

# Introduction à la sécurité informatique - 2

## Approches formelles de la privacité

F. Prost

[Frederic.Prost@insa-lyon.fr](mailto:Frederic.Prost@insa-lyon.fr)

INSA-Lyon

2026

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Anonymat/Identité dans un monde virtuel

- Rien ne se ressemble plus que les 0 et 1 d'une mémoire informatique.

# Anonymat/Identité dans un monde virtuel

- Rien ne se ressemble plus que les 0 et 1 d'une mémoire informatique.
- Problèmes duaux:
  - Anonymat: comment faire pour recevoir l'information (adresse de retour)...
  - Identité: comment prouver qu'on est soit dans un monde virtuel ?

# Anonymat/Identité dans un monde virtuel

- Rien ne se ressemble plus que les 0 et 1 d'une mémoire informatique.
- Problèmes duaux:
  - Anonymat: comment faire pour recevoir l'information (adresse de retour)...
  - Identité: comment prouver qu'on est soit dans un monde virtuel ?
- Le problème de l'intégrité : comment peut on assurer
  - Les messages ne sont pas modifiés ?
  - Les personnes/systèmes identifiés sont bien ceux auxquels on pense ?  
⇒ Identité électronique obligatoire ? (Starmer 26/9/2025)

# Anonymat/Identité dans un monde virtuel

- Rien ne se ressemble plus que les 0 et 1 d'une mémoire informatique.
- Problèmes duaux:
  - Anonymat: comment faire pour recevoir l'information (adresse de retour)...
  - Identité: comment prouver qu'on est soit dans un monde virtuel ?
- Le problème de l'intégrité : comment peut on assurer
  - Les messages ne sont pas modifiés ?
  - Les personnes/systèmes identifiés sont bien ceux auxquels on pense ?  
⇒ Identité électronique obligatoire ? (Starmer 26/9/2025)
- Replay attack: l'identité est par définition ce qui ne change pas, mais les preuves d'identités doivent changer !

# Comment prouver qu'on est soi ?

- Le problème de l'identité est complexe quand il n'y a pas coopération.  
On veut éviter que B ne se fasse passer pour A même après avoir vu des preuves d'identité de A.

# Comment prouver qu'on est soi ?

- Le problème de l'identité est complexe quand il n'y a pas coopération.  
On veut éviter que B ne se fasse passer pour A même après avoir vu des preuves d'identité de A.
- Trois niveaux traditionnels de protection :
  - ➊ Schémas d'Authentification : A peut prouver à B qu'il est A, et personne ne peut prouver à B qu'il est A.

# Comment prouver qu'on est soi ?

- Le problème de l'identité est complexe quand il n'y a pas coopération.  
On veut éviter que B ne se fasse passer pour A même après avoir vu des preuves d'identité de A.
- Trois niveaux traditionnels de protection :
  - ① Schémas d'Authentification : A peut prouver à B qu'il est A, et personne ne peut prouver à B qu'il est A.
  - ② Schémas d'Identification : A peut prouver à B qu'il est A, mais B ne peut pas prouver à quelqu'un d'autre qu'il est A.

# Comment prouver qu'on est soi ?

- Le problème de l'identité est complexe quand il n'y a pas coopération.  
On veut éviter que B ne se fasse passer pour A même après avoir vu des preuves d'identité de A.
- Trois niveaux traditionnels de protection :
  - ① Schémas d'Authentification : A peut prouver à B qu'il est A, et personne ne peut prouver à B qu'il est A.
  - ② Schémas d'Identification : A peut prouver à B qu'il est A, mais B ne peut pas prouver à quelqu'un d'autre qu'il est A.
  - ③ Schéma de Signature : A peut prouver à B qu'il est A, mais B ne peut même pas se prouver à lui-même qu'il est A.
- Différence entre 2 et 3 est quand on amène une affaire en justice et on cherche à montrer que l'identification était correcte (2). (3) est basé sur les ZKP et seulement les interactions directes permettent de faire une preuve de l'identité de A.

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ⇒ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ➡ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)
- Cela consiste en des **protocoles** qui utilisent des briques de bases enchaînées les unes autres pour obtenir un résultat spécifique (semblable à la programmation) :
  - Fonctions de hachage sécurisées,
  - Chiffrage symétrique/asymétrique,
  - Schémas de partage de secret,
  - Bit commitment,
  - Etc.

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ➡ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)
- Cela consiste en des **protocoles** qui utilisent des briques de bases enchaînées les unes autres pour obtenir un résultat spécifique (semblable à la programmation) :
  - Fonctions de hachage sécurisées,
  - Chiffrage symétrique/asymétrique,
  - Schémas de partage de secret,
  - Bit commitment,
  - Etc.
- En plus de ces primitives, on les compose en utilisant des techniques standard:
  - Schémas de challenge/réponse.

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ➡ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)
- Cela consiste en des **protocoles** qui utilisent des briques de bases enchaînées les unes autres pour obtenir un résultat spécifique (semblable à la programmation) :
  - Fonctions de hachage sécurisées,
  - Chiffrage symétrique/asymétrique,
  - Schémas de partage de secret,
  - Bit commitment,
  - Etc.
- En plus de ces primitives, on les compose en utilisant des techniques standard:
  - Schémas de challenge/réponse.
  - Utilisation de **nounces** et de l'aléa en général.

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ➡ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)
- Cela consiste en des **protocoles** qui utilisent des briques de bases enchaînées les unes autres pour obtenir un résultat spécifique (semblable à la programmation) :
  - Fonctions de hachage sécurisées,
  - Chiffrage symétrique/asymétrique,
  - Schémas de partage de secret,
  - Bit commitment,
  - Etc.
- En plus de ces primitives, on les compose en utilisant des techniques standard:
  - Schémas de challenge/réponse.
  - Utilisation de **nounces** et de l'aléa en général.
  - Cut and choose.

# Approche Crypto Traditionnelle de la sécurité

- Assurer la sécurité est plus compliqué que de juste utiliser des fonctions de cryptage.
  - ➡ Très souvent cela impose des contraintes contradictoires: vote électronique, cash électronique, authentification et attaque en replay...)
- Cela consiste en des **protocoles** qui utilisent des briques de bases enchaînées les unes autres pour obtenir un résultat spécifique (semblable à la programmation) :
  - Fonctions de hachage sécurisées,
  - Chiffrage symétrique/asymétrique,
  - Schémas de partage de secret,
  - Bit commitment,
  - Etc.
- En plus de ces primitives, on les compose en utilisant des techniques standard:
  - Schémas de challenge/réponse.
  - Utilisation de **nounces** et de l'aléa en général.
  - Cut and choose.
  - Etc.

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Fonctions de hachage cryptographiques

- Fonctions "presque" injectives, "dures à inverser" ...
- Équivalent informatique des empreintes digitales.
- Fonctions de hachage comme une machine à faire des steak hachés : impossible d'inverser le processus mais si les entrées sont changées le résultat change.

