} =$$

$$!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_H, c)\ .\,\mathsf{out}(c, < ag_H, \mathsf{pk}(ag_H), \mathsf{hon}(ag_H) >)$$

$$|\,\mathsf{new}\ ag_H^\star\ .\,\mathsf{out}(c_H^\star, < ag_H^\star, \mathsf{pk}(ag_H^\star), \mathsf{hon}(ag_H^\star) >)$$

$$|!\,\mathsf{new}\ c\ .\,\mathsf{out}(c_D, c)\ .\,\mathsf{out}(c, < ag_D, \mathsf{pk}(ag_D), \mathsf{dis}(ag_D), \mathsf{sk}(ag_D) >)$$

$$|K$$

Rémy's Theorem

Given a process P, we denote by $Ch(P)$ the set of public channels names occuring under a replication in P.

Theorem (Rémy)

Let P and Q be two simple protocols such that $Ch(P) = Ch(Q)$, and N be a set of names. We have that :

$$[\forall c \in Ch(P).\overline{P}^{N,c} \approx \overline{Q}^{N,c}] \Rightarrow P \approx Q$$

Rémy's Theorem

Given a process P, we denote by $Ch(P)$ the set of public channels names occuring under a replication in P.

Theorem (Rémy)

Let P and Q be two simple protocols such that $Ch(P) = Ch(Q)$, and N be a set of names. We have that :

$$[\forall c \in Ch(P).\overline{P}^{N,c} \approx \overline{Q}^{N,c}] \Rightarrow P \approx Q$$

From nonces to agents

- We apply this theorem on our model with $N = \{ag_H, ag_D\}$.
- It says that if there is an attack, then there is an attack with 3 agents.
- We need 4 agents (two honests and two dishonests)
- Agents are allowed to talk to themselves.
- The other limits are those of Rémy's theorem.

From nonces to agents

- We apply this theorem on our model with $N = \{ag_H, ag_D\}$.
- It says that if there is an attack, then there is an attack with 3 agents.
- We need 4 agents (two honests and two dishonests)
- Agents are allowed to talk to themselves.
- The other limits are those of Rémy's theorem.

From nonces to agents

- We apply this theorem on our model with $N = \{ag_H, ag_D\}$.
- It says that if there is an attack, then there is an attack with 3 agents.
- We need 4 agents (two honests and two dishonests)
- Agents are allowed to talk to themselves.
- The other limits are those of Rémy's theorem.

From nonces to agents

- We apply this theorem on our model with $N = \{ag_H, ag_D\}$.
- It says that if there is an attack, then there is an attack with 3 agents.
- We need 4 agents (two honests and two dishonests)
- Agents are allowed to talk to themselves.
- The other limits are those of Rémy's theorem.

From nonces to agents

- We apply this theorem on our model with $N = \{ag_H, ag_D\}$.
- It says that if there is an attack, then there is an attack with 3 agents.
- We need 4 agents (two honests and two dishonests)
- Agents are allowed to talk to themselves.
- The other limits are those of Rémy's theorem.

Table of contents

New result :

- Not only for simple protocols.
- More conditionals (disjonctions allowed).
- Errors outputed when tests fail.

New result :

- Not only for simple protocols.
- More conditionals (disjonctions allowed).
- Errors outputed when tests fail.

New result :

- Not only for simple protocols.
- More conditionals (disjonctions allowed).
- Errors outputed when tests fail.

Terms

Let $\Sigma = \Sigma_c \cup \Sigma_d$ be a signature.
Let $\mathcal{X}, \mathcal{W}$ be two sets of variables, and $\mathcal{N}$ be a set of names.

Messages

Let $\mathcal{M}_\Sigma$ be a set of ground constructor terms that stable by renaming (that is $\mathcal{M}_\Sigma \rho \subset \mathcal{M}_\Sigma$ for any renaming ρ). We call messages the element of $\mathcal{M}_\Sigma$.

Constructors, destructors

The set of rewriting rules $\mathcal{R}$ is a set of rules of the form $d(c_1, \ldots, c_n)$ where $d \in \Sigma_d$ and $c_1, \ldots, c_n \in \mathcal{T}(\Sigma_c, \mathcal{X})$.

Terms

Let $\Sigma = \Sigma_c \cup \Sigma_d$ be a signature.
Let $\mathcal{X}, \mathcal{W}$ be two sets of variables, and $\mathcal{N}$ be a set of names.

