Proof of security protocols

Bruno Blanchet
Vincent Cheval
Hubert Comon (comon@lsv.fr)
Adrien Koutsos

2021 – 2022

Table des matières

Première partie

Vérification symbolique

1 Introduction

The goals of the course are :
— Understanding what is a security protocol and what is a security property.
— Learning new proof techniques.
— And depending on the students, more cryptography, more proofs..
Cryptography proocols are complete protocols like https, TLS, AKA, Signal, Smart Cards, RFID...
These are distributed programs relying on crypto libraries (like RSA).

Exemple 1.1 **Crypto protocol**

A sends something to B : $\{secret\}_{\mathsf{pk}_B}$.
P_A : $\nu secret\,.\,\mathsf{out}(\{secret\}_{\mathsf{pk}_B})$
P_B : $\mathsf{in}(x)\,.\,\mathsf{let}\ s = \mathsf{dec}(x, sh_B)\ \mathsf{in}\ \ldots$
We want to prove the property φ. We run the two programs and check if they satisfy the property : $P_A \parallel P_B \stackrel{?}{\models} \varphi$.
The real question is when there is an attacker $\mathcal{A}$, is there $\mathcal{A} \parallel P_A \parallel P_B \stackrel{?}{\models} \varphi$.

So the question is the following : what is an attacker ?

Définition 1.1 Dolev Yao attacker (1983)

A Dolev Yao attacker is defined by a set of deduction rules.

Exemple 1.2 Dolev Yao attacker

$$\frac{\{x\}_{\mathsf{pk}} \qquad \mathsf{sk}}{x} \qquad\qquad \frac{x \qquad \mathsf{pk}}{\{x\}_{\mathsf{pk}}}$$

Définition 1.2 Computational attacker (Goldwasser Micali 1982)

A computational attacker is a probabilistic polytime Turing machine.
There is a security parameter η, which is the size of the numbers implied.

The security properties are not defined the same way for these attackers.

1.1 Security protocols syntax and semantics

The Needham-Schroeden protocol was invented in 1978 and attacked in 1995.

Définition 1.3 Needham-Schoeden protocol

The Needham-Schoeden protocols is the following :

$$\begin{aligned} A &\to B : \nu n_A \,.\, \{\langle A, n_A\rangle\}_{\mathsf{pk}_B} \\ B &\to A : \nu n_B \,.\, \{\langle n_A, n_B\rangle\}_{\mathsf{pk}_A} \\ A &\to B : \{n_B\}_{\mathsf{pk}_B} \end{aligned}$$

The attack is simple. C is controlled by the attacker. It is the following :

$$\begin{aligned} A &\to C : \nu n_A \,.\, \{\langle A, n_A\rangle\}_{\mathsf{pk}_C} \\ C^{(A)} &\to B : \{\langle A, n_A\rangle\}_{\mathsf{pk}_B} \\ B &\to A : \nu n_B \,.\, \{\langle n_A, n_B\rangle\}_{\mathsf{pk}_A} \\ A &\to C : \{n_B\}_{\mathsf{pk}_C} \\ C^{(A)} &\to B : \{n_B\}_{\mathsf{pk}_B} \end{aligned}$$

Exercice 1.1

Find an attack on secrecy of k.

$$\begin{aligned} A &\to B : \nu k \,.\, \langle A, \{k\}_{\mathsf{pk}_B}\rangle \\ B &\to A : \langle B, \{k\}_{\mathsf{pk}_A}\rangle \end{aligned}$$

Preuve du exercice 1.1.

$$\begin{aligned} A &\to B : \nu k \,.\, \langle A, \{k\}_{\mathsf{pk}_B}\rangle \\ C &\to B : \langle C, \{k\}_{\mathsf{pk}_B}\rangle \\ B &\to C : \langle B, \{k\}_{\mathsf{pk}_C}\rangle \\ C^{(B)} &\to A : \langle C, \{k\}_{\mathsf{pk}_A}\rangle \end{aligned}$$

Et là c'est le plus beau jour de ma vie.

Exercice 1.2

Assume there is no attack on secrecy of k.
Prove that A and B cannot safely use k as an encryption key.

$$\begin{aligned} A &\to B : \nu k \,.\, \{\langle A, k\rangle\}_{\mathsf{pk}_B} \\ B &\to A : \{\langle B, k\rangle\}_{\mathsf{pk}_A} \end{aligned}$$

Preuve du exercice 1.2.

$$A \to C : \nu k \,.\, \{\langle A, k\rangle\}_{\mathsf{pk}_C}$$
$$C^{(A)} \to B : \{\langle A, k\rangle\}_{\mathsf{pk}_B}$$
$$B \to A : \{\langle B, k\rangle\}_{\mathsf{pk}_A}$$

With this attack, A knows that the key is not safe because it has communicated with C. But B doesn't know it so it can try to use it in a protocol $B \to A : \nu s \,.\, \{s\}_k$.

Youpi !

Exercice 1.3

Find an attack on the secrecy of k. (more difficult)

$$A \to B : \nu k \,.\, \{\langle A, \{k\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_B}$$
$$B \to A : \{\langle B, \{k\}_{\mathsf{pk}_A}\rangle\}_{\mathsf{pk}_A}$$

Preuve du exercice 1.3.

$$A \to B : \nu k \,.\, \{\langle A, \{k\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_B}$$
$$C \to B : \{\langle C, \{\langle A, \{k\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_B}$$
$$B \to C : \{\langle B, \{\langle A, \{k\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_C}\rangle\}_{\mathsf{pk}_C}$$
$$C \to B : \{\langle C, \{k\}_{\mathsf{pk}_B}\rangle\}_{\mathsf{pk}_B}$$
$$B \to C : \{\langle C, \{k\}_{\mathsf{pk}_C}\rangle\}_{\mathsf{pk}_C}$$
$$C \to A : \{\langle B, \{k\}_{\mathsf{pk}_A}\rangle\}_{\mathsf{pk}_A}$$

cskifo

1.2 Messages and terms

There is a distinction between terms and messages.

Définition 1.4 Terms

The terms are the variables of $\mathcal{X}$, the names of $\mathcal{N}$ and the function symbols $\mathcal{F}$.

The function symbols are
- $\{\_\}_{\_}$
- $\mathsf{dec}(\_,\_)$ (such that $\mathsf{dec}(\{x\}^z_{\mathsf{pk}(y)}, y) \to x$)
- $\langle\_,\_\rangle$, $\pi_1(\_)$ (such that $\pi_1(\langle x, y\rangle) \to x$)
- $\pi_2(\_)$ (such that $\pi_2(\langle x, y\rangle) \to y$)
- $\mathsf{pk}(\_)$

Définition 1.5 Messages

The messages can be errors or terms in normal form in that variable.

2 Semantics of messages and terms

For the semantics of messages and terms, consider a model $\mathcal{M}$ and σ an assignment $\mathcal{X} \to \mathcal{M}$.

Notation 2.1 Interpretation

$[\![u]\!]^{\mathcal{M}}_{\sigma}$ is the interpretation of the term u in the environment σ and the model $\mathcal{M}$.

In the Dolev Yao model, the domain is the set of messages and errors. $[\![u]\!]^{\mathsf{DY}}_{\sigma} = u\sigma\downarrow$

Exemple 2.1

$$\left[\!\left[\{\langle\pi_2(\mathsf{dec}(x,\mathsf{sk}_y)),u_B\rangle\}^{M_B}_{\pi_1(\mathsf{dec}(x,\mathsf{sk}_B))}\right]\!\right]^{\mathsf{DY}}_{\{x\mapsto\{\langle\mathsf{pk}(\mathsf{sk}_A),n_a\rangle\}^{M_A}_{\mathsf{pk}(\mathsf{sk}_B)}\}}$$

$$= \left\{ \langle\pi_2(\underbrace{\mathsf{dec}(\{\langle\mathsf{pk}(\mathsf{sk}_A),n_A\rangle\}^{M_A}_{\mathsf{pk}(\mathsf{sk}_B)},\mathsf{sk}_B)}_{\langle\mathsf{pk}(\mathsf{sk}_A),n_A\rangle}),n_B \right\}^{M_B}_{\pi_1(\underbrace{\mathsf{dec}(\{\langle\mathsf{pk}(\mathsf{sk}_A),n_A\rangle\}^{M_A}_{\mathsf{pk}(\mathsf{sk}_B)},\mathsf{sk}_B)}_{\langle\mathsf{pk}(\mathsf{sk}_A),n_A\rangle})}$$

$$=\{\langle n_A,n_B\rangle\}^{n_B}_{\mathsf{pk}(\mathsf{sk}_B)}$$

2.1 Computational model

Suppose there is a security parameter $\eta\in\mathbb{N}$.
We have
— Crypto libraries : for every $f\in\mathcal{F}$, $[\![f]\!]$ is a library $(\{0,1\}^*)^\eta\to\{0,1\}^*$
— The interpretation of names $\tau:\mathcal{N}\to\{0,1\}^\eta$
— The environment $\sigma:\mathcal{X}\to\{0,1\}^*$
Then $[\![u]\!]^{M_C(\tau,\eta)}_\sigma$ is defined as follows :

$$[\![x]\!]^{M_C(\tau,\eta)}_\sigma=\sigma(x)$$
$$[\![n]\!]^{M_C(\tau,\eta)}_\sigma=\tau(n)$$
$$[\![f(u_1,...,u_n)]\!]^{M_C(\tau,\eta)}_\sigma=[\![f]\!]\left([\![u_1]\!]^{M_C(\tau,\eta)}_\sigma,...,[\![u_n]\!]^{M_C(\tau,\eta)}_\sigma\right)$$

2.2 Processes syntax

Définition 2.1 Process

The process is defined by a grammar :

$P ::=$	0	
	$\mid \mathsf{in}(x)\,.\,P$	x is unbounded in P, this bounds x
	$\mid \mathsf{out}(t)\,.\,P$	t is a term
	$\mid \nu n\,.\,P$	n is a name, this bounds n
	$\mid P\parallel P$	
	$\mid$ if $u=v$ then P else P	u, v terms
	$\mid$ let $x=u$ in P	$x\in\mathcal{X}$, u term

We assume
— $\parallel$ is associative and commutative : $P\parallel Q\parallel R\equiv P\parallel R\parallel Q$
— $(\nu n\,.\,P)\parallel Q\equiv\nu n\,.\,(P\parallel Q)$, where n is not free in Q
— $\nu n\,.\,P\equiv\nu n'\,.\,P\{n\mapsto n'\}$

Définition 2.2 Protocol

A protocol is a process in which every variable is bound exactly once, and every name is bound at most once.

Exemple 2.2

The protocol is

$$A \to B : \nu n_A \nu r_A \,.\, \{\langle \mathsf{pk}_A, n_A\rangle\}^{r_A}_{\mathsf{pk}_B}$$
$$B \to A : \nu n_B \nu r_B \,.\, \{\langle n_A, n_B\rangle\}^{r_B}_{\mathsf{pk}_A}$$
$$A \to B : \nu r'_A \,.\, \{n_B\}^{r'_A}_{\mathsf{pk}_B}$$

The process of A is

$$\begin{aligned} P_A(\mathsf{sk}_A, \mathsf{pk}_B) :=& \nu n_A \nu r_A \,.\, \mathsf{out}(\{\langle \mathsf{pk}(\mathsf{sk}_A), n_A\rangle\}^{x_A}_{\mathsf{pk}_B}) \\ & \mathsf{in}(z) \,.\, \mathsf{let}\ z_{n_A} = \pi_1(\mathsf{dec}(z, \mathsf{sk}_A))\ \mathsf{in} \\ & \mathsf{let}\ z_{n_B} = \pi_2(\mathsf{dec}(z, \mathsf{sk}_A))\ \mathsf{in} \\ & \nu r'_A \,.\, \mathsf{if}\ z_{n_A} = n_A\ \mathsf{then}\ \mathsf{out}(\{z_{n_B}\}^{r'_A}_{\mathsf{pk}'_B})\ \mathsf{else}\ 0 \end{aligned}$$

The process of B is

$$\begin{aligned} P_B(\mathsf{sk}_B) :=& \nu n_B \nu r_B \,.\, \mathsf{in}(x) \\ & \mathsf{let}\ y_{n_A} = \pi_2(\mathsf{dec}(x, \mathsf{sk}_B))\ \mathsf{in} \\ & \mathsf{let}\ y_{\mathsf{pk}_A} = \pi_1(\mathsf{dec}(x, \mathsf{sk}_B))\ \mathsf{in} \\ & \mathsf{out}(\{\langle y_{n_A}, n_B\rangle\}^{n_B}_{y_{\mathsf{pk}_A}}) \\ & \mathsf{in}(x')\mathsf{let}\ x'_{n_B} = \mathsf{dec}(x', \mathsf{sk}_B)\ \mathsf{in} \\ & \mathsf{if}\ x'_{n_B} = n_B\ \mathsf{then}\ 0\ \mathsf{else}\ 0 \end{aligned}$$

$$\nu \mathsf{sk}_A \nu \mathsf{sk}_B \,.\, (P_A(, \mathsf{pk}(\mathsf{sk}_C)) \parallel P_B(\mathsf{sk}_B) \parallel \mathsf{out}(\mathsf{pk}(\mathsf{sk}_A)) \,.\, \mathsf{out}(\mathsf{pk}(\mathsf{sk}_B)) \,.\, 0)$$

2.3 Operational semantics

We assume that the attacker controls the network.

Définition 2.3 **Configuration**

The configuration is made of three components (φ, σ, P) :

- The frame φ, which is the sequence of messages that have been sent on the network. It is of the form $\nu \overline{n} \,.\, \underbrace{t_1 \ldots t_m}_{\text{terms built on } \mathcal{X}, \mathcal{F}, \mathcal{N}}$.
- The environment σ, which is an assignment $\mathcal{X} \to \mathcal{D}_{\mathcal{M}}$.
- The process P.

The initial configuration is $(\emptyset, \emptyset, P_0)$.

Définition 2.4 **Rules of the semantics**

$$\frac{}{(\varphi, \sigma, Q \parallel \mathsf{in}(x)\, P) \underset{\mathcal{A}}{\to} (\varphi, \sigma \uplus \{x \mapsto m\}, Q \parallel P)} = \varphi \rhd_{\mathcal{A}} m$$

$$\frac{}{(\nu \overline{n} \,.\, \theta, \sigma, Q \parallel \mathsf{out}(t)\, P) \to (\nu \overline{n} \,.\, \theta \uplus [\![t]\!]^{\mathcal{M}}_{\sigma}, \sigma, Q \parallel P)}$$

$$\frac{}{(\nu \overline{n} \,.\, \theta, \sigma, \nu m \,.\, P) \to (\nu \overline{n} \nu m \,.\, \theta, \sigma, P)}\ \text{if } m \notin \text{freename}(\theta)$$

$$\frac{(\varphi, \sigma, P) \to (\varphi', \sigma', P')}{(\varphi, \sigma, \mathsf{if}\ s = t\ \mathsf{then}\ P\ \mathsf{else}\ Q) \to (\varphi', \sigma', P')}\ \text{if } [\![s]\!]^{\mathcal{M}}_{\sigma} = [\![t]\!]^{\mathcal{M}}_{\sigma} \notin \text{Err}$$

$$\frac{(\varphi, \sigma, Q) \to (\varphi', \sigma', Q')}{(\varphi, \sigma, \mathsf{if}\ s = t\ \mathsf{then}\ P\ \mathsf{else}\ Q) \to (\varphi', \sigma', Q')}\ \text{if } [\![s]\!]^{\mathcal{M}}_{\sigma} \neq [\![t]\!]^{\mathcal{M}}_{\sigma} \text{ or } [\![s]\!]^{\mathcal{M}}_{\sigma} \in \text{Err}$$

$$\frac{}{(\varphi, \sigma, \mathsf{let}\ x = s\ \mathsf{in}\ P) \to (\varphi, \sigma \uplus \{x \mapsto [\![s]\!]^{\mathcal{M}}_{\sigma}\}, P)}\ \text{if } [\![s]\!]^{\mathcal{M}}_{\sigma} \notin \mathrm{Err}$$

$$\frac{(\varphi, \sigma, P) \to (\varphi', \sigma', P')}{(\varphi, \sigma, P \,\|\, Q) \to (\varphi', \sigma', P' \,\|\, Q)}$$

Exercice 2.1

Using the operational semantics, find an attack on secrecy of k.

$$A \to B : \nu k\,.\,\langle A, \{k\}_{\mathsf{pk}_B}\rangle$$
$$B \to A : \langle B, \{k\}_{\mathsf{pk}_A}\rangle$$

Définition 2.5 Dolev Yao interpretation

The Dolev Yao interpretation $\triangleright_A$ is defined by the following rules.

$$\frac{}{\nu\overline{n}\theta \triangleright_A n}\ \text{if } n \notin \overline{n}$$

$$\frac{}{\nu\overline{n}t_1 \ldots t_n \triangleright_A t_i}$$

$$\frac{\varphi \triangleright_A \mathsf{enc}(t, \mathsf{pk}(u), r) \qquad \emptyset \triangleright_A u}{\varphi \triangleright_A t}$$

$$\frac{\varphi \triangleright_A t \qquad \varphi \triangleright_A \mathsf{pk}(u) \qquad \varphi \triangleright_A r}{\varphi \triangleright_A \mathsf{enc}(t, \mathsf{pk}(u), r)}$$

$$\frac{t \qquad u}{\langle t, u\rangle}$$

$$\frac{u}{\mathsf{pk}(u)}$$

$$\frac{\langle t, u\rangle}{t}$$

$$\frac{\langle t, u\rangle}{u}$$

$$\frac{\mathsf{enc}(t, k, r) \qquad k}{t}$$

Définition 2.6 Secrecy

s is secret in P iff for any sequence $(\emptyset, \emptyset, P) \to^* (\varphi, \sigma, Q)$, we have $\varphi \not\triangleright_A s$.

