Cryptologie 3

(Commitments, signatures numériques, cryptographie avancée)

Georg Fuchsbauer

www.di.ens.fr/~fuchsbau/cryptologie3.pdf

cours à l'ESGI 2018

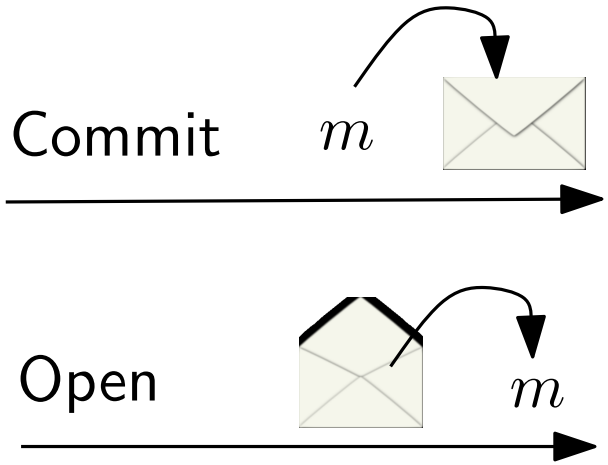
Mises en gage *(commitments)*

Mise en gage

Mise en gage – *commitment*

- "enveloppe numérique"

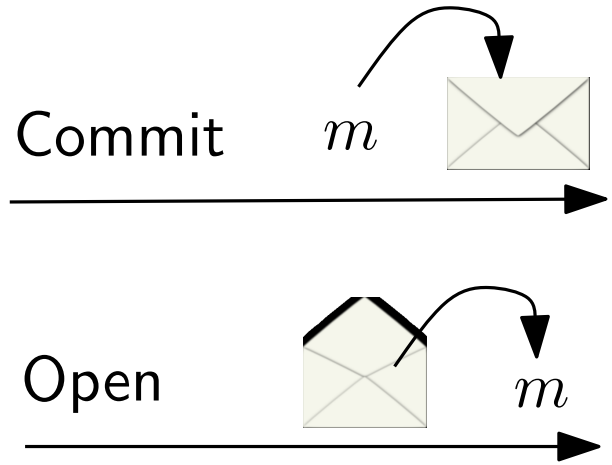
Brassard, Chaum, Crepeau : *Minimum Disclosure Proofs of Knowledge* (1988)



Mise en gage

Mise en gage – *commitment*

- "enveloppe numérique"

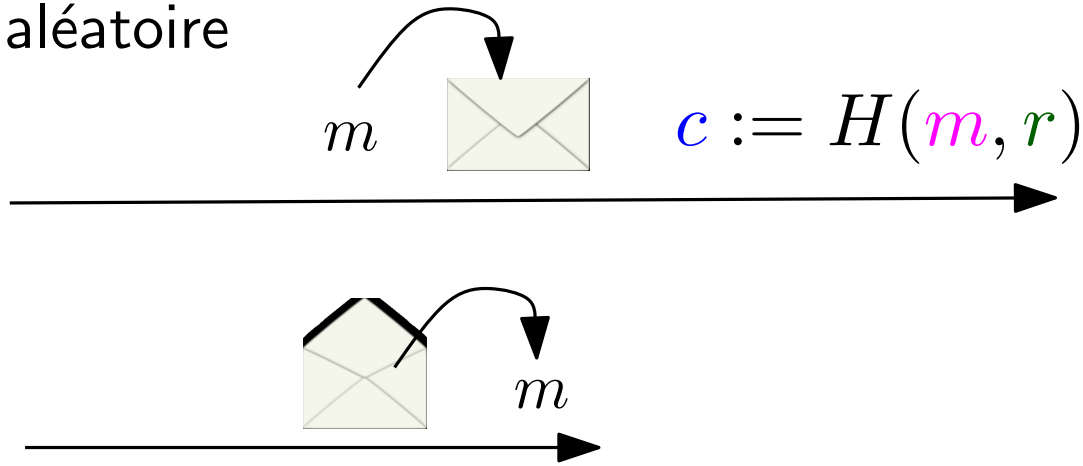


- **Dissimulation** *(hiding)* : le commitment ne révèle rien sur m
- **Liaison** *(binding)* : Alice ne peut sortir du commitment qu'une valeur

Mise en gage

Mise en gage – *commitment*

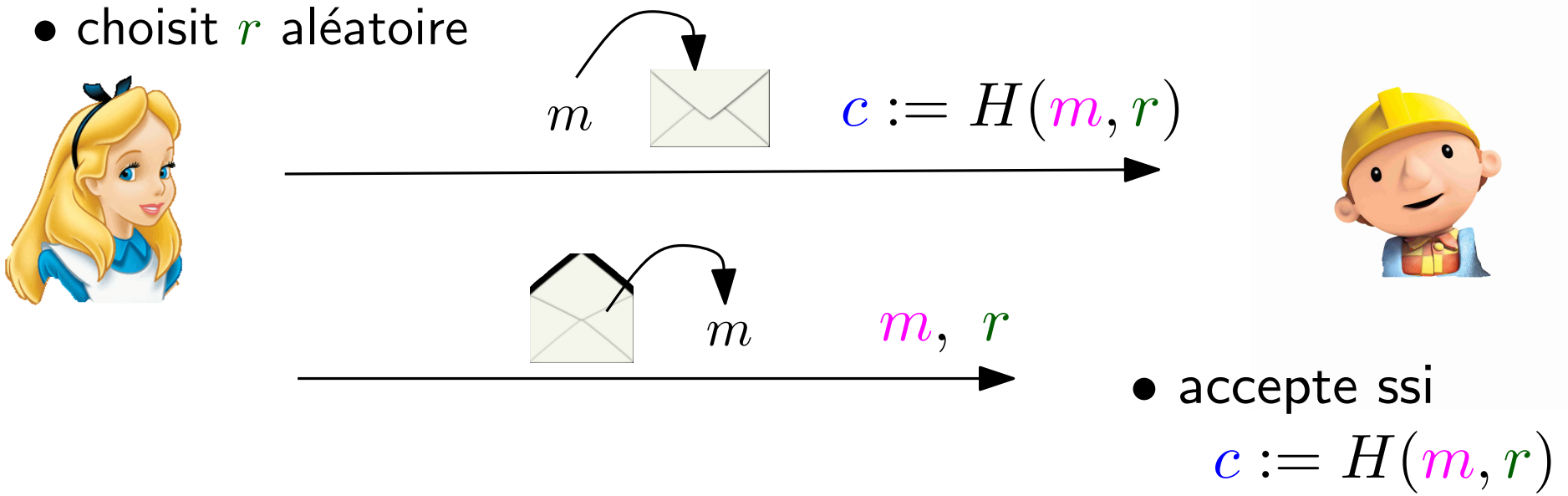
- “enveloppe numérique”
  - choisit r aléatoire



Mise en gage

Mise en gage – *commitment*

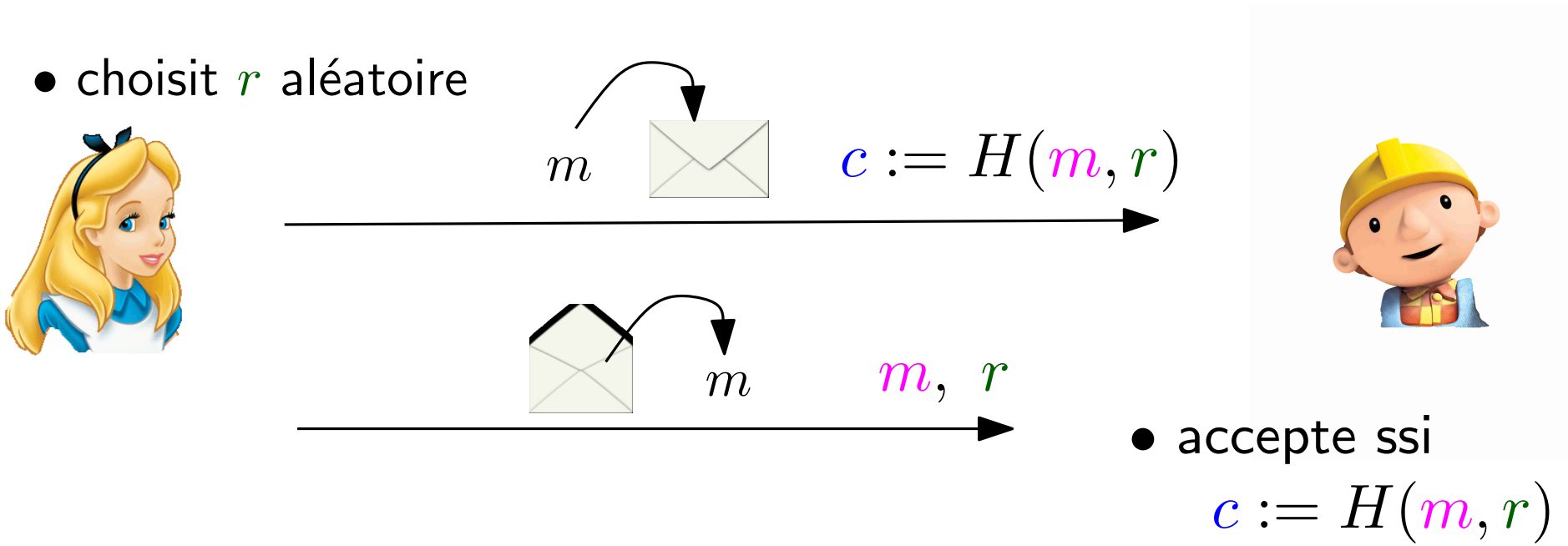
- “enveloppe numérique”



Mise en gage

Mise en gage – *commitment*

- “enveloppe numérique”



- Dissimulation : si H se comporte “de façon aléatoire”
- Liaison : si H résiste aux collisions

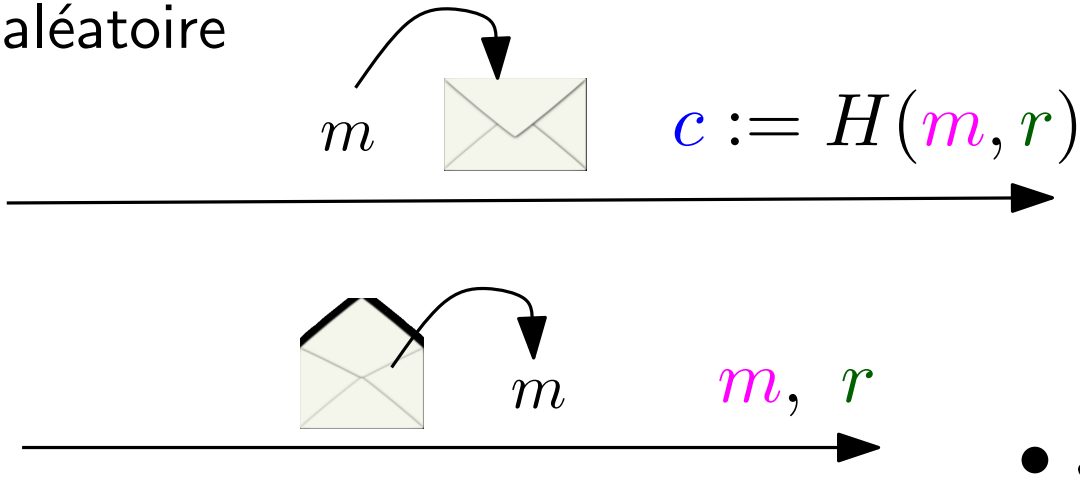
Mise en gage

Mise en gage – *commitment*

- "enveloppe numérique"

