# Cours25

## 1 Signatures Digitales

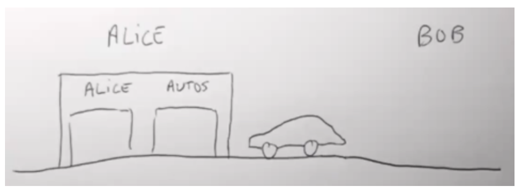


Figure 1: Concessionnaire d'Alice - Arnaud 2021

### 1.1 Rappel

**Non-Répudiation**: ne pas pouvoir dire que l'on a pas envoyé le message.

### 1.2 Signature

**msg**: Le message à envoyer
**sig**: La signature qui est créé à partir du message: msg$\xrightarrow{f(K_{prv},\, g)} sig$

Bob envoie une clef publique, ainsi qu'un message **concaténé** avec la signature (msg, sig). Ensuite, Alice a une algorithme de vérification qui prends ces trois valeurs et peux retourner Vrai ou Faux.

$$\text{Bob} \quad \begin{array}{|c|}\hline \text{msg} \\ \hline \text{sig} \\ \hline \end{array} \quad \begin{array}{c} \xrightarrow{K_{pub}} \\ \xrightarrow{(msg, sig)} \end{array} \quad \text{Alice} \quad \xrightarrow{ver(K_{pub},\, msg,\, sig)} \text{True/False}$$

À noter, que les variables utilisé lors de la vidéo étaient:
msg = x
sig = s

## 1.3 Signature avec RSA

Légende:
n = p*q où p et q sont deux grands nombre premier
d = $e^{-1} mod(\phi(n))$
x = message
s = signature

| Bob | | Alice |
|---|---|---|
| $K_{prv} = d$ | $\xrightarrow{RSAK_{pub}(n,e)}$ | |
| $S = Sig_{K_{prv}}(x^d mod(n))$ | $\xrightarrow{(x,s)}$ | $x' = S^e mod(n)$ |
| | | $x' = xmod(n) \rightarrow True$ |
| | | $x' \neq xmod(n) \rightarrow False$ |

## 1.4 Signature - Suite

Comment faire pour un algorithme de signature qui prend une entrée de taille maximale pour un message de taille N.

### 1.4.1 Naïf

$$K_{prv} \longrightarrow \left.\begin{array}{|c|c|c|c|c|}\hline x_1 & x_2 & x_3 & ... & x_n \\ \hline \downarrow & \downarrow & \downarrow & & \downarrow \\ \text{sig} & \text{sig} & \text{sig} & & \text{sig} \\ \downarrow & \downarrow & \downarrow & & \downarrow \\ \hline s_1 & s_2 & s_3 & ... & s_n \\ \hline \end{array}\right\}$$

**Aspects problématiques**
1- Pas efficace (nombre d'ops)
2- *Overhead* (quantité d'information)
3- Sécurité (message chiffré facilement modifiable)

### 1.4.2 Smarter way

$h$ : est une fonction de hachage

$$\underbrace{\begin{array}{|c|c|c|c|c|}\hline x_1 & x_2 & x_3 & ... & x_n \\ \end{array}}$$
$$\boxed{h}$$
$$\downarrow$$
$$\xrightarrow{K_{prv}} \boxed{sig}$$
$$\downarrow$$
$$\boxed{S}$$

Voici un exemple de communication:

$$\begin{array}{ccc} \text{Bob} & & \text{Alice} \\ z = h(x) & \xrightarrow{K_{pub}} & \\ S = Sig_{K_{prv}}(z) & \xrightarrow{(x,s)} & z' = h(x) \\ & & Ver_{K_{pub}}(z', s) = \text{True / False} \end{array}$$

# 2 Fonction de Hachage

| | |
|---|---|
| 1- One-Wayness / Résistance pré-image | h(x) $\neq$ x |
| 2- Résistance aux collisions faibles | $x_1,\ x_2 =?\ t.q.\ h(x_1) = h(x_2)$ |
| 3- Résistance aux collisions fortes | $x_1 =?,\ x_2 =?\ t.q.\ h(x_1) = h(x_2)$ |