Résultat 2.1

The problem Secrecy :
Data : φ, s
Question : $\varphi \overset{?}{\triangleright}_A s$
is NP-complete.

2.4 Computational semantics

Définition 2.7 **Attacker (computational semantics)**

The attacker is a (deterministic) probabilistic [a] polynomial time Turing machine (PPTTM).
In this model there is also a I/O tape to send and receive the messages (like an oracle tape).
To compute $\varphi \triangleright_A s$, the machine takes $[\![\varphi]\!]^{\mathcal{M}}_{\sigma}$ on the input tape and computes s on the output tape.

a. A probabilistic Turing machine is a Turing machine with a tape filled with random bits.

Définition 2.8 **Computational security property**

With η the security parameter (the length of $[\![n]\!]^{\mathcal{M}}$), ρ_A the attacker's random tape and τ the interpretation of names, the probability that the attacker can compute must be negligible :

$$\forall P \in \mathbb{R}[X], \forall \mathcal{A}, \exists N, \forall \eta > N, \mathbb{P}(\rho_A, \tau : (\emptyset, \emptyset, P) \to^* (\varphi, \sigma, Q) \text{ and } \varphi \triangleright_A s) < \frac{1}{P(\eta)}.$$

2.5 Exemple

On considère le scénario suivant dans le modèle Dolev Yao :

$$P_0 := \nu \mathsf{sk}_A \mathsf{sk}_B \, [P_A(\mathsf{sk}_A, \mathsf{pk}_C) \parallel P_B(\mathsf{sk}_B) \parallel \mathsf{out}(\mathsf{pk}_A)0 \parallel \mathsf{out}(\mathsf{pk}_B)0]$$

avec

$$\begin{aligned} P_A(\mathsf{sk}_A, \mathsf{pk}_C) :=& \nu n_A \nu r_A \nu r'_A \,.\, \mathsf{out}(\{\langle \mathsf{pk}(\mathsf{sk}_A), n_A\rangle\}^{r_A}_{\mathsf{pk}_C}) \\ & \mathsf{in}(x) \,.\, \mathsf{let}\ x_{n_B} = \pi_2(\mathsf{dec}(x, \mathsf{sk}_A))\ \mathsf{in} \\ & \mathsf{let}\ x_{n_A} = \pi_1(\mathsf{dec}(x, \mathsf{sk}_A))\ \mathsf{in} \\ & \mathsf{if}\ x_{n_A} = n_A\ \mathsf{then}\ \mathsf{out}(\{x_{n_B}\}^{r'_A}_{\mathsf{pk}'_C})\ \mathsf{else}\ 0 \end{aligned}$$

et

$$\begin{aligned} P_B(\mathsf{sk}_B) :=& \nu n_B \nu r_B \,.\, \mathsf{in}(y) \\ & \mathsf{let}\ y_{n_A} = \pi_2(\mathsf{dec}(y, \mathsf{sk}_B))\ \mathsf{in} \\ & \mathsf{let}\ y_{\mathsf{pk}_A} = \pi_1(\mathsf{dec}(y, \mathsf{sk}_B))\ \mathsf{in} \\ & \mathsf{out}(\{\langle y_{n_A}, n_B\rangle\}^{r_B}_{y_{\mathsf{pk}_A}}) \\ & \mathsf{in}(z) \mathsf{if}\ \mathsf{dec}(z, \mathsf{sk}_B) = n_B\ \mathsf{then}\ 0\ \mathsf{else}\ 0 \end{aligned}$$

On veut $(\emptyset, \emptyset, P_0) \to^* (\varphi, \sigma, Q)$ tel que $\varphi \triangleright_{\mathsf{DY}} n_B$.
Avec P^1_A le protocole P_A après le out et P^1_B le protocole après les ν,

$$(\emptyset, \emptyset, P_0) \to^* (\underbrace{\nu \overline{n} \,.\, \mathsf{pk}_A \,.\, \mathsf{pk}_B \{\langle \mathsf{pk}_A, n_A\rangle\}^{r}_{\mathsf{pk}_C}}_{\varphi_0}, \emptyset, P^1_A \parallel P^1_B)$$

À partir d'ici on veut savoir si on peut prouver $\varphi_0 \triangleright_{\mathsf{DY}} \{\langle \mathsf{pk}_A, n_A\rangle\}^{r_C}_{\mathsf{pk}_B}$, parce qu'on a

$$\dfrac{\dfrac{\dfrac{\overline{\varphi_0 \triangleright \{\langle \mathsf{pk}_A, n_A\rangle\}^{r_A}_{\mathsf{pk}_C}} \quad \overline{\varphi_0 \triangleright \mathsf{sk}_C}}{\varphi_0 \triangleright \langle \mathsf{pk}_A, n_A\rangle} \quad \overline{\varphi_0 \triangleright \mathsf{pk}_B} \quad \overline{\varphi_0 \triangleright r_C}}{\varphi_0 \triangleright \{\langle \mathsf{pk}_A, n_A\rangle\}^{r_C}_{\mathsf{pk}_B}}$$

Avec P^2_B le protocole P_B après le premier in ,

$$\begin{aligned} (\emptyset, \emptyset, P_0) &\to^* (\varphi_0, \{y \mapsto \{\mathsf{pk}_A, n_A\}^{r_C}_{\mathsf{pk}_B}\}, P^1_A \parallel P^2_B) \\ &\to (\varphi_0, \{y \mapsto \{\mathsf{pk}_A, n_A\}^{r_C}_{\mathsf{pk}_B}, y_{n_A} \mapsto\}, P^1_A \parallel P^3_B) \end{aligned}$$

Exercice 2.2

Finir l'exemple.

2.6 Correction du protocole

En 1995, Lowe trouve une attaque sur le protocole NS et propose une modification : le protocole NSL.

$$\begin{aligned} A \to B :& \nu n_A \nu r_A \nu r'_A \,.\, \{\langle \mathsf{pk}_A, n_A\rangle\}^{r_A}_{\mathsf{pk}_B} \\ B \to A :& \{\langle \mathsf{pk}_B, n_A\rangle, n_B\}_{\mathsf{pk}_A} \\ A \to B :& \{n_B\}_{\mathsf{pk}_B} \end{aligned}$$

Il prouve que le protocole est sûr, mais une attaque utilisant le modèle calculatoire est trouvée par la suite.

Définition 2.9 **Chiffrement El Gamal**

Soit G un groupe cyclique d'ordre d et de générateur g.
Une clé secrète est un nombre aléatoire sk, et la clé publique associée est $\mathsf{pk} = g^{\mathsf{sk}}$.
Le chiffré g^m, avec un nombre aléatoire r, est la paire $(g^r, \mathsf{pk}^r \cdot g^m)$.
Pour déchiffrer (u, v) on calcule v/u^{sk}.

L'attaque est alors la suivante. L'adversaire calcule $\{\langle\langle pk_C, n_A\rangle n_B\rangle\}^{n_B}_{\mathsf{pk}_A}$ à partir de $\{\langle\langle \mathsf{pk}_B, n_A\rangle n_B\rangle\}^{n_B}_{\mathsf{pk}_A}$.
$(g^{r_B}, g^{\mathsf{sk}_A \cdot r_B} \cdot g^{\mathsf{sk}_B + 2^{|\mathsf{sk}_B|} n_A + 2^{|\mathsf{sk}_B|+|n_A|} n_B})$

3 Proverif

3.1 Propriété d'autentification

Si A démarre le protocole en voulant parler à B et que l'exécution du protocole s'est bien déroulée, alors A est certain d'avoir parlé à B.

3.1.1 Events

Définition 3.1 **Process**

On ajoute dans les processus les events :

$$\begin{aligned} P ::= \;& 0 \\ & | \,\mathsf{in}(x) \,.\, P \\ & | \,\mathsf{out}(x) \,.\, P \\ & | \, P \| P \\ & | \,\mathsf{ev}(\vec{v}) \,.\, P \end{aligned}$$

où $\mathsf{ev} \in \mathcal{E}\mathrm{v}$ est une fonction event, et $\vec{v}$ est une séquence de termes.
L'ensemble des events est noté $\mathcal{E}v$.

Leur sémantique est la suivante.

$$\frac{}{(\phi, \sigma, \mathsf{ev}(\vec{v}) \,.\, P, \varepsilon) \to (\phi, \sigma, P, [\![\mathsf{ev}(\vec{v})]\!]^{\mathcal{M}}_{\sigma} . \varepsilon)}$$

la séquence des évènements exécutés pendant l'exécution

Et pour toutes les autres règles, ε est inchangé.

3.1.2 Property

On considère les processus suivants.

$$\begin{aligned} A(\mathsf{pk},\mathsf{sk}) :=& \nu n \nu r \,.\, \mathsf{out}(\{\langle \mathsf{pk}(\mathsf{sk})_a, n\rangle\}^r_{\mathsf{pk}_b}) \\ & \mathsf{in}(z) \,.\, \mathsf{let}\ z_1 = \mathsf{dec}(z, \mathsf{sk}_a)\ \mathsf{in} \\ & \mathsf{if}\ z_1 = n\ \mathsf{then}\ \mathsf{ev}_a(n, \mathsf{pk}_b, \mathsf{pk}(\mathsf{sk}_a))\ \mathsf{else}\ 0 \end{aligned}$$

$$\begin{aligned} B(\mathsf{sk}_b) :=& \nu r' \,.\, \mathsf{in}(x) \,.\, \mathsf{let}\ y = \mathsf{dec}(x, \mathsf{sk}_b)\ \mathsf{in} \\ & \mathsf{let}\ y_1 = \pi_1(y)\ \mathsf{in}\ \mathsf{let}\ y_2 = \pi_2(y)\ \mathsf{in} \\ & \mathsf{out}(\{y_2\}^{r'}_{y_1}) \,.\, \mathsf{ev}_b(y_2, \mathsf{pk}(\mathsf{sk}_b), y_1) \end{aligned}$$

Alors on a comme propriété que $\forall x, y, z, \mathsf{ev}_a(x, y, z) \Rightarrow \mathsf{ev}_b(x, y, z) \equiv \varphi$.
De manière générale, on a $\forall x ..., \mathsf{ev}_1(\vec{u_1}) \wedge ... \wedge \mathsf{ev}_n(\vec{u_n}) \Rightarrow \Phi(\wedge, \vee, u = v, u \neq v, \mathsf{ev}(\vec{v}))$.

$\mathrm{attacker}(u)$ signifie que l'attaquant peut déduire u. On a par exemple
— $\mathsf{ev}(u) \Rightarrow \mathrm{attacker}(u)$
— $\mathrm{attacker}(k) \wedge \mathsf{ev}_{\mathsf{honnest\ key}}(k) \Rightarrow \bot$ prouve que les clés honêtes ne sont pas déductibles.
Dans le protocole

$$\nu \mathsf{sk} \nu \mathsf{sk}_b \,.\, \mathsf{out}(\mathsf{pk}(\mathsf{sk}_a), \mathsf{pk}(\mathsf{sk}_b)) \,.\, A(\mathsf{sk}_a, \mathsf{pk}(\mathsf{sk}_a)) \| B(\mathsf{sk}_b)$$

Φ est faux car on peut toujours s'arrêter dans B à l'exécution de $\mathsf{out}(\{y_2\}^{r'}_{y_1})$.
Pour corriger cela on déplace l'event le plus tôt possible, donc on corrige B en B' :

$$\begin{aligned} B'(\mathsf{sk}_b) :=& \nu r' \,.\, \mathsf{in}(x) \,.\, \mathsf{let}\ y = \mathsf{dec}(x, \mathsf{sk}_b)\ \mathsf{in} \\ & \mathsf{let}\ y_1 = \pi_1(y)\ \mathsf{in}\ \mathsf{let}\ y_2 = \pi_2(y)\ \mathsf{in} \\ & \mathsf{ev}_b(y_2, \mathsf{pk}(\mathsf{sk}_b), y_1) \,.\, \mathsf{out}(\{y_2\}^{r'}_{y_1}) \end{aligned}$$

et alors ϕ est vraie.

3.1.3 Réplication

On ajoute dans la grammaire des processus $!P$ qui représente un nombre non borné de copies de P.

Sa sémantique est $!P \equiv !P \| P$.

3.2 Écrire en Proverif

3.2.1 Types

Les types aident à modéliser. On peut demander à Proverif de vérifier les querries avec un attaquant typé, mais c'est moins fort que le non typé.

Syntaxe | **Type**

```
type my_type
```

Les types natifs sont
— `bitstring`
— `bool`
— `nat`
— `channel`

3.2.2 Name and function

Syntaxe

```
fun g(ty1,...,tyn):ty
free n:ty
free n:ty [private]
```

3.2.3 Rewrite rules

Exemple 3.1 — Chiffrement randomisé asymétrique

```
dec(aenc(x, pk(y), z), y) → x
fun dec(bitstring, skey) : bitstring
reduc forall x:bitstring, y:skey, z:rand; dec(aenc(x,pk(y),z),y)=x
```

Syntaxe

$P :=\ 0$	$P :=\ 0$
$\mid \mathsf{in}(x).P$	$\mid$ `in(c, x:ty); P` avec c un canal
$\mid \mathsf{out}(x).P$	$\mid$ `out(c,u); P`
$\mid$ if $u = v$ then P else P	$\mid$ `if a=v then P else P`
$\mid$ let $y = u$ in P	$\mid$ `let y:ty=u in P (else P)` optionnel au cas où u est invalide
$\mid \nu \overline{n}.P$	$\mid$ `new n:ty; P`
$\mid P \| P$	$\mid P \mid P$
$\mid \mathsf{ev}(\overline{u}).P$	$\mid$ `event ev(v); P`
$\mid !P$	$\mid$ `!P`

3.2.4 Queries

3.3 Output of Proverif

3.3.1 Overview

3.3.2 Lire la trace d'attaque

Des autres instructions possibles sont `insert` et `get` :

Syntaxe **Table**

`insert my_tbl(u1,...,un).` P où `my_tbl`$\in \mathcal{T}$ est un symbole de fonction table
`get my_tbl(x1,...,xn) in` P `else` P

La sémantique du `insert` est la suivante :

$$(\phi, \sigma, \texttt{insert tbl(u1,...,un)}\ P, \varepsilon, T) \to (\phi, \sigma, P, \varepsilon, T \cup \{\texttt{tbl}([\![\texttt{u1}]\!]^{\mathcal{M}}_{\sigma}, ..., [\![\texttt{un}]\!]^{\mathcal{M}}_{\sigma}\}$$

La sématique du `get` est la suivante :

$$\frac{}{(\phi, \sigma, \texttt{get tbl(x1,...,xn)}\ P, \varepsilon, T) \to (\phi, \sigma\{\texttt{ti/xi}\}_{i=1}^{n}, P, \varepsilon, T)} \text{ si } \texttt{tbl(t1,...,tn)} \in T$$

Exemple 3.2

```
let Gen = new sk; insert Honestkey(sk); out(c,pk(sk)).
let runA = get Honestkey(xskA) in in(c,pkB); A(xskA,pkB).
let runB = get Honestkey(xskB) in B(xskB).
process
 !Gen | !runA | runB
```

3.3.3 Good practices

Exemple 3.3 `letfun`

```
letfun enc(x,y) = new r:rand; aenc(x,y,r).
```

Exemple 3.4 Sanity check

On peut obtenir des queries `false` qui correspondent à des exécutions normales en déclarant `event Sanity.` à un point particulier du protocole.