Application :
jeu de pile ou face
à distance

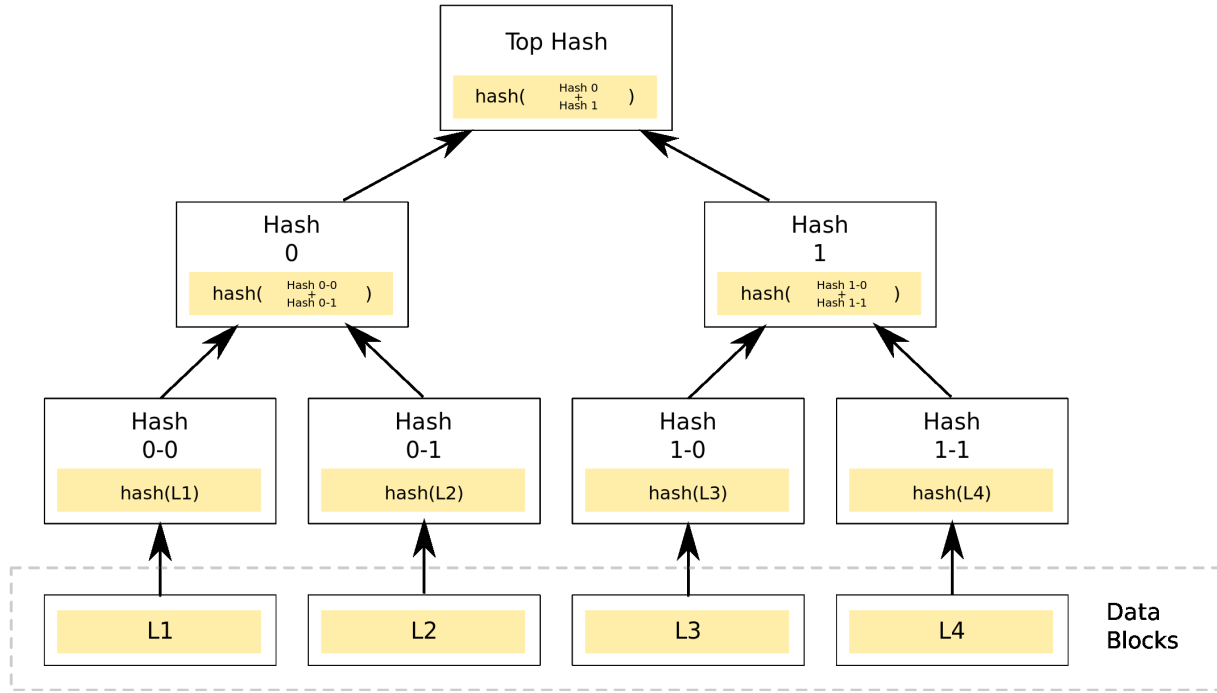
- choisit r aléatoire



- accepte ssi
$c := H(m, r)$

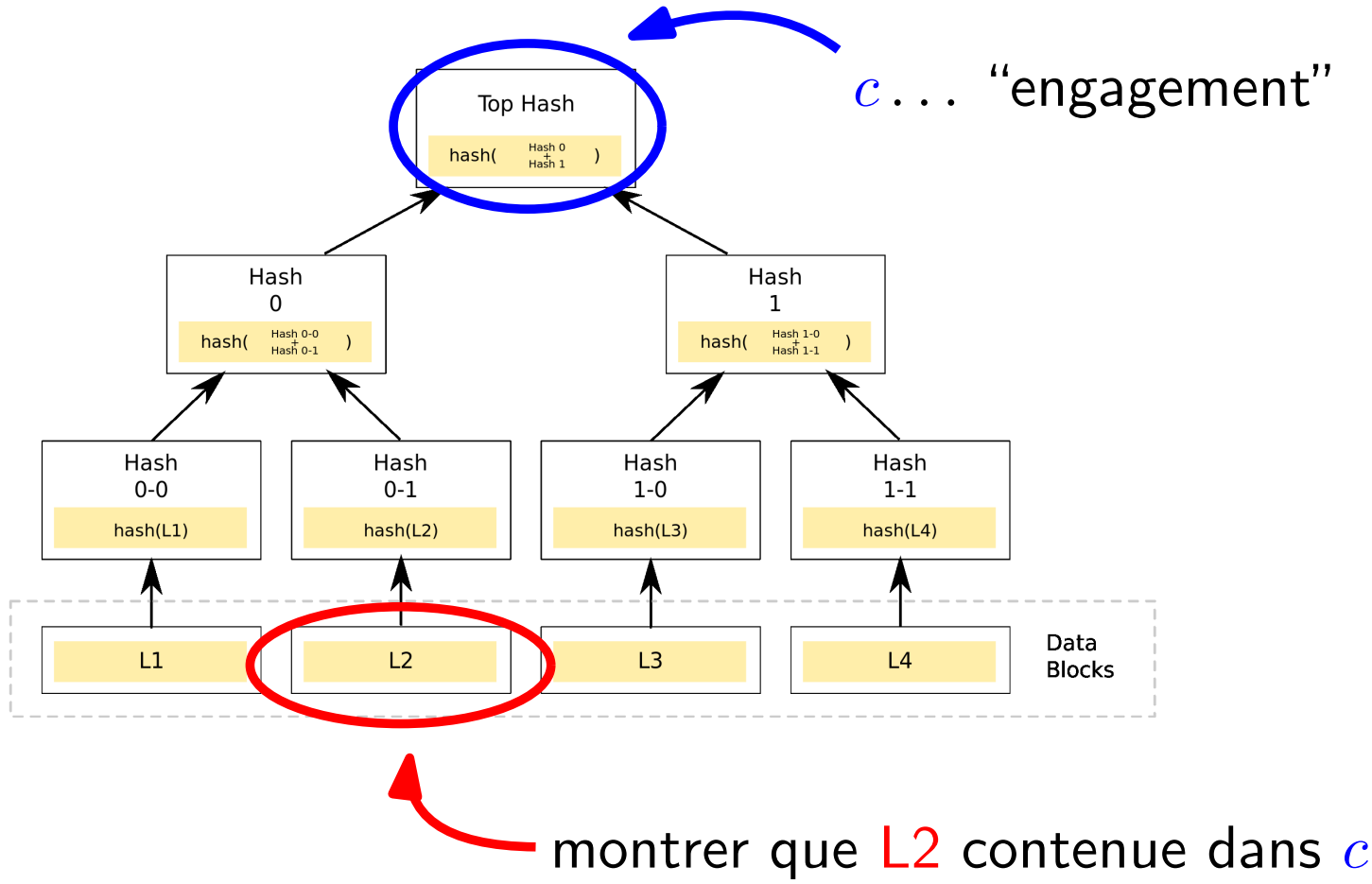
- Dissimulation : si H se comporte "de façon aléatoire"
- Liaison : si H résiste aux collisions

Arbres de Merkle



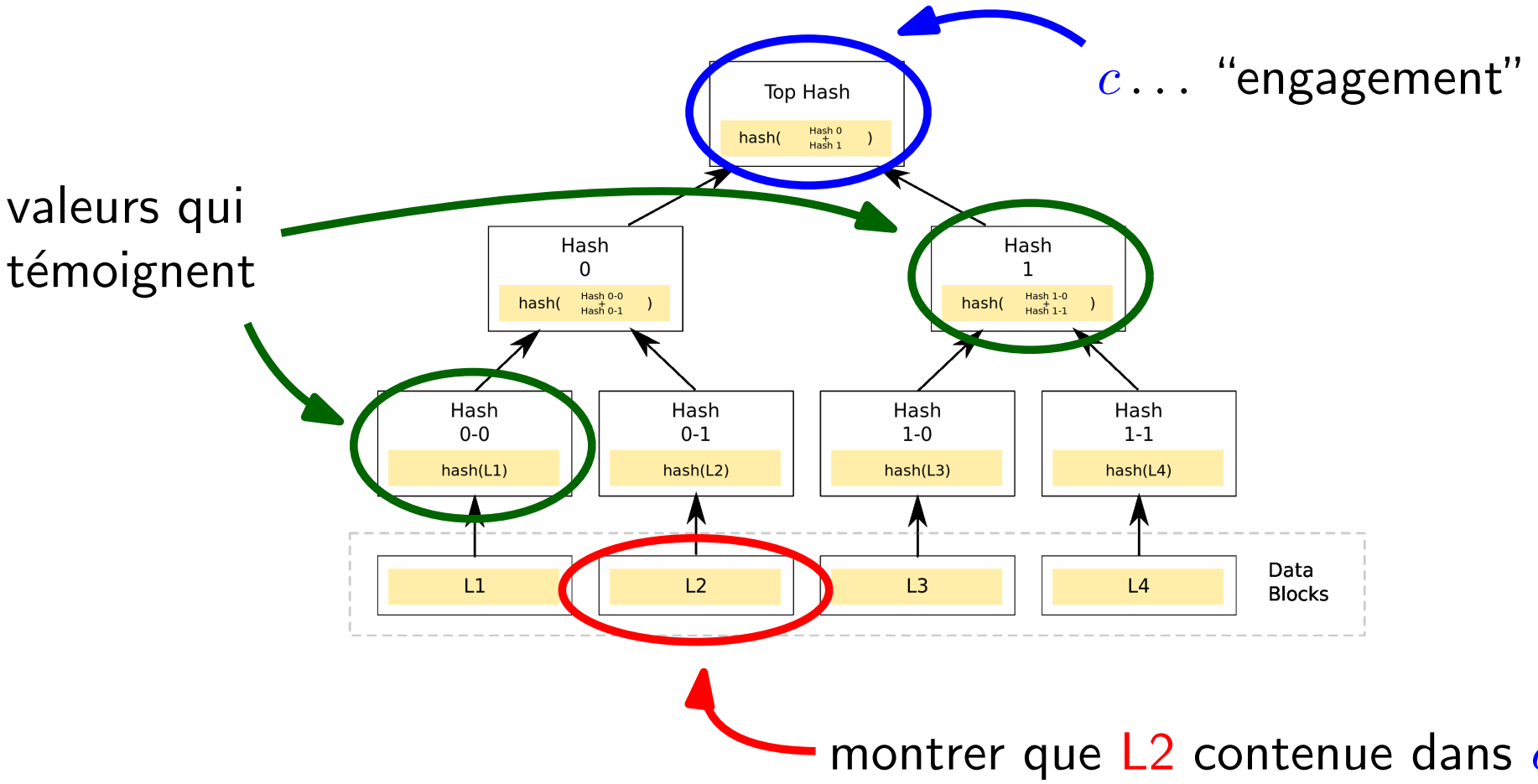
- Arbre de Merkle permet d'**ouvrir** une **partie** de la valeur mise en gage de façon **efficace**

Arbres de Merkle



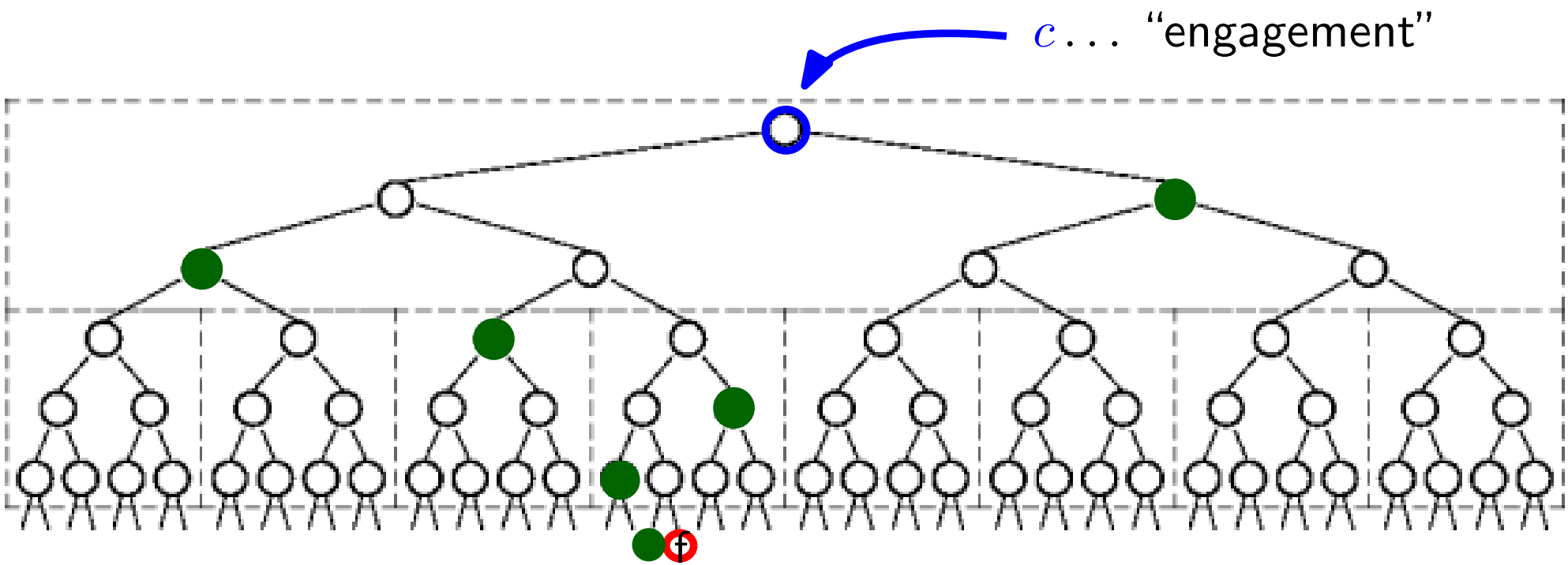
- Arbre de Merkle permet d’**ouvrir** une **partie** de la valeur mise en gage de façon **efficace**

