Cryptographie à clef publique
Cours 6

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

31/03/2022

Aujourd'hui

Plan

Plan

RSA-FDH : *full domain hash*

Signature RSA : KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où q et q sont deux grand nombres premiers aléatoires
2. Choisir e et d tels que $ed \equiv 1 \mod \phi(n)$
3. La clé publique est $\texttt{pk} = (e, n)$, la clé privée est $\texttt{sk} = (d, \phi(n))$.

L'espace des messages est $\mathcal{M} = \{0,1\}^*$ et celui des signatures est $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

Signature RSA-FDH : Sign$(\boldsymbol{m}, \texttt{sk})$

On suppose que $\boldsymbol{m} \in \{0,1\}^*$ et que $H : \{0,1\}^* \to \mathbb{Z}/n\mathbb{Z}$ est une fonction de hachage.

1. Hacher $\boldsymbol{m}$, c'est-à-dire calculer $h := H(\boldsymbol{m})$
2. Calculer et retourner $s = h^d \mod n$.

Signature RSA-FDH : Verif$(\boldsymbol{m}, s, \texttt{pk})$

1. Calculer $h' = s^e \mod n$.
2. Hacher $\boldsymbol{m}$, c'est-à-dire calculer $h := H(\boldsymbol{m})$
3. Faire le test $h' = h$ et retourner le booléen associé.

Signature ElGamal

On se place dans le groupe multiplicatif d'un corps fini $\mathbb{F}_p$, où p est premier. On note g un générateur de $\mathbb{F}_p^\times$.

Signature ElGamal : KeyGen

1. Choisir aléatoirement $a \in \mathbb{F}_p^\times$.
2. Calculer $\alpha = g^a \mod p$.
3. La clé publique est $\texttt{pk} = \alpha$, la clé privée est $\texttt{sk} = a$.

L'espace des messages est $\mathcal{M} = \mathbb{F}_p^\times$ et celui des signatures est $\mathcal{S} = \mathbb{F}_p^\times \times \mathbb{Z}/(p-1)\mathbb{Z}$.

Signature ElGamal : Sign$(m, \texttt{sk})$

1. Choisir aléatoirement $k \in (\mathbb{Z}/(p-1)\mathbb{Z})^\times$.
2. Calculer $b = g^k \mod p$.
3. Calculer $c = (m - ab)k^{-1} \mod (p-1)$.
4. Retourner $s = (b, c)$.

Signature ElGamal : Verif$(m, s, \texttt{pk})$

1. Calculer $x = \alpha^b \cdot b^c \mod p$ et $y = g^m \mod p$.
2. Faire le test $x = y$ et retourner le booléen associé.

Plan

Description de DSA

DSA : *Digital Signature Algorithm*. Proposé par la NSA en 1991, selectionné par le NIST comme standard (processus non-public...).

C'est essentiellement une **variante de la signature ElGamal** :

1. avec un module d'exposant plus petit,
2. où l'on **hache** le message m qui intervient dans l'exposant.

Remarques. À propos de la signature ElGamal :

- il faut garder un groupe $\mathbb{F}_p^\times$ de taille 2048 bits pour que le problème du logarithme reste difficile,
- **mais** le nombre d'exposants "possibles" peut être plus petit : 2^{224} exposants permettent d'avoir 112 bits de sécurité sur la fonction de hachage (à cause des collisions par le paradoxe des anniversaires)

Intérêt principal : signatures plus courtes !

Description de DSA

Paramètres du système.

1. Choisir q un nombre premier de 224 bits minimum.
2. Choisir p un nombre premier de 2048 bits minimum, tel que
   - $p-1$ est divisible par q,
   - le logarithme discret dans $\mathbb{F}_p^\times$ est difficile.
3. Calculer g un générateur d'un sous-groupe cyclique $\mathbb{G}$ d'ordre q de $\mathbb{F}_p^\times$.
4. On se donne également une fonction de hachage H sur 224 bits.