3.3.4 Conclusion

Ce qu'il faut retenir quand on modélise un protocole :

1. Définir un process par rôle
   - Identifier les rôles jouées par les agents honnêtes
   - Identifier les connaissances initiales
   - Identifier les valeurs apprises pendant l'exécution
2. Identifier les connaissances initiales de l'attaquant
   - Public ?
   - Privé ?
   - Corrompu ?
3. Scénarios
   - Est-ce que l'attaquant peut participer ?
   - Laisser l'attaquant choisir qui peut parler à qui

3.4 Vérification indécidable (pour un nombre non borné de sessions)

Définition 3.2 Problème de Post

Input : $u_1, ..., u_n, v_1, ..., v_n \in \{a, b\}^*$
Question : $\exists k \geqslant 1, \exists i_1, ..., i_k \in [\![1, n]\!], u_{i_1}...u_{i_k} = v_{i_1}...v_{i_k}$

Résultat 3.1

Le problème de Post est indécidable.

Définition 3.3

Encodage : $\forall u \in \{a, b\}^*, \tilde{u}(t) = (\ell_1, (\ell_2(..., (\ell_m, t)...)))$ où $u = \ell_1...\ell_m$.
Init : $P_A^i = \text{out}(\{(\tilde{u}_i(0), \tilde{v}_i(0))\}_k)$
Next : $P_B^i = \text{in}(y)\,.\,\text{let}\ (y_1, y_2) = \text{dec}(y, k)\ \text{in}\ \text{out}(\{(u_i(y_1), v_i(y_2))\}_k)$
Test : $P_C = \text{in}(y)\,.\,\text{let}\ (y_1, y_2) = \text{dec}(y, k)\ \text{in if}\ y_1 = y_2\ \text{then out(bad)}$
System : `new k,bad;` P_A^1 | ... | P_A^n | !P_B^1 | ... | !P_B^n | P_C
Question : est-ce que bad est secret ?

Proposition 3.2

Ce problème est indécidable.

Preuve du proposition 3.2.

Observation 1

k n'apparaît qu'en position clé du chiffrement, donc k reste secret

Observation 2

L'observation 1 implique que tout terme $\{w\}_k$ connu de l'attaquant est output par un P_A^i ou P_B^i avec $w = (w_1, w_2)$ pour certains w_1, w_2.

Observation 3

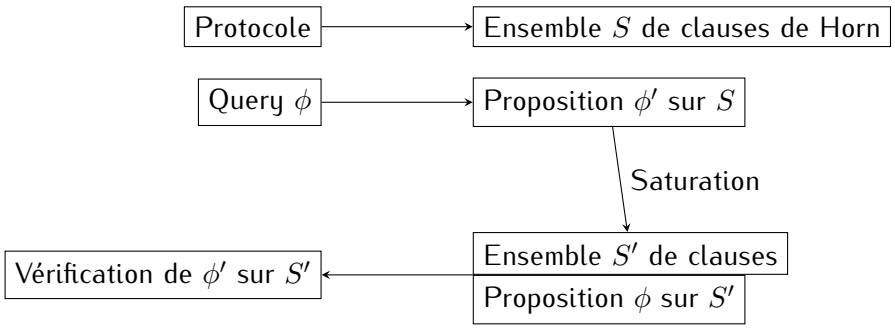
L'observation 2 implique que tout terme $\{w\}_k$ connu de l'attaquant est de la forme $w = (t_1, t_2)$ avec $t_1 = \tilde{u}_{i_1}(\tilde{u}_{i_2}, ..., \tilde{u}_{i_k}(0), ...))$ et $t_2 = \tilde{v}_{i_1}(\tilde{v}_{i_2}, ..., \tilde{v}_{i_k}(0), ...))$ et $i_1, ..., i_k \in [\![1, n]\!]$.

Observation 4

bad n'est pas secret ssi l'attaquant connaît un terme $\{(t_1, t_2)\}_k$ avec $t_1 = t_2$.

Et là c'est le plus beau jour de ma vie.

4 Procedure interne de Proverif



ϕ' vrai sur S implique ϕ vrai sur P
ϕ' vrai sur S ssi ϕ' vrai sur S'
La saturation simplifie les clauses, donc ϕ' est facilement vérifiable sur S'.

4.1 Représentation en clauses de Horn

Définition 4.1

$F_1 \wedge ... \wedge F_n \rightarrow C$, où les F_i et C sont des facts de la forme $p(u_1, ..., u_k)$.

La sémantique est que si les F_i sont vrais alors C est vrai.

Exemple 4.1 Exemples de prédicats/facts

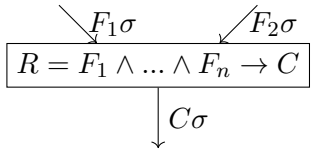
$\mathrm{attacker}(t) \equiv$ l'attaquant connaît t
$\mathrm{event}(\mathrm{ev}) \equiv$ l'event ev a été exécuté
$t \neq t' \equiv t$ est différent de t'

Exemple 4.2 Exemples de clauses

R_1 $\text{attacker}(x) \wedge \text{attacker}(y) \rightarrow \text{attacker}((x, y))$
R_2 $\text{attacker}(\mathsf{aenc}(x, \mathsf{pk}(y), z)) \wedge \text{attacker}(y) \rightarrow \text{attacker}(x)$
R_3 $\text{attacker}((x, y)) \rightarrow \text{attacker}(x)$

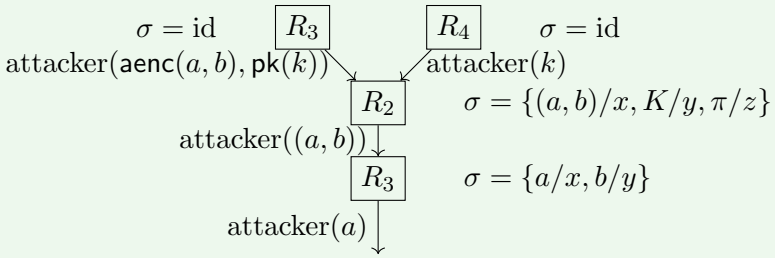
4.1.1 Dérivabilité d'un fact

$\phi \equiv \mathsf{secret}(m)$ dans $P \Rightarrow \phi' \equiv$ est-ce que $\text{attacker}(m)$ est dérivable dans S
Une dérivation de F dans S est un arbre où
— les nœuds sont labellisés par une clause de S
— les arêtes sont labellisées par un fact clos (sans variable)
— l'arête sortante de la racine est labellisée par F
— pour tout nœud il existe un σ tel que



Exemple 4.3

$S = \{R_1, R_2, R_3, \rightarrow \text{attacker}(k), \rightarrow \text{attacker}(\mathsf{aenc}((a, \mathsf{pk}), \mathsf{pk}(k), r))\}$
$\text{attacker}(a)$ est dérivable dans S.



4.1.2 Représentation du protocole en clauses de Horn

L'idée est que si le rôle A output t après avoir reçu $t_1, ..., t_n$ alors on crée une clause $\text{attacker}(t_1) \wedge \text{attacker}(t_n) \rightarrow \text{attacker}(t)$.

Exemple 4.4

Sur le k handshake, $A(\mathsf{sk}_a, \mathsf{pk}(\mathsf{sk}_b)) = \nu n \nu r \,.\, \mathsf{out}(\mathsf{aenc}((\mathsf{pk}(\mathsf{sk}_a), n), \mathsf{pk}(\mathsf{sk}_b), r))... \rightarrow \text{attacker}(\mathsf{aenc}((\mathsf{pk}(\mathsf{sk}_a), a), \mathsf{pk}(\mathsf{sk}_b), r))$
$B(\mathsf{sk}_b) = \nu r' \,.\, \mathsf{in}(x)$
let $y = \mathsf{dec}(x, \mathsf{sk}_b)$ in
let $y_1 = \pi_1(y)$ in let $y_2 = \pi_2(y)$ in
$\mathsf{out}(\mathsf{aenc}(y_2, y_1, r'))$
Il faut prendre en compte les let

1. Pour que le let $y = \mathsf{dec}(x, \mathsf{sk}_b)$ in réussisse, il faut $\sigma_1 = \{x \rightarrow \mathsf{aenc}(y, \mathsf{pk}(\mathsf{sk}_b), z)\}$
2. Pour le let $y_1 = \pi_1(y)$ in il faut $\sigma_2 = \{y \rightarrow (y_1, z)\}$
3. Pour le let $y_2 = \pi_2(y)$ in il faut $\sigma_3 = \{z \rightarrow y_2\}$

On génère $\text{attacker}(x, \sigma_1\sigma_2\sigma_3) \rightarrow \text{attacker}(\mathsf{aenc}(y_2, y_1, r')\sigma_1\sigma_2\sigma_3) \Rightarrow \text{attacker}(\mathsf{aenc}((y_1, y_2), \mathsf{pk}(\mathsf{sk}_b), z)) \rightarrow \text{attacker}(\mathsf{aenc}(y_2, y_1, r'))$.

Remarque

Dans les clauses de Proverif, $F_1 \wedge ... \wedge F_n \rightarrow C$, on a $\text{vars}(C) \subseteq \text{vars}(F_1, ..., F_n)$.

4.1.3 Représentation des nonces

Dans $\text{attacker}(\text{aenc}((y_1, y_2), \text{pk}(\text{sk}_B), z_r)) \to \text{attacker}(\text{aenc}(y_2, y_1, r'))$, sk_B et r' sont des nonces vus par Proverif comme des constantes. Si B est exécuté deux fois alors la clause modélise le fait que B utiliserait deux fois le même random r'. Ça ne va pas !

On va skolemiser les nonces pour intégrer les inputs précédents : $r' \to \overline{r'}[t_1, ..., t_n]$, avec $t_1, ..., t_n$ tous les inputs avant la déclaration de `new r'`.

1. Avec $B(\text{sk}_B) = \nu r' \,.\, \text{in}(x) \,.\, \text{let } y = ... \text{ in }$, on a $r' \to \overline{r'}[]$.
2. Avec $B(\text{sk}_B) = \text{in}(x) \,.\, \nu r' \text{let } y \,.\, \text{let } y = ... \text{ in }$, on a $r' \to \overline{r'}[x]$.

On a le même problème avec la réplication.

Exemple 4.5

Avec $B(\text{sk}_B) = \nu r \,.\, \text{in}(y)...$,
$\quad !\nu \text{sk}_B \,.\, B(\text{sk}_B)$
$\to^* !\nu \text{sk}_B B(\text{sk}_B) \mid \nu \text{sk}_B B(\text{sk}_B) \mid \nu \text{sk}_B B(\text{sk}_B)$
$\to^* !\nu \text{sk}_B B(\text{sk}_B) \mid B(\text{sk}'_B) \mid B(\text{sk}'_B)$
$\text{attacker}(\text{aenc}((y_1, y_2), \text{pk}(\overline{\text{sk}}B[]), z, r)) \to \text{attacker}(\text{aenc}(y_2, y_1, \overline{r}'[]))$: on perd la fraîcheur de sk_B.

L'idée va être de rajouter des variables qui indicenc la réplication : $!^i \nu \text{sk}_B B(\text{sk}_B) \Rightarrow \text{sk}_B \to \overline{\text{sk}}_B[i]$ et $B(\text{sk}_B) = \text{in}(x) \,.\, \nu r' ... \Rightarrow r' \to \overline{r}'[i, x]$.

4.1.4 Génération formelle d'un Process en clauses

Gestion des destructeurs "let $x = t$ in P"

Si $t = g(t_1, ..., t_n)$, $t \Downarrow (u, \sigma) \equiv t$ peut se réduire en u en instanciant les variables par σ.

— Si $t \in \mathcal{X} \cup \mathcal{N}$ (les noms), $t \Downarrow (t, \emptyset)$

— Si $t = g(t_1, ..., t_n)$ avec g qui n'apparaît en symbole de tête d'une règle de réécriture, alors g ne se réduit pas et $t \Downarrow g(u_1, ..., u_n), \sigma$ où $(t_1, ..., t_n) \Downarrow (u_1, ..., u_n), \sigma$

— Si g est un symbole de tête d'une règle $g(u_1, ..., u_n) \to u$ (variable fraîche), $t \Downarrow (u\sigma', \sigma\sigma')$ où $(t_1, ..., t_n) \Downarrow ((v_1, ..., v_n), \sigma)$.

— $(t_1, ..., t_n) \Downarrow (v_1\sigma', ..., v_{n-1}\sigma, v_n), \sigma\sigma'$ où $(t_1, ..., t_n) \Downarrow ((v_1, ..., v_{n-1}), \sigma)$ et $t_n\sigma \Downarrow (v_n, \sigma')$.

On cherche θ tel que $u_1\theta = v_1\theta \wedge ... \wedge u_n\theta = v_n\theta$.

Exemple 4.6

let $x = \underbrace{\text{dec}(\pi_1(x), h(\pi_2(y)))}_{t}$ in

On cherche u et σ tels que $t \Downarrow (u, \sigma)$
$t = \text{dec}(\underbrace{\pi_1(x)}_{t_1}, \underbrace{h(\pi_2(y))}_{t_2})$
On a $x \Downarrow (x, \emptyset)$
$\text{mgu}(x = (x', y')) = \{x \to (x', y')\} = \sigma_1$ donc $t_1 \Downarrow (x'\sigma_1, \sigma_1) = (x', \sigma_1)$
Et même chose pour t_2 : $y \Downarrow (y, \emptyset)$ donc $\pi_2(y) \Downarrow (y_2, \sigma_2)$, avec $\sigma_2 = \{y \to (y_1, y_2)\}$, donc $h(\pi_2(y)) \Downarrow (h(y_2), \sigma_2)$.
$(t_1, t_2) \Downarrow ((x', h(y_2)), \underbrace{\sigma_1\sigma_2}_{\sigma_3})$, donc $\sigma_3 = \{x \to (x', y'), y \to (y_1, y_2)\}$.

$\text{dec}(\text{aenc}(x_1, \text{pk}(y_3), z), y_3) \to x_1$
$\text{mgu}(x' = \text{aenc}(x_1, \text{pk}(y_3), z), y_3 = h(y_2)) = \{x' \to \text{aenc}(x_1, \text{pk}(y_3), z), y_3 = h(y_2)\} = \sigma_4$
$t \Downarrow (x_1, \underbrace{\sigma_3\sigma_4}_{\sigma_5})$

Définition 4.2 **Calcul de** $\mathrm{mgu}(u_1 = v_1, ..., u_n = v_n)$

— Si $u_1 = g(u'_1, ..., u'_k)$ et $v_1 = g'(v'_1, ..., vk'')$, et si $g \neq g'$, alors il n'y a pas d'unifier.
Sinon, $g = g'$ et $k = k'$ et $\mathrm{mgu}(u_1 = v_1, ..., u_n = v_n) = \mathrm{mgu}(u'_1 = v'_1, ..., u'_k = v'_k, u_2 = v_2, ..., u_n = v_n)$.
— Si $u_1 = v_1$ syntaxiquement, alors $\mathrm{mgu}(u_1 = v_1, ..., u_n = v_n) = \mathrm{mgu}(u_2 = v_2, ..., u_n = v_n)$.
— Si $u_1 \in \mathcal{X}$, et si $u_1 \in \mathrm{vars}(v_1)$, alors il n'y a pas d'unifier.
Sinon $u_1 \notin \mathrm{vars}(v_1)$ et $\mathrm{mgu}(u_1 = v_1, ..., u_n = v_n) = \{u_1 \to v_1\}\mathrm{mgu}(u_2 = v_2, ..., u_n = v_n)$.