Arbres de Merkle

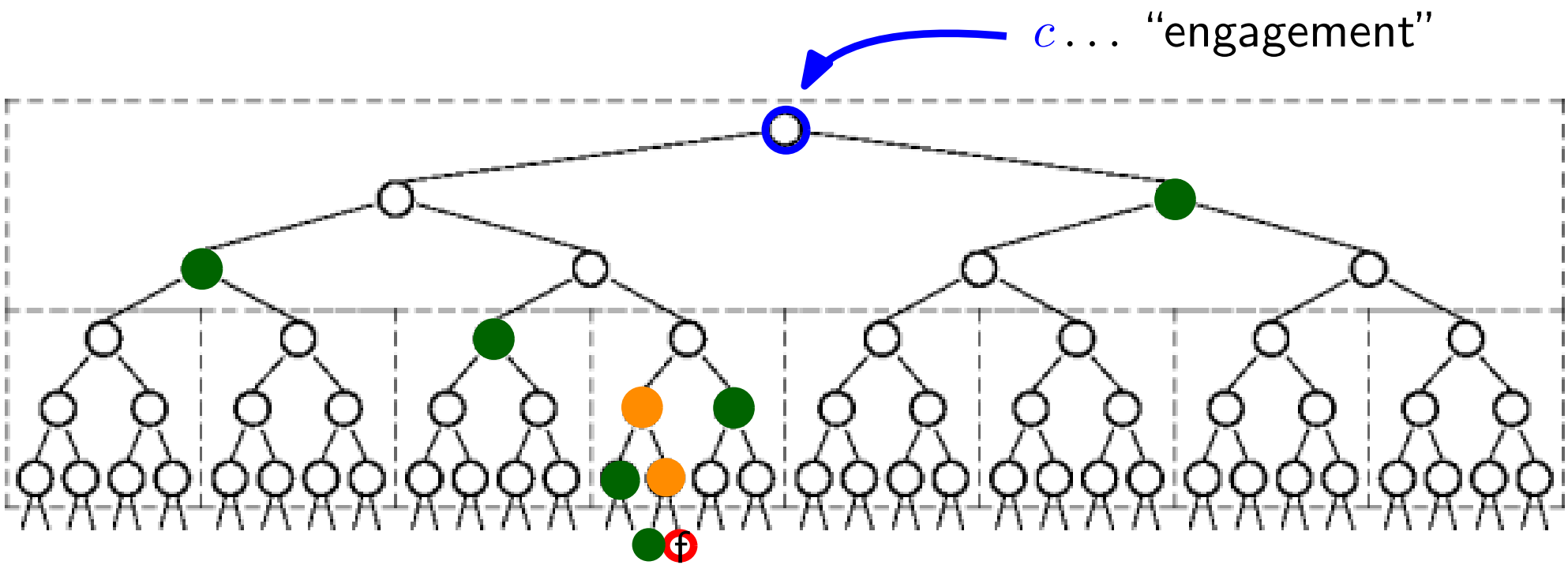


- Arbre de Merkle permet d’**ouvrir** une **partie** de la valeur mise en gage de façon **efficace**

Arbres de Merkle



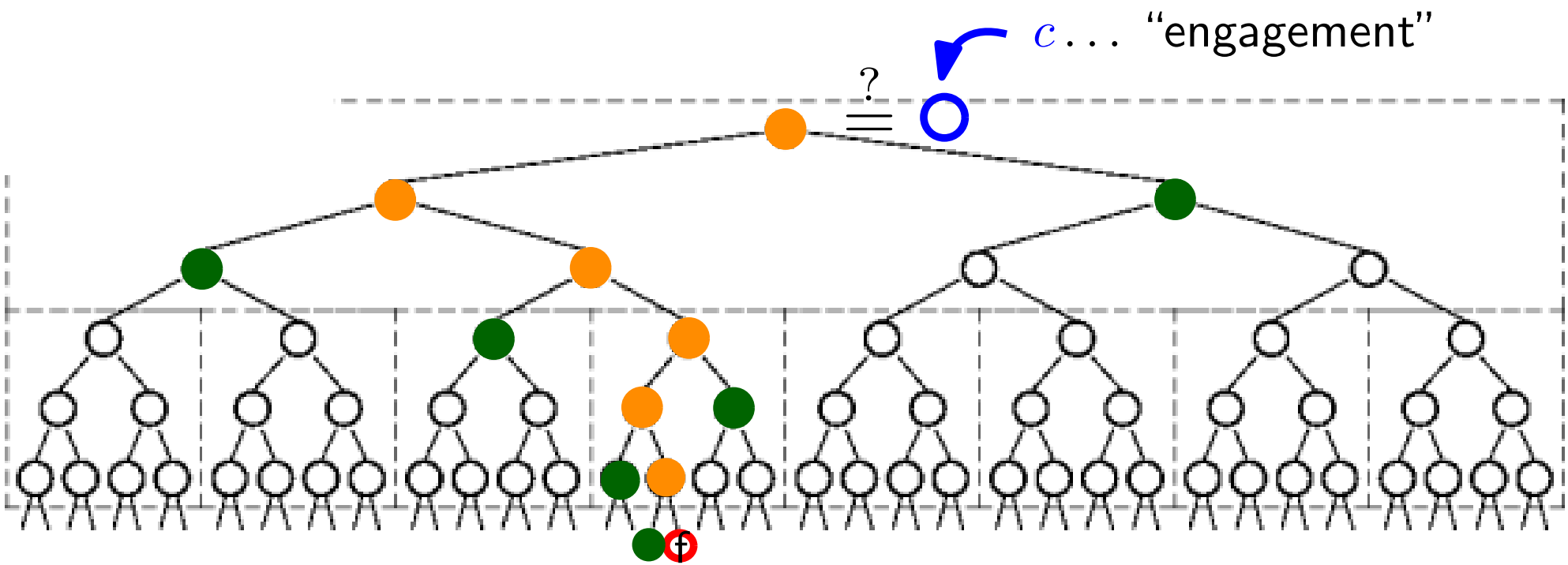
Arbres de Merkle



● . . . valeurs
qui témoignent

● . . . valeurs
qu’on peut calculer

Arbres de Merkle

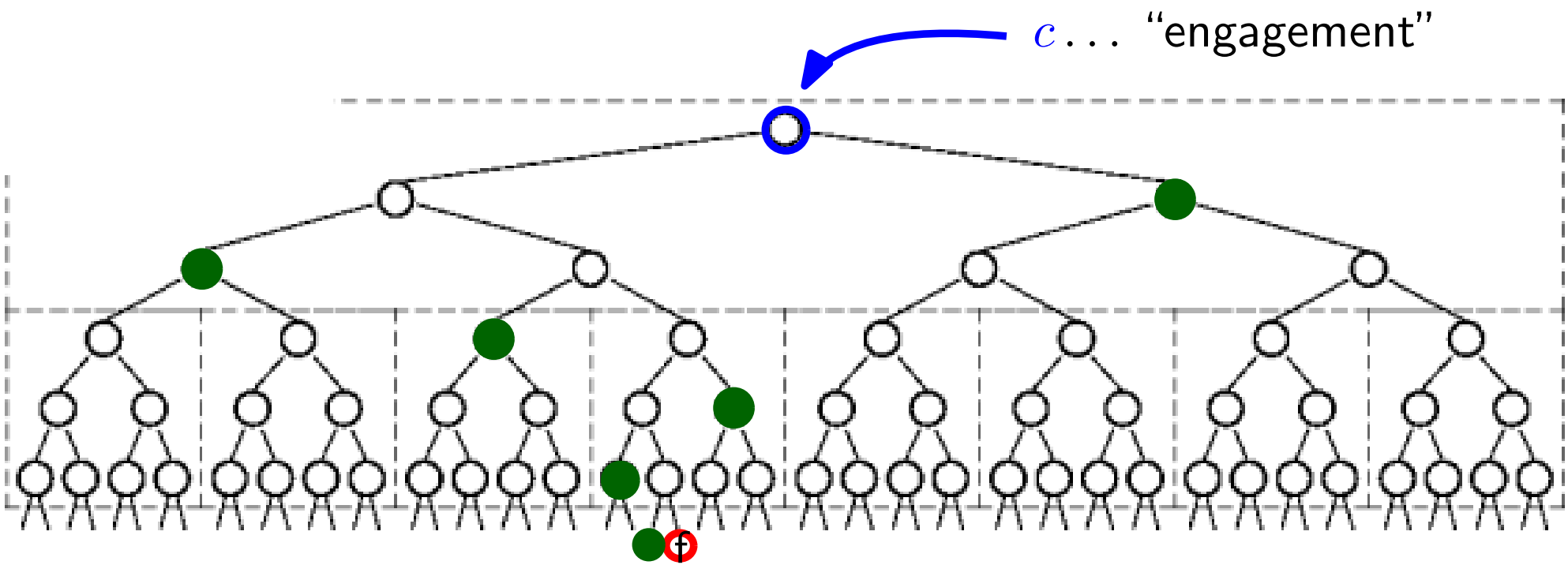


● . . . valeurs
qui témoignent

montrer que (f) contenue dans c

● . . . valeurs
qu’on peut calculer

Arbres de Merkle



Nombre de “fichiers” (feuilles) : 2^n

Nombre de “témoignes” : n (= hauteur de l’arbre)

Preuve de travail

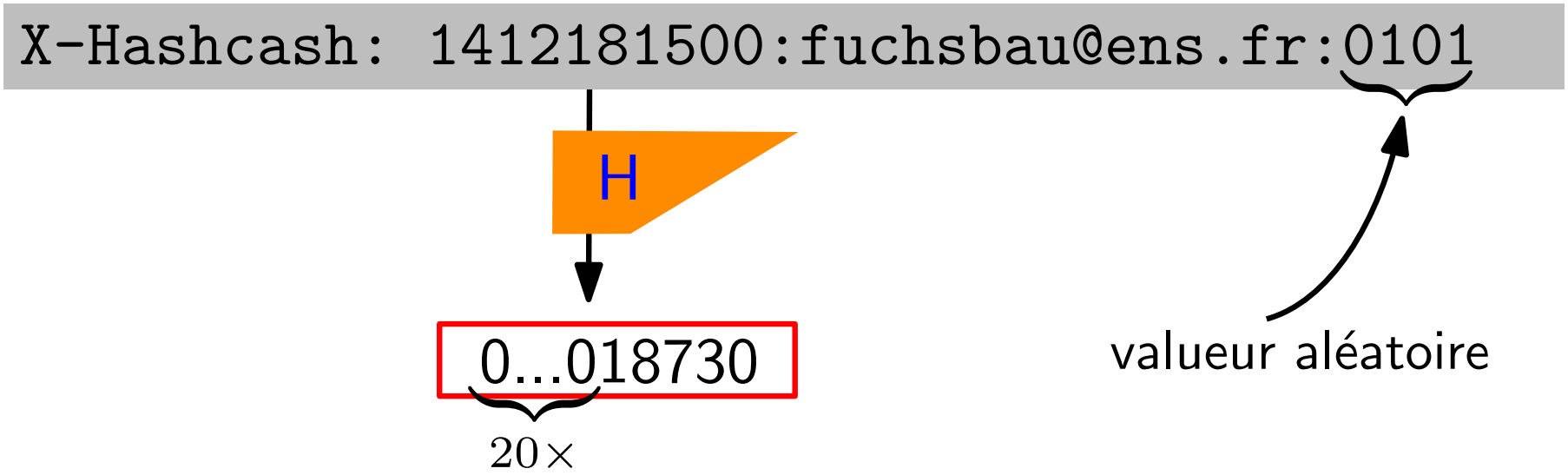
(proof of work)

- prouver qu'on a travaillé
- par ex. : dissuader le pourriel *(spam)* Hashcash

Preuve de travail

(proof of work)

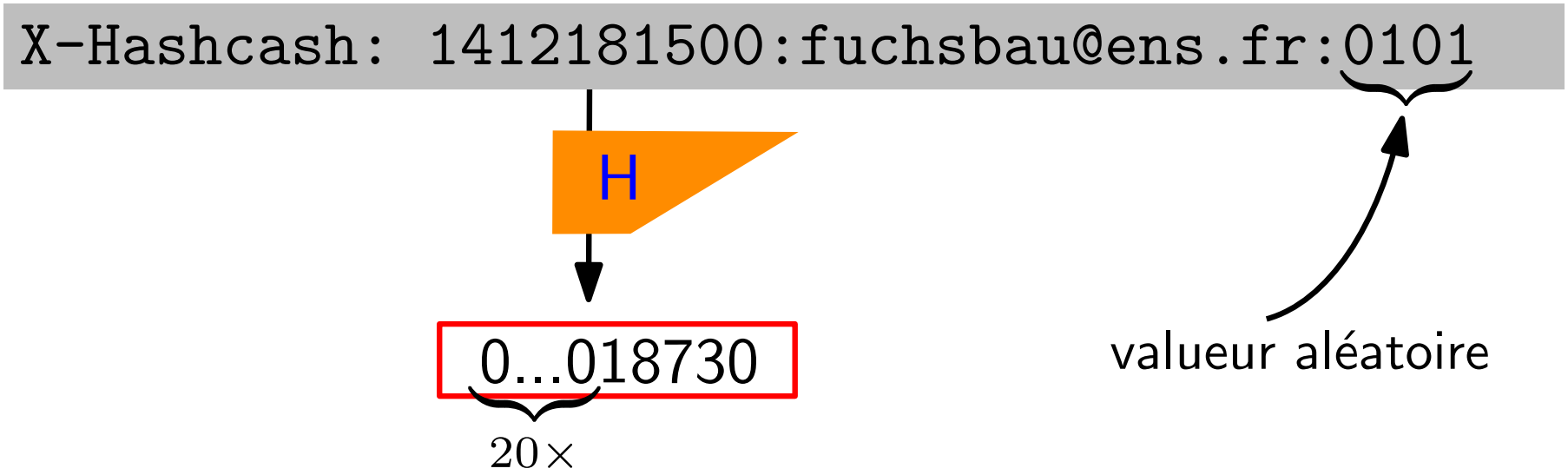
- prouver qu'on a travaillé
- par ex. : dissuader le pourriel *(spam)* Hashcash