# Fonctions de hachage cryptographiques

- Fonctions "presque" injectives, "dures à inverser" ...
- Équivalent informatique des empreintes digitales.
- Fonctions de hachage comme une machine à faire des steak hachés : impossible d'inverser le processus mais si les entrées sont changées le résultat change.
- L'idée est que si  $h(x) = y$  et  $y$  est enregistré, alors si  $x$  est changé en  $x'$ , on a  $h(x') = y' \neq y$ . (clef numéro de sécu)

# Fonctions de hachage cryptographiques

- Fonctions "presque" injectives, "dures à inverser" ...
- Équivalent informatique des empreintes digitales.
- Fonctions de hachage comme une machine à faire des steak hachés : impossible d'inverser le processus mais si les entrées sont changées le résultat change.
- L'idée est que si  $h(x) = y$  et  $y$  est enregistré, alors si  $x$  est changé en  $x'$ , on a  $h(x') = y' \neq y$ . (clef numéro de sécu)
- La manière la plus simple de faire un MAC (Message Authentication Code), avec une fonction de hachage paramétrée:  
A et B partagent  $k$ , A envoie  $(x, y = h_k(x))$  à B. Alors C ne peut changer  $x$  en  $x'$  et envoyer  $(x', h_k(x'))$  sans connaître  $k$ .

# Formalisation des fonctions de hachage

## Definition (famille de hash)

Une famille de hash  $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$

- $\mathcal{X}$ : ensemble de messages
- $\mathcal{Y}$ : ensemble des hashs/tags d'authentification
- $\mathcal{K}$ : ensemble des clefs
- $\mathcal{H}$ : pour chaque  $k \in \mathcal{K}$ , il y a  $h_k \in \mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$

# Formalisation des fonctions de hachage

## Definition (famille de hash)

Une famille de hash  $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$

- $\mathcal{X}$ : ensemble de messages
- $\mathcal{Y}$ : ensemble des hashs/tags d'authentification
- $\mathcal{K}$ : ensemble des clefs
- $\mathcal{H}$ : pour chaque  $k \in \mathcal{K}$ , il y a  $h_k \in \mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$

### ① Preimage:

In  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $y \in \mathcal{Y}$

Out  $x \in \mathcal{X}$  s.t.  $h(x) = y$

# Formalisation des fonctions de hachage

## Definition (famille de hash)

Une famille de hash  $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$

- $\mathcal{X}$ : ensemble de messages
- $\mathcal{Y}$ : ensemble des hashs/tags d'authentification
- $\mathcal{K}$ : ensemble des clefs
- $\mathcal{H}$ : pour chaque  $k \in \mathcal{K}$ , il y a  $h_k \in \mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$

### ① Preimage:

In  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $y \in \mathcal{Y}$

Out  $x \in \mathcal{X}$  s.t.  $h(x) = y$

### ② Second Preimage:

In  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $x \in \S$

Out  $x' \in \mathcal{X}$  s.t.  $h(x) = h(x')$  and  $x \neq x'$

# Formalisation des fonctions de hachage

## Definition (famille de hash)

Une famille de hash  $(\mathcal{X}, \mathcal{Y}, \mathcal{K}, \mathcal{H})$

- $\mathcal{X}$ : ensemble de messages
- $\mathcal{Y}$ : ensemble des hashs/tags d'authentification
- $\mathcal{K}$ : ensemble des clefs
- $\mathcal{H}$ : pour chaque  $k \in \mathcal{K}$ , il y a  $h_k \in \mathcal{H} : \mathcal{X} \rightarrow \mathcal{Y}$

### ① Preimage:

**In**  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $y \in \mathcal{Y}$

**Out**  $x \in \mathcal{X}$  s.t.  $h(x) = y$

### ② Second Preimage:

**In**  $h : \mathcal{X} \rightarrow \mathcal{Y}$ , and  $x \in \S$

**Out**  $x' \in \mathcal{X}$  s.t.  $h(x) = h(x')$  and  $x \neq x'$

### ③ Collision:

**In**  $h : \mathcal{X} \rightarrow \mathcal{Y}$

**Out**  $x, x' \in \mathcal{X}$  s.t.  $h(x) = h(x')$  and  $x \neq x'$

# Les propriétés des fonctions de hachage: Random Oracle Model

- Bellare and Rogaway in 1995 [Bellare and Rogaway, 1995].
- L'idée est de capturer l'essence d'une fonction de hachage "idéale":
  - $h : \mathcal{X} \rightarrow \mathcal{Y}$  est choisi aléatoirement.
  - $h$  est vu comme une boite noire: accède par un Oracle.

# Les propriétés des fonctions de hachage: Random Oracle Model

- Bellare and Rogaway in 1995 [Bellare and Rogaway, 1995].
- L'idée est de capturer l'essence d'une fonction de hachage "idéale":
  - $h : \mathcal{X} \rightarrow \mathcal{Y}$  est choisi aléatoirement.
  - $h$  est vu comme une boite noire: accède par un Oracle.
- On peut analyser les fonctions de hachage indépendamment des particularités de certaines fonctions de hachage.
- Notion de *randomized algorithms*.

# Les propriétés des fonctions de hachage: Random Oracle Model

- Bellare and Rogaway in 1995 [Bellare and Rogaway, 1995].
- L'idée est de capturer l'essence d'une fonction de hachage "idéale":
  - $h : \mathcal{X} \rightarrow \mathcal{Y}$  est choisi aléatoirement.
  - $h$  est vu comme une boite noire: accède par un Oracle.
- On peut analyser les fonctions de hachage indépendamment des particularités de certaines fonctions de hachage.
- Notion de *randomized algorithms*.
- Notion d' $\epsilon$  probabilité de succès en moyenne. relativement aux nombres de requêtes  $Q$  à l'oracle:  $(\epsilon, Q)$ .

# Pré-image

```
Find_PreImage(h,y,Q):  
    choose X0 subset of X, |X0|=Q  
    for all x in X0  
        do if h(x)=y then return (x)  
    return (fail)
```

# Pré-image

```
Find_PreImage(h, y, Q):  
    choose X0 subset of X, |X0|=Q  
    for all x in X0  
        do if h(x)=y then return (x)  
    return (fail)
```

## Theorem

*Si  $|X| = M$  la moyenne des succès de Find\_Preimage est :*

$$\epsilon = 1 - (1 - 1/M)^Q$$

## Deuxième Pré-image

Find\_Second\_PreImage( $h, x, Q$ ):

```
y := h(x)
choose X0 subset of X\{x}, |X0|=Q-1
for all x0 in X0
    do if h(x0)=y then return (x0)
return (fail)
```

## Deuxième Pré-image

Find\_Second\_PreImage( $h, x, Q$ ):

```
y := h(x)
choose X0 subset of X\{x}, |X0|=Q-1
for all x0 in X0
    do if h(x0)=y then return (x0)
return (fail)
```