Exemple 4.7 **Calcul de** mgu

$\theta = \mathrm{mgu}(\{h(x, (a, z)) = h(g(z), y)\})$
$= \mathrm{mgu}(x = g(z), (a, z) = y)$
$= \{x \to g(z)\}\mathrm{mgu}((a, z) = y)$
$= \{x \to g(z), y \to (a, z)\}$

Exemple 4.8 **Calcul de** $t \Downarrow (u, \sigma)$

$t = \pi_1((\mathsf{dec}(x_1, a), a)$
— $a \Downarrow (a\emptyset)$
— $x_1 \Downarrow (x_1, \emptyset)$
— $\mathsf{dec}(\mathsf{aenc}(x, \mathsf{pk}(y), z)y) \to x$, $(x_1, a) \Downarrow (x_1, a), \emptyset$
$\sigma' = \mathrm{mgu}(x_1 = \mathsf{aenc}(x, \mathsf{pk}(y), z), a = y) = \{y \to a, x_1 \to \mathsf{aenc}(x, \mathsf{pk}(a), z)\}$
$\mathsf{dec}(x_1, a) \Downarrow (x, \sigma')$
$\pi_1(\mathsf{dec}(x_1, a), a) \Downarrow (x_1\sigma', \sigma'\sigma'') = (x_1, \sigma'\sigma'')$
$\pi_1((z_1, z_1)) \to z_1$, $\sigma'' = \mathrm{mgu}(z_1 = x, z_1 = a)$
$(\mathsf{dec}(x_1, a), a) \Downarrow (x, a)\sigma''$

Pour un process P, sa traduction en clauses est noté $[\![P]\!]_{\rho,H,I}$ où
— ρ est le renomage des noms (skolemisation)
— H est les hypothèses de la clause
— I est les précédents inputs et les variables de réplication.

Définition 4.3 **Traduction d'un Process en clauses**

— $[\![\mathtt{0}]\!]_{\rho,H,I} = \emptyset$
— $[\![\mathtt{new\ s.}\ P]\!]_{\rho,H,I} = [\![P]\!]_{\rho\{s \to \bar{s}[I]\},H,I}$
— $[\![\mathtt{out(t).}\ P]\!]_{\rho,H,I} = [\![P]\!]_{\rho,H,I} \cup \{H \to \mathrm{att}(t\rho)\}$
— $[\![\mathtt{in(x).}\ P]\!]_{\rho,H,I} = [\![P]\!]_{\rho,H \wedge \mathrm{att}(x),I \cdot x}$
— $[\![\mathtt{!}P]\!]_{\rho,H,I} = [\![P]\!]_{\rho,H,I \cdot i}$ avec i fraîche
— $[\![P\ \mathtt{|}\ Q]\!]_{\rho,H,I} = [\![P]\!]_{\rho,H,I} \cup [\![Q]\!]_{\rho,H,I}$
— $[\![\mathtt{if\ u=v\ then}\ P\ \mathtt{else}\ Q]\!] = [\![P]\!]_{\rho,H\sigma,I} \cup [\![Q]\!]_{\rho,H \wedge \neg\sigma,I}$, avec $\sigma = \mathrm{mgu}(u\rho, u) = \{x_1 \to t_1\}_{i=1}^n$ et $\neg\sigma = \bigvee_{i=1}^n x_i \neq t_i$.
— $[\![\mathtt{let\ x=t\ in}\ P]\!]_{\rho,H,I} = [\![P]\!]_{\rho',H'\rho',I\rho'}$ avec $\rho' = \rho\sigma\{x \to u\}$, $t\rho \Downarrow (u, \sigma)$.
Plus généralement, $\bigcup_{t\rho \Downarrow (u,\sigma), \rho' = \rho\{x \to u\}} [\![P]\!]_{\rho',H\rho',I\rho'}$.

Il reste à traiter la traduction du pouvoir de l'attaquant.
— $\forall f$ d'arité n qui n'apparaît pas en tête d'une règle de réécriture

$$\mathrm{att}(x_1) \wedge ... \wedge \mathrm{att}(x_2) \to \mathrm{att}(f(x_1, ..., x_n))$$

— $\forall g(t_1, ..., t_m) \to t$,

$$\mathrm{att}(t_1) \wedge ... \wedge \mathrm{att}(t_n) \to \mathrm{att}(t)$$

— les noms frais de l'attaquant

$$\to (\overline{n_{\mathrm{att}}}[i])$$

où n_{att} n'apparaît pas dans le protocole.

On note $\mathcal{C}_{\text{att}}$ les clauses de l'attaquant.

4.1.5 Expression du secret

Soit s un nom généré au début du protocole (i.e. $s \to \overline{s}[]$) : `free s. my_type [private]≡ new s. p.`

Théorème 4.1

Si $\text{att}(\overline{s}[])$ n'est pas dérivable dans $[\![P]\!]_{\emptyset,\emptyset,\emptyset} \cup \mathcal{C}_{\text{att}}$, alors s est secret dans P.

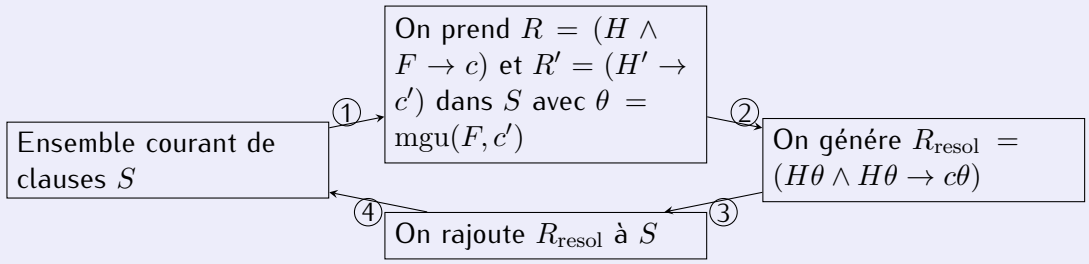
4.2 Saturation precedure

4.2.1 Résolution

La règle de résolution est

$$\frac{R = (H \wedge F \to c) \qquad R' = (H' \to c')}{H\theta \wedge H'\theta \to c\theta} \text{ si } \theta = \text{mgu}(F, c')$$

Procédure v1



On répète le cycle jusqu'à ce que S reste inchangé.

Pourquoi ça marche bien ?

— Supposons que F_{att} est dérivable dans $\text{saturation}(S)$ par $\mathcal{D}$.
On peut alors obtenir une dérivation de F_{goal} strictement plus petite que dans $\text{Saturate}(S)$.
Donc $\text{att}(\overline{s}[])$ est dérivable dans $[\![P]\!]_{\emptyset,\emptyset,\emptyset} \cup \mathcal{C}_{\text{att}}$ alors $(\emptyset \to \text{att}(\overline{s}[])) \in \text{Saturated}([\![P]\!]_{\emptyset,\emptyset,\emptyset} \cup \mathcal{C}_{\text{att}})$.

Attention

Cette v1 ne termine pas.

Exemple 4.9

$R_0 = (F(x) \wedge F'(y) \to F(y))$.
Appliquer résolution sur R_0 et R_0 renommé.

$$\frac{F(x) \wedge F'(y) \to F(y) \qquad F(x') \wedge F'(y') \to F(y')}{R_1 = F(x) \wedge F'(x') \wedge F'(y') \to F(y')} \theta = \text{mgu}(F(y), F(x')$$

Appliquer résolution sur R_0 et $R1$: $R_2 = F(x) \wedge F'(x') \wedge F'(y') \wedge F'(z') \to F(z')$...

L'idée va être de simplifier les clauses générées par la règle de réécriture. Mais il faut assurer la soundess ! Si F est dérivable dans $S \cup \{R\}$ alors F est dérivable dans $S \cup \{\text{simplify}(R)\}$.

4.2.2 Exemples de simplifications

On peut simplifier

— les tautologies : si $R = H \wedge F \to F$ alors $\text{simplify}(R) = \emptyset$.

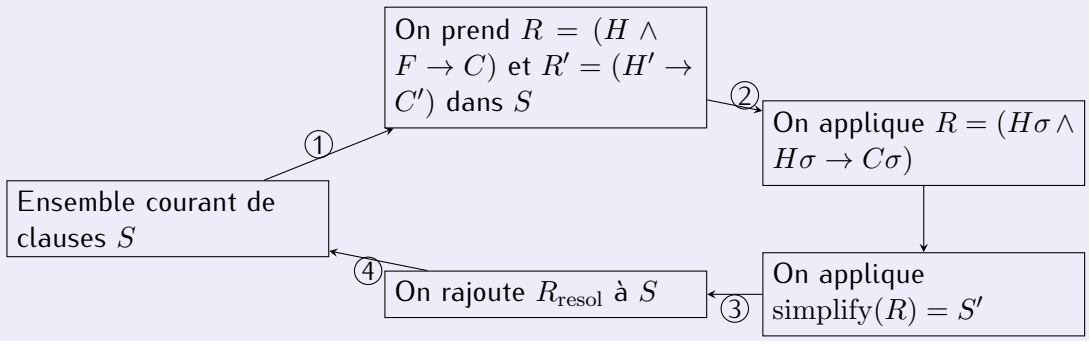
— any attacker term : si $R = H \wedge \text{att}(x) \to C$ alors on peut supprimer $H(x)$ si $x \notin \text{vars}(H, C)$.
Plus généralement, $\text{simplify}(\wedge H \to C) = \{H \to C\}$ est sound.
Attention ça peut rendre dérivables des faits qui ne l'étaient pas.
Dans le cas $F = \text{att}(x)$ avec $x \notin \text{vars}(H, C)$ on peut construire $\sigma' = \sigma_{|X|x}\{x \to t\}$ pour n'importe quel terme t. En particulier on peut prendre $\sigma' = \sigma_{x \setminus S}\{x \to \overline{n_{\text{att}}}[i]\}$ comme $\to \text{att}(\overline{n_{\text{att}}}[i]) \in S$.

— les redondances

Définition 4.4 Redondance

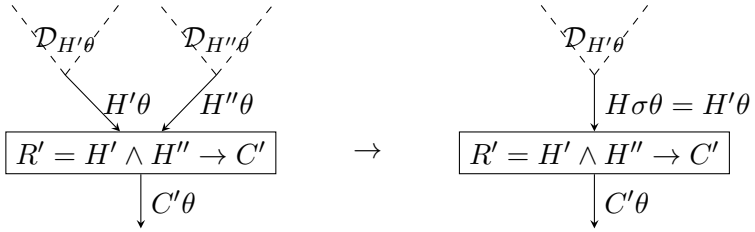
Si $RH \wedge H' \to C$ avec $H\sigma \subseteq H'$ et $\text{vars}(H', C) \cap \text{vars}(H) = \emptyset$, alors $\text{simplify}(R) = \{H' \to C\}$.

Procédure v2



4.2.3 Subsumption

On considère $R = H \to C$ et $R' = (H' \wedge H'' \to C')$.
S'il existe σ tel que $H\sigma = H' \wedge C\sigma = C'$, alors on dit que R subsume R'.
L'idée est que si R subsume R' alors on supprime R'.



4.2.4 Fonction de sélection

Il reste des chaînes infinies malgré la subsumption et les simplifications. On définit alors une fonction de sélection :

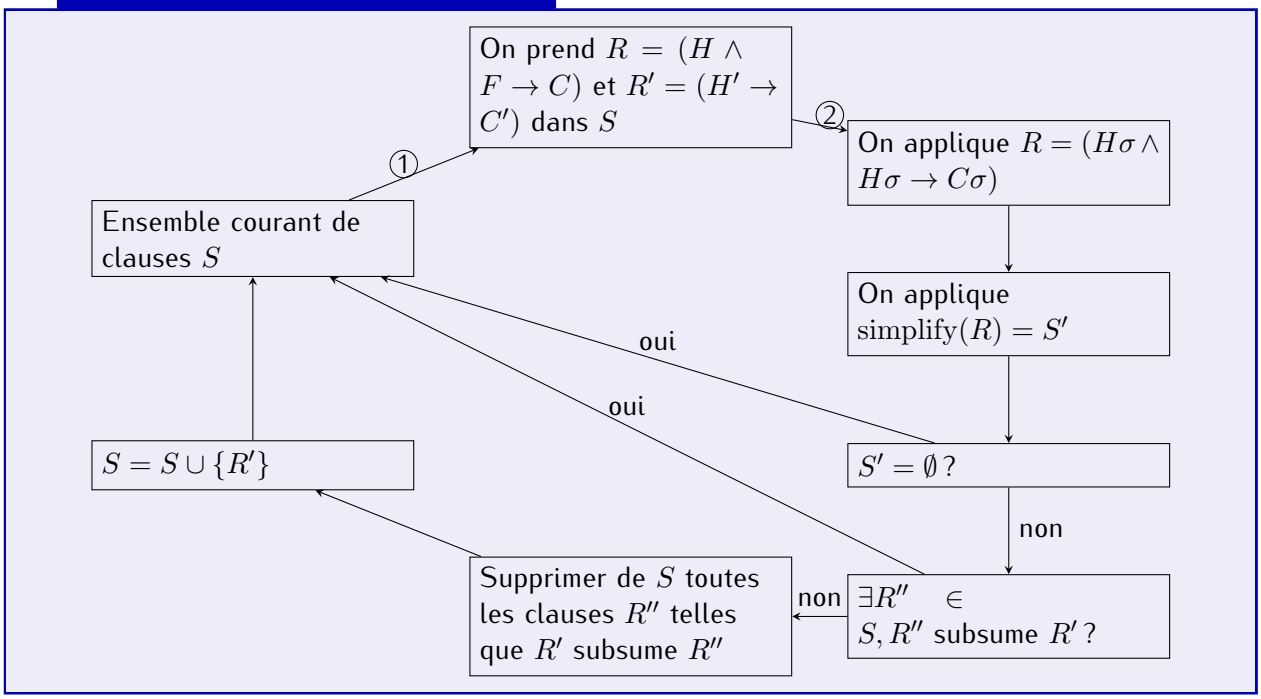
$$\text{sel}(H \to C) \subseteq H \setminus \{\text{att}(x) \mid x \text{ est une variable}\}.$$

Cette fonction indique sur quels faits on peut appliquer les résolutions.
C'est pour préciser sur quels faits ne pas appliquer de résulution.

La règle de résolution devient alors

$$\frac{H \wedge F \to C \qquad H' \to C'}{H\sigma \wedge H'\sigma \to C\sigma} \begin{array}{l}\sigma = \text{mgu}(F, C') \\ F \in \text{sel}(H \wedge F \to C) \\ \text{sel}(H' \to C') \neq \emptyset\end{array}$$

Procédure v4 (oui j'ai pas noté la v3)



4.2.5 Soundness de la saturation

Critère

Si F est dérivable de S_{init} alors F est dérivable de $\text{Saturated}(S_{\text{init}})$.

C'est vrai car subsumption et simplification sont sound.
L'ancien critère était le suivant.

Critère

Si F est dérivable de S_{init} alors $\emptyset \to F \in \text{Saturated}(S_{\text{init}})$.

Mais il n'est plus vrai.
Le problème maintenant est de trouver un nouveau critère simple.

Proposition 4.2 **Critère**

Si F est dérivable dans $\text{Saturated}(S_{\text{init}})$ alors il existe σ et $H \to C \in S_\emptyset := \{H' \to C' \in \text{Saturated}(S_{\text{init}}) \mid \text{sel}(H' \to C') = \emptyset\}$ tels que $C\sigma = F$.

Preuve du proposition 4.2.

On va montrer que si F est dérivable dans $\text{Saturated}(S_{\text{init}})$ par $\mathcal{D}$ alors il existe $\mathcal{D}'$ de $S_\emptyset$ qui dérive F'. On prouve ça par l'absurde.
On prend une mesure sur $\mathcal{D}$: la paire $m(\mathcal{D}) =$ (nombre de nœuds de $\mathcal{D}$ labellisés par R avec $\text{sel}(R) \neq \emptyset$, nombre de nœuds de $\mathcal{D}$).
On suppose par l'absurde qu'il existe une plus petite dérivation $\mathcal{D}$ donnant F dans $\text{Saturated}(S_{\text{init}})$, et on note $m(\mathcal{D}) = (n, m)$ avec $n > 0$.
[...]

Voilà !

5 Propriétés d'équivalence

Définition 5.1 **Secret fort**

Un secret s dans un processus P est un secret fort s'il est indistingable d'un secret aléatoire n. $P(s)$ est indistingable de $P(n)$.

Exemple 5.1 **Respect de la vie privée**

On ne peut pas distinguer les propriétés d'une personne avec des propriétés aléatoires.

Exemple 5.2 **Anonymat de vote**

Si A vote v_1 et B vote v_2, alors $P(A, v_1)\|P(B, v_2)$ est indistingable de $P(A, v_2)\|P(B, v_1)$.