Messages

Let $\mathcal{M}_\Sigma$ be a set of ground constructor terms that stable by renaming (that is $\mathcal{M}_\Sigma\rho \subset \mathcal{M}_\Sigma$ for any renaming ρ). We call messages the element of $\mathcal{M}_\Sigma$.

Constructors, destructors

The set of rewriting rules $\mathcal{R}$ is a set of rules of the form $d(c_1, \ldots, c_n)$ where $d \in \Sigma_d$ and $c_1, \ldots, c_n \in \mathcal{T}(\Sigma_c, \mathcal{X})$.

Terms

Let $\Sigma = \Sigma_c \cup \Sigma_d$ be a signature.
Let $\mathcal{X}, \mathcal{W}$ be two sets of variables, and $\mathcal{N}$ be a set of names.

Messages

Let $\mathcal{M}_\Sigma$ be a set of ground constructor terms that stable by renaming (that is $\mathcal{M}_\Sigma\rho \subset \mathcal{M}_\Sigma$ for any renaming ρ). We call messages the element of $\mathcal{M}_\Sigma$.

Constructors, destructors

The set of rewriting rules $\mathcal{R}$ is a set of rules of the form $d(c_1, \ldots, c_n)$ where $d \in \Sigma_d$ and $c_1, \ldots, c_n \in \mathcal{T}(\Sigma_c, \mathcal{X})$.

Syntax

Let Ch_0 and Ch^{fresh} be two disjoint sets of public channel names.

Processes

Let Ch be an infinite set of channels. We consider processes built using the following grammar :

$$P, Q := 0 \mid !^{c}_{c'}P \mid (P|Q) \mid \text{ new } n.P \mid \text{ in}(c, u).P$$
$$\mid \text{ out}(c, u).P \mid \text{ let } y = v \text{ in } P \text{ else } E \mid \text{if } cond \text{ then } P \text{ else } E$$

where $u \in \mathcal{T}(\Sigma_c, \mathcal{N} \cup \mathcal{X})$,$v \in \mathcal{T}(\Sigma, \mathcal{N} \cup \mathcal{X})$, $n \in \mathcal{N}$, $e \in \mathcal{E}$, $\mathcal{E}$ disjoint from $\mathcal{C}$ and Σ_0 and $c, c' \in Ch$, and :

$$E = 0 \mid \text{out}(c, e)$$

Semantics
Configurations

Configuration

A configuration is a pair $(\mathcal{P}, \phi)$ where $\mathcal{P}$ is a multiset of ground processes and ϕ is a substitution $\phi = \{w_1 \triangleright m_1; \ldots; w_n \triangleright m_n\}$ where $w_1, \ldots, w_n \in \mathcal{W}$ and $m_1, \ldots, m_n$ are messages, that is terms in $\mathcal{M}_\Sigma$.

Sometimes, we may denote $\{P\}$ by P and $(\mathcal{P}, \emptyset)$ by $\mathcal{P}$.

Semantics (1)

The semantics is given by the following rules :

- $(0 \cup \mathcal{P}, \phi) \rightarrow^{\tau} (\mathcal{P}, \phi)$
- $(!^{c}_{c'} P \cup \mathcal{P}, \phi) \rightarrow^{\mathsf{sess}(c, ch_i)} (P\{^{ch_i}/_{c'}\} \cup !^{c}_{c'} P \cup \mathcal{P}, \phi)$ with ch_i a new fresh channel name.
- $((P|Q) \cup \mathcal{P}, \phi) \rightarrow^{\tau} (P \cup Q \cup \mathcal{P}, \phi)$
- $(\mathsf{new}\, n.P, \phi) \rightarrow^{\tau} (P\{^{n'}/_{n}\} \cup \mathcal{P}, \phi)$ with n' fresh.
- $(\mathsf{in}(c, u).P, \phi) \rightarrow^{\mathsf{in}(c,R)} (P\sigma \cup \mathcal{P}, \phi)$ when $R\phi\downarrow$ and u are unifiable and where σ is the most general unifier of $R\phi\downarrow$ and u.
- $(\mathsf{out}(c, u).P, \phi) \rightarrow^{\mathsf{out}(c,w)} (P \cup \mathcal{P}, \phi \cup \{w \ \triangleright \ u\phi\downarrow\})$

Semantics (2)