### Theorem

*Si  $|X| = M$  la moyenne des succès de Find\_Second\_Preimage est :*

$$\epsilon = 1 - (1 - 1/M)^{Q-1}$$

# Collision

```
Find_Collision(h,x,Q):
```

```
    Choose X0 subset of X\{x}, |X0|=Q-1
```

```
    for all x in X0
```

```
        do y[x] := h(x)
```

```
    if y[x]=y[x'] for some x <> x'
```

```
        then return (x,x')
```

```
    else return (fail)
```

# Collision

```
Find_Collision(h,x,Q):
```

```
    Choose X0 subset of X\{x}, |X0|=Q-1
```

```
    for all x in X0
```

```
        do y[x] := h(x)
```

```
    if y[x]=y[x'] for some x <> x'
```

```
        then return (x,x')
```

```
    else return (fail)
```

## Theorem

*La probabilité de succès moyen de Find\_Collision est, en supposant que  $|X| = M$*

$$\epsilon = 1 - \left(\frac{M-1}{M}\right)\left(\frac{M-2}{M}\right) \dots \left(\frac{M-Q+1}{M}\right)$$

## Quelques chiffres

- Le Paradoxe des anniversaires et Find\_Second\_Preimage :  
Dans un groupe de 23 personnes il y a 1/2 que deux personnes soient nées le même jour :  $Q = 23$  et  $M = 365$ .

## Quelques chiffres

- Le Paradoxe des anniversaires et Find\_Second\_Preimage :  
Dans un groupe de 23 personnes il y a 1/2 que deux personnes soient nées le même jour :  $Q = 23$  et  $M = 365$ .
- Le théorème de l'analyse de Find\_Collision 4 donne la probabilité qu'il n'y ait pas de collision:

$$\prod_{i=1}^{Q-1} \left(1 - \frac{i}{M}\right)$$

quand  $x \rightarrow 0$  on a  $1 - x \simeq \exp^{-x}$ , d'où

$$\begin{aligned} \prod_{i=1}^{Q-1} \left(1 - \frac{i}{M}\right) &\simeq \prod_{i=1}^{Q-1} \exp^{\frac{-i}{M}} \\ &= \exp^{-\sum_{i=1}^{Q-1} \frac{i}{M}} \\ &= \exp^{\frac{-Q(Q-1)}{2M}} \end{aligned}$$

$$Q \simeq \sqrt{2M \log\left(\frac{1}{1-\epsilon}\right)}$$

pour  $\epsilon = 1/2$  on a  $Q \simeq 1.17\sqrt{M}$ .

# Collision en utilisant la deuxième Préimage

- On peut facilement trouver une collision en utilisant l'algorithme de second-preimage:

```
choose random x in X  
if Find_Second_Preimage (h,x,Q)=x'  
    then return (x,x')  
else return failure
```

- Attention à la taille de shash (paradoxe des anniversaires) !

Avec une proba 1/2 un hash de 40-bits n'a besoin que de  $2^{20}$  hash aléatoires.

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Philosophie de l'identité

- Problème philosophique ancien et complexe : social vs. le sentiment d'identité personnelle (Platon, déclaration "soyez vous même" etc.).

# Philosophie de l'identité

- Problème philosophique ancien et complexe : social vs. le sentiment d'identité personnelle (Platon, déclaration "soyez vous même" etc.).
- Plus techniquement, on identifie trois manières de prouver son identité :
  - Ce que vous savez: mot de passe, preuve d'un théorème, etc.

# Philosophie de l'identité

- Problème philosophique ancien et complexe : social vs. le sentiment d'identité personnelle (Platon, déclaration "soyez vous même" etc.).
- Plus techniquement, on identifie trois manières de prouver son identité :
  - Ce que vous savez: mot de passe, preuve d'un théorème, etc.
  - Ce que vous avez: clef, carte, smartphone (via sms), email

# Philosophie de l'identité

- Problème philosophique ancien et complexe : social vs. le sentiment d'identité personnelle (Platon, déclaration "soyez vous même" etc.).
- Plus techniquement, on identifie trois manières de prouver son identité :
  - Ce que vous savez: mot de passe, preuve d'un théorème, etc.
  - Ce que vous avez: clef, carte, smartphone (via sms), email
  - Ce que vous êtes: biométrie.

# Philosophie de l'identité

- Problème philosophique ancien et complexe : social vs. le sentiment d'identité personnelle (Platon, déclaration "soyez vous même" etc.).
- Plus techniquement, on identifie trois manières de prouver son identité :
  - Ce que vous savez: mot de passe, preuve d'un théorème, etc.
  - Ce que vous avez: clef, carte, smartphone (via sms), email
  - Ce que vous êtes: biométrie.
- Chaque méthode à ses avantages et ses inconvénients.

# Challenge-Response et Randomization

- L'identification est un processus quotidien qui va du "c'est moi" à l'interphone jusqu'à la "nuclear football" des puissances atomiques.
- C'est maintes fois répété (souvent des machines entre elles: photocopieuse/ordi etc.).

# Challenge-Response et Randomization

- L'identification est un processus quotidien qui va du "c'est moi" à l'interphone jusqu'à la "nuclear football" des puissances atomiques.
- C'est maintes fois répété (souvent des machines entre elles: photocopieuse/ordi etc.).
- Schéma non sûr: supposons qu'Alice et Bob partagent un secret  $k$  (un mot de passe)
  - ① Bob choisit un nombre aléatoire  $r$ , et l'envoie à Alice.
  - ② Alice calcule  $y = h_K(r)$  et envoie  $y$  à Bob.
  - ③ Bob calcule  $y' = h_K(r)$ . Si  $y = y'$  Bob accepte, sinon il rejette.

# Challenge-Response et Randomization

- L'identification est un processus quotidien qui va du "c'est moi" à l'interphone jusqu'à la "nuclear football" des puissances atomiques.
- C'est maintes fois répété (souvent des machines entre elles: photocopieuse/ordi etc.).
- Schéma non sûr: supposons qu'Alice et Bob partagent un secret  $k$  (un mot de passe)
  - ① Bob choisit un nombre aléatoire  $r$ , et l'envoie à Alice.
  - ② Alice calcule  $y = h_K(r)$  et envoie  $y$  à Bob.
  - ③ Bob calcule  $y' = h_K(r)$ . Si  $y = y'$  Bob accepte, sinon il rejette.
- Attaque (session parallèle) de ce schéma:
  - ① Bob choisit aléatoirement  $r$ , l'envoie à Alice.

# Challenge-Response et Randomization

- L'identification est un processus quotidien qui va du "c'est moi" à l'interphone jusqu'à la "nuclear football" des puissances atomiques.
- C'est maintes fois répété (souvent des machines entre elles: photocopieuse/ordi etc.).
- Schéma non sûr: supposons qu'Alice et Bob partagent un secret  $k$  (un mot de passe)
  - ① Bob choisit un nombre aléatoire  $r$ , et l'envoie à Alice.
  - ② Alice calcule  $y = h_K(r)$  et envoie  $y$  à Bob.
  - ③ Bob calcule  $y' = h_K(r)$ . Si  $y = y'$  Bob accepte, sinon il rejette.
- Attaque (session parallèle) de ce schéma:
  - ① Bob choisit aléatoirement  $r$ , l'envoie à Alice.
  - ② Oscar intercepte  $r$  et le renvoie à Bob
  - ③ Bob pensant qu'il a reçu une requête d'identification d'Alice calcule  $y = h_K(r)$  et renvoie  $y$ .
  - ④ Oscar peut alors se faire passer pour Alice en renvoyant  $y$  qu'il a reçu à Bob.