5.1 Équivalence statique

Symbolique

Les messages sont des termes sans variables.
On note $\mathcal{E}$ l'ensemble des équations, ou règles de réécriture

Exemple 5.3

$$\pi_1(\langle x, y\rangle) \to x$$
$$\pi_2(\langle x, y\rangle) \to y$$
$$\mathsf{dec}(\{x\}^z_y, y) \to x$$

Les frames sont de la forme $\nu\overline{n}.\{x_1 \mapsto \nu_1, ..., x_n \mapsto \nu_n\}$.
Si φ est une frame et s un terme, $s\varphi$ est le terme $s\{x_1 \mapsto s'_1, ..., x_n \mapsto s'_n\}$ où $\nu\overline{n}'.s'_1...s'_n = \varphi$ tel que $\mathrm{Noms}(s) \cap \overline{n}' = \emptyset$.

Notation 5.1

On note $\mathcal{P}$ l'ensemble fini des symboles de prédicats à interprétation dans les messages.
Par défaut $\mathcal{P}$ contient l'égalité, interprétée par $s =_\Sigma t$.

Définition 5.2 **Équivalence statique**

φ_1 est statiquement équivalent à φ_2 si pour tous termes, $t_1, ..., t_n$ tout prédicat $P \in \mathcal{P}$,

$$P^I(t_1\varphi_1, ..., t_n\varphi_1) \text{ ssi } P^I(t_1\varphi_2, ..., t_n\varphi_2).$$

On note alors $\varphi_1 \sim \varphi_2$.

Exemple 5.4 **Exemples avec seulement l'égalité**

On considère le système de réécriture de l' exemple 5.3.

- $\nu n.\{x \mapsto n\} \sim \nu n'.\{x \mapsto n'\}$
- $\underbrace{\nu n.\{x \mapsto n, y \mapsto n\}}_{\varphi_1} \not\sim \underbrace{\nu n.\nu n'.\{x \mapsto n, y \mapsto n'\}}_{\varphi_2}$:
  Il faut trouver t_1 et t_2 tels que $t_1\varphi_1 \downarrow = t_2\varphi_1 \downarrow$ et $t_1\varphi_2 \downarrow \neq t_2\varphi_2 \downarrow$, ou $t_1\varphi_1 \downarrow \neq t_2\varphi_1 \downarrow$ et $t_1\varphi_2 \downarrow = t_2\varphi_2 \downarrow$. Donc on peut prendre $t_1 = x$ et $t_2 = y$.
- $\nu k.x \mapsto \langle\{n\}_k, k\rangle \not\sim \nu k.\nu k'.x \mapsto \langle\{n\}_k, k'\rangle$ en prenant $t_1 = n$ et $t_2 = \mathsf{dec}(\pi_1(x), \pi_2(x))$.
- $\underbrace{\nu n.\nu k.x \mapsto \langle\{n\}_k, k\rangle}_{\varphi_1} \sim \underbrace{\nu n.\nu k.\nu k'.x \mapsto \langle\{n\}_k, k'\rangle}_{\varphi_2}$ car il n'y a que des égalités triviales dans les deux cas.
  Mais si on ajoute le prédicat M tel que $M^I = \{t \mid t \text{ ne contient pas de déchiffrement ni de projection}\}$, alors $\varphi_1 \not\sim \varphi_2$ avec

$t_1 = \mathsf{dec}(\pi_1(x), \pi_2(x))$ car $t_1\varphi_1 \downarrow \in M^I$ et $t_2\varphi_2 \downarrow \notin M^I$.

— $\underbrace{\nu n.\nu k.\nu r.x \mapsto \langle\langle\{n\}_k^r, k\rangle, n\rangle}_{\varphi_1} \not\sim \underbrace{\nu n.\nu k.\nu k'.\nu r.x \mapsto \langle\langle\{n\}_k^r, k'\rangle, n\rangle}_{\varphi_2}$ en prenant $t_1 = \pi_2(x)$ et $t_2 = \mathsf{dec}(\pi_1(\pi_1(x)), \pi_2(\pi_1(x)))$.

Exercice 5.1

On considère le système de réécriture de l' exemple 5.3.

$$\varphi_1 = \nu k_1.\nu k_2.\nu r_1.\nu r_2.\nu r_3.x \mapsto \{\{k_1\}_{k_2}^{r_1}\}_{k_1}^{r_2}$$
$$y \mapsto k_1$$
$$z \mapsto \{k_2\}_{k_1}^{r_3}$$

$$\varphi_2 = \nu k_1.\nu k_2.\nu r_1.\nu r_2.\nu r_3.x \mapsto \{\{k_1\}_{k_1}^{r_1}\}_{k_2}^{r_2}$$
$$y \mapsto k_1$$
$$z \mapsto \{k_2\}_{k_1}^{r_3}$$

Dire si $\varphi_1 \sim \varphi_2$ ou non.

Preuve du exercice 5.1.

On pose $t_1 = y$ et $t_2 = \mathsf{dec}(\mathsf{dec}(x, y), \mathsf{dec}(y, y))$.
Alors $t_1\varphi_1 \downarrow = k_1$ et $t_2\varphi_1 \downarrow = k_1$, mais $t_1\varphi_2 \downarrow = k_1$ et $t_2\varphi_2 \downarrow \neq k_1$.
Donc $\varphi_1 \not\sim \varphi_2$.

Voilà !

Définition 5.3 Équivalence statique

Soit $\varphi_1 = \nu\overline{n_1}\Theta_1$ et $\varphi_2 = \nu\overline{n_2}\Theta_2$. $\varphi_1 \sim \varphi_2$ ssi

1. Θ_1 et Θ_2 ont même domaine
2. $\forall P \in \mathcal{P}, \forall t_1, ..., t_n,$
$$P^I(t_1\varphi_1, ..., t_n\varphi_1) \text{ ssi } P^I(t_1\varphi_2, ..., t_n\varphi_2).$$

Définition 5.4 Prédicat Ek

$$Ek^I = \{(s,t) \mid s \in M^I, t \in M^I, \exists u, v, r_1, r_2, k, s = \{u\}_k^{r_1}, t = \{v\}_k^{r_2}\}$$

Théorème 5.1

Si $\mathcal{E}$ est défini par un système de réécriture sous-terme convergent (et les interprétations P^I sont dans $PT(M^E)$), alors l'équivalent statique est décidable en temps polynomial.

Preuve Idée de la preuve.

On a la propriété de petite attaque.

1. Saturer les frames (par exemple si $\{s\}_k^r$ et k sont dans la frame on ajoute s)
2. Identités obtenues par des petites recettes (sur les recettes saturées)
3. On conclut à l'équivalence ssi ces observations sont les mêmes sur les deux frames

Youpi !

5.2 Équivalence de trace

On a deux protocoles P et Q et on veut savoir quand ils sont indistingable. C'est un raffinement de la sémantique.

$$(\mathsf{in}(x).P, \varphi, \sigma, ...) \xrightarrow{u} (P, \varphi, \sigma \uplus \{x \mapsto u\}...).$$

Définition 5.5 Équivalence de trace

P est indistingable de Q ssi

1. $\forall (P, \varphi_0, \emptyset, ...) \xrightarrow{u_1} ... \xrightarrow{u_n} (P', \varphi_n, \sigma_n, ...)$,
   $\exists (Q, \varphi'_0, \emptyset, ...) \xrightarrow{u_1} ... \xrightarrow{u_n} (Q', \varphi'_n, \sigma'_n, ...)$,
   $\varphi_n \sim \varphi'_n$.
2. $\forall (Q, \varphi_0, \emptyset, ...) \xrightarrow{u_1} ... \xrightarrow{u_n} (Q', \varphi_n, \sigma_n, ...)$,
   $\exists (P, \varphi'_0, \emptyset, ...) \xrightarrow{u_1} ... \xrightarrow{u_n} (P', \varphi'_n, \sigma'_n, ...)$,
   $\varphi_n \sim \varphi'_n$.

Les deux programmes produisent un état dans lequel on ne peut pas distinquer le résultat.
On note alors $P \sim Q$.

Exemple 5.5

$P = \nu k.\nu n.\nu r.\nu n'.\nu r'$

$[\mathsf{out}(\{n\}^r_k) \,.\, \mathsf{in}(y) \,.\, \mathsf{let}\ y'_n = \pi_1(\mathsf{dec}(y,k))\ \mathsf{in}\ \mathsf{if}\ y'_n = n\ \mathsf{then}\ \mathsf{ev}_a(n)\ \mathsf{else}\ 0$

$\| \mathsf{in}(x) \mathsf{let}\ y_n = \mathsf{dec}(x,k)\ \mathsf{in}\ \mathsf{out}(\{\langle y_n, x' \rangle\}^{r'}_k) \,.\, \mathsf{ev}_b(y_n) 0]$

Q est la même chose que P mais en remplaçant n par m. Alors $P \sim Q$.

Exercice 5.2 Pour la semaine prochaine

Définition 5.6 Secret faible

s est un secret faible si $\forall (P(s), \emptyset, \emptyset) \to^* (P_n, \varphi_n, \sigma_n), \varphi_n \not\vdash s$.

Définition 5.7 Secret fort

s est un secret fort si $P \sim \nu s' \,.\, P\{s \mapsto s'\}$.

1. Trouver un exemple de protocole où s est secret faible et s n'est pas secret fort.
2. Montrer que secret fort implique secret faible.

Preuve du exercice 5.2.

On va prendre s de la forme $\langle k_1, 0 \rangle$, et

$$P(s) = \nu r.\nu k \,.\, \mathsf{out}(\{s\}^r_k) \,\|\, \mathsf{in}(x) \,.\, \mathsf{out}(\pi_2(\mathsf{dec}(x,k))).$$

Youpi !

5.3 Sécurité calculatoire

Définition 5.8 Adversaire

Les adversaires sont des machines de Turing probabilistes polynomiales avec oracle : $\mathcal{A}^{\mathcal{O}}$. L'oracle correspond à l'interaction avec le protocole.
Plus précisément c'est une machine de Turing à 3 rubans : une bande d'aléatoire tirée au début du calcul (et donc les transitions sont ensuite déterministe), une bande de calcul et une bande d'interaction avec l'oracle.

L'oracle est aussi randomisé : $\mathcal{O} : \{0,1\}^* \times \{0,1\}^\omega \longrightarrow \{0,1\}^*$. Avec une entrée x, l'oracle la remplace par $\mathcal{O}(x \,|\, R)$ sur la bande d'oracle.

Définition 5.9 — Chiffrement symétrique

On a $\mathcal{E}$ et $\mathcal{D}$ tels que
- $\mathcal{D}(\mathcal{E}(x,k,r),k) = x$
- Les clefs ont toutes la même longueur
- Si $|x| = |y|$ alors $|\mathcal{E}(x,k,r)| = |\mathcal{E}(y,k,r)|$.

Définition 5.10 — Fonction négligeable

Une fonction $f : \mathbb{N} \to \mathbb{R}$ est négligeable si $\forall P \in \mathbb{N}[X], \exists N, \forall \eta > N, f(\eta) < \frac{1}{P(\eta)}$.

Définition 5.11 — Indécidabilité pour attaque choisie (IND-CPA)

- On utilise un oracle de chiffrement gauche/droite : $\mathcal{O}_k^\ell(x\#y,r) = \mathcal{E}(x,k,r)$ et $\mathcal{O}_k^r(x\#y,r) = \mathcal{E}(y,k,r)$, si $|x| = |y|$, et sinon il renvoit 0.
- On fait deux appels successifs à un oracle qui utilisent des aléas différents. Si on appelle deux fois l'oracle avec les mêmes données les résultats peuvent être différents.
- On a un paramètre de sécurité $\eta \in \mathbb{N}$, qui est la longueur des clefs et des aléas utilisés dans le chiffrement.
- $\forall \mathcal{A}$ PPT, $\mathrm{Adv}(\mathcal{A},\eta)$ est négligeable, où

$$\mathrm{Adv}(\mathcal{A},\eta) = |\mathbb{P}\{R,R_0,k : \mathcal{A}^{\mathcal{O}_k^\ell(\_,R_0)}(0^\eta,R) = 1\} - \mathbb{P}(R,R_0,k : \mathcal{A}^{\mathcal{O}_k^r(\_,R_0)}(0^\eta,R) = 1\}|$$

Exemple 5.6

1. $\mathcal{E}(x,k,r) = x$. Il n'est pas IND-CPA.
   $\mathcal{A}$ soumet $0\#1$ à l'oracle. S'il répond 0 alors $\mathcal{A}$ s'arrête et répondant 1, sinon 0. $\mathcal{A}$ est en temps constant.
$$\mathbb{P}\{R,R_0,k : \mathcal{A}_k^{\mathcal{O}_k^\ell(\_,R_0)}(0^\eta,R) = 1\} = 1$$
$$\mathbb{P}\{R,R_0,k : \mathcal{A}_k^{\mathcal{O}_k^r(\_,R_0)}(0^\eta,R) = 1\} = 0$$
2. $\mathcal{E}(x,k,r)$ ne dépend pas de r. Il n'est pas IND-CPA.
   $\mathcal{A}$ soumet $0\#0$ à l'oracle, et reçoit $\mathcal{E}(0,k)$. Il soumet ensuite $0\#1$. S'il reçoit le même message il répond 1 et sinon il répond 0.
$$\mathbb{P}\{R,R_0,k : \mathcal{A}_k^{\mathcal{O}_k^\ell(\_,R_0)}(0^\eta,R) = 1\} = 1$$
$$\mathbb{P}\{R,R_0,k : \mathcal{A}_k^{\mathcal{O}_k^r(\_,R_0)}(0^\eta,R) = 1\} = 0 \qquad \text{par propriété de déchiffrement}$$
3. Si on change la définition de IND-CPA en $\forall P, \exists N, \forall \eta > N, \forall \mathcal{A}, \mathrm{Adv}(\mathcal{A},\eta) < \frac{1}{P(\eta)}$ (on a mis le $\forall \mathcal{A}$ deux places en arrière), alors il n'y a aucun chiffrement qui soit IND-CPA.
   L'idée est que $\mathcal{A}$ peut énumérer toutes les clefs et les essayer.
   Il soumet $0\#1$ à l'oracle et reçoit m, calcule $\mathcal{D}(m,k)$ pour toutes les clés k. Si pour une clé il obtient 0 alors il répond 1 et sinon il répond 0.

Exercice 5.3

Si on change la définition de IND-CPA par $\forall P \in \mathbb{N}[X], \exists N, \forall \mathcal{A}, \forall \eta > N, \mathrm{Adv}(\mathcal{A},\eta) < \frac{1}{P(\eta)}$, alors aucun chiffrement n'est IND-CPA.

Preuve du exercice 5.3.

On montre $\exists P \in \mathbb{N}[X], \forall N, \exists \mathcal{A}, \exists \eta > N, \mathrm{Adv}(\mathcal{A}, \eta) \geqslant \frac{1}{P(\eta)}$.
$\mathcal{A}$ peut énumérer toutes les clefs (en temps constant). $\mathcal{A}$ soumet $0\#1$ à l'oracle. Il reçoit $m = \mathcal{E}(0, k, r)$ ou $\mathcal{E}(1, k, r)$.
Si $\mathcal{A}$ trouve k_1 tel que $\mathsf{dec}(m, k_1) = 0$, il se pourrait que $\mathsf{dec}(\mathcal{E}(1, r, r), k_1) = 0$.
La solution va être que $\mathcal{A}$ soumet $m_1\#m_2$ pour tous $m_1, m_2 \in \{0,1\}^{N+1}$. Si $\mathcal{A}$ se trompe, par exemple il dit gauche alors que c'est droit, ça veut dire qu'il a trouvé k_1 tel que pour tous $m_1, m_2 \in \{0,1\}^{N+1}$, $\mathsf{dec}(\mathcal{E}(m_2, k, r_{m_1,m_2}), k_1) = m_1$.

Et là c'est le plus beau jour de ma vie.

Exercice 5.4

Si on change la définition de négligeable par $\exists N, \forall \eta > N, f(\eta) < \frac{1}{2^n}$, alors aucun chiffrement n'est IND-CPA.

Preuve du exercice 5.4.

$\mathcal{A}$ tire une clef. C'est la bonne avec probabilité $\frac{1}{2^\eta}$.

cskifo

Exercice 5.5

Avec $\mathrm{Adv}'(d, \eta) = |2\mathbb{P}\{b \xleftarrow{u} \{\ell, r\}, R, R_0, k : \mathcal{A}^{\mathcal{O}_k^b(\_, R_0)}(\_, R) = 1\} - 1|$, un chiffrement est IND-CPA ssi $\mathrm{Adv}'(\mathcal{A}, \eta)$ est négligeable.