- $(\text{let } x = v \text{ in } P \text{ else } E \cup \mathcal{P}, \phi) \rightarrow^{\tau} (P\{^{v\downarrow}/_{x}\} \cup \mathcal{P}, \phi)$ when $v \downarrow \in \mathcal{M}_{\Sigma}$.
- $(\text{let } x = v \text{ in } P \text{ else } E \cup \mathcal{P}, \phi) \rightarrow^{\tau} (E \cup \mathcal{P}, \phi)$ when $v \downarrow \notin \mathcal{M}_{\Sigma}$.
- $(\text{if } t \text{ then } P \text{ else } E \cup \mathcal{P}, \phi) \rightarrow^{\tau} (P \cup \mathcal{P}, \phi)$ where $t = b_1 \vee \cdots \vee b_n$ evaluates as true.
- $(\text{if } t \text{ then } P \text{ else } E \cup \mathcal{P}, \phi) \rightarrow^{\tau.\text{out}(c,e)} (\mathcal{P}, \phi)$ where $t = b_1 \vee \cdots \vee b_n$ evaluates as false.

Simple protocols

Basic process

Definition

Let B be a process and c a channel name. We say that B is a basic process built on c if it is written in the following grammar :

$$B := 0 \mid \mathsf{in}(c, u).B \mid \mathsf{out}(c, u).B \mid \mathsf{let}\, x = v \,\mathsf{in}\, B \,\mathsf{else}\, E$$

$$\mid \mathsf{if}\, t \,\mathsf{then}\, P \;\mathsf{else}\, E \mid \mathsf{new}\, n.B$$

Simple protocols

Definition

Let P be a protocol. We say that P is simple if :

$$P = !\,\mathsf{new}\, c'_1.\,\mathsf{out}(c_1, c'_1).B_1 \mid \ldots \mid !\,\mathsf{new}\, c'_m.\,\mathsf{out}(c_m, c'_m).B_m$$

$$\mid B_{m+1} \mid \ldots \mid B_p$$

where each B_i for $1 \leq i \leq m$ is a basic process built on channel c'_i, and each B_i for $m+1 \leq i \leq p$ is a basic process built on channels c_i.

Hypothesis

Definition (Action-Determinism)

Let P be a protocol. We say that P is action-deterministic if for every trace tr such that $P \Rightarrow^{\mathsf{tr}} (\mathcal{P}, \phi)$, if there are $Q_1, Q_2 \in \mathcal{P}$, when the visible actions act, act' pass respectively in Q_1 and Q_2 then act $\neq$ act', or they occur on channels $c \neq c'$.

Definition (Adequate theories)

Let $\Sigma = \Sigma_c \cup \Sigma_d$ be a signature, $\mathcal{R}$ be a convergent rewriting system and $\mathcal{M}_\Sigma$ be a set of messages. We say that the theory $\mathcal{M}_\Sigma$ is adequate w.r.t. $\mathcal{M}_\Sigma$ when for any term in normal form, there exist n_1, n_2 such that for any renaming ρ, $n_1\rho \neq n_2\rho$ implies $t\rho \downarrow \notin \mathcal{M}_\Sigma$.

Hypothesis

Definition (Action-Determinism)

Let P be a protocol. We say that P is action-deterministic if for every trace tr such that $P \Rightarrow^{\mathsf{tr}} (\mathcal{P}, \phi)$, if there are $Q_1, Q_2 \in \mathcal{P}$, when the visible actions act, act' pass respectively in Q_1 and Q_2 then act $\neq$ act', or they occur on channels $c \neq c'$.

Definition (Adequate theories)

Let $\Sigma = \Sigma_c \cup \Sigma_d$ be a signature, $\mathcal{R}$ be a convergent rewriting system and $\mathcal{M}_\Sigma$ be a set of messages. We say that the theory $\mathcal{M}_\Sigma$ is adequate w.r.t. $\mathcal{M}_\Sigma$ when for any term in normal form, there exist n_1, n_2 such that for any renaming ρ, $n_1\rho \neq n_2\rho$ implies $t\rho \downarrow \notin \mathcal{M}_\Sigma$.

New transformation

Let P be a protocol in the form :

$$P = P_S | P_G$$

where P_S is simple.