# Version corrigée

- Version sécurisée:
  - ➊ Bob choisit aléatoirement  $r$ , l'envoie à Alice.

# Version corrigée

- Version sécurisée:

- ① Bob choisit aléatoirement  $r$ , l'envoie à Alice.
- ② Alice calcule  $y = h_K(ID(Alice), r)$  et envoie  $y$  à Bob.
- ③ Bob calcule  $y' = h_K(ID(Alice), r)$ . If  $y = y'$  Bob accepte, sinon il rejette.

# Version corrigée

- Version sécurisée:
  - ➊ Bob choisit aléatoirement  $r$ , l'envoie à Alice.
  - ➋ Alice calcule  $y = h_K(ID(Alice), r)$  et envoie  $y$  à Bob.
  - ➌ Bob calcule  $y' = h_K(ID(Alice), r)$ . If  $y = y'$  Bob accepte, sinon il rejette.
- Si Oscar essaie de lancer une session parallèle, il le fait avec une mauvaise identité: savoir  $h_K(ID(Bob), r)$  n'aide pas pour le calcul de  $h_k(ID(Alice), r)$ .

# Version corrigée

- Version sécurisée:
  - ① Bob choisit aléatoirement  $r$ , l'envoie à Alice.
  - ② Alice calcule  $y = h_K(ID(Alice), r)$  et envoie  $y$  à Bob.
  - ③ Bob calcule  $y' = h_K(ID(Alice), r)$ . If  $y = y'$  Bob accepte, sinon il rejette.
- Si Oscar essaie de lancer une session parallèle, il le fait avec une mauvaise identité: savoir  $h_K(ID(Bob), r)$  n'aide pas pour le calcul de  $h_k(ID(Alice), r)$ .
- Y a t il d'autres attaques ????

# Version corrigée

- Version sécurisée:
  - ① Bob choisit aléatoirement  $r$ , l'envoie à Alice.
  - ② Alice calcule  $y = h_K(ID(Alice), r)$  et envoie  $y$  à Bob.
  - ③ Bob calcule  $y' = h_K(ID(Alice), r)$ . If  $y = y'$  Bob accepte, sinon il rejette.
- Si Oscar essaie de lancer une session parallèle, il le fait avec une mauvaise identité: savoir  $h_K(ID(Bob), r)$  n'aide pas pour le calcul de  $h_k(ID(Alice), r)$ .
- Y a t il d'autres attaques ????  
⇒ Que suppose-t-on ?
  - Clef secrète.
  - Random Challenges.
  - MAC Security.

# Signature en un transparent

- Comment authentifier qu'on est à l'origine d'un message?
- Signature électronique/numérique (pas comme un dessin sur un pdf):  
     $\Rightarrow$  est basé sur la cryptographie asymétrique public/privé.
- Cryptographie à clef publique repose sur une paire  $k, \bar{k}$ .  $k$  est privée et  $\bar{k}$  est publique.  
Typiquement RSA, El Gammal, courbes elliptiques, vecteurs discrets.
- Tout le monde peut crypter avec  $\bar{k}$  mais seul celui qui connaît  $k$  peut déchiffrer.
- Si le chiffrement et le déchiffrement commutent on a  
 $k(\bar{k}(m)) = \bar{k}(k(m)) = m$ .
- un schéma de signature simplissime est Bob envoie à Alice  $m, k(m)$ .  
Alice peut calculer  $\bar{k}(k(m))$  et vérifier que cela donne bien  $m$ .  
Seul Bob, réputé être le seul à connaître  $k$  a pu engendrer un tel message.

# Conclusion

- Il existe des livres entiers de MAC/protocoles d'identification/schéma de signatures.
- Différents protocoles pour différents usages:
  - One time passwords.
  - Tickets avec une durée de vie limitée (typiquement les login de mail).
  - Avec ou sans autorité centrale.
  - etc.
- Il est très difficile d'avoir de "preuves convaincantes" de leur correction.

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Communiquer sans révéler son identité

- Problème de la vie courante:
  - Confession,
  - Tests médicaux anonymes,
  - Consultations médicales anonymes,
  - Etc.
- Semble paradoxale: il faut bien une adresse de retour pour s'avoir où envoyer la réponse sur internet !

# Communiquer sans révéler son identité

- Problème de la vie courante:
  - Confession,
  - Tests médicaux anonymes,
  - Consultations médicales anonymes,
  - Etc.
- Semble paradoxale: il faut bien une adresse de retour pour s'avoir où envoyer la réponse sur internet !
- Plusieurs points de vue:
  - Sender anonymity.
  - Receiver anonymity.
  - Observateur externe vs Interne.

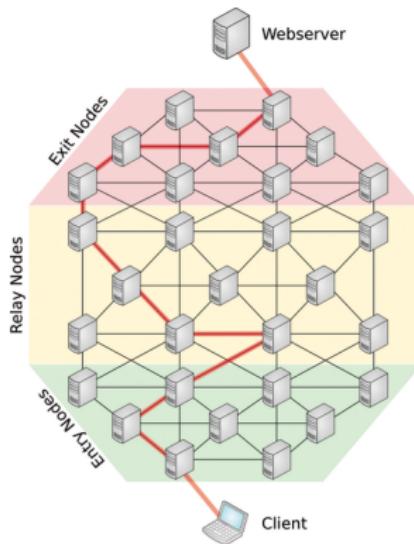
# Sender Anonymity

- On veut que celui qui envoie la requête reste anonyme tout en permettant de lui envoyer une réponse.
  - Utile pour les revues de papiers scientifiques.
  - Web surfing dans un pays totalitaire..
  - Etc.
- L'adresse de retour devrait permettre de toujours retrouver l'émetteur de la requête.
- On passe par l'empilement de couches d'encodage (en utilisant un cryptage asymétrique qui commute) :

$$K(\overline{K}(M)) = \overline{K}(K(M)) = M$$

# Chaum's mix nets [Chaum, 1981], L'idée

- L'idée est de mettre plusieurs enveloppes autour du message de façon à ce que chaque intermédiaire ne connaissent que deux liens sur le chemin.