Preuve du exercice 5.5.

$$\begin{aligned} &2\mathbb{P}(b = \ell)\mathbb{P}(\mathcal{A}^{\mathcal{O}^\ell}(0^\eta) = \ell) + \mathbb{P}(b = r)\mathbb{P}(\mathcal{A}^{\mathcal{O}^r}(0^\eta) = r) - 1 \\ &\quad = \mathbb{P}(\mathcal{A}^{\mathcal{O}^\ell}(0^\eta) = \ell) + (1 - \mathbb{P}(\mathcal{A}^{\mathcal{O}^r}(0^\eta) = \ell) - 1 \end{aligned}$$

Voilà !

Théorème 5.2

On suppose le déchiffrement est IND-CPA et d'autres hypothèses.
Si $\varphi \sim \psi$ alors $[\![\varphi]\!] \approx [\![\psi]\!]$.

Définition 5.12 — Distributions indistingables

Deux familles de distributions $\{D_\eta\}_{\eta\in\mathbb{N}} \approx \{D'_\eta\}_{\eta\in\mathbb{N}}$ sont indistingables si pour tout adversaire PPT $\mathcal{A}$,

$$\mathcal{E}(\mathcal{A}, \eta) = |\mathbb{P}(R, x \leftarrow D_\eta : \mathcal{A}(x, 0^\eta, R) = 1\} - \mathbb{P}\{R, x \leftarrow D'_\eta : \mathcal{A}(x, 0^\eta, R) = 1|$$

est négligeable.

Exercice 5.6

La distribution de Dirac : 0 avec proba 1 et $w\#0$ avec proba 0 ($w \in \{0,1\}$) et la distribution uniforme sont distingables.

Exercice 5.7 — Pas obligatoire

Avec D_η^k la distribution uniforme sur $\{0,1\}^\eta$ et D'^k_η la distribution sur $\{0,1\}^\eta$ définie par $\mathbb{P}(x \leftarrow D_\eta^k : x = a) = 0$ si $a = b0^{n-k}$ pour un certain b, et $\frac{1}{2^n - 2^k}$ sinon, alors D_η^k et D'^k_η sont indistingables.

Soit t un terme en forme normale (sans dec, π_1, π_2). Étant donné un tirage $\tau : \mathcal{N} \to \{0,1\}^\eta$.

$$[\![t]\!]^\tau_\eta = \begin{cases} \tau(t) & \text{si } t \in \mathcal{N} \\ p([\![t_1]\!]^\tau_\eta, [\![t_2]\!]^\tau_\eta) & \text{si } t = \langle t_1, t_2 \rangle \\ \mathcal{E}([\![t]\!]^\tau_\eta, [\![u]\!]^\tau_\eta, [\![v]\!]^\tau_\eta) & \text{si } t = \{t\}^v_u \end{cases}$$

avec $p : (\{0,1\}^*)^2 \to \{0,1\}^*$ tel que $|x| = |x'|$ et $|y| = |y'|$ implique $|p(x,y)| = |p(x',y')|$.

$[\![]\!]_\eta$ est une distribution sur $\{0,1\}^*$ et $[\![t_1, ..., t_n]\!]$ est une distribution sur $(\{0,1\}^*)^n$.

Définition 5.13 **Prédicats (pour l'équivalence statique)**

1. $M : M^I$ est l'ensemble des termes dont la forme normale
   (a) ne contient ni dec ni π_1, ni π_2
   (b) toutes les occurrences de $\{u\}^w_v$ sont telles que $v, w \notin \mathcal{N}$ ("CLEFS ATOMIQUES")
2. EQ :
$$\mathrm{EQ}^I = \{(s,t) \mid s\downarrow = t\downarrow \text{ et } M^I(s) \text{ et } M^I(t)\}$$
   Par exemple $\mathrm{dec}(k,k) \notin \mathrm{EQ}^I$.
3. EK :
$$\mathrm{EK}^I = \{(s,t) \mid \exists u, v, \exists k, r, r' \in \mathcal{N}, s\downarrow = \{u\}^r_k, t\downarrow = \{v\}^{r'}_k \text{ et } M^I(u) \text{ et } M^I(v)\}$$
4. EL :
$$\mathrm{EL} = \{(s,t) \mid \exists u, v \in M^I, \exists k, k', r, r' \in \mathcal{N}, s\downarrow = \{u\}^r_k, t\downarrow = \{v\}^{r'}_{k'}, \forall \eta, \ell(u,\eta) = \ell(v,\eta)\}$$

On pourrait distinguer $[\![\{u\}^r_k, \{v\}^{r'}_k]\!]_\eta$ et $[\![\{u\}^r_k, \{v\}^{r'}_{k'}]\!]_\eta$.

On a fait l'hypothèse de "régularité" des longueurs :

$$|m| = |m'| \Longrightarrow |\mathcal{E}(m,k,r)| = |\mathcal{E}(m',k,r')|$$

$$|m_1| = |m'_1| \text{ et } |m_2| = |m'_2| \Longrightarrow |p(m_1,m_2)| = |p(m'_1,m'_2)|$$

$\ell(u,\eta)$ est défini par
— $\ell(u,\eta) = \eta$ si $u \in \mathcal{N}$
— $\ell(u,\eta) = f(\ell(v,\eta))$ si $u = \{v\}^r_k$
— $\ell(u,\eta) = g(\ell(v,\eta), \ell(w,\eta))$ si $u = \langle v, w \rangle$
où f et g sont des paramètres.

Exercice 5.8 **Problème des cycles de clefs**

Si $u = k$, $\{k\}^r_k$. On suppose que $\mathcal{E}_0$ est IND-CPA. On pose $\mathcal{E}(x,k,r) = \mathcal{E}_0(x,k,r)$ si $x \neq k$, k sinon. Montrer que $\mathcal{E}$ est IND-CPA.

Théorème 5.3

Avec un chiffrement IND-CPA et en supposant que

1. il n'y a pas de cycles de clefs : la relation $<_{s_1...s_n}$ sur les clefs definie par $k_1 <_{s_1...s_n} k_2$ si $s_1...s_n$ contient un sous-terme $\{u\}^r_{k_2}$ el que k_1 est un sous-terme de u (on ne peut chiffrer avec k que des messages qui n'utilisent que des clefs strictement plus petites) est un ordre strict
2. les random seeds des chiffrements sont utilisées seulement une fois
3. $s_1, ..., s_m, t_1, ..., t_m \in M^I$.

Alors $\nu\overline{n}\,.\,s_1...s_n \sim \nu\overline{n}\,.\,t_1...t_n$ implique $[\![s_1...s_n]\!]_\eta \approx [\![t_1...t_n]\!]_\eta$.

Preuve du théorème 5.3.

Lemme 5.4

Si k, r n'apparaissent pas dans u alors $[\![\{u\}_k^r]\!] \approx [\![\{0^{\ell(u,\eta)}\}_k^r]\!]_\eta$.

On a même un lemme plus général.

Lemme 5.5

Avec un chiffrement IND-CPA, si $k, r_1, ..., r_m$ n'apparaissent pas dans $u_1, ..., u_m$ alors $[\![\{u_1\}_k^{r_1}, ..., \{u_m\}_k^{r_m}]\!] \approx [\![\{0^{\ell(u_1,\eta)}\}_k^{r_1}, ..., \{0^{\ell(u_m,\eta)}\}_k^{r_m}]\!]_\eta$.

Preuve du lemme 5.5.

Si $\mathcal{A}$ est un adversaire sur l'indistinguabilité, on construit $\mathcal{B}$ adversaire sur IND-CPA.
On prend $\tau : \mathcal{N} \setminus \{k, r_1, ..., r_m\}/to\{0,1\}^*$.
On montre que $[\![\{u_1\}_k^{r_1}, ..., \{u_m\}_k^{r_m}]\!]_\eta^\tau \approx [\![\{0^{\ell(u_1,\eta)}\}, ..., \{0^{\ell(u_m,\eta)}\}]\!]_\eta^\tau$.
$\mathcal{B}$ va

- Lire les noms $k, r_1, ..., r_m$ (τ stocké quelque part).
- Calculer $[\![u_i]\!]_\eta^\tau$, ce qui est possible car $k, r_1, ..., r_m$ n'apparaissent pas dans $u_1, ..., u_m$.
- Faire m requêtes $[\![u_i]\!]_\eta^\tau \# 0^{\ell(u_i,\eta)}$. Les réponses sont soit $[\![\{u_1\}_k^{r_1}, ..., \{u_m\}_k^{r_m}]\!]$ pour l'oracle gauche, soit $[\![\{0^{\ell(u_1,\eta)}\}_k^{r_1} ... \{0^{\ell(u_m,\eta)}\}_k^{r_m}]\!]_\eta^\tau$ pour l'oracle droit.

$\mathrm{Adv}(\mathcal{B}, \eta)$ est l'avantage de $\mathcal{A}$ dans la distinction des deux distributions.

cskifo

La preuve du théorème se fait par récurrence sur $\|s_1\| + ... + \|s_m\| + \|t_1\| + ... + \|t_m\|$, où $\|t\|$ est défini par

- $\|t\| = 0$ si t est constante (ou un nom qui n'est pas lié)
- $\|u\| = 1$ si $u \in \overline{n}$
- $\|\{u\}_k^r\| = 1 + \|u\|$
- $\|\langle u_1, u_2\rangle\| = 1 + \|u_1\| + \|u_2\|$

Cas de base. $s_1, ..., s_m \sim t_1, ..., t_m$
Le test $\mathrm{EQ}(x_i, s_i)$ est satisfait par $s_1, ..., s_m$, donc $\mathrm{EQ}(x_i, s_i)$ est satisfait par $t_1, ..., t_m$, donc $t_i = s_i$, donc $[\![s_1, ..., s_m]\!]_\eta \approx [\![t_1, ..., t_m]\!]_\eta$.

Récurrence

Cas 1. $\exists i \neq j, s_i = s_j$ (resp. $t_i = t_j$) et s_i n'est pas une constante.
On suppose $i = 1$. Alors $\nu\overline{n}\,.\,s_2, ..., s_m \sim \nu\overline{n'}\,.\,t_2, ..., t_m$. Par hypothèse de récurrence, $[\![s_2 ... s_m]\!]_\eta \approx [\![t_2 ... t_m]\!]_\eta$. Donc $[\![s_j, s_2, ..., s_m]\!]_\eta \approx [\![t_j, t_2, ..., t_m]\!]_\eta$.
Or par équivalence statique, $s_1 = s_j \Leftrightarrow t_1 = t_j$. D'où $[\![s_1, s_2, ..., s_m]\!]_\eta \approx [\![t_1, t_2, ..., t_m]\!]_\eta$.

Cas 2. $\exists i, s_i = \langle s_{i_1}, s_{i_2}\rangle$ (on prend $i = 1$).
$\pi_1(s_1) \in M^I$ donc par équivalence statique, $\pi_1(t_1) \in M^I$, donc $t_i = \langle t_{i_1}, t_{i_2}\rangle$.
$\nu\overline{n}, s_{11}, s_{12}, s_2, ..., s_m \sim \nu\overline{n'} t_{11}, t_{12}, t_2, ..., t_m$ recipe sur $s_{11}, ..., s_m$: u donne un recipe v sur $s_1, ..., s_m$ par $v = u\{x_{11} \mapsto \pi_1(x_1), x_{12} \mapsto \pi_2(x_1)\}$.
Par hypothèse de récurrence, $[\![s_{11}, s_{12}, ..., s_m]\!]_\eta \approx [\![t_1, ..., t_m]\!]_\eta$ et donc $[\![s_1, ..., s_m]\!]_\eta \approx [\![t_1, ..., t_m]\!]_\eta$.

Cas 3. $s_i = \{s_i'\}_{s_j}^{r_i}$
Alors $t_i = \{t_i'\}_{t_j}^{r_i'}$ par équivalence statique.
On remplace s_i par s_i' et t_i par t_i'. $\nu\overline{n}\,.\,s_1, ..., s_i', ..., s_m \sim \nu\overline{n'}\,.\,t_1, ..., t_i', ..., t_m$.
Par hypothèse de récurrence, $[\![s_1, ..., s_i', ..., s_m]\!] \approx [\![t_1, ..., t_i', ..., t_m]\!]_\eta$ donc comme r_i (resp. r_i' n'apparaît qu'une seule fois, on a $[\![s_1, ..., s_m]\!]_\eta \approx [\![t_1, ..., t_m]\!]_\eta$.

Cas 4. Soient $\{s'_1\}^{r_1}_{k_1}$,...,$\{s'_\ell\}^{r_\ell}_{k_\ell}$ les chiffrés de la séquence $s_1, ..., s_m$, et au moins un s'_i n'est pas une clé.
Soit k une clef maximale dans $k_1, ..., k_\ell$ telle que les chiffrés par k sont $\{u_1\}^{r_1}_k$,...,$\{u_p\}^{r_p}_k$ t $u_1, ..., u_p$ ne sont pas tous des constantes.
Donc par lemme 5.5 (et k n'apparaît pas dans $u_1, ..., u_p$), $[\![\{u_1\}^{r_1}_k, ..., \{u_p\}^{r_p}_k]\!]_\eta \approx [\![\{0^{\ell(u_1,\eta)}\}^{r_1}_k, ..., \{0^{\ell(u_p,\eta)}\}^{r_p}_k]\!]_\eta$.
Si on remplace $\{u_i\}^{r_i}_k$ par $\{0^{\ell(u_i,\eta)}\}^{r_i}_k$ dans $s_1, ..., s_m$, on obtient une séquence statiquement équivalente (k n'apparaît pas dans $s_1, ..., s_m$ autrement que dans $\{u_i\}^{r_i}_k$).
$\nu\overline{n}\,.\,s'_1..., s'_m \sim \nu\overline{n'}\,.\,t_1, ..., t_m$
Comme au moins un u_i tel que u_i non constant, par hypothèse de récurrence,

$$\begin{aligned}[\![s'_1, ..., s'_m]\!]_\eta &\approx [\![t_1, .., t_m]\!]_\eta \\ &\approx [\![s_1, ..., s_m]\!]_\eta\end{aligned}$$

C'est ce que je voulais !

Deuxième partie

Adrien Koutsos

6 Introduction

6.1 Protocols as sequences of terms

Dans cette partie on va voir des autres preuves calculatoires, avec des autres hypothèses que IND-CPA.

Les termes sont des noms ou des symboles de fonction. On a en plus par rapport à avant des symboles logiques $\dot{\wedge}$, $\dot{\vee}$, $\dot{\to}$.

Ici on n'autorise pas les réplications sans limite avec ! parce qu'on veut des séquences de termes finies.

Exemple 6.1 Private Authentification v1

1. $A \to B : \nu n_A \,.\, \mathsf{out}(c_A, \{\langle \mathsf{pk}_A, n_A\rangle\}_{\mathsf{pk}_B})$
2. $B \to A : \nu n_B, \mathsf{in}(c_A, x) \,.\, \mathsf{out}(c_B, \{\langle \pi_2(x), n_B\rangle\}_{\mathsf{pk}_A})$

Pour modéliser l'attaquant on ajoute un terme $\mathrm{att}(t)$ qui peut représenter n'importe quelle machine de Turing en temps polynomial. On ajoute même des symboles de fonctions spécifiques à l'adversaire.

6.2 Protocol branching

Exemple 6.2 Private Authentification v2

1. $A \to B : \nu n_A \,.\, \mathsf{out}(c_A, \{\langle \mathsf{pk}_A, n_A\rangle\}_{\mathsf{pk}_B})$
2. $B \to A : \nu n_B, \mathsf{in}(c_A, x) \,.\, \mathsf{if}\ \pi_1(x) \doteq \mathsf{pk}_A\ \mathsf{then}\ \mathsf{out}(c_B, \{\langle \pi_2(x), n_B\rangle\}_{\mathsf{pk}_A})\ \mathsf{else}\ \mathsf{out}(c_B, \{0\}_{\mathsf{pk}_A})$

On a alors comme termes

— $t_1 \equiv \{\langle \mathsf{pk}_A, n_A\rangle\}_{\mathsf{pk}_B}$

— $t_2 \equiv \mathsf{if}\ \pi_1(x) \doteq \mathsf{pk}_A\ \mathsf{then}\ \{\langle \pi_2(x), n_B\rangle\}_{\mathsf{pk}_A}\ \mathsf{else}\ \{0\}_{\mathsf{pk}_A}$

On ne va autoriser que des processus déterministes. Si on doit faire un choix alors on laisse l'adversaire le faire et de notre point de vue c'est comme si on avait du non-déterminisme.

