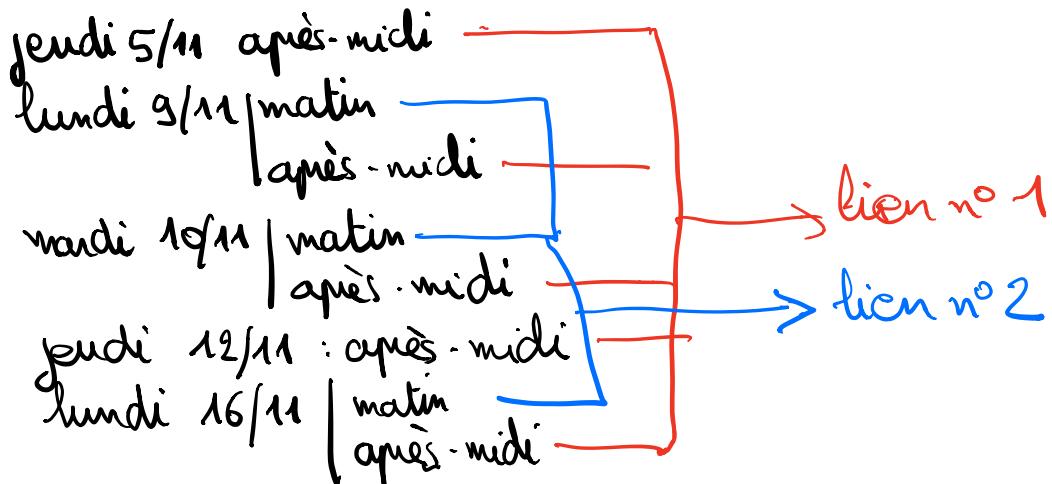


Cryptographie : compléments et applications

Examen "Méthodes algorithmiques pour la crypto"
→ jeudi 12/11 à 9^h30



Thèmes du cours :

- 1) Signature électronique et fonctions de hachage
- 2) Diffie-Hellman et courbes elliptiques
- 3) Authentification et zero-knowledge

ch 1: Signature électronique et fonctions de hachage

I] Signature RSA

① Rappels sur RSA

- on choisit e exposant impair (ex: $e=3, e=17, \dots$)
- deux entiers premiers p et q (choisis aléatoirement)

tels que $\boxed{\text{pgcd}(e, \varphi(n)) = 1}$

avec $\varphi(n) = \underbrace{(p-1)(q-1)}_{\text{pair}}$

Comment choisir p aléatoirement ?

- choisir un entier a aléatoire impair
- tester si a est premier
 - si oui \square
 - Si non : soit on recommence soit on fait $a := a+2$

Comment tester si a est premier ?

- Test de Fermat : On teste, pour différents entiers x , si

$$\rightarrow \boxed{x^{a-1} \equiv 1 \pmod{a}}$$

Rappel : a premier $\Rightarrow \forall x \neq 0 \pmod{a}$

$$a \mid x^{a-1} - 1$$

(petit théorème de Fermat)

Réiproquement, est-il vrai que

$(\forall x \neq 0 \pmod{a}, \boxed{x^{a-1} \equiv 1 \pmod{a}})$

~~\Rightarrow~~ a premier ?

Non : il existe des entiers a non premiers qui pourtant passent le test de Fermat ($\forall x$)

Les entiers a s'appellent les nombre de Carmichael

Remarque: On peut démontrer qu'il existe une infinité de nombres de Carmichael

- Test de Solovay. Strassen | raffinements des
- Test de Miller. Rabin | test de Fermat

Remarque: Pour effectuer ces tests, le calcul se ramène essentiellement à des exponentiations modulaires

Savoir si $x^{a-1} = 1 \pmod{a}$
nécessite de calculer $\boxed{x^{a-1} \pmod{a}}$

on utilise "square and multiply"
dont la complexité
est $O(k^3)$ où k est
la taille (= nb de bits) de