Signature DSA : KeyGen

1. Choisir a aléatoirement dans $(\mathbb{Z}/q\mathbb{Z})^\times$.
2. Calculer $\alpha = g^a$.
3. La clé publique est $\texttt{pk} = \alpha$, la clé privée est $\texttt{sk} = a$.

Description de DSA

- L'espace des messages est $\mathcal{M} = \{0,1\}^*$.
- L'espace des signatures est $\mathcal{S} = (\mathbb{Z}/q\mathbb{Z})^\times \times (\mathbb{Z}/q\mathbb{Z})^\times$.

Signature DSA : $\mathsf{Sign}(\boldsymbol{m}, \mathtt{sk})$

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir k aléatoirement dans $(\mathbb{Z}/q\mathbb{Z})^\times$.
3. Calculer $b = (g^k \mod p) \mod \mathbf{q}$.
4. Calculer $c = (h + ab)k^{-1} \mod \mathbf{q}$.
5. Si $b = 0$ ou $c = 0$, revenir à l'étape 2.
6. Sinon, retourner $s = (b, c)$.

Signature DSA : $\mathsf{Verif}(\boldsymbol{m}, s, \mathtt{pk})$

1. Calculer $h = H(\boldsymbol{m})$.
2. Calculer $x = g^{hc^{-1} \bmod q}\, \alpha^{bc^{-1} \bmod q}$.
3. Faire le test $x \equiv b \mod q$ et retourner le booléen associé.

Validité, sécurité et performance de DSA

Validité. Modulo p on a :

$$x = g^{hc^{-1} \mod q} \alpha^{bc^{-1} \mod q} = g^{(h+ab)c^{-1} \mod q} = g^{k \mod q} = g^k$$

Ainsi,

$$x \equiv (g^k \mod p) \mod q \qquad (\equiv b)$$

Théorème. Sous les hypothèses suivantes :

- le problème du logarithme discret est difficile,
- on se plaace dans le modèle de l'oracle aléatoire (fonction de hachage "idéales"),
- le générateur d'aléa est parfait,

la signature DSA est EUF-CMA (= *résistante aux attaques de falsification existentielle à message choisi*).

Performances :

- **Calcul efficace :** un haché du message + $O(1)$ exponentiations dans $\mathbb{Z}/p\mathbb{Z}$ + $O(1)$ calculs dans $\mathbb{Z}/q\mathbb{Z}$.
- **Taille de clefs :** $\log_2 p = 2048$ bits pour la clé publique, $\log_2 q = 224$ bits pour la clé privée.
- **Taille de signature :** $2\log_2 q = 448$ bits.

Description de ECDSA

On peut étendre DSA aux **groupes de points de courbes elliptiques**. Le standard « ECDSA » apparaît en 2000.

C'est essentiellement le même algorithme que DSA !

Paramètres du système.

1. Choisir une courbe elliptique E sur $\mathbb{F}_p$ pour laquelle
   - l'ordre n du groupe de points est divisible par un grand nombre premier q (typiquement ≥ 224 bits),
   - le logarithme discret dans le sous-groupe d'ordre q est difficile.
2. Calculer P un générateur du sous-groupe cyclique d'ordre q.
3. On se donne également une fonction de hachage H sur 224 bits.

Signature ECDSA : KeyGen

1. Choisir a aléatoirement dans $\mathbb{Z}/q\mathbb{Z}^\times$.
2. Calculer $A = aP$.
3. La clé publique est $\texttt{pk} = A$, la clé privée est $\texttt{sk} = a$.

.

Description de ECDSA

Signature ECDSA : $\mathsf{Sign}(\boldsymbol{m}, \mathtt{sk})$

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir $k \in (\mathbb{Z}/q\mathbb{Z})^\times$ aléatoirement.
3. Calculer $M = kP$ et $b = x_M \mod q$.
4. Calculer $c = (h + ab)k^{-1} \mod q$.
5. Si $b = 0$ ou $c = 0$, revenir à l'étape 2.
6. Sinon, retourner $s = (b, c)$.