Preuve de travail

(proof of work)

- prouver qu'on a travaillé
- par ex. : dissuader le pourriel *(spam)* Hashcash



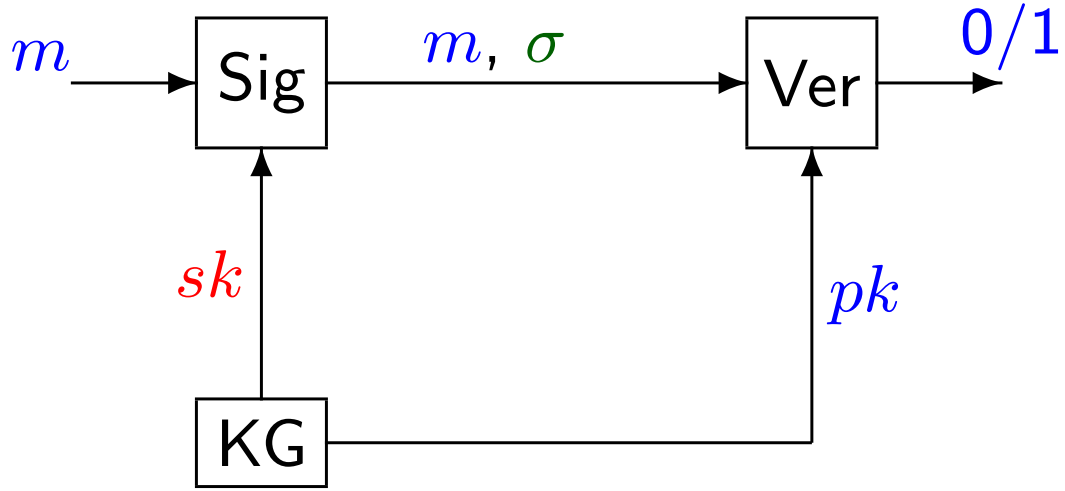
- essayer $\approx 2^{20}$ valeurs ($\sim$1s)
- facile à vérifier ($\sim 1\mu$s)

Signatures numériques

Motivation

- Protection de l'intégrité des données :
  - Le message vient-il du bon émetteur ?
  - Le message n'a-t-il pas été modifié en transit ?

Alice



Bob

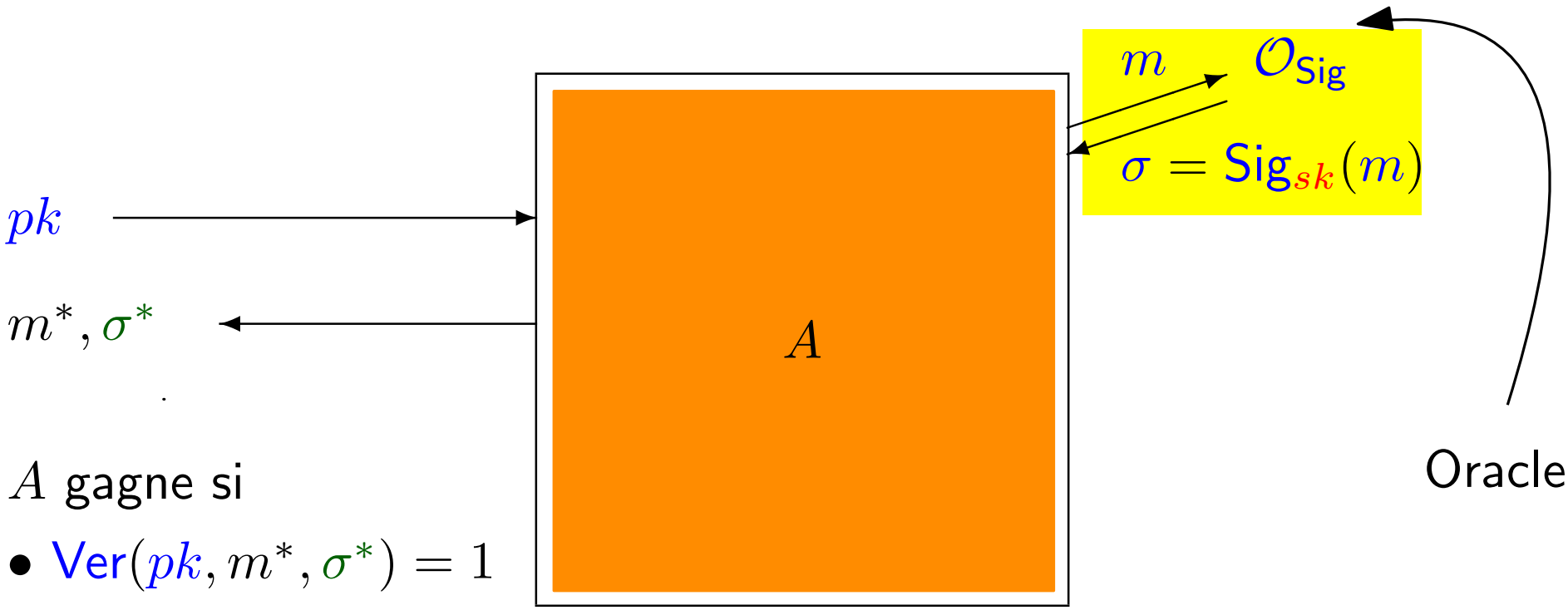
- Tout utilisateur a paire de clés (pk, sk) liées "mathématiquement"
- pk est rendue publique (par ex. disponible dans un annuaire)

Motivation

- Exemples :
  - Mise à jour signée par l'éditeur de logiciels
  - Certificats de clé publiques (navigateur web)
  - Signer des transactions dans Bitcoin
- Signatures vs. MACs:
  - Gestion de clés
  - Vérifiabilité *publique*
  - *Non-répudiation*

Sécurité de signatures numériques

(existential unforgeability under chosen-message attack)



A gagne si

- $\mathsf{Ver}(pk, m^*, \sigma^*) = 1$
- $m^* \notin \{m_1, m_2, \ldots\}$

Infalsifiabilité

(même notion que pour les MACs)

Signature RSA naïve

Rappel Chiffrement RSA naïf

- Génération de clé :
  - Choisir p et q deux "grands" nombres premiers
  - Calculer $N = p \cdot q$ *Rappel :* $\phi(N) = (p-1)(q-1)$
  - Choisir $e \in \mathbb{N}$ tel que pgcd$(e, \phi(N)) = 1$
  - Calculer d tel que $e \cdot d \equiv 1 \pmod{\phi(N)}$
  - $pk = (N, e)$ $sk = (N, d)$

Signature RSA naïve

Rappel Chiffrement RSA naïf

- Génération de clé :
  - Choisir p et q deux "grands" nombres premiers
  - Calculer $N = p \cdot q$ *Rappel :* $\phi(N) = (p-1)(q-1)$
  - Choisir $e \in \mathbb{N}$ tel que pgcd$(e, \phi(N)) = 1$
  - Calculer d tel que $e \cdot d \equiv 1 \pmod{\phi(N)}$
  - $pk = (N, e)$ $sk = (N, d)$

- Chiffrement : $c = m^e \bmod N$

- Déchiffrement : $m = c^d \bmod N$

Rappel : $m^{ed} \equiv m \pmod{N}$

Signature RSA naïve

Rappel Chiffrement RSA naïf

- Génération de clé :
  - Choisir p et q deux "grands" nombres premiers
  - Calculer $N = p \cdot q$ *Rappel :* $\phi(N) = (p-1)(q-1)$
  - Choisir $e \in \mathbb{N}$ tel que $\text{pgcd}(e, \phi(N)) = 1$
  - Calculer d tel que $e \cdot d \equiv 1 \pmod{\phi(N)}$
  - $pk = (N, e)$ $sk = (N, d)$

- Chiffrement : $c = m^e \bmod N$
- Déchiffrement : $m = c^d \bmod N$

Rappel : $m^{ed} \equiv m \pmod{N}$

Signature RSA naïve

- Signature : $\sigma = m^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} m \pmod{N}$

Attaques contre signature RSA naïve

Attaque sans requête à l'oracle

- Étant donné $pk = (N, e)$, choisir σ au hasard ;
  calculer $m = \sigma^e \bmod N$
- (m, σ) valable car $\sigma^e \equiv m \pmod N$

Attaques contre signature RSA naïve

Attaque sans requête à l'oracle

- Étant donné $pk = (N, e)$, choisir σ au hasard ;
  calculer $m = \sigma^e \bmod N$
- (m, σ) valable car $\sigma^e \equiv m \pmod N$

Falsification pour message choisi

- Pour falsifier une signature sur m, choisir m_1, m_2 tels que

  $$m \equiv m_1 \cdot m_2 \bmod N$$

- Obtenir des signatures σ_1 et σ_2 sur m_1 et m_2

Attaques contre signature RSA naïve

Attaque sans requête à l'oracle

- Étant donné $pk = (N, e)$, choisir σ au hasard ;
  calculer $m = \sigma^e \bmod N$
- (m, σ) valable car $\sigma^e \equiv m \pmod N$

Falsification pour message choisi

- Pour falsifier une signature sur m, choisir m_1, m_2 tels que

  $$m \equiv m_1 \cdot m_2 \bmod N$$

- Obtenir des signatures σ_1 et σ_2 sur m_1 et m_2
- Alors $\sigma := \sigma_1 \cdot \sigma_2 \bmod N$ est valable pour M

  Pourquoi ?

Attaques contre signature RSA naïve

Attaque sans requête à l'oracle

- Étant donné $pk = (N, e)$, choisir σ au hasard ;
  calculer $m = \sigma^e \bmod N$
- (m, σ) valable car $\sigma^e \equiv m \pmod N$

Falsification pour message choisi

- Pour falsifier une signature sur m, choisir m_1, m_2 tels que

  $$m \equiv m_1 \cdot m_2 \bmod N$$

- Obtenir des signatures σ_1 et σ_2 sur m_1 et m_2
- Alors $\sigma := \sigma_1 \cdot \sigma_2 \bmod N$ est valable pour M

  Pourquoi ?

$$\sigma^e \equiv (\sigma_1 \cdot \sigma_2)^e \equiv {\sigma_1}^e \cdot {\sigma_2}^e \equiv m_1 \cdot m_2 \equiv m \pmod N$$

Signature RSA-FDH

(full-domain hash)

Solution : utiliser fonction de hachage $\approx$ PKCS #1 v2.1

- **Génération de clé :**
  - Générer une paire de clé RSA $pk = (N, e)$, $sk = (N, d)$
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_N^*$

Signature RSA-FDH

(full-domain hash)

Solution : utiliser fonction de hachage $\approx$ PKCS #1 v2.1

- **Génération de clé :**
  - Générer une paire de clé RSA $pk = (N, e)$, $sk = (N, d)$
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_N^*$
- **Signature :** $\sigma = H(m)^d \bmod N$
- **Verification :** $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

Exercice. Quelles propriétés de H sont nécessaires pour que RSA-FDH soit sûr ?

Signature RSA-FDH

(full-domain hash)

Solution : utiliser fonction de hachage $\approx$ PKCS #1 v2.1

- **Génération de clé :**
  - Générer une paire de clé RSA $pk = (N, e)$, $sk = (N, d)$
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_N^*$
- **Signature :** $\sigma = H(m)^d \bmod N$
- **Verification :** $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

Exercice. Quelles propriétés de H sont nécessaires pour que RSA-FDH soit sûr ?

Théorème. Si le problème RSA est dur et H est modélisée comme fonction aléatoire, alors RSA-FDH est sûr.