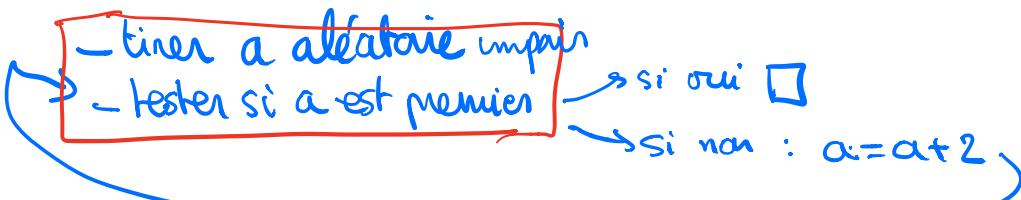
x et de a

Remarque: L'avantage de Solovay. Strassen et Miller. Rabin sur Fermat)
algorithme probabilité

est que cette fois on peut aussi garantir la primalité avec une probabilité d'erreur aussi petite qu'on veut.

Remarque: Il existe des algorithmes déterministes qui donnent une garantie de primalité avec 100% de certitude

ex: Algorithme AKS (AKS, Keyal, Sareen, 2002)



→ question : quelle est la fréquence des nombres premiers

$$\pi(x) = \text{Card} \left\{ n \in \mathbb{N}^*, \begin{array}{l} n \text{ est premier} \\ \text{et } 1 \leq n \leq x \end{array} \right\}$$

Théorème des Nombres Premiers (Hadamard, de la Vallée Poussin)

$$\pi(x) \underset{x \rightarrow \infty}{\sim} \frac{x}{\ln x}$$

Interprétation : la "densité" des nombres premiers au voisinage de x

$$\text{est } \approx \frac{1}{\ln x}$$

ex : Si $x \approx 2^{512}$

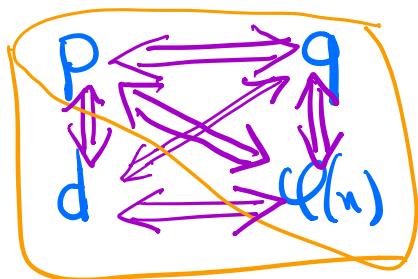
$$\frac{1}{\ln x} \approx \frac{1}{\ln 2^{512}} \approx \frac{1}{512 \ln 2} \approx \frac{1}{300}$$

Consequence : trouver un entier p premier de 512 bits nécessitera en moyenne d'effectuer ≈ 150 tests (puce de Miller-Rabin)

Une fois les paramètres p et q choisis

[clé publique] : $n (= p \times q)$, e
[clé secrète] : $P, q, d, \varphi(n)$
où d (exposant secret) est obtenu
par la formule : $d = e^{-1} \bmod \varphi(n)$
algo d'Euclide étendu
(complexité $O(k^2)$)
où k est la taille
en bits de $\varphi(n)$

Remarque : on peut montrer que la connaissance
d'un des 4 nombres $p, q, d, \varphi(n)$
permet d'en déduire les 3 autres



$$q = \frac{n}{p} \xrightarrow{\text{connu}}$$

$$\begin{aligned}\varphi(n) &= (p-1)(q-1) \\ &= (p-1)\left(\frac{n}{p}-1\right)\end{aligned}$$

$$d = e^{-1} \bmod \varphi(n)$$

Si on connaît $\varphi(n)$, peut-on
retrouver p et q ?