Signature ECDSA : $\mathsf{Verif}(\boldsymbol{m}, s, \mathtt{pk})$

1. Calculer $h = H(\boldsymbol{m})$.
2. Calculer le point $Q = h(c^{-1} \mod q)P \oplus (bc^{-1} \mod q)A$, ainsi que son abscisse x_Q.
3. Faire le test $x_Q \equiv b \mod q$ et retourner le booléen associé.

Sécurité essentiellement identique à DSA.

Avantage : taille de la clé publique 2048 bits $\to$ 224 bits.

Inconvénient : un peu plus lent (à modérer).

D'autres signatures

Beaucoup d'**autres signatures** existent.

- Signature de **Schnorr** (voir plus tard, après avoir introduit les schémas d'identification).
- Signatures à base de **fonctions de hachage** (ex : signature de Lamport), ne reposent pas sur des problèmes de théorie des nombres.
- Signatures **post-quantiques**, notamment fondées sur des problèmes sur les réseaux euclidiens, systèmes multivariés, codes correcteurs, isogénies, ... (voir séances sur la cryptographie post-quantique).

Plan

Plan

Motivations

Contexte. Alice et Bob font partie d'un grand réseau de communication, dans lequel chaque participant engendre une paire de clefs publique/privée (pour du chiffrement ou de la signature).

Les clefs publiques sont émises publiquement. Pour valider ces clefs, on a besoin d'une **infrastructure à clef publique** (*public-key infrastructure*, PKI).

Une solution est d'adoption d'une **autorité de certification** (*certification authority*, CA).

- C'est un organisme externe auquel tout participant fait confiance.
- Cette autorité va produire des **certificats** d'authenticité, par exemple pour les clefs publiques des utilisateurs.

Autres types de PKI :

- **Toile de confiance** (*web of trust*), utilisé dans PGP (*pretty good privacy*) : modèle complètement décentralisé où chaque entité peut certifier les clefs d'autres entités.
- Plus récemment : ***blockchains*** (ex : CertCoin).

Certificats

Un **certificat** est une structure de données reliant
- l'identité d'une personne (nom, adresse email, etc.),
- une information à certifier (clef publique)
- et la signature d'une autorité de confiance.

On peut y ajouter des informations additionnelles.

Exemple. La norme X.509 est utilisée pour les certificats dans TLS/SSL (protocoles de sécurisation). La structure de données contient (dans le désordre) :

- L'identifiant du signataire et/ou du détenteur du certificat
- Algorithme de chiffrement
- Clé publique
- La signature de l'émetteur du certificat
- La version du certificat
- Le numéro de série
- Le nom de l'autorité de certification
- La date de début et de fin de validité
- L'objet de l'utilisation du certificat
- Les extensions au certificat
- L'algorithme de signature

Protocole de création/vérification d'un certificat

L'**autorité de certification** (CA) émet publiquement un algorithme $\mathsf{Verif}_{\mathrm{CA}}$ de vérification de sa signature. Elle garde secrètement l'algorithme de signature associé $\mathsf{Sign}_{\mathrm{CA}}$.

Supposons qu'Alice veuille **faire certifier une clé publique**. Deux cas possibles :

1. C'est l'autorité de certification (CA) qui crée la paire de clefs. Puis la CA transmet la clé privée $\mathtt{sk}_A$ à Alice par un moyen sécurisé, authentifié et privé.
2. C'est Alice qui crée la paire de clefs. Puis elle transmet la clef publique $\mathtt{pk}_A$ à la CA par un moyen sécurisé. Alice fournit également une **preuve de connaissance de la clef privée**.