# Chaum's mix nets [Chaum, 1981], les spécifications

- La clef publique de  $A$  est assimilée à son nom, et sa clef privée à  $\bar{A}$ .
- Les relais sont appelés des “mixes” ils reçoivent des messages, les mélange puis les renvoie.
- La structure d'un message envoyé à un mix  $K$  est

$$K(R, K_{next}(R_{next}, M), K_{next})$$

Le mix  $K$  peut le décrypter et envoyer la seconde partie du message ( $K_{next}(R_{next}, M)$ ) à  $K_{next}$

# Chaum's mix nets [Chaum, 1981], les spécifications

- La clef publique de  $A$  est assimilée à son nom, et sa clef privée à  $\bar{A}$ .
- Les relais sont appelés des “mixes” ils reçoivent des messages, les mélange puis les renvoie.
- La structure d'un message envoyé à un mix  $K$  est

$$K(R, K_{next}(R_{next}, M), K_{next})$$

Le mix  $K$  peut le décrypter et envoyer la seconde partie du message ( $K_{next}(R_{next}, M)$ ) à  $K_{next}$

- La procédure peut être récursivement répétée :

$$K_n(R_n, K_{n-1}(R_{n-1}, \dots, K_2(R_2, K_1(R_1, B(R_0, M), B), K_1) \dots), K_{n-1})$$

## Chaum's mix nets [Chaum, 1981], Adresse de retour

- Une adresse de retour anonyme peut être ajoutée au message:

$$K_1(R_1, A), K_A$$

avec  $K_A$  une clef de session,  $A$  étant l'adresse d'Alice. Bob renvoie:

$$K_1(R_1, A), K_A(R_0, M)$$

et mix  $K_1$  décrypte la première couche de cryptage du message et envoie:

$$R_1(K_A(R_0, M))$$

à  $A$  en utilisant  $R_1$  comme clef d'encryption.

- On peut faire une construction récursive ::

$$K_1(R_1, K_2(R_2, \dots, K_n(R_n, A) \dots)), K_A(R_0, M)$$

- A la fin Alice reçoit:

$$R_n(R_{n-1}(\dots R_2(R_1(K_A(R_0, M)) \dots)))$$

# Mix Nets dans la vie réelle

- Attaque des mix nets:

# Mix Nets dans la vie réelle

- Attaque des mix nets:
  - Noyer le réseaux de faux messages.

# Mix Nets dans la vie réelle

- Attaque des mix nets:

- Noyer le réseaux de faux messages.
- Timing attacks entre les noeuds d'entrées et de sorties.

# Mix Nets dans la vie réelle

- Attaque des mix nets:

- Noyer le réseaux de faux messages.
- Timing attacks entre les noeuds d'entrées et de sorties.
- Posséder (hacker) beaucoup de relais.
- etc. ref <http://freehaven.net/anonbib/>

# Mix Nets dans la vie réelle

- Attaque des mix nets:
  - Noyer le réseaux de faux messages.
  - Timing attacks entre les noeuds d'entrées et de sorties.
  - Posséder (hacker) beaucoup de relais.
  - etc. ref <http://freehaven.net/anonbib/>
- Mix nets en pratique: The Onion routing, aka Tor.
  - Pas exactement des mix nets mais idées similaires.
  - NSAproof (Ils essayent d'autres attaques quand vous utilisez TOR).
  - More than 2.5 Million Users et 6,500 relais:  
<https://metrics.torproject.org/>

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Monnaie Electronique Anonyme

- Paiement sans mécanisme d'audit (typiquement l'opposé de BitCoin)

...

# Monnaie Electronique Anonyme

- Paiement sans mécanisme d'audit (typiquement l'opposé de BitCoin)  
...  
● Problème de base à résoudre :
  - fausse monnaie ?
  - double spending ? (un fichier reste inchangé quand on le lit/transmet).

# Monnaie Electronique Anonyme

- Paiement sans mécanisme d'audit (typiquement l'opposé de BitCoin)  
...
- Problème de base à résoudre :
  - fausse monnaie ?
  - double spending ? (un fichier reste inchangé quand on le lit/transmet).
- Problème politique: les gouvernements n'aiment pas lacher le contrôle de la monnaie.
- Protocoles complexes mêlant plusieurs ingrédients cryptos: bit commitment, secret sharing et Blind signatures.

# Blind Signature

- Normalement on sait ce qu'on signe...

# Blind Signature

- Normalement on sait ce qu'on signe...
- Pour respecter la privavité cela pourrait être une bonne idée de ne pas le savoir.

# Blind Signature

- Normalement on sait ce qu'on signe...
  - Pour respecter la privavité cela pourrait être une bonne idée de ne pas le savoir.
  - Completely blind signature:
    - ① Alice prend le message à signer et le multiplie par un nombre aléatoire (blinding factor).
    - ② Alice envoie le blinded message à Bob.
    - ③ Bob signe le blinded message.
    - ④ Alice divise par le blinding factor.
- ⇒ La signature la multiplication doivent communter  $f(m * r) = r * f(m)$  (par exemple RSA).

# Bit Commitment

- Problème: s'engager à une prédition sans avoir à la réveler. Le vérificateur veut être sûr que la prédition ne peut pas être changée une fois qu'elle a été faite.

# Bit Commitment

- Problème: s'engager à une prédition sans avoir à la réveler. Le vérificateur veut être sûr que la prédition ne peut pas être changée une fois qu'elle a été faite.
- Solution avec cryptographie symétrique :
  - ① Bob engendre  $R$ , l'envoie à Alice
  - ② Alice fait sa prédition (un bit)  $b$  et envoie  $K(R, b)$
  - ③ Quand le temps est venu de rendre la prévision publique Alice envoie  $K$  à Bob.
  - ④ Bob décrypte le message reçu et rend public le bit de prédition et vérifie  $R$ .

# Bit Commitment

- Problème: s'engager à une prédition sans avoir à la réveler. Le vérificateur veut être sûr que la prédition ne peut pas être changée une fois qu'elle a été faite.
- Solution avec cryptographie symétrique :
  - ① Bob engendre  $R$ , l'envoie à Alice
  - ② Alice fait sa prédition (un bit)  $b$  et envoie  $K(R, b)$
  - ③ Quand le temps est venu de rendre la prévision publique Alice envoie  $K$  à Bob.
  - ④ Bob décrypte le message reçu et rend public le bit de prédition et vérifie  $R$ .
- Solution avec des fonctions hachage sûres:
  - ① Alice engendre  $R_1, R_2$  et envoie  $H(R_1, R_2, b), R_1$  à Bob.
  - ② Quand le temps est venu de rendre la prévision publique Alice envoie à Bob. ( $R_1, R_2, b$ )
  - ③ Bob calcule le hash et le compare à celui reçu ainsi que  $R_1$

# Secret Sharing

- Version crypto de protocoles implantés dans des environnements ultra sécurisés: banques, lancement de missiles nucléaires et. Plusieurs personnes doivent se mettre d'accord pour faire une action.

# Secret Sharing

- Version crypto de protocoles implantés dans des environnements ultra sécurisés: banques, lancement de missiles nucléaires et. Plusieurs personnes doivent se mettre d'accord pour faire une action.