$$\begin{aligned}\varphi(n) &= (p-1)(q-1) \\ &= \underbrace{pq}_{=n} - (p+q) + 1\end{aligned}$$

$$p+q = n - \varphi(n) + 1$$

$$\begin{cases} p+q = n - \varphi(n) + 1 \\ p \times q = n \end{cases}$$

\Rightarrow p et q sont les solutions de l'équation

$$(X-p)(X-q)=0, c'est à dire X^2 - (p+q)X + pq = 0$$

ou encore : $X^2 - (n - \varphi(n) + 1)X + n = 0$

$$\{p, q\} = \frac{n - \varphi(n) + 1 \pm \sqrt{\Delta}}{2}$$

où $\Delta = (n - \varphi(n) + 1)^2 - 4n$

Reste à montrer que si on connaît d, alors on connaît p, q, $\varphi(n)$

Supposons qu'on connaît d

et prenons un entier x arbitraire

et calculons $y = \frac{x^{ed-1}}{2}$ entier mod n

$$(ed = 1 \text{ mod } \varphi(n))$$

On sait que $y^2 = x^{ed-1} \text{ mod } n$ avec $ed-1 = \lambda \varphi(n)$

$$\Rightarrow y^2 = x^{\lambda \varphi(n)} \text{ mod } n \quad (\text{car } ed=1 \text{ mod } \varphi(n))$$

$$\boxed{y^2 = 1 \text{ mod } n} \quad (\text{car } x^{\varphi(n)} = 1 \text{ mod } n) \quad \text{Théorème d'Euler}$$

$$\rightarrow \boxed{y^2 = 1 \text{ mod } n} \iff \begin{cases} y = 1 \text{ mod } n & (\text{et } y = 1 \text{ mod } p) \\ \text{ou} \\ y = -1 \text{ mod } n & (\text{et } y = -1 \text{ mod } p) \\ \text{ou} \\ y \equiv \alpha \text{ mod } n & (\text{et } y = 1 \text{ mod } p) \\ \text{ou} \\ y \equiv -\alpha \text{ mod } n & (\text{et } y = -1 \text{ mod } p) \end{cases} \quad \begin{cases} y = 1 \text{ mod } q \\ \text{et } y = -1 \text{ mod } q \end{cases}$$

α est un entier tel que
 $\begin{cases} \alpha \equiv 1 \text{ mod } p \\ \alpha \equiv -1 \text{ mod } q \end{cases}$ (th. des restes chinois)

 $y^2 = 1 \text{ mod } n \iff \begin{cases} y^2 \equiv 1 \text{ mod } p \\ \text{et} \\ y^2 \equiv 1 \text{ mod } q \end{cases} \iff \begin{cases} y \equiv 1 \text{ mod } p \\ \text{ou} \\ y \equiv -1 \text{ mod } p \\ \text{et} \\ y \equiv 1 \text{ mod } q \\ \text{ou} \\ y \equiv -1 \text{ mod } q \end{cases}$

Si le y qu'on aurait obtenu est $\neq \pm 1$

alors par exemple $\begin{cases} y \equiv 1 \text{ mod } p \\ \text{et} \\ y \equiv -1 \text{ mod } q \end{cases} \Rightarrow \begin{cases} y-1 \equiv 0 \text{ mod } p \\ \text{et} \\ y-1 \not\equiv 0 \text{ mod } q \end{cases}$

et $\boxed{\text{pgcd}(y-1, n) = p}$

divisible par p
mais pas par q

$= pq$

Si le y obtenu est ± 1

on recommence avec un autre choix de α
jusqu'à obtenir un $y \neq \pm 1$

Clé secrète : $p, q, d, \varphi(n)$

② Sécurité de RSA

a) Attaquant qui cherche à retrouver la clé secrète

→ factoriser : à partir de n , retrouver p et q

Meilleur algorithme connu :

GNFS (General Number Field Sieve , 1994)

complexité :

$$O(e^{c(\ln n)^{1/3} (\ln \ln n)^{2/3}})$$

$$c \approx 1,92$$

↓
algorithme
sous-exponentiel

ex: sécurité de 2^{80}

$$e^{c(\ln n)^{1/3} (\ln \ln n)^{2/3}}$$

$$2^{80}$$

supposé
suffisant
pour empêcher
les attaques

On peut aussi obtenir la condition

$$n \geq 2^{1024}$$

$$|\ln n| \geq 1024 \text{ bits}$$

→ 2048 ou 4096 bits
($\Leftrightarrow 2^{128}$)