Protocoles d'authentification

Alice, qui a une clé publique pk, veut s'authentifier auprès de Bob

sk

Protocole d'authentification

pk

Client/Prouveur

Serveur/Vérifieur

Protocoles d'authentification

Alice, qui a une clé publique pk, veut s'authentifier auprès de Bob

sk

Protocole d'authentification

pk

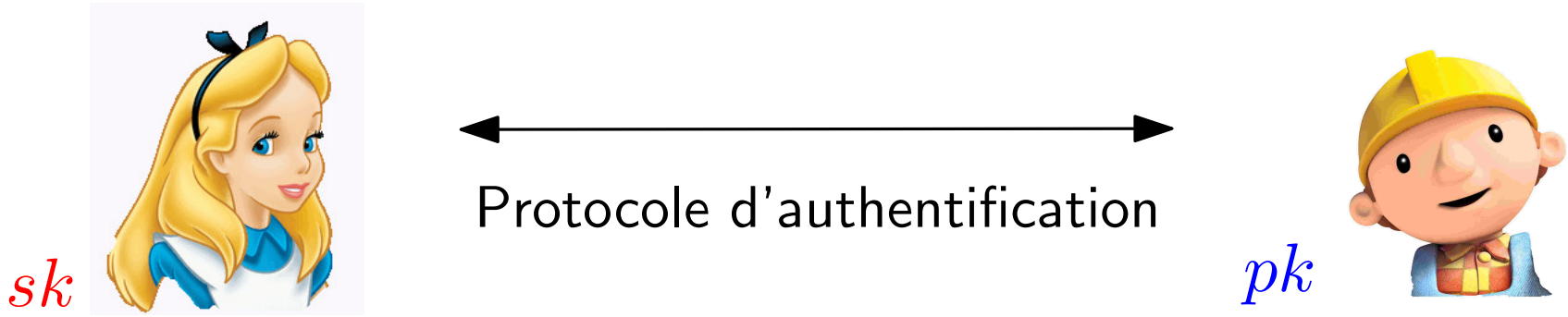
Client/Prouveur

Serveur/Vérifieur

- En utilisant sk, Alice convainc Bob
- *Sans* sk, personne n'arrive à s'authentifier
  - même après avoir *observé* des authentifications

Protocoles d'authentification

Alice, qui a une clé publique pk, veut s'authentifier auprès de Bob



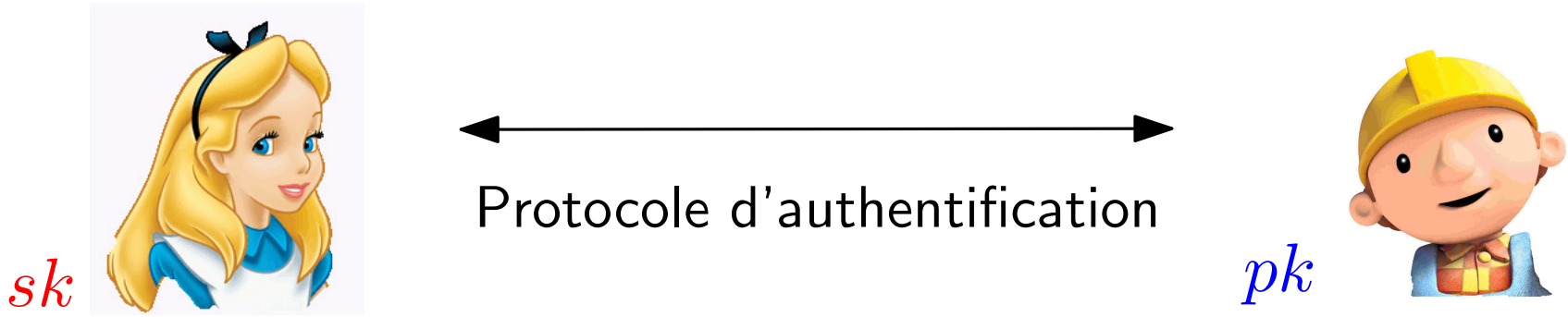
Propriétés. Le protocole doit être :

- **correct** : Si Alice connaît sk, alors elle convaincra Bob
- **robuste** *(sound)* : Si Alice convainc Bob, alors elle connaît sk.

  Formellement : D'un prouver convaincant on peut **extraire** la clé sk ($\Rightarrow$ alors elle doit connaître sk).

Protocoles d'authentification

Alice, qui a une clé publique pk, veut s'authentifier auprès de Bob



Propriétés. Le protocole doit être :

- **correct** : Si Alice connaît sk, alors elle convaincra Bob
- **robuste** *(sound)* : Si Alice convainc Bob, alors elle connaît sk.
- ne **pas révéler** de l'**information** *(zero-knowledge)* : Une transcription du protocole (c-à-d les message échangés) peut être **simulée** sans connaître la clé sécrète ($\Rightarrow$ le protocole ne peut divulguer de l'information sur sk).

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

sk 

pk 

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$$I = g^k \longrightarrow$$

pk

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$$I = g^k$$

$$r$$

- choisit r aléatoire

pk

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r

$s = r \cdot x + k$

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r

$s = r \cdot x + k$

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

correct : $g^s \cdot y^{-r} = g^{rx+k} \cdot (g^x)^{-r} = g^k = I$

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r

$s = r \cdot x + k$

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"robuste" *

* Si Alice arrive à convaincre Bob, alors elle doit vraiment connaître x

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r_1

s_1

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"robuste" *

* Si Alice arrive à convaincre Bob, alors elle doit vraiment connaître x

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r_1

s_1

r_2

s_2

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"robuste" *

* Si Alice arrive à convaincre Bob, alors elle doit vraiment connaître x

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r_1

s_1

r_2

s_2

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"robuste"*: $g^{s_1} \cdot y^{-r_1} = I = g^{s_2} \cdot y^{-r_2}$

$$\Rightarrow \quad g^{s_1 - s_2} = y^{r_1 - r_2}$$

* Si Alice arrive à convaincre Bob, alors elle doit vraiment connaître x

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r_1

s_1

r_2

s_2

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"robuste"*: $g^{s_1} \cdot y^{-r_1} = I = g^{s_2} \cdot y^{-r_2}$

$$\Rightarrow \quad g^{s_1 - s_2} = y^{r_1 - r_2} \quad [\hat{}\,(r_1 - r_2)^{-1}]$$

$$\Rightarrow \quad x = (s_1 - s_2) \cdot (r_1 - r_2)^{-1} \pmod{|\mathbb{G}|}$$

* Si Alice arrive à convaincre Bob, alors elle doit vraiment connaître x

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$$I = g^k$$

$$r$$

$$s = r \cdot x + k$$

- choisit r aléatoire

pk

- accepte ssi

$$g^s \cdot y^{-r} = I$$

"zero-knowledge" *

* À un observateur extérieur, rien n'est révélé

Protocole d'authentification de Schnorr

Schnorr: groupe $\mathbb{G}$, générateur g $\quad sk = x, \quad pk = y = g^x$

- choisit k aléatoire

sk

$I = g^k$

r

$s = r \cdot x + k$

- choisit r aléatoire

pk

- accepte ssi $g^s \cdot y^{-r} = I$

"zero-knowledge" * **:**

Transcription du protocole (I, r, s) peut être simulé :

- choisir r, s aléatoires
- calculer $I = g^s \cdot y^{-r}$ (distribués comme dans le protocole)

* À un observateur extérieur, rien n'est révélé

Signature Schnorr

Problème : protocole d'authentification *interactif* ($\neq$ signature)

$\Rightarrow$ laissons **fonction de hachage** jouer le rôle du vérifieur !

Signature Schnorr

Problème : protocole d'authentification *interactif* ($\neq$ signature)

$\Rightarrow$ laissons fonction de hachage jouer le rôle du vérifieur !

- **Génération de clé :**
  - Choisir groupe $(\mathbb{G}, \cdot)$ d'ordre q ; choisir x aléatoire
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_q$
  - $pk = (\mathbb{G}, H, y = g^x)$, $sk = (pk, x)$

Signature Schnorr

Problème : protocole d'authentification *interactif* ($\neq$ signature)

$\Rightarrow$ laissons fonction de hachage jouer le rôle du vérifieur !

- **Génération de clé :**
  - Choisir groupe $(\mathbb{G}, \cdot)$ d'ordre q ; choisir x aléatoire
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_q$
  - $pk = (\mathbb{G}, H, y = g^x)$, $sk = (pk, x)$
- **Signature** pour m:
  - choisir k aléatoire
  - calculer $I = g^k$
  - calculer $r = H(I, m)$
  - calculer $s = r \cdot x + k \bmod q$
  - retourner signature (r, s)

Signature Schnorr

Problème : protocole d'authentification *interactif* ($\neq$ signature)

$\Rightarrow$ laissons fonction de hachage jouer le rôle du vérifieur !

- **Génération de clé :**
  - Choisir groupe $(\mathbb{G}, \cdot)$ d'ordre q ; choisir x aléatoire
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_q$
  - $pk = (\mathbb{G}, H, y = g^x)$, $sk = (pk, x)$
- **Signature** pour m:
  - choisir k aléatoire
  - calculer $I = g^k$
  - calculer $r = H(I, m)$
  - calculer $s = r \cdot x + k \bmod q$
  - retourner signature (r, s)
- **Verification :**
  - étant donnés $pk = y$, m, et signature (r, s)
  - calculer $I = g^s \cdot y^{-r}$
  - accepter ssi $H(I, m) = r$

Signature Schnorr

Problème : protocole d'authentification *interactif* ($\neq$ signature)

$\Rightarrow$ laissons fonction de hachage jouer le rôle du vérifieur !

- **Génération de clé :**
  - Choisir groupe $(\mathbb{G}, \cdot)$ d'ordre q ; choisir x aléatoire
  - Spécifier une fonction de hachage H avec image $\mathbb{Z}_q$
  - $pk = (\mathbb{G}, H, y = g^x)$, $sk = (pk, x)$
- **Signature** pour m:
  - choisir k aléatoire
  - calculer $I = g^k$
  - calculer $r = H(I, m)$
  - calculer $s = r \cdot x + k \bmod q$
  - retourner signature (r, s)
- **Verification :**
  - étant donnés $pk = y$, m, et signature (r, s)
  - calculer $I = g^s \cdot y^{-r}$
  - accepter ssi $H(I, m) = r$

Théorème. Si le problème DL est dur pour $\mathbb{G}$ et H est modélisée comme oracle aléatoire, alors la signature Schnorr est sûre

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire
- choisit α et r aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

pk

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire
- choisit α et r aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

pk

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$I = g^k$

$\alpha, \quad r$

$s = k^{-1}(\alpha + xr)$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{r s^{-1}} = I$

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = I$

correct : $g^{\alpha s^{-1}} \cdot (g^x)^{rs^{-1}} = g^{(\alpha + xr)s^{-1}}$

$$= g^{(\alpha+xr)\cdot k\cdot(\alpha+xr)^{-1}} = I$$

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire
- choisit α et r aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = I$

correct : $g^{\alpha s^{-1}} \cdot (g^x)^{rs^{-1}} = g^{(\alpha+xr)s^{-1}}$

$$= g^{(\alpha+xr)\cdot k\cdot(\alpha+xr)^{-1}} = I$$

Robuste et zero-knowledge comme Schnorr

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{r s^{-1}} = I$

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = I$

Signature : définir
- $\alpha = H(m)$ (H fonction de hachage)
- $r = F(I)$ ($F\colon \mathbb{G} \to \mathbb{Z}_q$ fctn simple)

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$$I = g^k$$

$$\alpha, \quad r$$

$$s = k^{-1}(\alpha + xr)$$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{r s^{-1}} = I$

Signature : définir
- $\alpha = H(m)$ (H fonction de hachage)
- $r = F(I)$ ($F\colon \mathbb{G} \to \mathbb{Z}_q$ fctn simple)

DSA : $\mathbb{G}$ sous-groupe de $\mathbb{Z}_p^*$

ECDSA : $\mathbb{G}$ groupe sur courbe elliptique

DSA / ECDSA

Digital Signature Standard (NIST)

groupe $\mathbb{G}$ $\quad sk = x, \quad pk = y = g^x$

- choisit $k \neq 0$ aléatoire

sk

$I = g^k$

$\alpha, \quad r$

$s = k^{-1}(\alpha + xr)$

- choisit α et r aléatoire

pk

- accepte ssi $s \neq 0$ et $g^{\alpha s^{-1}} \cdot y^{rs^{-1}} = I$

Signature : définir
- $\alpha = H(m)$ (H fonction de hachage)
- $r = F(I)$ ($F\colon \mathbb{G} \to \mathbb{Z}_q$ fctn simple)