Definition

Let $\mathcal{C} = (c_1, \ldots, c_k)$ be a multiset of channels of P. We define $\overline{P}^{N,\mathcal{C}}$ as :

$$\overline{P}^{N,\mathcal{C}} = \overline{P}^N | B_1^\star(c_1^\star) | \ldots | B_k^\star(c_k^\star)$$

where B_i is the basic process corresponding to the channel c_i in P_S.

Result

Let P be a protocol. We denote by k_P the maximal size of a disjonction occuring in P.

Theorem

Let P and Q be two action-deterministic protocols. If there an attack against $P_{Header}|P \approx P_{Header}|Q$, then there is an attack against $P|\overline{P}^{N,\mathcal{C}}_{Header} \approx Q|\overline{P}^{N,\mathcal{C}}_{Header}$ for some multiset $\mathcal{C}$ built on $\{c_H, c_D\}$ of size less than $k_P + k_Q$.

Result

Let P be a protocol. We denote by k_P the maximal size of a disjonction occuring in P.

Theorem

Let P and Q be two action-deterministic protocols. If there an attack against $P_{Header}|P \approx P_{Header}|Q$, then there is an attack against $P|\overline{P}^{N,\mathcal{C}}_{Header} \approx Q|\overline{P}^{N,\mathcal{C}}_{Header}$ for some multiset $\mathcal{C}$ built on $\{c_H, c_D\}$ of size less than $k_P + k_Q$.

Table of contents

How to build a counter-example ?

- Take a protocol such that there is an attack iff an instance of PCP has a solution.
- The adversary is allowed to add a tile iff he gives a nonce (or an agent name).
- These names are stocked in a list.
- Find a way to do that the attack is possible iff the elements of the lists are pairwise distinct.

How to build a counter-example ?

- Take a protocol such that there is an attack iff an instance of PCP has a solution.
- The adversary is allowed to add a tile iff he gives a nonce (or an agent name).
- These names are stocked in a list.
- Find a way to do that the attack is possible iff the elements of the lists are pairwise distinct.

How to build a counter-example ?

- Take a protocol such that there is an attack iff an instance of PCP has a solution.
- The adversary is allowed to add a tile iff he gives a nonce (or an agent name).
- These names are stocked in a list.
- Find a way to do that the attack is possible iff the elements of the lists are pairwise distinct.

How to build a counter-example ?

- Take a protocol such that there is an attack iff an instance of PCP has a solution.
- The adversary is allowed to add a tile iff he gives a nonce (or an agent name).
- These names are stocked in a list.
- Find a way to do that the attack is possible iff the elements of the lists are pairwise distinct.

Forbid the agents to speak to themselves.

Initialisation

We assume that the protocol is never played with $A = B$.

$$B \to S : A, B$$
$$S \to B : \mathrm{enc}(< A, B >, K_{diff}),$$
$$\mathrm{enc}(<< u_{init}, v_{init} >, < B, end >>, K_{PCP})$$
$$B \to S : \mathrm{enc}(<< x, y >, z_\ell >, K_{PCP})$$

For a honest execution, the only possibility would be for B to forward the second message, but it is not checked by the server.

Forbid the agents to speak to themselves.

Initialisation

We assume that the protocol is never played with $A = B$.

$$
\begin{aligned}
&B \rightarrow S : A, B \\
&S \rightarrow B : \mathrm{enc}(< A, B >, K_{diff}), \\
&\mathrm{enc}(<< u_{init}, v_{init} >, < B, end >>, K_{PCP}) \\
&B \rightarrow S : \mathrm{enc}(<< x, y >, z_\ell >, K_{PCP})
\end{aligned}
$$

For a honest execution, the only possibility would be for B to forward the second message, but it is not checked by the server.

Forbid the agents to speak to themselves.

Initialisation

We assume that the protocol is never played with $A = B$.

$$
\begin{aligned}
&B \rightarrow S : A, B \\
&S \rightarrow B : \mathrm{enc}(< A, B >, K_{diff}), \\
&\mathrm{enc}(<< u_{init}, v_{init} >, < B, end >>, K_{PCP}) \\
&B \rightarrow S : \mathrm{enc}(<< x, y >, z_{\ell} >, K_{PCP})
\end{aligned}
$$

For a honest execution, the only possibility would be for B to forward the second message, but it is not checked by the server.

Forbid the agents to speak to themselves.

Add a tile.