Dans tous les cas, la clef publique $\mathtt{pk}_A$ est ensuite **certifiée** par la CA. Si $\mathtt{ID}(\text{Alice})$ représente l'identité d'Alice :

$$\begin{cases} s = \mathsf{Sign}_{\mathrm{CA}}\left(\mathtt{ID}(\text{Alice}) \,\|\, \mathtt{pk}_A\right) \\ \mathtt{CERT}(\text{Alice}, \mathtt{pk}_A) = \left[\, \mathtt{ID}(\text{Alice}) \,\|\, \mathtt{pk}_A \,\|\, s \,\right] \end{cases}$$

Puis, lorsque Bob souhaite **vérifier la certification**, il vérifie simplement que

$$\mathsf{Verif}_{\mathrm{CA}}\left(s,\ \mathtt{ID}(\text{Alice}) \,\|\, \mathtt{pk}_A\right) = \mathtt{true}$$

Démo !

Exemple en ligne

Plan

Une première proposition de chiffrement authentifié

Objectif. Alice veut envoyer un message m chiffré et **authentifié** à Bob (= Bob est convaincu que c'est bien Alice qui a produit le message).

Notation.

- $\mathsf{Enc}_B / \mathsf{Dec}_B$ sont les algorithmes de chiffrement public / déchiffrement privé de Bob.
- $\mathsf{Sign}_A / \mathsf{Verif}_A$ sont les algorithmes de signature privé / vérification publique d'Alice.

Première idée (*sign-and-encrypt*) :

1. Alice signe $s = \mathsf{Sign}_A(m)$.
2. Alice calcule $c = \mathsf{Enc}_B(m \,\|\, s)$ et envoie c à Bob.
3. Bob déchiffre $\mathsf{Dec}_B(c) = m \,\|\, s$ puis vérifie que $\mathsf{Verif}_A(m, s) = \texttt{true}$.

Problème. Bob lui-même peut briser l'authentification du message, en **réutilisant la signature** :

1. Bob déchiffre $\mathsf{Dec}_B(c) = m \,\|\, s$, puis chiffre $m \,\|\, s$ avec l'algorithme de chiffrement Enc_C public associé à Charlie, une autre personne.
2. Charlie croit que Alice lui a envoyé un message $\mathsf{Enc}_C(m \,\|\, s)$, car après déchiffrement la signature s est celle d'Alice

Une seconde proposition

Seconde idée (*encrypt-and-sign*) :

1. Alice calcule $c = \mathsf{Enc}_B(\boldsymbol{m})$.
2. Alice signe $s = \mathsf{Sign}_A(c)$ et envoie (c, s) à Bob.
3. Bob vérifie la signature $\mathsf{Verif}_B(c, s) = \texttt{true}$ puis déchiffre $\mathsf{Dec}_B(c) = \boldsymbol{m}$.

Problème. Oscar peut procéder à une attaque par le milieu :

1. Oscar observe (c, s) et remplace s par sa signature $s' = \mathsf{Sign}_O(c)$.
2. Oscar envoie (c, s') à Bob et lui fait croire que le message $\boldsymbol{m}$ est le sien...

Remarque : dans ce cas, Oscar ne connaît pas le message $\boldsymbol{m}$.

Solution 1

À Alice et Bob sont associés des identifiants publics $\texttt{ID}_A$ et $\texttt{ID}_B$.

PROTOCOLE « SIGN-THEN-ENCRYPT »

Pour envoyer un message authentifié $\boldsymbol{m}$ à Bob, Alice :

1. calcule une signature $s = \mathsf{Sign}_A(\boldsymbol{m} \,\|\, \texttt{ID}_B)$
2. calcule un chiffré $c = \mathsf{Enc}_B(\boldsymbol{m} \,\|\, s \,\|\, \texttt{ID}_A)$

Pour déchiffrer et authentifier le message reçu c, Bob :

1. déchiffre $\mathsf{Dec}_B(c) = \boldsymbol{m} \,\|\, s \,\|\, \texttt{ID}_A$
2. reconnaît l'identité d'Alice dans le message déchiffré
3. vérifie la signature associée est correcte $\mathsf{Verif}_A(\boldsymbol{m} \,\|\, \texttt{ID}_B,\ s) = \texttt{true}$.