DSA : $\mathbb{G}$ sous-groupe de $\mathbb{Z}_p^*$

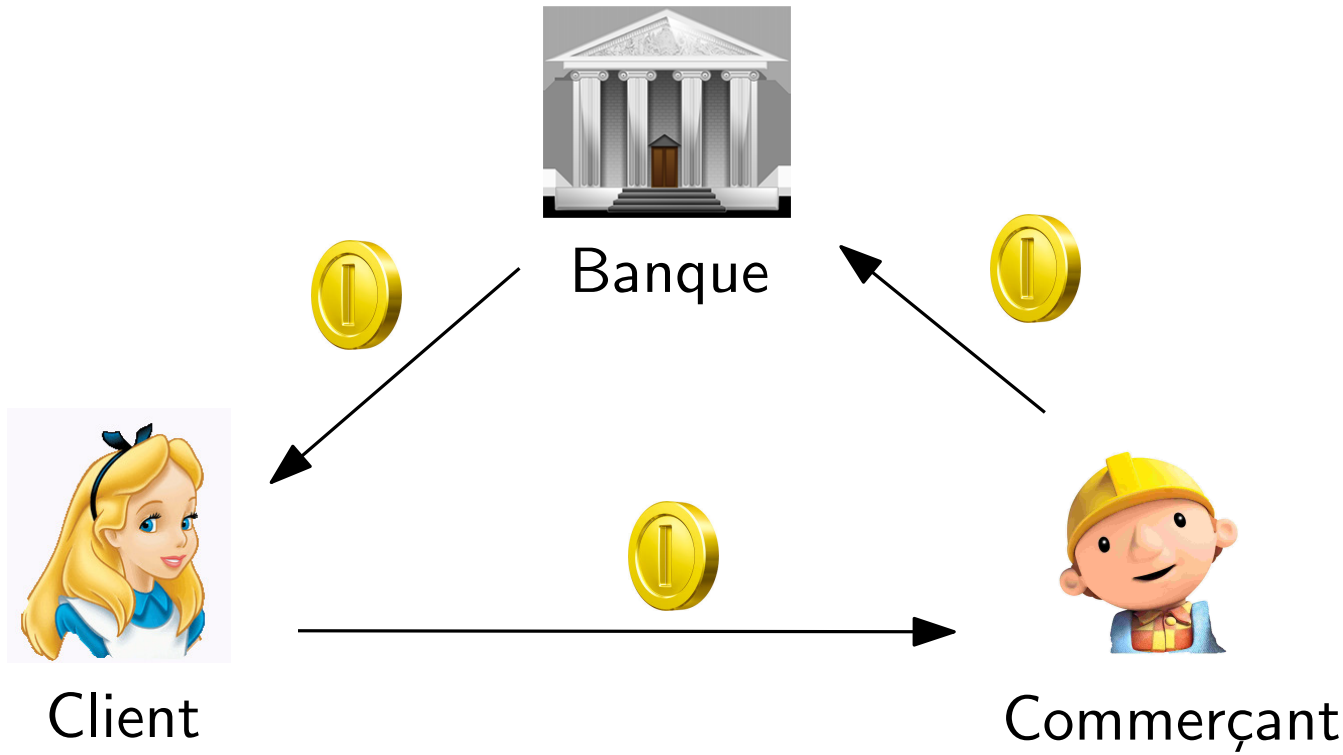
ECDSA : $\mathbb{G}$ groupe sur courbe elliptique

Important : nouveau k pour toute signature (PlayStation 2010)

Cryptographie avancée

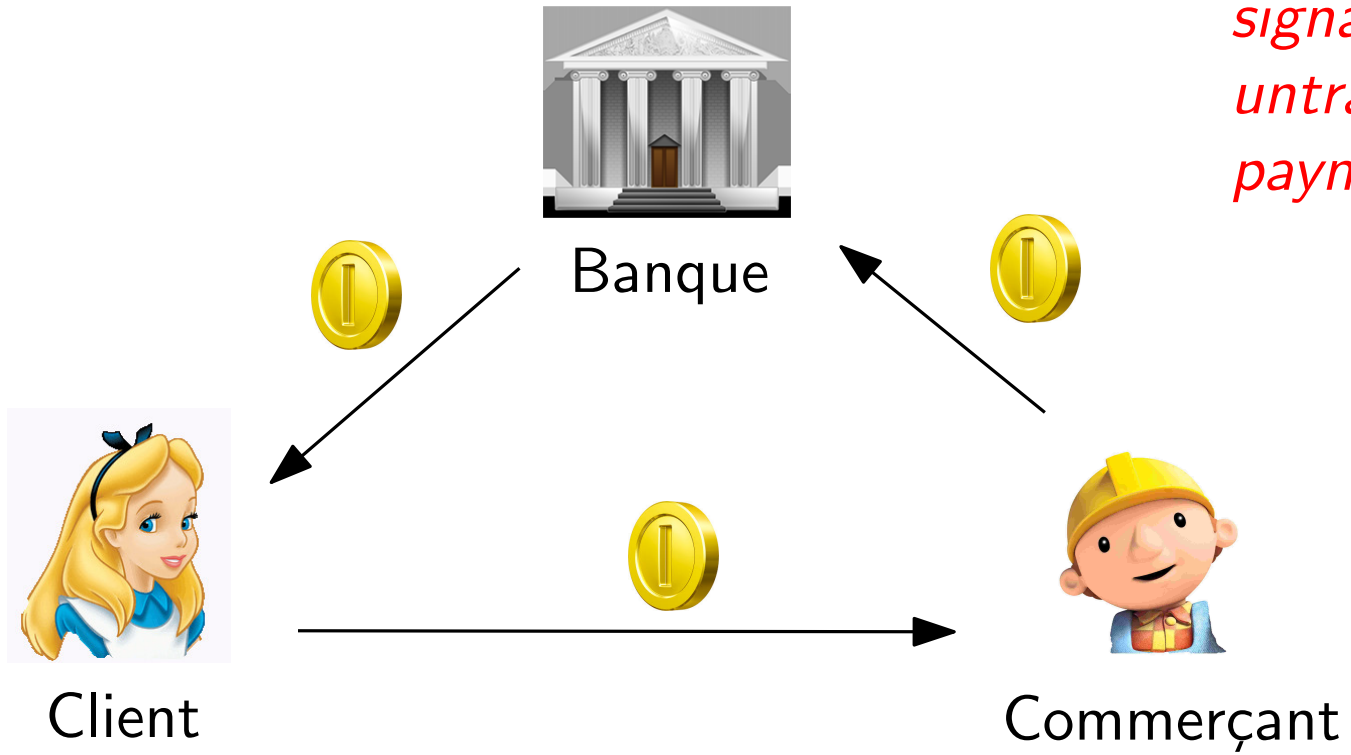
Monnaie électronique

(e-cash)



Monnaie électronique

Chaum : *Blind signatures for untraceable payments* (1982)



Pièces . . . infalsifiables

$\Rightarrow$ signature numérique !

- Double-dépense ?
- **Anonymat** ?

Signature en blanc

(blind signatures)

- Alice veut obtenir une signature sur m sans le révéler
- Même si la banque voit la signature après, elle ne peut l'associer avec Alice

m

Protocole de signature

pk

sk

signataire

σ

Signature en blanc

Rappel **RSA** :

- Clés : $pk = (N, e), \quad sk = (N, d)$
- Signature : $\sigma = H(m)^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

m

Protocole de signature

pk

sk

signataire

σ

Signature en blanc

Rappel **RSA** :

- Clés : $pk = (N, e), \quad sk = (N, d)$
- Signature : $\sigma = H(m)^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

- choisit r aléatoire

m 

$$M := H(m) \cdot r^e \bmod N \longrightarrow$$

pk  sk

signataire

Signature en blanc

Rappel **RSA** :

- Clés : $pk = (N, e), \quad sk = (N, d)$
- Signature : $\sigma = H(m)^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

- choisit r aléatoire

m 

$$M := H(m) \cdot r^e \bmod N \longrightarrow$$

$$\sigma' := M^d \longleftarrow$$

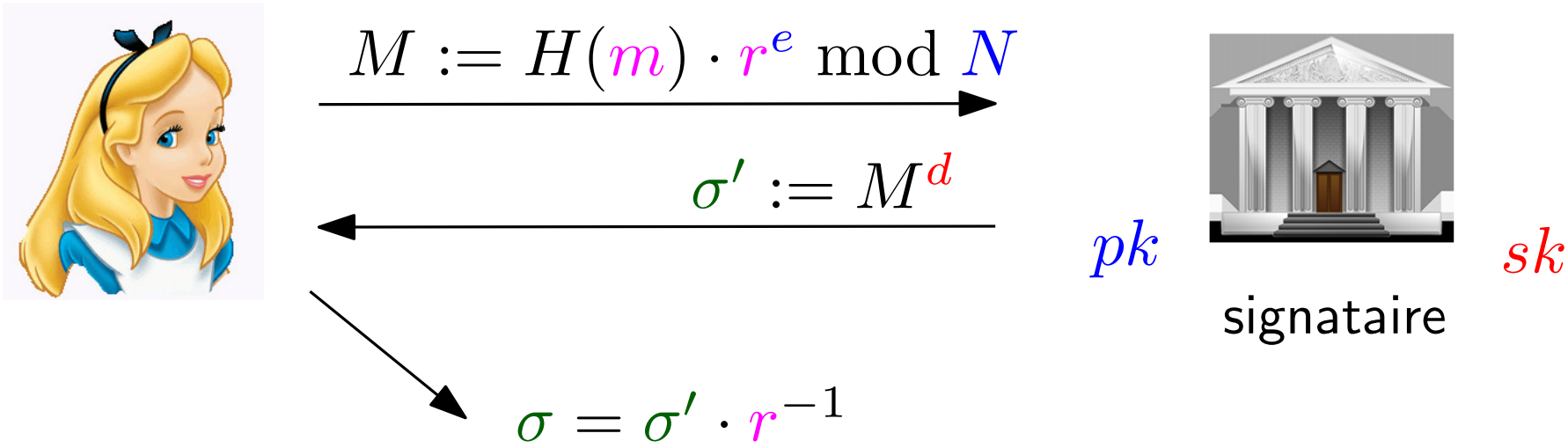
pk  sk

signataire

Signature en blanc

Rappel **RSA** :

- Clés : $pk = (N, e), \quad sk = (N, d)$
- Signature : $\sigma = H(m)^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$



Signature en blanc

Rappel **RSA** :

- Clés : $pk = (N, e)$, $sk = (N, d)$
- Signature : $\sigma = H(m)^d \bmod N$
- Verification : $\sigma^e \stackrel{?}{\equiv} H(m) \pmod N$

- choisit r aléatoire

m

$$M := H(m) \cdot r^e \bmod N$$

$$\sigma' := M^d$$

$$\sigma = \sigma' \cdot r^{-1}$$

pk

signataire

sk

correct : $\sigma' = H(m)^d \cdot r^{e \cdot d} = \sigma \cdot r$

Monnaie électronique

Banque

1 choisit n^o
aléatoire

Client

2 obtient
σ sur n^o
"en blanc"

Commerçant

Monnaie électronique

Banque

1 choisit n^o aléatoire

Client

2 obtient σ sur n^o

3 donne n^o et σ

Commerçant

Monnaie électronique

Banque

1 choisit n^o aléatoire

Client

2 obtient σ sur n^o

3 donne n^o et σ

4 demande si n^o dépensé

Commerçant

Monnaie électronique

Banque

4 demande si n^o dépensé

1 choisit n^o aléatoire

2 obtient σ sur n^o

3 donne n^o et σ

5 donne marchandise

Client

Commerçant

Monnaie électronique

Banque

4 demande si n^o dépensé

1 choisit n^o aléatoire

2 obtient σ sur n^o

3 donne n^o et σ

5 donne marchandise

Client

Commerçant

- Double-dépense ?
- **Anonymat** ?

Monnaie électronique

Banque

1 choisit n^o aléatoire

2 obtient σ sur n^o

4 demande si n^o dépensé

3 donne n^o et σ

5 donne marchandise

Client

Commerçant

- Double-dépense ?

  vérification *en ligne*

- **Anonymat** ?

  grâce à signature en blanc

Zero Knowledge, Zerocoin, Zerocash, Zcash

Preuves zero-knowledge

Preuve zero-knowledge

- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations

Preuves zero-knowledge

Preuve zero-knowledge de connaissance

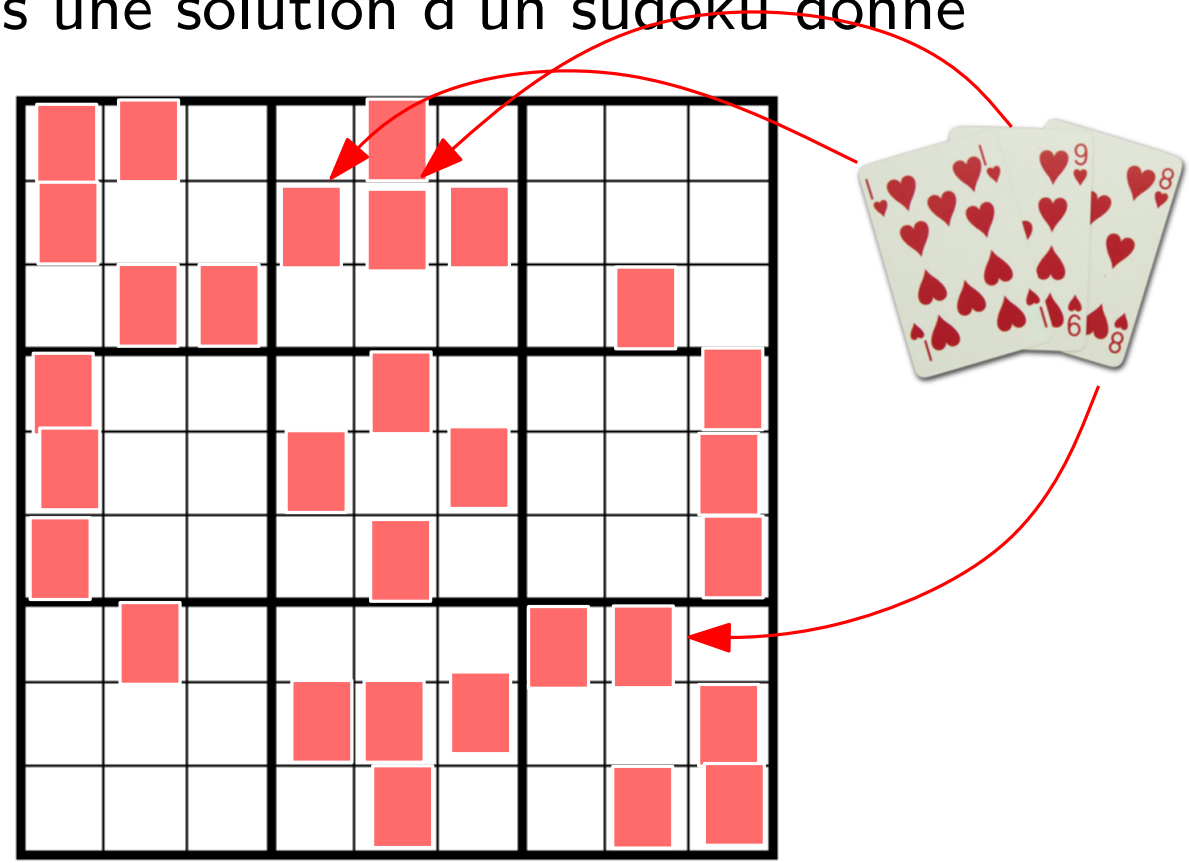
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Preuves zero-knowledge