Calculs faisables dans le domaine ciel	Calculs faisables dans le domaine gauvemental	Calculs infaisables
$\frac{2020}{20 \times 18 \text{ mois}} \approx 2^{70}$ 2^{90}	$\approx 2^{90}$ 2^{110}	2^{128} 2^{148}

Loi de Moore : la puissance de calcul double tous les 18 mois

b) Attaquant qui veut "juste" inverser la fonction RSA

$$f : \begin{cases} \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z} \\ x \mapsto y = x^e \text{ mod } n \end{cases}$$

Théorème : Si p et q sont premiers, $p \neq q$, $n = pq$
 (Euler) e est un entier, avec $\text{pgcd}(e, \varphi(n)) = 1$
 $d = e^{-1} \text{ mod } \varphi(n)$

alias $f : \begin{cases} \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z} \\ x \mapsto y = x^e \text{ mod } n \end{cases}$

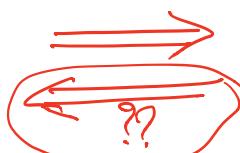
est une bijection

et $f^{-1} : \begin{cases} \mathbb{Z}/n\mathbb{Z} \rightarrow \mathbb{Z}/n\mathbb{Z} \\ y \mapsto x = y^d \text{ mod } n \end{cases}$

→ Pb de la racine e -ième mod n :

à partir de $y (= x^e \text{ mod } n)$, peut-on retrouver x ?

Pb de la factorisation

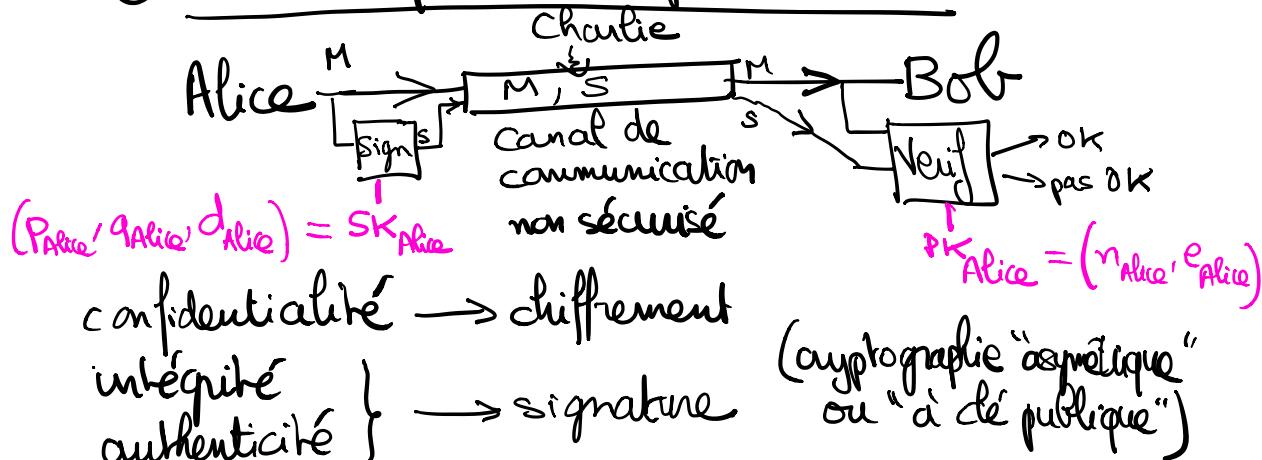


Pb d'extraire une racine e -ième



III] Signature RSA et fonctions de hachage

① Problématique de la signature RSA



Idee "naïve" pour produire la signature s du message M

$$s = M^d \bmod n \quad M = \text{entier quelconque}$$

pb 1 : on doit restreindre l'espace des messages

$$(\text{sinon : } \underline{M} \text{ et } \underline{M}' = M + n$$

auraient la même signature

$$\text{car } s' = M'^d \bmod n = (\underline{M} + n)^d \bmod n$$

(deux messages différents ayant la même signature)

$$= M^d \bmod n = s$$

→ On doit imposer par exemple que les messages doivent être tels que

$$0 \leq M < n$$

($f : x \mapsto x^e \bmod n$ est une bijection sur $\{0, \dots, n-1\}$)