## Definition

Soient  $t, w$  des entiers positifs,  $t \leq w$ . Un  $(t, w)$ -threshold schémas est une méthode pour partager une clef  $K$  parmi  $w$  participants de telle manière à ce que tout sous ensemble de taille  $t$  peut calculer  $K$  mais aucun groupe plus petit ne le puisse.

# Secret Sharing

- Version crypto de protocoles implantés dans des environnements ultra sécurisés: banques, lancement de missiles nucléaires et. Plusieurs personnes doivent se mettre d'accord pour faire une action.

## Definition

Soient  $t, w$  des entiers positifs,  $t \leq w$ . Un  $(t, w)$ -threshold schémas est une méthode pour partager une clef  $K$  parmi  $w$  participants de telle manière à ce que tout sous ensemble de taille  $t$  peut calculer  $K$  mais aucun groupe plus petit ne le puisse.

- Le Shamir scheme est inconditionnellement sûr (ne dépend pas de la puissance de calcul de l'adversaire)
- Enormément d'applications en crypto.

## Secret Sharing [Shamir, 1979], définition

$D$  est le dealer.  $P_i, 1 \leq i \leq w$  sont les participants.  $K \in \mathbb{Z}_p$  est le secret à partager ( $p > w$ ).

# Secret Sharing [Shamir, 1979], définition

$D$  est le dealer.  $P_i, 1 \leq i \leq w$  sont les participants.  $K \in \mathbb{Z}_p$  est le secret à partager ( $p > w$ ).

## Definition (Shamir $(t, w)$ -Threshold Scheme)

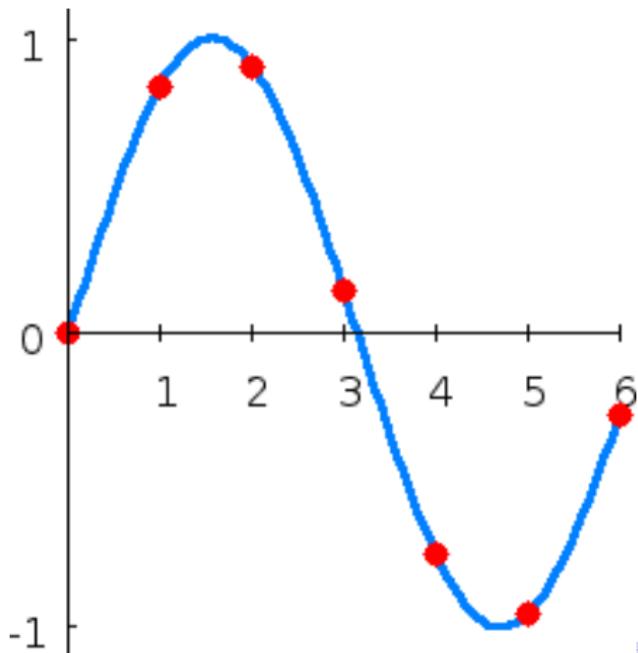
- ① **Initialization Phase:**  $D$  choisit  $w$  distincts, non-zéro éléments de  $\mathbb{Z}_p$  :  $x_i, 1 \leq i \leq w$ . Pour  $i \in \{1, \dots, w\}$ ,  $D$  donne  $x_i$  à  $P_i$ .  $x_i$  sont des valeurs publiques.
- ② **Share Distribution:**  $D$  choisit de manière aléatoire et secrète  $t - 1$  éléments de  $\mathbb{Z}_p$  :  $a_1, \dots, a_{t-1}$ .
- ③ Pour  $1 \leq w \leq w$ ,  $D$  calcule

$$y_i = a(x_i) = K + \sum_{j=1}^{t-1} a_j x_i^j \mod p$$

- ④ Pour  $1 \leq i \leq w$ ,  $D$  donne  $y_i$  à  $P_i$

# Secret Sharing [Shamir, 1979], interprétation géométrique

- Le schéma repose sur le polynôme d'interpolation de Lagrange: il n'y a qu'un seul polynôme de degré  $t - 1$  qui passe par  $t$  points différents !



## Secret Sharing [Shamir, 1979], calcul du secret

- Supposons  $P_{i_1}, \dots, P_{i_t}$  veulent retrouver le secret. Ils savent que  $y_{i_j} = a(x_{i_j})$
- Comme  $a(x)$  a un degré au plus  $t - 1$ :

$$a(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1}$$

- On se retrouve avec  $t$  équations linéaires à  $t$  inconnues  $a_0 + \dots + a_{t-1}$ , il n'y a qu'une solution et  $a_0$  est la clef !

# Secret Sharing [Shamir, 1979], calcul du secret

- Supposons  $P_{i_1}, \dots, P_{i_t}$  veulent retrouver le secret. Ils savent que  $y_{i_j} = a(x_{i_j})$
- Comme  $a(x)$  a un degré au plus  $t - 1$ :

$$a(x) = a_0 + a_1x + \dots + a_{t-1}x^{t-1}$$

- On se retrouve avec  $t$  équations linéaires à  $t$  inconnues  $a_0 + \dots + a_{t-1}$ , il n'y a qu'une solution et  $a_0$  est la clef !
- Plus facile d'utiliser la formule Lagrangienne d'interpolation (il suffit de calculer  $a(0)$ ):

$$K = \sum_{j=1}^t \left( y_{i_j} \prod_{1 \leq k \leq t, k \neq j} \frac{x_{i_k}}{x_{i_k} - x_{i_j}} \right)$$

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$.
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$.
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe e tva la dépenser chez le marchant.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$.
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok l'accepte et le transfert à la banque.

# E-cash Protocole 1 ( à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$.
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$.
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok l'accepte et le transfert à la banque.
- ⑦ La banque vérifie la signature et crédite 1000 \$ sur le compte du marchant.

# E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ ajoute à chaque billet un nombre aléatoire distinct.

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$ **qui ont tous des nombres aléatoires distincts.**

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok l'accepte et le transfert à la banque.

## E-cash Protocole 2 (à partir de [Chaum, 1982])

- ① Alice prépare 100 billets anonyme pour 1000 \$ **ajoute à chaque billet un nombre aléatoire distinct.**
- ② Alice masque les 100 billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 enveloppes et confirme que ce sont bien tous des billets de 1000 \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte 1000\$ du compte d'Alice.
- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok l'accepte et le transfert à la banque.
- ⑦ La banque vérifie la signature **et vérifie que le nombre aléatoire n'a jamais été utilisé**, crédite 1000 \$ sur le compte du marchant, et **enregistre le nombre aléatoire.**

# E-cash [Chaum, 1982]

- ① Alice prépare  $n$  billets anonymes de la forme

$$\langle A, X, I_1, \dots, I_n \rangle$$

tels que  $A$  est le montant en \$,  $X$  est un grand nombre aléatoire et chaque  $I_j = (I_{jL}, I_{jR})$  est une information d'identité qui est partagée en 2 (en utilisant un Shamir scheme de partage de secret). De plus Alice commit sur chacune des parties.