Preuve zero-knowledge de connaissance

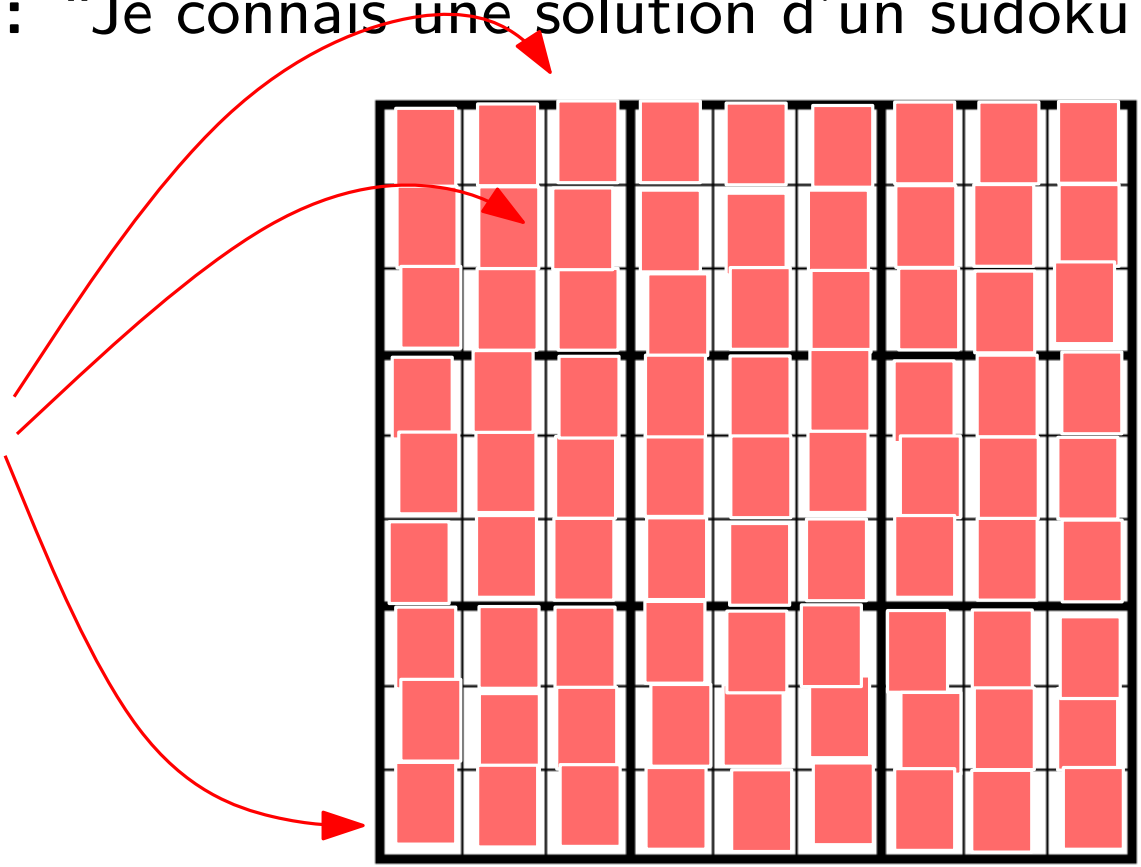
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"
  - Verifier



Preuves zero-knowledge

Preuve zero-knowledge de connaissance

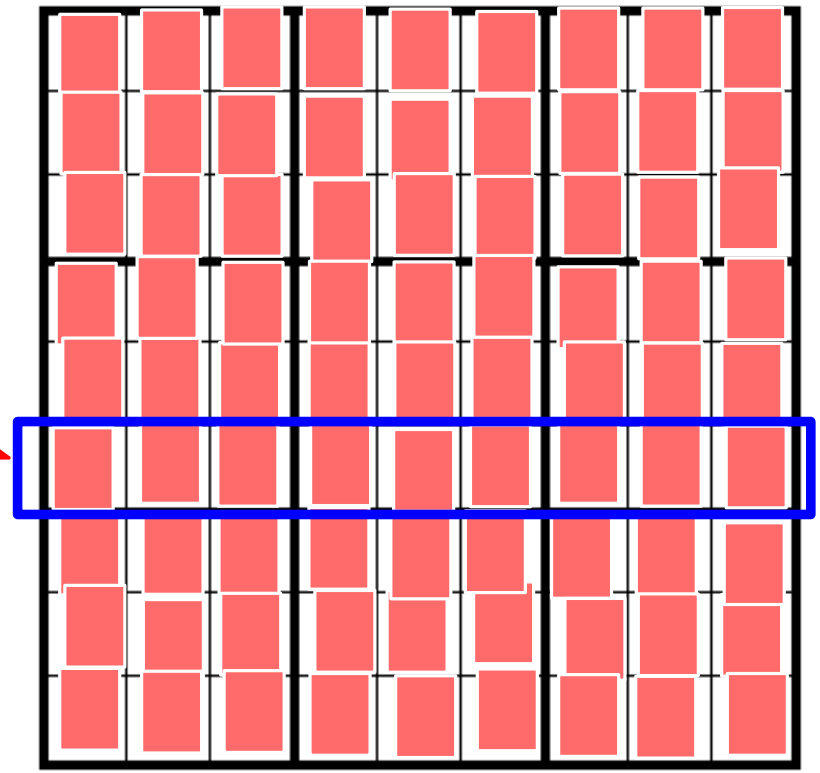
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"
  - Verifier
  - Prover



Preuves zero-knowledge

Preuve zero-knowledge de connaissance

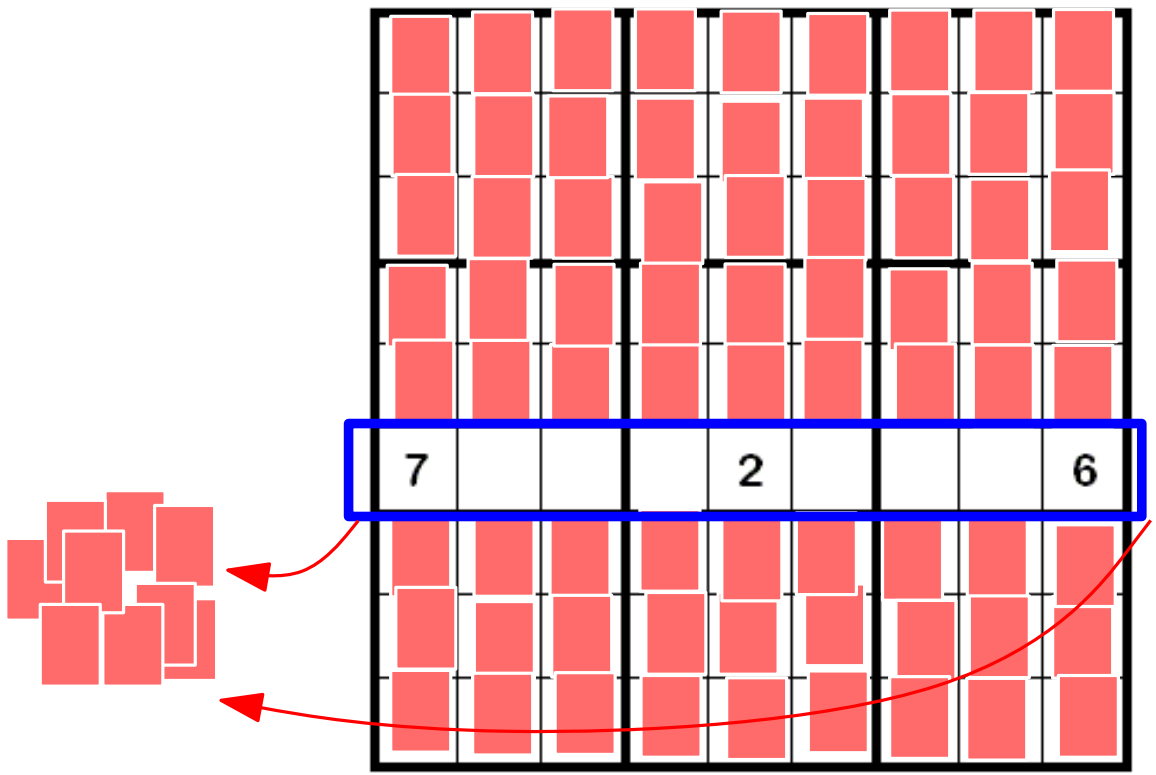
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"
  - Verifier
  - Prover
  - Verifier



Preuves zero-knowledge

Preuve zero-knowledge de connaissance

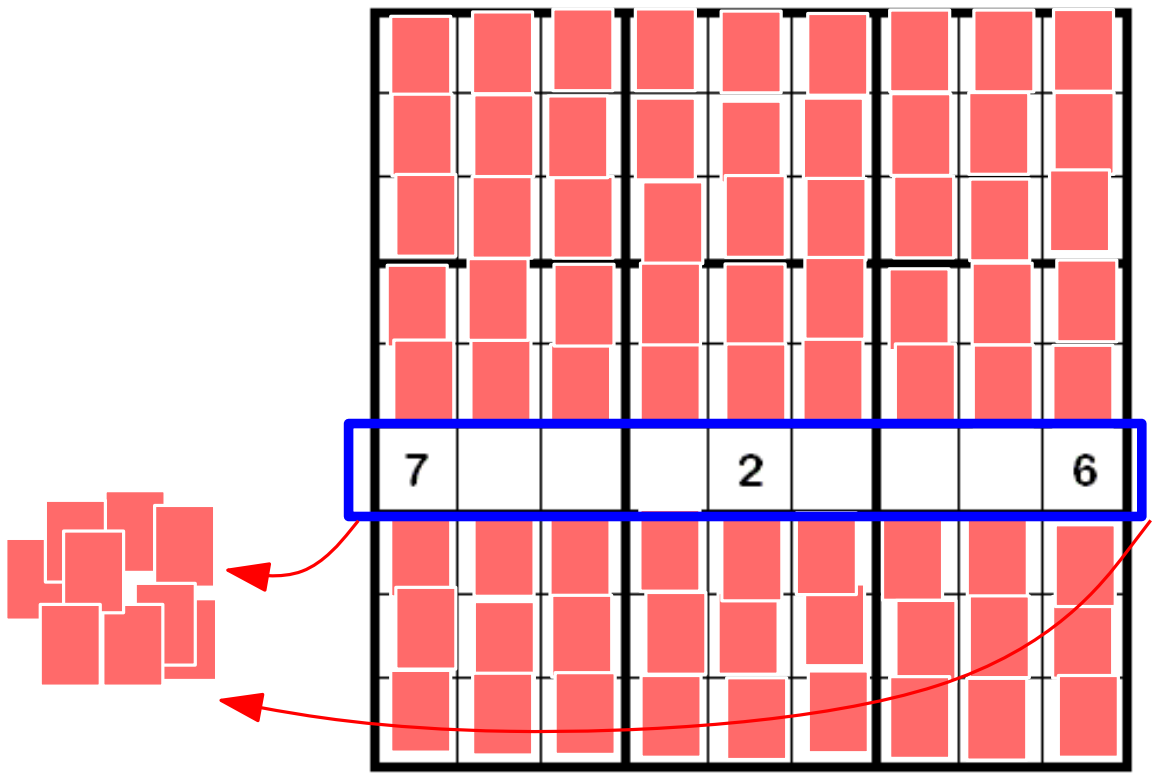
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"
  - Verifier
  - Prover
  - Verifier
  - Prover



Preuves zero-knowledge

Preuve zero-knowledge de connaissance

- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"