Rémarque: on pouvait quand même définir un mécanisme pour signer des messages de taille quelconque, en se ramenant au cas de messages $< n$

Ex: pour M quelconque, on peut écrire M "en base n ", c'est à dire :

$$M = \dots + \underbrace{M_2}_{0 \leq M_2 < n} n^2 + \underbrace{M_1}_{0 \leq M_1 < n} n + \underbrace{M_0}_{0 \leq M_0 < n}$$

puis signer successivement M_0, M_1, M_2, \dots



$$\begin{aligned} S_0 &= \text{signature de } M_0 \\ S_1 &= \text{signature de } M_1 \\ S_2 &= \frac{\dots}{M_2} \end{aligned}$$

pb: Charlie peut par exemple
 - supprimer un élément du message (M_i)
 - permuter des éléments ($M_i \leftrightarrow M_j$)

⇒ il faut en plus protéger la "logique qui lie les blocs entre eux" ~~entre~~ du point de vue de l'intégrité
 ⇒ il faut ajouter d'autres signatures

pb : compliqué à mettre en place
 - calcul de signature est long lent car il faut appliquer autant de fois RSA qu'il y a de blocs

pb 2: Sécurité de la signature "naïve" RSA
 supposons qu'Alice signe deux messages M et M'

$$S = M^d \bmod n$$

$$S' = M'^d \bmod n$$

$$\boxed{S \times S' = (M \times M')^d \bmod n}$$

$$S'' = M''^d \bmod n$$

Remarque: Si Alice n'a envoyé que M, S à Bob

Charlie peut envoyer

$$\frac{M \times M}{M^2}, \frac{S \times S}{S^2} \text{ à Bob}$$

(à quelque)

$$\lambda^e \times M, \lambda \times S$$

$$\text{signature} = (\lambda^e \times M)^d = \underbrace{(\lambda^e)^d}_{=\lambda} \times \underbrace{M^d}_{=S} = \lambda \times S$$

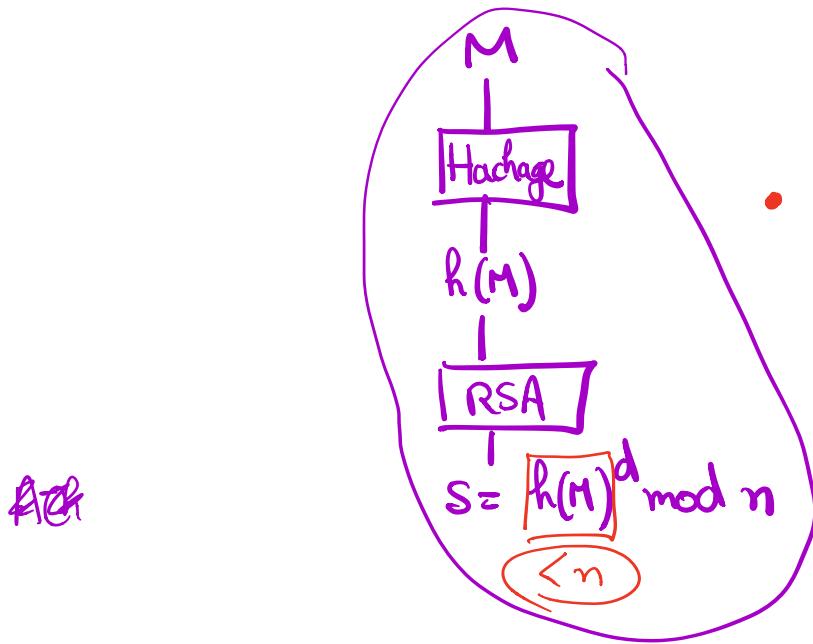
Remarque: Si Alice n'a rien envoyé du tout (!)