# E-cash [Chaum, 1982]

- ① Alice prépare  $n$  billets anonymes de la forme

$$\langle A, X, I_1, \dots, I_n \rangle$$

tels que  $A$  est le montant en \$,  $X$  est un grand nombre aléatoire et chaque  $I_j = (I_{jL}, I_{jR})$  est une information d'identité qui est partagée en 2 (en utilisant un Shamir scheme de partage de secret). De plus Alice commit sur chacune des parties.

- ② Alice blinds les  $n$  billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 envelopes et confirme que ce sont bien tous des billets de  $A$  \$ **qui ont tous des nombres aléatoires distincts.**

## E-cash [Chaum, 1982]

- ① Alice prépare  $n$  billets anonymes de la forme

$$\langle A, X, I_1, \dots, I_n \rangle$$

tels que  $A$  est le montant en \$,  $X$  est un grand nombre aléatoire et chaque  $I_j = (I_{jL}, I_{jR})$  est une information d'identité qui est partagée en 2 (en utilisant un Shamir scheme de partage de secret). De plus Alice commit sur chacune des parties.

- ② Alice blinds les  $n$  billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 envelopes et confirme que ce sont bien tous des billets de  $A$  \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte  $A$  \$ du compte d'Alice.

## E-cash [Chaum, 1982]

- ① Alice prépare  $n$  billets anonymes de la forme

$$\langle A, X, I_1, \dots, I_n \rangle$$

tels que  $A$  est le montant en \$,  $X$  est un grand nombre aléatoire et chaque  $I_j = (I_{jL}, I_{jR})$  est une information d'identité qui est partagée en 2 (en utilisant un Shamir scheme de partage de secret). De plus Alice commit sur chacune des parties.

- ② Alice blinds les  $n$  billets et les envoie à la banque.
- ③ La banque ouvre (en demandant à Alice) 99 envelopes et confirme que ce sont bien tous des billets de  $A$  \$ **qui ont tous des nombres aléatoires distincts.**
- ④ La banque signe en aveugle la dernière enveloppe et la renvoie à Alice. La banque décompte  $A$  \$ du compte d'Alice.

# Anonymous E-cash [Chaum, 1982]

- 5 Alice ouvre l'enveloppe et va la dépenser chez le marchant.

# Anonymous E-cash [Chaum, 1982]

- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok, et demande à Alice de révéler soit le côté gauche soit le côté droit de chaque  $I_j$ 's (c'est un vecteur de  $n$  bits) et transfert le tout à la banque.

# Anonymous E-cash [Chaum, 1982]

- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok, et demande à Alice de révéler soit le côté gauche soit le côté droit de chaque  $I_j$ 's (c'est un vecteur de  $n$  bits) et transfert le tout à la banque.
- ⑦ La banque vérifie la signature et vérifie que le nombre aléatoire n'a jamais été utilisé, crédite 1000 \$ sur le compte du marchant, enregistre le nombre aléatoire et toutes les informations d'identité.

# Anonymous E-cash [Chaum, 1982]

- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok, et demande à Alice de révéler soit le côté gauche soit le côté droit de chaque  $I_j$ 's (c'est un vecteur de  $n$  bits) et transfert le tout à la banque.
- ⑦ La banque vérifie la signature et vérifie que le nombre aléatoire n'a jamais été utilisé, crédite 1000 \$ sur le compte du marchant, enregistre le nombre aléatoire et toutes les informations d'identité.
- ⑧ Si le nombre d'unicité est déjà dans la base, la banque peut comparer les informations d'identité associées  
⇒ Si c'est le même, alors le Marchand a triché

# Anonymous E-cash [Chaum, 1982]

- ⑤ Alice ouvre l'enveloppe et va la dépenser chez le marchant.
- ⑥ Le marchant prend l'argent, vérifie la signature de la banque et si c'est ok, et demande à Alice de révéler soit le côté gauche soit le côté droit de chaque  $I_j$ 's (c'est un vecteur de  $n$  bits) et transfert le tout à la banque.
- ⑦ La banque vérifie la signature et vérifie que le nombre aléatoire n'a jamais été utilisé, crédite 1000 \$ sur le compte du marchant, enregistre le nombre aléatoire et toutes les informations d'identité.
- ⑧ Si le nombre d'unicité est déjà dans la base, la banque peut comparer les informations d'identité associées
  - ⇒ Si c'est le même, alors le Marchand a triché
  - ⇒ Si ce n'est pas le même, alors le deuxième Marchand a sûrement sélectionné un vecteur de  $n$  bits différent. La banque a donc les parties gauches et droites d'un certain  $I_i$  et peut retrouver l'identité d'Alice.

# Plan

- 1 Introduction
- 2 Intégrité des données, Fonction de hash sécurisées
- 3 Prouver son Identité
- 4 Communications anonymes
- 5 Electronic Cash
- 6 Preuves Zero-Knowledge et application

# Introduction

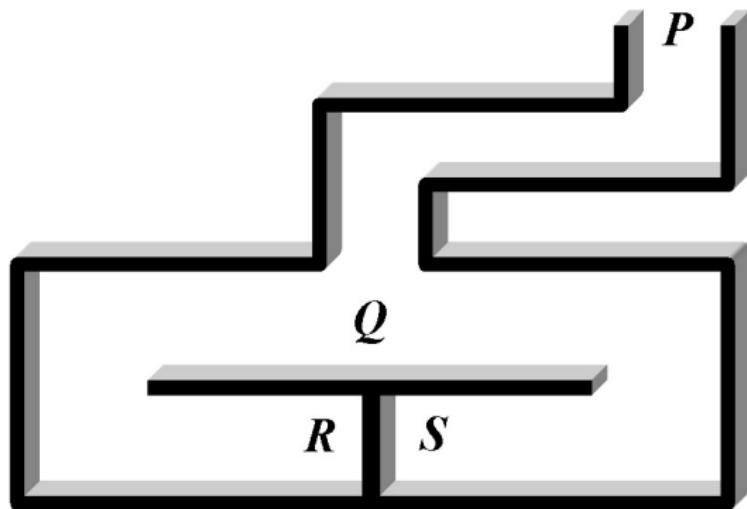
- Concept très important en crypto pour prouver l'absence de fuite d'information.
- Au cœur de l'idée de confidentialité.
- Souvent mal compris.
- A l'origine se trouvent les systèmes de preuves interactives: un preveur essaye de convaincre un vérificateur qu'il sait qu'une certaine formule logique est vraie au delà de tout doute raisonnable au travers d'une discussion.
- Preuves interactives, deux approches:
  - Soundness (problème original): le preuveur essaye de berner le vérificateur.