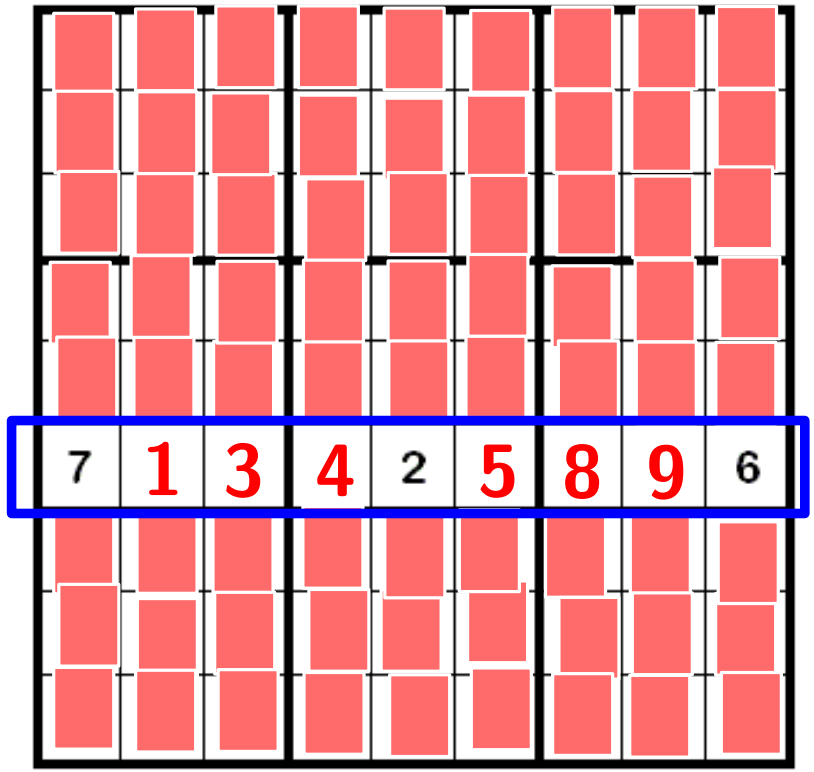
- Verifier
- Prover
- Verifier
- Prover

vérifier : $\{1, \ldots, 9\}$

Preuves zero-knowledge

Preuve zero-knowledge de connaissance

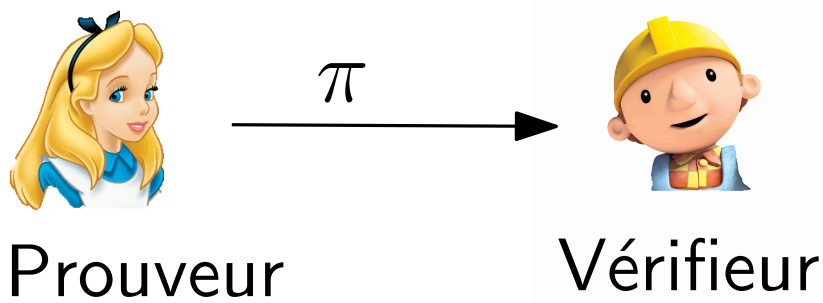
- convaincre quelqu'un qu'une proposition est vrai sans révéler d'autres informations
- **Exemple :** "Je connais une solution d'un sudoku donné"
- il existe un **simulateur** qui sait simuler des preuves **sans** connaître une solution



zk-SNARKs

zero-**k**nowledge
Succinct
Non-interactive
ARgument of **K**nowledge

- prouveur peut convaincre vérifieur de façon *non-interactive*



zk-SNARKs

zero-**k**nowledge

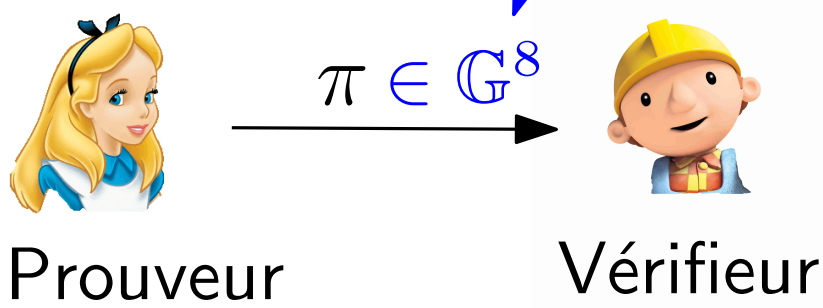
Succinct

Non-interactive

ARgument of **K**nowledge

- taille de la preuve est **constante**

- prouveur peut convaincre vérifieur de façon *non-interactive*



zk-SNARKs

zero-**k**nowledge

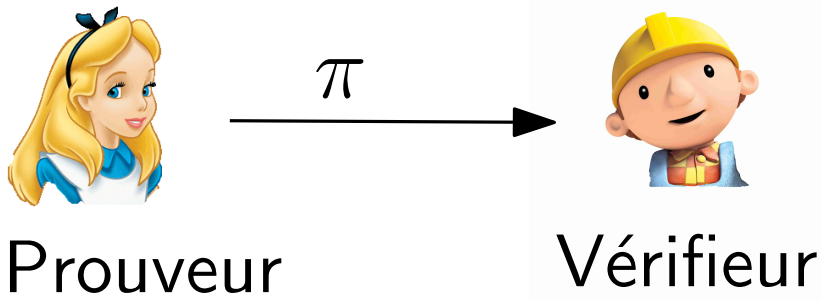
Succinct

Non-interactive

ARgument of **K**nowledge

- prouveur peut convaincre vérifieur de façon *non-interactive*

simulation?



zk-SNARKs

zero-**k**nowledge

Succinct

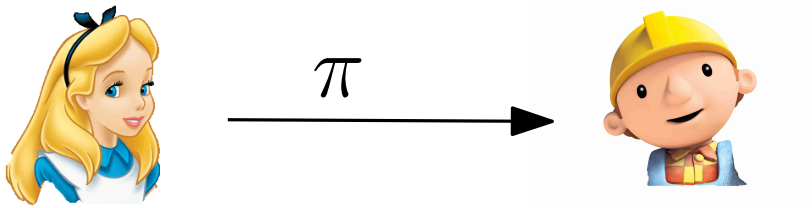
Non-interactive

ARgument of **K**nowledge

- prouveur peut convaincre vérifieur de façon *non-interactive*
- nécessite **paramètres de confiance**

$$par: \quad g^x, g^{x^2}, \ldots$$

simulation?



Prouveur Vérifieur

zk-SNARKs

zero-**k**nowledge

Succinct

Non-interactive

ARgument of **K**nowledge

- prouveur peut convaincre vérifieur de façon *non-interactive*
- nécessite **paramètres de confiance**

$$par: \ \ g^x, g^{x^2}, \ldots$$

x **permet de simuler des preuves**

déchet toxique

π

Prouveur

Vérifieur

Cryptomonnaies anonymes

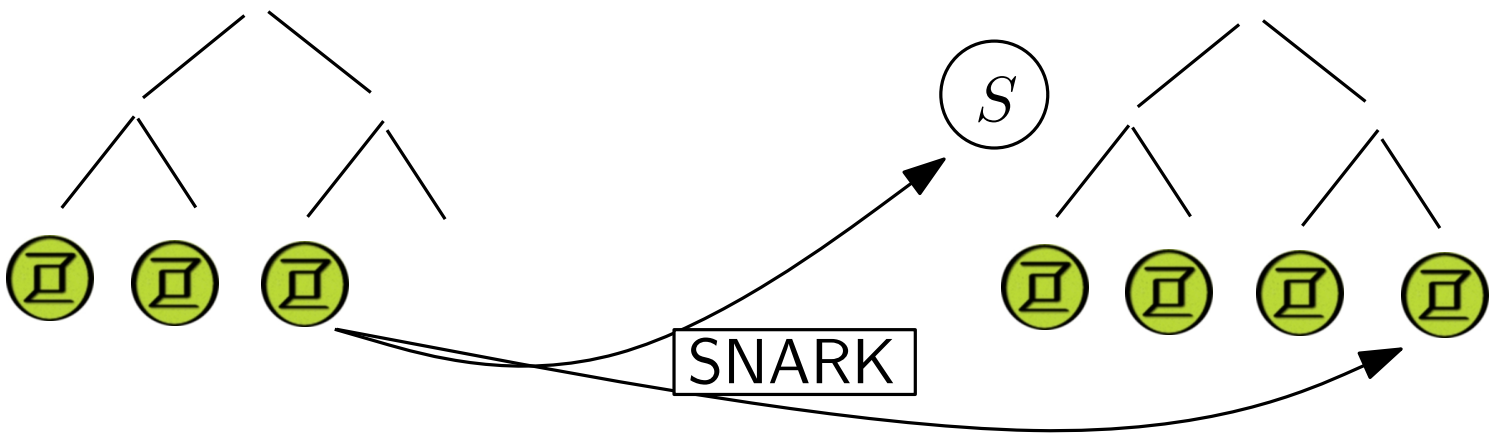
Zerocash [BCGGMTV'14]

- pièce est un "engagement" *(commitment)* sur **numéro de série**

Cryptomonnaies anonymes

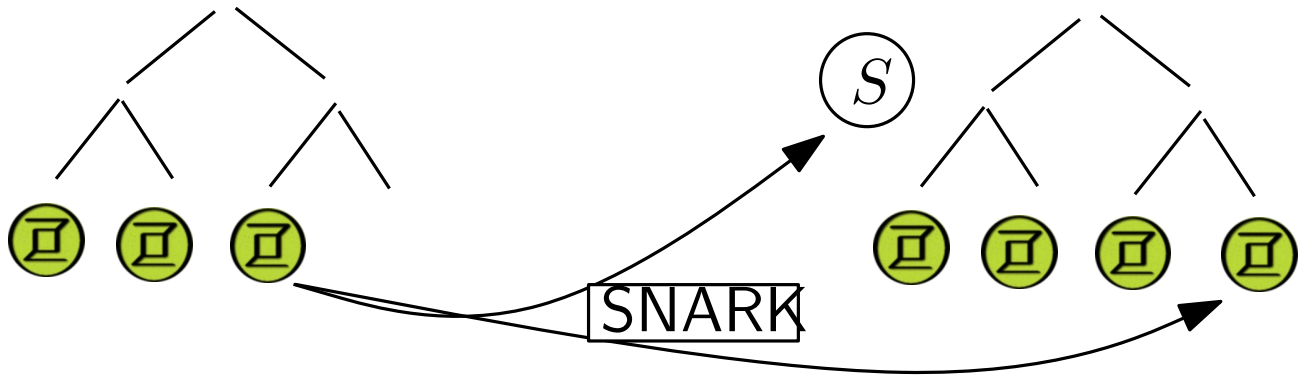
Zerocash [BCGGMTV'14]

- pièce est un "engagement" *(commitment)* sur **numéro de série**
- transaction – crée des nouvelles pièces
  - **révèle** n^0 de série des pièces dépensées
  - **prouve** que tout fait correctement (zk-SNARK)



Cryptomonnaies anonymes

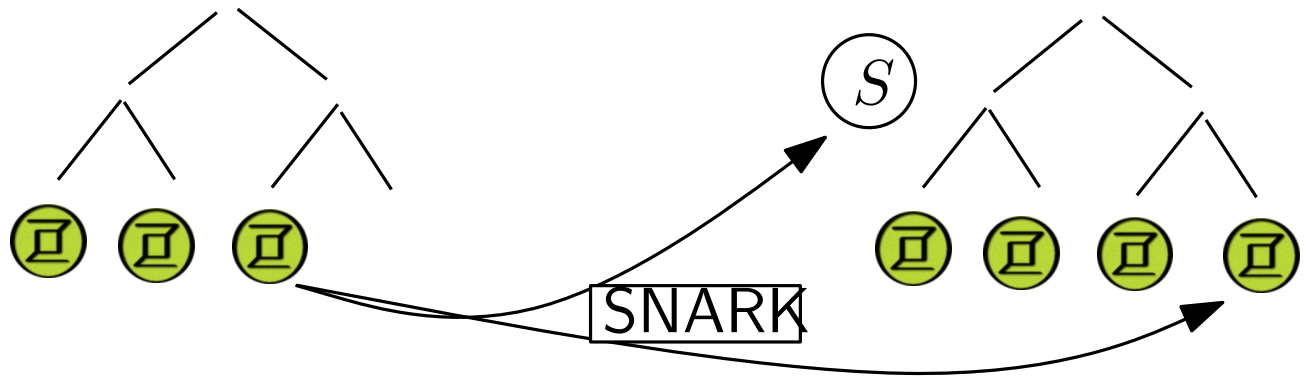
Zerocash



- complètement anonyme !
- création de SNARKs est lent 
- nécessite des paramètres de confiance 

Cryptomonnaies anonymes

Zerocash



- complètement anonyme !
- création de SNARKs est lent 
- nécessite des paramètres de confiance 
  - connaissance du secret permet contrefaction de pièces !
  - anonymat ? **subversion-resistant SNARKs** [F'17]
    SNARKs qui garantissent l'anonymat même quand paramètres sont truqués