$$\text{Alice } \xrightarrow{M=1, S=1} \text{Bob}$$

$$\text{Charlie } \xrightarrow{\lambda^e, \lambda} \text{Bob}$$

signature
message

② Paradigme "Hash and Sign"



Avec

Remarque : (1) et (2)
peuvent être vus
comme contradictoires

En effet si (dans (2)) l est
très petit, (1) ne peut pas
être vrai

ex: $l = 10$ \rightarrow l'ensemble
d'image de h
a un cardinal $2^{10} = 1024$

Hypothèses

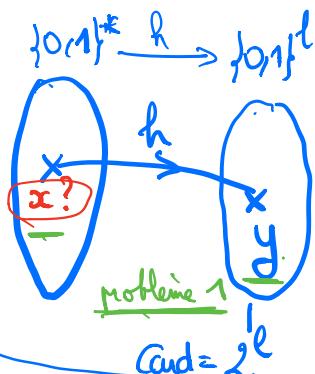
- $h: \{0,1\}^* \rightarrow \{0,1\}^l$
avec l fixé
- $$\{0,1\}^* = \bigcup_{n \geq 0} \underbrace{\{0,1\}^n}_{\text{messages de } n \text{ bits}}$$

- Il ne doit pas arriver, en pratique, que deux messages M et M' ($M \neq M'$) aient le même "haché"
(c'est à dire que $h(M) = h(M')$)
 - avec l suffisamment petit pour garantir que $\forall M, h(M) < n$
- (1)
- (2)

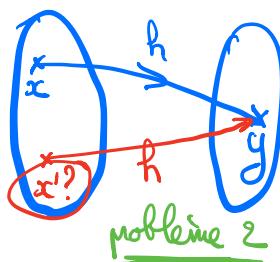
(3) Fonctions de hachage

Def: Une fonction $h: \{0,1\}^* \rightarrow \{0,1\}^l$ (l fixé)
est appelée fonction hachage ssi elle vérifie
les 3 propriétés suivantes

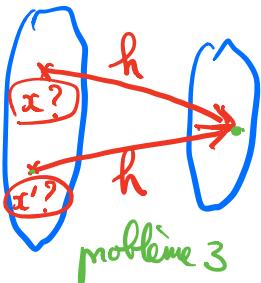
P1 : h est à sens unique (one-way)
 pour tout $y \in \{0,1\}^l$, il est
calculatoirement difficile de trouver
 $x \in \{0,1\}^k / h(x) = y$



P2 : h est à collisions faibles difficiles
 (second - préimage résistant) :
 pour tout $y \in \{0,1\}^*$, tel que $h(x) = y$,
 il est calculatoirement difficile de
 trouver $x' \in \{0,1\}^*$ tel que
 $\begin{cases} x' \neq x \\ h(x') = y \end{cases}$



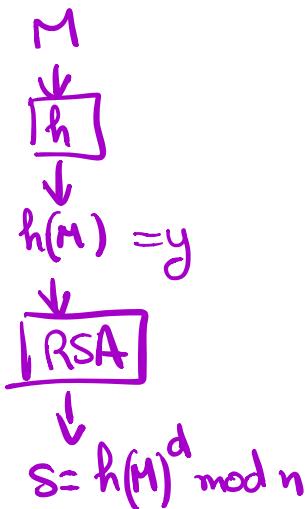
P3 : h est à collisions fortes difficiles
 (collision - résistant) :
 il est calculatoirement difficile de
 trouver $x \in \{0,1\}^*$ et $x' \in \{0,1\}^*$
 tels que $\begin{cases} x' \neq x \\ h(x) = h(x') \end{cases}$