Usage des hash: Mots de passe
Attaque: Rainbow Table (contré par le salage)

## 2.1 Collisions faibles

| Bob | | | | Alice |
|---|---|---|---|---|
| $z = h(x_1)$<br>$S = Sig_{K_{prv}}(z)$ | $\xrightarrow{K_{pub}}$<br>$\xrightarrow{(x_1,s)}$ | Oscar subs | $\xrightarrow{x_2,s}$ | $z' = h(x_2)$ |
| | | | | $Ver_{K_{pub}}(z', s) = \text{True}$ |

$|h()|\}\#$ bits d'output possibilités.
**Principe du pigeonnier**, si on essaie $2^n + 1$ options, il y aura forcément une collision.
Dans la réalité: $n = 80,\ \ 2^{80}$

## 2.2 Collisions Fortes

$x_1$= Transfer 10$ into Oscar's account
$x_1$= Transfer 10 000$ into Oscar's account

Oscar veut envoyer $x_2$ à Alice, en prétendant que le message vienne de Bob.
$Bob \xrightarrow{(x_1,s)} Oscar \xrightarrow{(x_2,s)} Alice$

Son but est donc de modifier le message en **gardant la sémantique**. par exemple en ajoutant des espaces, changeant la tabulation, etc.
Si il y a 64 positions où il est possible de faire une modification $\rightarrow 2^{64}$ versions.
**Collision Faible**: $2^80$ messages pour trouver une collision (garantie) si $|h(x)| = 80 = n$.
**Collision Forte**: $2^{40}$ messages

## 3 Birthday paradox et collisions

https://en.wikipedia.org/wiki/Birthday_problem

P($\varnothing$ collision entre 2 personnes) = (1-$\frac{1}{365}$)
P($\varnothing$ collision entre 3 personnes) = (1-$\frac{1}{365}$)(1-$\frac{2}{365}$)
t = 366  P($\varnothing$ collision pour t=366) = 0
P(au moins 1 collisions) = 1- P($\varnothing$ collision)  = 1- [$(1-\frac{1}{365})(1-\frac{t-1}{365})$]
t=23,  p > 50%
t=40,  p > 90%

Il est donc très probable d'avoir une collision même avec un nombre d'essais beaucoup plus petit qu'on ne le penserait.
Voici aussi, avec la fonction de hachage (sur $2^n$ bits)
P($v\varnothing$ collision) $= (1-\frac{1}{2^n})(1-\frac{2}{2^n})...(1-\frac{t-1}{2^n}) = \prod_{i=1}^{t-1}(1-\frac{i}{2^n})$

Rappel: **Série de Taylor**
$e^{-x} = 1 - x + \frac{x^2}{2!} - \frac{x^3}{3!} + ...$ x << 1
Quand x < 1, tout les termes $\frac{x^2}{2!} - \frac{x^3}{3!} + ...$ sont négligible.
On approxime donc à: $e^{-x} \approx 1 - x$

P($\varnothing$ collision) $\approx \prod_{i=1}^{t-1} e^{-\frac{1}{2^n}} = e^{\frac{-1+2+3+...+t-1}{2^n}}$
Rappel: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$
P($\varnothing$ collision) $\approx e^{\frac{-t(t-1)}{2*2^n}}$ $\rightarrow$ déterminer la valeur de t
$\lambda \approx 1 - e^{\frac{-t(t-1)}{2^{n+1}}}$ $\rightarrow$ lambda est la probabilité qu'il y aie une collision
t(t-1) $\approx 2^{n+1} \ln \frac{1}{1-\lambda}$
t(t-1) $\approx t^2$
t $\approx \sqrt{2^{n+1} \ln \frac{1}{1-\lambda}}$
t $\approx 2^{\frac{n+1}{2}} \sqrt{\ln \frac{1}{1-\lambda}}$

Donc on peux voir pourquoi pour les collisions fortes vs faibles: (le divisé par 2)
$2^{80} \rightarrow 2^{40}$