Définition 6.1 Processus action-déterministe

Une configuration (φ, σ, P) est action-déterministe si pour toute configuration atteignable (φ', σ', P') à partir de laquelle pour toute action observable on ne peut obtenir que des configurations identiques.

Exemple 6.3

1. $\mathsf{out}(c, t_1) \parallel \mathsf{in}(c, x) \,.\, \mathsf{out}(c, t_2)$ n'est pas action-déterministe car après avoir input à droite on a une configuration dans laquelle on ne sait pas si on va output à gauche ou à droite
2. $\mathsf{if}\ b\ \mathsf{then}\ \mathsf{out}(c, t_1)\ \mathsf{else}\ \mathsf{in}(c, x) \,.\, \mathsf{out}(c, t_2)$ est action-déterministe, car une fois qu'on a testé si b est vrai ou faux il n'y a plus qu'un seul out
3. $\mathsf{out}(c, t_1) \parallel \mathsf{if}\ b\ \mathsf{then}\ \mathsf{out}(c, t_2)\ \mathsf{else}\ \mathsf{in}(c, x) \,.\, \mathsf{out}(c_0, t_3)$ n'est pas action-déterministe car si b est vrai on peut avoir du non déterminisme

6.3 Folding

Définition 6.2 — Folding configuration

Une configuration de folding est un tuple $(\varphi, \sigma, j, \Pi_1, ..., \Pi_l)$ où

1. φ est une séquence de termes, c'est la frame, la séquence de termes envoyés depuis le début
2. σ est une séquence finie de mapping, il donne la valeur des variables
3. $j \in \mathbb{N}$ compte le nombre d'inputs depuis le début
4. pour tout i, $\Pi_i = (P_i, b_i)$ avec P_i un terme et b_i un booléen, $\Pi_1, ..., \Pi_l$ représente le processus en cours

Un observable dans un folding est soit $\mathsf{in}(c)$ soit $\mathsf{out}(c)$.

Exemple 6.4

1. $\mathsf{in}(c, x) \,.\, \mathsf{out}(c, t) \Rightarrow t[x \mapsto \mathrm{att}_0()]$
2. $\mathsf{out}(c, t_1) \parallel \mathsf{in}(c_0, x) \,.\, \mathsf{out}(c_0, t_2) \Rightarrow \mathsf{out}(c), \mathsf{in}(c_0), \mathsf{out}(c_0), t_1, t_2[x \mapsto \mathrm{att}_0(t_1)]$
   $\mathsf{out}(c, t_1) \parallel \mathsf{in}(c_0, x) \,.\, \mathsf{out}(c_0, t_2) \Rightarrow \mathsf{in}(c_0), \mathsf{out}(c_0), \mathsf{out}(c), t_2[x \mapsto \mathrm{att}_0()], t_1$
   $\mathsf{out}(c, t_1) \parallel \mathsf{in}(c_0, x) \,.\, \mathsf{out}(c_0, t_2) \Rightarrow \mathsf{in}(c_0), \mathsf{out}(c), \mathsf{out}(c_0), t_1, t_2[x \mapsto \mathrm{att}_0()]$ (cas inutile)
3. if b then $\mathsf{out}(c, t_1)$ else $\mathsf{out}(c, t_2) \Rightarrow \mathsf{out}(c)$, if b then t_1 else if $\neg b$ then t_2 else error
4. if b then $\mathsf{out}(c_1, t_1)$ else $\mathsf{out}(c_2, t_2) \Rightarrow \mathsf{out}(c_1), \mathsf{out}(c_2)$, if b then t_1 else error, if $\neg b$ then t_2 else error
   if b then $\mathsf{out}(c_1, t_1)$ else $\mathsf{out}(c_2, t_2) \Rightarrow \mathsf{out}(c_2), \mathsf{out}(c_1)$, if b then t_1 else error, if $\neg b$ then t_2 else error

6.4 Semantics of termes

Les termes correspondent intuitivement à des distributions de bitstring calculables en temps polynomial.

Une interprétation $[\![t]\!]^\sigma_\mathcal{M}$ est paramétrée par une valuation $\sigma : \mathcal{X} \to \mathcal{D}$ comme machine de Turing en temps polynomial et d'un modèle calculatoire $\mathcal{M}$ qui interprète les symboles de fonction.

L'interprétation de $[\![t]\!]^\sigma_\mathcal{M}(1^\eta, (\rho_p, \rho_a))$ est

1. $[\![t]\!]^\sigma_\mathcal{M} = \sigma(x)$
2. $[\![m]\!]^\sigma_\mathcal{M} = M_m(1^\eta, \rho_p)$ $\forall m, m', M_m(1^\eta, \rho_p)$ extrait des parties diszointes de ρ_p de longueur η si $m \neq m'$
3. $[\![f(t_1, ..., t_n)]\!]^\sigma_\mathcal{M} = [\![f]\!]^\sigma_\mathcal{M}([\![t_1]\!]^\sigma_\mathcal{M}, ..., [\![t_n]\!]^\sigma_\mathcal{M})$
4. $[\![g(t_1, ..., t_n)]\!]^\sigma_\mathcal{M} = [\![g]\!]^\sigma_\mathcal{M}([\![t_1]\!]^\sigma_\mathcal{M}, ..., [\![t_n]\!]^\sigma_\mathcal{M}, \rho_a)$ avec $g \in \mathcal{G}$

6.5 A first order logic for indistinguishability

On a un des prédicats $t_1, ..., t_n \sim t_{n+1}, ..., t_{2n}$ qui représente l'indistingabilité entre termes.

On aura $[\![t_1, ..., t_n \sim s_1, ..., s_n]\!]^\sigma_\mathcal{M}$ ssi pour toute machine de Turing en temps polynomial A, avec

$$f(\eta) = |\mathbb{P}_\rho(A(([\![t_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho))_{1 \leqslant i \leqslant n}, \rho_a)) - \mathbb{P}_\rho(A(([\![s_i]\!]^\sigma_\mathcal{M}(1^\eta, \rho))_{1 \leqslant i \leqslant n}, \rho_a))|,$$

f est une fonction négligeable.

Définition 6.3 — Formule satisfaite

Une formule est satisfaite par un modèle calculatoire $\mathcal{M}$, noté $\mathcal{M} \vDash \varphi$ si $[\![\varphi]\!]^\sigma_\mathcal{M}$ est vraie pour tout σ.

Exercice 6.1

1. $\nvDash \bot \sim \top$
2. $\vDash n_0 \sim n_0$

3. $\vDash n_0 \sim n_1$ car l'adversaire ne peut pas les distinguer, il ne voit pas les deux
4. $\vDash n_0 \doteq n_1 \sim \bot$ car les noms sont tirés uniformément donc la probabilité d'avoir une collision est $1/2^\eta$ donc c'est négligeable.
5. $\nvDash n_0, n_0 \sim n_0, n_1$
6. $\vDash f(n_0) \sim f(n_1)$ avec $f \in \mathcal{F} \cup \mathcal{G}$
7. $\nvDash \pi_1(\langle (n_0, n_1) \rangle) = n_0 \sim \top$

Définition 6.4 Indistingabilité de protocoles

Deux protocoles P et Q sont indistingables, écrit $P \approx Q$ si pour tout τ, $\vDash \mathrm{fold}(P, \tau) \sim \mathrm{fold}(Q, \tau)$.

Exercice 6.2

1. $\mathsf{out}(c, t_1) \approx \mathsf{out}(c, t_2)$ si $\vDash t_1 \sim t_2$
2. $\mathsf{out}(c, t) \approx \mathsf{null}$ si $\vDash t \sim \mathsf{error}$

6.6 Structural rules

L'indistingabilité est une relation d'équivalence :

$$\frac{}{u \sim u}\ \text{Refl}$$

$$\frac{u \sim v}{v \sim u}\ \text{Sym}$$

$$\frac{u \sim v \qquad v \sim w}{u \sim w}\ \text{Trans}$$

Et il y a beaucoup d'autres règles pour beaucoup d'autres choses.

7 Cryptographic reduction

Pour montrer la sécurité d'un protocole S, on fait des réductions vers un problème H : $S \leqslant_{\mathsf{red}} H$.
Et H peut être IND-CPA, IND-CCA$_1$, DLOG$_\mathcal{G}$, DDH...
Pour faire ça on montre que

$$\forall A \text{ PPTM}, \exists B \text{ PPTM}, \mathrm{Adv}^\eta_S(A) \leqslant \mathrm{Adv}^\eta_h(B).$$

Définition 7.1 IND-CCA$_1$

Soit A une PPTM avec deux oracles :
— un oracle de déchiffrement $O^n_{\mathsf{dec}}(x) := \mathsf{dec}(x, \mathsf{sk}(n))$
— un oracle de chiffrement $O^{n,b,\eta}_{\mathsf{LR}}(m_0, m_1) := \begin{cases} r \xleftarrow{\$} \{0,1\}^\eta, \{m_b\}^r_{\mathsf{pk}(n)} & \text{si } \mathrm{len}(m_0) = \mathrm{len}(m_1) \\ 0 & \text{sinon.} \end{cases}$

A ne peut appeler O_{LR} qu'une seule fois, et ne peut plus appeler O_{dec} après avoir appelé O_{LR}.
L'avantage est alors

$$\mathrm{Adv}^\eta_{\text{IND-CCA}_1}(A) = \left| \mathbb{P}_n \left(A^{O^n_{\mathsf{dec}}(\cdot), O^{n,1,\eta}_{\mathsf{LR}}(\cdot,\cdot)}(1^\eta, \mathsf{pk}(n)) \right) - \mathbb{P}_n \left(A^{O^n_{\mathsf{dec}}(\cdot), O^{n,0,\eta}_{\mathsf{LR}}(\cdot,\cdot)}(1^\eta, \mathsf{pk}(n)) \right) \right|$$

Le schéma est IND-CCA$_1$ si $\forall A, \mathrm{Adv}^\eta_{\text{IND-CCA}_1}(A) \in \mathrm{negl}(\eta)$.

IND-CCA$_1$ est le schéma de règles d'inférences closes :

$$\frac{\mathrm{len}(m_0) = \mathrm{len}(m_1)}{\vec{u}, \{m_0\}^r_{\mathsf{pk}(n)} \sim \vec{u}, \{m_1\}^r_{\mathsf{pk}(n)}}$$

quand :
— n apparaît dans $\vec{v}, m_0, m_1$ dans des sous-termes de la forme $\mathsf{pk}(n), \mathsf{dec}(\_, \mathsf{sk}(n))$
— $r \notin \mathrm{st}(\vec{u}, m_0, m_1)$.

Lemme 7.1

Les règles IND-CCA$_1$ sont correctes dans tout modèle calculatoire où le schéma de chiffrement est IND-CCA$_1$.

Preuve du lemme 7.1.

Soient $\mathcal{M}, \vec{u}, m_0, m_1$ une instance de IND-CCA$_1$.
On suppose que $\mathcal{M} \vDash \text{len}(m_0) = \text{len}(m_1)$ et que $\mathcal{M} \nvDash \underbrace{\vec{u}, \{m_0\}^r_{\mathsf{pk}(n)} \sim \vec{u}, \{m1\}^r_{\mathsf{pk}(n)}}_{\varphi}$.

Soit A tel que $\text{Adv}^\eta_\varphi(A) \notin \text{negl}(\eta)$. On va construire B, un adversaire gagnant contre IND-CCA$_1$.
$B^{\dashrightarrow}(1^\eta, w_{\mathsf{pk}(n)})$ fait :

- lazy sampling de ρ_a et $\rho'_p = \rho_p[n \mapsto 0, r \mapsto 0]$
- calcule $w_{\vec{u}}, w_{m_0}, w_{m_1} = [\![\vec{u}, m_0, m_1]\!]_{\mathcal{M}}(1^\eta, \rho)$
- calcule $w_{\mathsf{LR}} = O^{n,b,\eta}_{\mathsf{LR}}(w_{m_0}, w_{m_0})$
  Comme $\mathcal{M} \vDash \text{len}(m_0) = \text{len}(m_1)$, alors $w_{\mathsf{LR}} = [\![\{w_{m_b}\}^r_{\mathsf{pk}(n)}]\!]_{\mathcal{M}}(1^\eta, \rho)$ (à négligeable près)
- renvoie $A(1^\eta, w_\sigma, w_{\mathsf{LR}}, \rho_a)$

On a $\text{Adv}^\eta_{\text{IND-CCA}_1}(B) = \text{Adv}^\eta_\varphi(A)$, à quantité négligeable près.

Voilà !

Lemme 7.2

Pour tout $s \in \text{st}(\vec{u}, m_0, m_1)$

- soit s est un sous-terme interdit $\mathcal{M}, n, \mathsf{sk}(n)$
- soit B peut calculer en temps polynomial $w_s := [\![s]\!]_{\mathcal{M}}(1^\eta, \rho)$.

Preuve du lemme 7.2.

Montrons le par induction sur $\vec{u}, m_0, m_1$.

- si $s \in \mathcal{N} \setminus \{r, n\}$, on calcule w_s directement de ρ'_p
- si $s \in \{r, n\}$ ok
- si $s \equiv f(t_1, ..., t_n)$ alors
  - si $t_1, ..., t_n$ n'est pas interdit, par hypothèse d'induction on peut calculer $w_{t_i} = [\![t_i]\!]_{\mathcal{M}}(1^\eta, \rho)$ pour tout i. Donc $[\![f]\!]_{\mathcal{M}}(w_{t_1}, ..., w_{t_n})$ (avec éventuellement ρ_a si $f \in \mathcal{G}$).
  - si un des t_i est interdit,
    - $s \equiv \mathsf{pk}(n)$, ok
    - $s \equiv \mathsf{dec}(m, \mathsf{sk}(n))$ et m pas interdit, $w_m = [\![m]\!]\mathcal{M}(1^\eta, \rho)$, on calcule $w_s := O^n_{\mathsf{dec}}(w_m)$

Pour conclure, on remarque que $\vec{u}, m_0, m_1$ ne sont pas interdits, donc on peut calculer en temps polynomial $w_{\vec{u}, w_{m_0}, w_{m_1}}$.

Youpi !

Exemple 7.1

Soit $\varphi = \vDash \mathsf{pk}(n), \{0\}^r_{\mathsf{pk}(n)} \sim \mathsf{pk}(n), \{\mathsf{sk}(n)\}^r_{\mathsf{pk}(n)}$.
Montrer que s'il existe un schéma IND-CCA$_1$, il en existe un autre qui ne satisfasse pas φ.

Proposition 7.3

$$\frac{\forall i, \text{len}(t_1) = \text{len}(s_i)}{\vec{u}, \{t_1\}^{r_1}_{\mathsf{pk}(n)}, ..., \{t_n\}^{r_n}_{\mathsf{pk}(n)} \sim \vec{u}, \{s_1\}^{r_1}_{\mathsf{pk}(n)}, ..., \{s_n\}^{r_n}_{\mathsf{pk}(n)}}$$

où n est utilisé comme IND-CCA$_1$ et $r_1, ..., r_n \notin \text{st}(\vec{u}, t_1, ..., t_n)$.

Définition 7.2 KP-CCA$_1$

Soit A une PPTM avec oracles :

- $O^{m_0, m_1, b, \eta}_{\mathsf{LR}}(m) := r \xleftarrow{\$} \{0,1\}^\eta, \{m\}^r_{\mathsf{pk}(m_b)}$

— $O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_0))}$
— $O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_1))}$

A ne peut appeler O_{LR} qu'au plus une fois, et ne peut plus appeler un O_{dec} après avoir appelé O_{LR}. L'avantage est

$$\begin{aligned}\mathrm{Adv}^{\eta}_{\text{KP-CCA}_1}(A) = &|\mathbb{P}\left(A^{O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_0))},O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_1))},O_{\mathsf{LR}}^{m_0,m_1,0,\eta}}(1^\eta,\mathsf{pk}(m_0),\mathsf{pk}(m_1))\right)\\ &-\mathbb{P}\left(A^{O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_0))},O_{\mathsf{dec}(\cdot,\mathsf{sk}(m_1))},O_{\mathsf{LR}}^{m_0,m_1,1,\eta}}(1^\eta,\mathsf{pk}(m_0),\mathsf{pk}(m_1))\right)|\end{aligned}$$

Le schéma de chiffrement est KP-CCA$_1$ si $\forall A$ PPTM, $\mathrm{Adv}^{\eta}_{\text{KP-CCA}_1}(A) \in \mathrm{negl}(\eta)$.