Savoir résoudre le pb 1 \Rightarrow Savoir résoudre le pb 2 \Rightarrow Savoir résoudre le pb 3

P1 \iff P2 \iff P3

Remarque : Importance de P1 pour la signature



Si P1 n'était pas vrai
 un attaquant pourrait
 - choisir s aléatoirement
 - calculer $y = s^e \bmod n$
 [puis trouver x tel que
 $h(x) = y$]

~~Bob~~
 Charlie $\xrightarrow{x, s}$ Bob

$$\begin{aligned}
 & \text{Le message } M = x \\
 & \text{aurait bien pour signature} \\
 & \quad \cancel{s} = h(M)^d \bmod n \\
 & \quad = y^d \bmod n \\
 & \quad = (s^e)^d \bmod n \\
 & \quad \cancel{s} = s
 \end{aligned}$$

• Impératifs de P2

Si P2 n'était pas vrai

à partir de la connaissance

$$M \text{ et } s = h(M)^d \bmod n$$

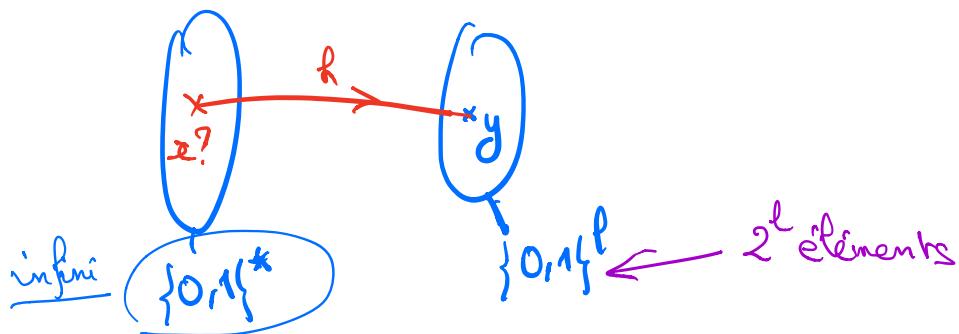
Charlie pourrait trouver M' tel que $\underbrace{h(M)}_{\neq M} = h(M') = y$

Charlie $\xrightarrow{M', s}$ Bob
 avec s signature valide de M'

Alice $\xrightarrow{M, s}$ Bob

- Pareil pour P3

Complexité d'un algorithme générique pour résoudre le pb 1



Algorithme : Générer x aléatoirement jusqu'à ce que l'on obtienne $h(x) = y$ (y fixé)

Probabilité d'avoir $h(x) = y$

$$P = \frac{1}{2^l}$$

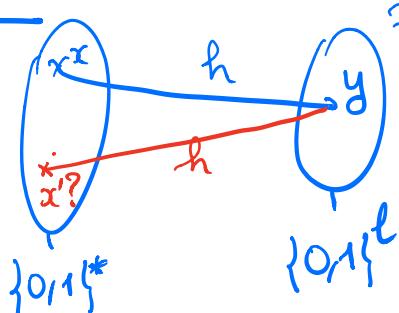
\Rightarrow nb d'essais (en moyenne) = 2^l (loi des grands nombres)

\Rightarrow complexité de l'algo = $O(2^l)$ (en moyenne)

En pratique on doit choisir l pour que $2^l \geq 2^{80}$

$$\Leftrightarrow l \geq 80$$

• Pb 2 :



\approx Même Algo : ~~(x et y sont fixés)~~

Générer x' aléatoirement ($x' \neq x$)

jusqu'à obtenir

$$h(x') = y$$

\rightarrow Complexité $O(2^l)$ en moyenne

\rightarrow même contrainte : on doit prendre $l \geq 80$