Recall the last step :

$$
\begin{aligned}
&B \to S : \mathrm{enc}(<< x, y >, z_\ell >, K_{PCP}) \\
&S \to A : \\
&\mathrm{enc}(<< xu_1, yv_1 >, < A, z_\ell >>, K_{PCP}), \\
&\ldots, \\
&\mathrm{enc}(<< xu_n, yv_n >, < A, z_\ell >>, K_{PCP})
\end{aligned}
$$

Forbid the agents to speak to themselves.

Get the encrypted secret.

$$
\begin{aligned}
& S \rightarrow A : \\
& \mathrm{enc}(\mathrm{enc}(SECRET, \mathrm{enc}(< A, z_\ell >, K_{approved})), \\
& \mathrm{enc}(<< xu_1, xu_1 >, < A, z_\ell >>, K_{PCP})), \\
& \ldots, \\
& \mathrm{enc}(\mathrm{enc}(SECRET, \mathrm{enc}(< A, z_\ell >, K_{approved})), \\
& \mathrm{enc}(<< xu_n, xu_n >, < A, z_\ell >>, K_{PCP}))
\end{aligned}
$$

Forbid the agents to speak to themselves.

Check the list

$$
\begin{aligned}
&? \rightarrow S :< A, end > \\
&S \rightarrow ? : \mathrm{enc}(< A, end >, K_{appproved}) \\
&? \rightarrow S : \mathrm{enc}(< x', z'_\ell >, K_{approved}), \\
&\mathrm{enc}(< y', z'_\ell >, K_{approved}), \mathrm{enc}(< x', y' >, K_{diff}) \\
&S \rightarrow ? : \mathrm{enc}(< x', < y', z'_\ell >>, K_{approved})
\end{aligned}
$$

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Forbid the agents to speak to themselves.

Sketch of proof

- We have to get $SECRET$.
- It is encrypted twice : $\text{enc}(\text{enc}(SECRET, K_2), K_1)$.
- We can obtain the first key iff we have a solution of PCP : $K_1 = \text{enc}(<< x, x >, z_\ell >, K_{PCP})$.
- We can obtain the second key iff we have the corresponding list of identities encrypted by $K_{approved}$: $K_2 = \text{enc}(z_\ell, K_{approved})$
- This can be done iff the elements of the list are pairwise distinct.
- The length of the list is not allways calculable.

Other negative results.

- A fortiori : else branches.
- Model with XOR and pair (or only dec, enc with explicit destructors).
- Remove the action-determinism.

Other negative results.

- A fortiori : else branches.
- Model with XOR and pair (or only dec, enc with explicit destructors).
- Remove the action-determinism.

Other negative results.

- A fortiori : else branches.
- Model with XOR and pair (or only dec, enc with explicit destructors).
- Remove the action-determinism.

Model with XOR and pair

- There are two trees that are equals whenever the elements of a list are not pairwise distinct.
- These trees can be built inductively from the list.
- So we can get almost the same counter-example.

Model with XOR and pair

- There are two trees that are equals whenever the elements of a list are not pairwise distinct.
- These trees can be built inductively from the list.
- So we can get almost the same counter-example.

Model with XOR and pair

- There are two trees that are equals whenever the elements of a list are not pairwise distinct.
- These trees can be built inductively from the list.
- So we can get almost the same counter-example.

Remove the action-determinism

- It is also impossible to remove this hypothesis.
- Problem : A lot of different frames
- It is impossible to bound their number, even between equivalent frames.

Remove the action-determinism

- It is also impossible to remove this hypothesis.
- Problem : A lot of different frames
- It is impossible to bound their number, even between equivalent frames.

Remove the action-determinism

- It is also impossible to remove this hypothesis.
- Problem : A lot of different frames
- It is impossible to bound their number, even between equivalent frames.

Conclusion

- Rémy's theorem give a solution for simple protocols without else branch.
- We can add error branches, tests of disjonctions of equalities, and exchange simple for action-deterministic.
- They are counter-examples for some extensions : else branches, determinacy and model with XOR and pair.

Conclusion

- Rémy's theorem give a solution for simple protocols without else branch.
- We can add error branches, tests of disjonctions of equalities, and exchange simple for action-deterministic.
- They are counter-examples for some extensions : else branches, determinacy and model with XOR and pair.

Conclusion

- Rémy's theorem give a solution for simple protocols without else branch.
- We can add error branches, tests of disjonctions of equalities, and exchange simple for action-deterministic.
- They are counter-examples for some extensions : else branches, determinacy and model with XOR and pair.