Les règles de la logique sont

$$\frac{}{\vec{u},\{m\}^r_{\mathsf{pk}(m_0} \sim \vec{u},\{m\}^r_{\mathsf{pk}(m_1)}}$$

7.1 Private authentification

Définition 7.3 Private authentification

$I_X : \nu r, n_I \,.\, \mathsf{out}(c_I, \{\langle \mathsf{pk}_I, n_I\rangle\}^r_{\mathsf{pk}_S})$
$S_X : \nu r_S, n_S \,.\, \mathsf{in}(c_I, x) \,.\, \mathsf{if}\ \pi_1(d) \doteq \mathsf{pk}_A \mathbin{\dot\wedge} \mathrm{len}(\pi_2(d)) \doteq \mathrm{len}(n_S)$
$\qquad\qquad \mathsf{then}\ \mathsf{out}(c_S\{\langle \pi_2(d), n_S\rangle\}^{r_S}_{\mathsf{pk}_A})$
$\qquad\qquad \mathsf{else}\ \mathsf{out}(t_S, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A})$
où $d \equiv \mathsf{dec}(x, \mathsf{sk}_s)$.

On suppose que le chiffrement est IND-CCA$_1$ et KP-CCA$_1$.

Anonymat : $A \neq B$
Avec $C[P] \equiv \nu n_A, n_B, n_S \,.\, \mathsf{out}(c, \langle \mathsf{pk}_A, \langle \mathsf{pk}_B, \mathsf{pk}_S\rangle\rangle) \,.\, P$ on a

$$\underbrace{C[I_A \parallel S_A]}_{L} \approx \underbrace{C[I_B \parallel S_A]}_{R}.$$

Montrons que pour toute trace d'observables τ on a $\mathrm{fold}(L,\tau) \sim \mathrm{fold}(R,\tau)$ valide dans tout modèle $\mathcal{M}$ tel que
— le chiffrement est IND-CCA$_1$ et KP-CCA$_1$
— il existe un symbole $c_{\langle \_,\_\rangle}$ tel que $\mathcal{M} \vDash \mathrm{len}(\langle x,y\rangle) = c_{\langle \_,\_\rangle}(\mathrm{len}(x),\mathrm{len}(y))$.
On admet qu'il suffit de considérer $\tau = \mathsf{out}(c) \,.\, \mathsf{out}(c_I) \,.\, \mathsf{in}(c_I) \,.\, \mathsf{out}(c_S)$.

$$\underbrace{\langle \mathsf{pk}_A, \langle \mathsf{pk}_B, \mathsf{pk}_S\rangle\rangle, \{\langle \mathsf{pk}_A, n_A\rangle\}^r_{\mathsf{pk}_S}}_{\varphi_A}, out^A_A \sim \varphi_B, out^A_B$$

où $out^X_Y \equiv$ if $\pi_1(d_Y) = \mathsf{pk}_X \mathbin{\dot\wedge} \mathrm{len}(\pi_2(d_x)) = \mathrm{len}(n_S)$
$\qquad$ then $\{\langle \pi_2(d_X), n_S\rangle\}^{r_S}_{\mathsf{pk}_X}$
$\qquad$ else $\{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_X}$

$$\dfrac{\dfrac{\dfrac{\dots}{\mathrm{len}(m^A_A) = \mathrm{len}(\langle n_S, n_S\rangle)}}{\varphi_A \,.\, out^A_A \sim \varphi_A, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A}}\ \text{IND-CCA}_1 \qquad \dfrac{}{out^A_A = out^A_A}\ \text{FO}}{\varphi_A \,.\, out^A_A \sim \varphi_A, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A}}\ \text{R}$$

$$\dfrac{\dfrac{\mathsf{ok}}{\varphi_A, out^A_A \sim \varphi_A, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A}} \qquad \dfrac{\dots \qquad \dfrac{\dfrac{\dfrac{}{\mathsf{pk}_A, \varphi_A \sim \mathsf{pk}_A, \varphi_B}}{\varphi_A, n_S, r_S, \mathsf{pk}_A \sim \varphi_B, n_S, r_S, \mathsf{pk}_A}\ \text{Fresh}}{\varphi_A, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A} \sim \varphi_B, \{\langle n_S, n_S\rangle\}^{r_S}_{\mathsf{pk}_A}}\ \text{FA}^2\text{+Dup}}{\varphi_A, \{\langle n_S, n_S\rangle\}^{n_S}_{\mathsf{pk}_A} \sim \varphi_B, out^A_B}\ \text{Trans}}{\varphi_A, out^A_A \sim \varphi_B, out^A_B}\ \text{Trans}$$

Donc $I_A \parallel S_A \approx I_B \parallel S_B$.

Définition 7.4 KCL

$R : \nu n_R \,.\, \mathsf{out}(c_R, n_R)$
$T_X : \mathsf{in}(c_R, y) \,.\, \nu n_T \,.\, \mathsf{out}(c_T, \langle X \oplus m_T, m_T \oplus H(y, K_X)\rangle)$
où K_X est une clé symmétrique du tag X, connue par X et R, et H est un PRF.

Ce protocole est traçable, on peut retrouver l'identité de R en interagissant avec le protocole : $T_A, T_B \not\approx T_A, T_A$.

Définition 7.5 Non-traçabilité

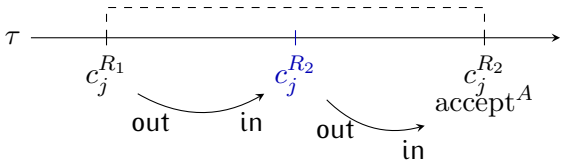
On considère un protocole $P(\vec{K}_{\mathrm{id}})$ paramétré par du matériel de clé secrète $\vec{K}_{\mathrm{id}}$.
P est non-traçable si $\forall N \in \mathbb{N}, \forall M \in \mathbb{N}$,

$$\nu \vec{K}_1, ..., \vec{K}_N \,.\, !_{i\leqslant N} !_{j \leqslant M} P(\vec{K}_i, c_{i,j}) \approx \nu \vec{K}_{1,1}, ..., \vec{K}_{N,M} !_{i\leqslant N} !_{j\leqslant M} P(\vec{K}_{i,j}, c_{i,j}).$$

La première partie est le scénario du monde réel, dans lequel on a plein de scénarios pour chaque identité, et le deuxième est le scénario du monde idéal, dans lequel chaque identité n'utilise qu'une seule fois le protocole.

8 Hash-lock

On a
$T(A,i) : \nu n_{T,i} \,.\, \mathsf{in}(c^T_{A,i}, x) \,.\, \mathsf{out}(c^T_{A,i}, \langle n_{T,i}, H(\langle x, n_{T,i}\rangle, K_A)\rangle)$
$R(j) : \nu n_{R,j} \,.\, \mathsf{in}(c^{R_1}_j, \_) \,.\, \mathsf{out}(c^{R_1}_j, n_{R,j})$
$\quad \mathsf{in}(c^{R_2}_j, y) \,.$
$\quad \mathsf{if} \bigvee_{A\in I} \pi_2 y = H(\langle n_{R,j}, \pi_1 y\rangle, k_A)$
$\quad \mathsf{then}\ \mathsf{out}(c^{R_2}_j, ok)$
$\quad \mathsf{else}\ \mathsf{out}(c^{R_2}_j, ko)$
Et on lance $\nu(k_A)_{A\in I} \,.\, (!_{A\in I} !_{\leqslant M} T(A,\cdot)) \parallel (!_N R(j))$.
On veut que $\forall \tau, \forall c^{R_1}_j < c^{R_2}_j \in \tau, \mathrm{accept}^A @ c^{R_2}_j, \exists c^{R_1}_j < c^T_{A,i} < c^{R_2}_j$,



8.1 Préliminaires

Soit τ_1, τ_2 des traces d'observables.
— $\tau_1 \leqslant \tau_2$ ssi $\exists \tau'_1, \tau_2 = \tau_1 @ \tau'_1$
— $\tau \Diamond c$ si τ finit par une output sur c, i.e. $\exists \tau', \tau = \tau', out(c)$.
Soit $T_{IO} = \{\tau \,|\, \forall c, \tau \Diamond c \Rightarrow \exists \tau', \tau = \tau', \mathsf{in}(c), \mathsf{out}(c)\}$.

Soit P un protocole action-déterministe, $\tau \in T_{IO}$ tel que τ contient j inputs : $\mathrm{fold}(P, \tau) = t_1, ..., t_n$.
— $\mathsf{out}_P @ \tau = \begin{cases} t_n & \text{si } \exists c, \tau \Diamond c \\ \mathsf{empty} & \text{sinon} \end{cases}$
— $\mathrm{frame}_P @ \tau = \begin{cases} \langle \mathrm{frame}_P @ \mathrm{pred}(\tau), \mathsf{out}_P @ \tau & \text{si } \tau \neq \varepsilon \\ \mathsf{empty} & \text{sinon} \end{cases}$
— $\mathrm{input}_P @ \tau = \mathrm{att}_j(\mathrm{frame}_P @ \mathrm{pred}(\tau)$ si $\exists c, \tau \Diamond c$
— $\mathrm{accept}^A_P @ \tau = (\pi_2(\mathrm{input} @ \tau) \doteq H(\langle n_{R,j}, \pi_1(\mathrm{input} @ \tau)\rangle, k_A))$ quand $\tau \Diamond c^{R_2}_j$
Le protocole HL garantie l'authentification de T par R si

$$\forall \tau \in T_{IO}, \forall A \in I, \forall \tau_1 \Diamond c^{R_1}_j, \tau_3 \Diamond c^{R_2}_j, \tau_1 < \tau_2 \leqslant \tau$$

$$\text{accept}^A@\tau_3 \Rightarrow \dot{\bigvee}_{\substack{\tau_2 \Diamond c^T_{A,i} \\ \tau_1 < \tau_2 < \tau_3}} \text{out}@\tau_1 \doteq \text{in}@\tau_2 \mathbin{\dot{\wedge}} \text{out}@\tau_2 \doteq \text{in}@\tau_3 \sim \text{true}$$

8.2 Termes booléens

Définition 8.1 — Terme booléen

Un terme booléen est un terme t tel que pour tout modèle calculatoire $\mathcal{M}$, pour tous σ, η, ρ,

$$[\![t]\!]^\sigma_{\mathcal{M}}(1^\eta, \rho) \in \{0, 1\}.$$

Exemple 8.1

is_even_1 est un terme booléen.
Si $\mathcal{C} = \{\mathcal{M} \text{ modèle calculatoire} \mid [\![\text{is\_even}]\!]_{\mathcal{M}} \text{ est ce à quoi on s'attend}\}$, $\text{is\_even}(x) \mathbin{\dot{\vee}} (x = x + 1)$ est un terme booléen relatif à $\mathcal{C}$.

Définition 8.2 — Jugement d'accessibilité

Un jugement d'accessibilité est un élément $\Gamma \vdash t$ où Γ est une séquence de termes $t_1, ..., t_n$, et $t, t_1, ..., t_n$ sont des termes booléens.

Définition 8.3 — Jugement valide

Un jugement $\Gamma \vdash t$ est valide $t_1 \Rightarrow ... \Rightarrow t_n \Rightarrow t \sim \text{true}$ est valide.

Proposition 8.1

Les règles propositionnelles standard sont correctes relativement à ces jugements.

Exemple 8.2

$$\frac{\Gamma, t_1 \vdash t \qquad \Gamma, t_2 \vdash t}{\Gamma, t_1 \vee t_2 \vdash t}$$

Remarque 8.1

$\vee$ n'est pas la même chose que $\dot{\vee}$.
$t_1 \sim \text{true} \vee t_2 \sim \text{true}$ valide implique $\emptyset \vdash t_1 \mathbin{\dot{\vee}} t_2$ valide, mais l'inverse n'est pas vrai.

Troisième partie

Bruno Blanchet

9 Game-based proofs

On utilise le modèle calculatoire, qui est une machine de Turing probabiliste. Ce n'est qu'un modèle, on ne prend pas en compte toutes les informations qui pourraient fuiter autre part sur les calculs.

Une approche consiste à utiliser un théorème de computational soundness comme dans la partie Comon, mais il faut des hypothèses supplémentaires (comme pas de cycle de clés).

Une autre approche consiste à faire les preuves directement dans le modèle calculatoire. C'est ce qu'on va faire dans cette partie.

9.1 Provable security

On va faire des réductions entre le problème de casser un schéma cryptographique donné et des problèmes difficiles connus (factorisation d'entiers, logarithme discret, 3-SAT...).

Exemple 9.1 Problème RSA

Donnés $n = pq$, e et $y \in \mathbb{Z}_n^*$ on veut trouver x tel que $y = x^e \mod n$.

Exemple 9.2 Problème RSA flexible

Donnés $n = pq$, et $y \in \mathbb{Z}_n^*$ on veut trouver x et $e > 1$ tel que $y = x^e \mod n$.

Exemple 9.3 Problème Diffie-Hellman calculatoire

Donnés un groupe $\mathbb{G} = \langle g \rangle$ d'ordre q et $X = g^x$, $Y = g^y$, on veut trouver $Z = g^{xy}$.

On caractérise la qualité d'un attaquant par sa probabilité à trouver la solution. La difficulté du problème est alors le max des probabilités de succès des attaquants en temps t.

Exemple 9.4 Problème Diffie-Hellman décisionel

Donnés $\mathbb{G} = \langle g \rangle$ d'ordre q, $X = g^x$, $Y = g^y$ et $Z \in \mathbb{G}$, on veut savoir si trouver $Z = g^{xy}$.

Proposition 9.1

DDH $\leqslant$ CDH $\leqslant$ DLP

Définition 9.1 Indistingabilité

Soient $\mathcal{D}_0$ et $\mathcal{D}_1$ deux distributions sur un ensemble fini X.

— $\mathcal{D}_0$ et $\mathcal{D}_1$ sont parfaitement indistingables si leur distance est nulle :

$$\mathrm{Dist}(\mathcal{D}_0, \mathcal{D}_1) := \sum_{x \in X} |\mathbb{P}_{a \in \mathcal{D}_1}(a = x) - \mathbb{P}_{a \in \mathcal{D}_0}(a = x)| = 0.$$

— $\mathcal{D}_0$ et $\mathcal{D}_1$ sont statistiquement indistingables si leur distance est négligeable.

Définition 9.2 Indistingabilité calculatoire

Soient $\mathcal{D}_0$ et $\mathcal{D}_1$ deux distributions sur un ensemble fini X.

— Un distingueur $\mathcal{A}$ entre $\mathcal{D}_0$ et $\mathcal{D}_1$ est caractérisé par son avantage

$$\mathrm{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A}) := \mathbb{P}_{a\in\mathcal{D}_1}(\mathcal{A}(a)=1) - \mathbb{P}_{a\in\mathcal{D}_0}(\mathcal{A}(a)=1).$$

— La distingabilité calculatoire de $\mathcal{D}_0$ et $\mathcal{D}_1$ est

$$\mathrm{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(t) := \max_{|\mathcal{A}|\leqslant t}\{\mathrm{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A})\}.$$

Théorème 9.2

Pour tout attaquant $\mathcal{A}$, $\mathrm{Adv}^{\mathcal{D}_0,\mathcal{D}_1}(\mathcal{A}) \leqslant \mathrm{Dist}(\mathcal{D}_0,\mathcal{D}_1)$.

Théorème 9.3 Argument hybride

$\forall t, \mathrm{Adv}^{\mathcal{D}_A,\mathcal{D}_B}(t) \leqslant n \times \mathrm{Adv}^{\mathsf{ddh}}_{\mathbb{G}}(t + 2(n-1)_{\tau_{\mathsf{exp}}})$

9.2 Encryption

Définition 9.3 OW-CPA

Définition 9.4 IND-CPA

9.3 Game-based proofs

En général il faut utiliser plusieurs problèmes sous-jacents, donc on va diviser la preuve en plusieurs étapes qui utilisent chacunes une réduction.

9.4 Signatures

Définition 9.5 EUF-NMA

Définition 9.6 EUF-CMA

Définition 9.7 SUF-CMA

10 CryptoVerif

10.1 Proof technique

Avec C un contexte, $C[Q]$ est le terme dans lequel on a remplacé tous les $[]$ par Q. Par exemple si $C_0 = [] \,\|\, Q_0$, alors $C_0[Q_1] = Q_1 \,\|\, Q_0$.

Index des définitions

Index des résultats