# Introduction

- Concept très important en crypto pour prouver l'absence de fuite d'information.
- Au cœur de l'idée de confidentialité.
- Souvent mal compris.
- A l'origine se trouvent les systèmes de preuves interactives: un preveur essaye de convaincre un vérificateur qu'il sait qu'une certaine formule logique est vraie au delà de tout doute raisonnable au travers d'une discussion.
- Preuves interactives, deux approches:
  - Soundness (problème original): le preuveur essaye de berner le vérificateur.
  - Et si vous ne faites pas confiance au vérificateur?  
    ⇒ fuite d'information, eg unix passwords...
- On veut une preuve interactive qui convainc un vérificateur de l'assertion mais qui n'apporte aucune information au vérificateur.

# ZKP pour les enfants [Quisquater et al., 1989]

- Comment prouver que vous connaissez un secret sans avoir à le révéler ?
- Image d'Alibaba et la caverne magique.



# ZKP: Analyse

- Basé sur une approche "cut and choose".

# ZKP: Analyse

- Basé sur une approche "cut and choose".
- On peut ajuster la fiabilité du ZKP par

# ZKP: Analyse

- Basé sur une approche "cut and choose".
- On peut ajuster la fiabilité du ZKP par
  - Caverne plus compliquée (différents tunnels).
  - More repetitions of the challenge.
- Les interactions peuvent être faites en parallèle pour accélérer le processus.

# ZKP: Analyse

- Basé sur une approche "cut and choose".
- On peut ajuster la fiabilité du ZKP par
  - Caverne plus compliquée (différents tunnels).
  - More repetitions of the challenge.
- Les interactions peuvent être faites en parallèle pour accélérer le processus.
- La preuve de connaissance ne peut pas être transmise (facile de tricher en sélection les tests positifs).

# Preuves d'identités

- Envoyer login plus password expose à des attaques par rejeu.

# Preuves d'identités

- Envoyer login plus password expose à des attaques par rejet.
- Une manière d'éviter ce problème est d'utiliser un schéma avec randomisation et challenge/response.

# Preuves d'identités

- Envoyer login plus password expose à des attaques par rejet.
- Une manière d'éviter ce problème est d'utiliser un schéma avec randomisation et challenge/response.
- Il reste que le serveur sait qui s'est connecté.

# Preuves d'identités

- Envoyer login plus password expose à des attaques par rejet.
- Une manière d'éviter ce problème est d'utiliser un schéma avec randomisation et challenge/response.
- Il reste que le serveur sait qui s'est connecté.
- Une des caractéristiques des ZKP est le fait qu'elles sont "non-transmissible".
- ZKP permet de simuler des clefs physiques de ce point de vue: vous pouvez montrer que vous avez les accréditations sans avoir à les montrer. Personne ne peut savoir si une clef physique a été utilisée ou pas.

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.
- Trouver un circuit Hamiltonien est NP-complet (dur).

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.
- Trouver un circuit Hamiltonien est NP-complet (dur).
- La ZKP interactive associée est :
  - Alice produit un graphe isomorphe à  $G$  (permutation des sommets) :  $H$

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.
- Trouver un circuit Hamiltonien est NP-complet (dur).
- La ZKP interactive associée est :
  - Alice produit un graphe isomorphe à  $G$  (permutation des sommets) :  $H$
  - Alice commit  $H$  en utilisant un commitment scheme.

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.
- Trouver un circuit Hamiltonien est NP-complet (dur).
- La ZKP interactive associée est :
  - Alice produit un graphe isomorphe à  $G$  (permutation des sommets) :  $H$ .
  - Alice commit  $H$  en utilisant un commitment scheme.
  - Bob choisit une question parmi:
    - ➊ Prouve qu'il y a un isomorphisme entre  $G$  et  $H$ .
    - ➋ Donne le chemin hamiltonien dans  $H$ .

# Preuves d'identité basées sur l'isomorphisme de graphes

- Alice veut prouver son identité en montrant qu'elle connaît un chemin hamiltonien dans un grand graphe  $G$  (facile à construire) sans révéler le chemin.
- Trouver un circuit Hamiltonien est NP-complet (dur).
- La ZKP interactive associée est :
  - Alice produit un graphe isomorphe à  $G$  (permutation des sommets) :  $H$ .
  - Alice commit  $H$  en utilisant un commitment scheme.
  - Bob choisit une question parmi:
    - ① Prouve qu'il y a un isomorphisme entre  $G$  et  $H$ .
    - ② Donne le chemin hamiltonien dans  $H$ .
  - Alice répond en révélant son commitment  $H$  puis:
    - ① soit donne l'isomorphisme (la permutation utilisée).
    - ② soit donne l'image par la permutation des noeuds qui forment le circuit Hamiltonien.

# Bibliography I

 Abadi, M., Lampson, B. W., and Lévy, J. (1996).  
Analysis and caching of dependencies.

In *Proceedings of the 1996 ACM SIGPLAN International Conference on Functional Programming (ICFP '96), Philadelphia, Pennsylvania, May 24-26, 1996.*, pages 83–91.

 Bellare, M., Pointcheval, D., and Rogaway, P. (2000).  
Authenticated key exchange secure against dictionary attacks.

In *Advances in Cryptology - EUROCRYPT 2000, International Conference on the Theory and Application of Cryptographic Techniques, Bruges, Belgium, May 14-18, 2000, Proceeding*, pages 139–155.

## Bibliography II

-  Bellare, M. and Rogaway, P. (1995).  
Provably secure session key distribution: the three party case.  
In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 57–66.
-  Berardi, S. (1996).  
Pruning simply typed lambda-terms.  
*J. Log. Comput.*, 6(5):663–681.
-  Chaum, D. (1981).  
Untraceable electronic mail, return addresses, and digital pseudonyms.  
*Commun. ACM*, 24(2):84–88.

# Bibliography III



Chaum, D. (1982).

Blind signatures for untraceable payments.

In Chaum, D., Rivest, R. L., and Sherman, A. T., editors, *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982.*, pages 199–203.



Chaum, D. (1988).

The dining cryptographers problem: Unconditional sender and recipient untraceability.

*J. Cryptology*, 1(1):65–75.



Feige, U., Fiat, A., and Shamir, A. (1988).

Zero-knowledge proofs of identity.

*J. Cryptology*, 1(2):77–94.

# Bibliography IV



Golle, P. and Juels, A. (2004).

Dining cryptographers revisited.

In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 456–473.



Panti, M., Spalazzi, L., and Tacconi, S. (2002).

Attacks on cryptographic protocols: A survey.

Technical report, Instituto di Infomatrica, University of Ancona.

# Bibliography V



Quisquater, J., Quisquater, M., Quisquater, M., Quisquater, M., Guillou, L. C., Guillou, M. A., Guillou, G., Guillou, A., Guillou, G., Guillou, S., and Berson, T. A. (1989).

How to explain zero-knowledge protocols to your children.

In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 628–631.



Shamir, A. (1979).

How to share a secret.

*Commun. ACM*, 22(11):612–613.

# Bibliography VI



Volpano, D. M. and Smith, G. (1997).

A type-based approach to program security.

In *TAPSOFT'97: Theory and Practice of Software Development, 7th International Joint Conference CAAP/FASE, Lille, France, April 14-18, 1997, Proceedings*, pages 607–621.