La première attaque ne fonctionne plus, car Alice a lié sa signature s à l'identité du destinataire, Bob.

Solution 2

À Alice et Bob sont associés des identifiants publics $\texttt{ID}_A$ et $\texttt{ID}_B$.

PROTOCOLE « ENCRYPT-THEN-SIGN »

Pour envoyer un message authentifié $\boldsymbol{m}$ à Bob, Alice :

1. calcule un chiffré $c = \mathsf{Enc}_B(\boldsymbol{m} \,\|\, \texttt{ID}_A)$,
2. signe $s = \mathsf{Sign}_A(c \,\|\, \texttt{ID}_B)$ et envoie $(c, s, \texttt{ID}_A)$ à Bob.

Pour déchiffrer et authentifier le message reçu, Bob :

1. vérifie que $\mathsf{Verif}_A(c \,\|\, \texttt{ID}_B, s)$,
2. déchiffre c et obtient $\boldsymbol{m}$ et l'identité $\texttt{ID}_A$,
3. vérifie que l'identité correspond à ce qu'il a reçu précédemment.

La seconde attaque ne fonctionne plus, car Oscar ne peut pas intégrer son identité personnelle ID_O au chiffré c afin de prétendre avoir chiffré le message $\boldsymbol{m}$.

Plan

Plan

Schémas d'identification : motivation

Objectif d'un schéma d'identification : prouver à l'autre son identité.

Motivations :

- essentiellement pour du **droit d'accès** : à un réseau, à un site web, à des ressources, ou même à un lieu physique (passeport)
- brique de base pour d'autres protocoles (ex : signature)

Essentiellement 3 manières de procéder à une **vérification d'identité**.

1. Par ce que l'on **est**.
   $\rightarrow$ Biométrie : empreinte digitale, reconnaissance faciale, etc.
2. Par ce que l'on **a**.
   $\rightarrow$ Documents d'identité, clés, etc.
3. Par ce que l'on **sait**.
   $\rightarrow$ Mots de passe.

Schémas d'identification : définitions

Définition. Un **schéma d'identification** implique un **prouveur** $\mathcal{P}$ et un **vérifieur** $\mathcal{V}$. Il est constitué d'un algorithme de génération de clefs et d'un protocole de vérification.

- Lors de la **génération de clefs** KeyGen, le prouveur engendre une paire de clefs publique/privée et émet la clé publique.
- Lors du **protocole de vérification** noté $[\mathcal{P} \leftrightarrow \mathcal{V}]$, le vérifieur retourne un booléen $b \in \{\mathsf{true}, \mathsf{false}\}$ suivant s'il est convaincu ou non de l'identité du prouveur

La propriété de **validité** (ou **correction**) est la suivante :

si le prouveur est **honnête** (*i.e.* il possède la clé privée et suit le protocole), alors le vérfieur retourne $b = \mathsf{true}$ avec probabilité 1.

Sauf si le contraire est indiqué : dans ces slides, Alice = prouveur et Bob = vérifieur.

Schémas d'identification : modèle de sécurité

Pour la **sécurité**, le but d'une attaque est l'**imposture**, *i.e.* se faire passer pour le prouveur $\mathcal{P}$ auprès d'un vérifieur.

Il y a différents modèles d'attaques, mais toutes se caractérisent par deux étapes.

1. **Étape d'apprentissage.** L'attaquant observe des échanges entre un prouveur $\mathcal{P}$ et un vérifieur $\mathcal{V}$ lors de plusieurs itérations du protocole de vérification $[\mathcal{P} \leftrightarrow \mathcal{V}]$.
2. **Étape d'imposture.** L'attaquant produit une simulation de prouveur $\widetilde{\mathcal{P}}$. L'attaque est réussie si, lors d'une exécution de $[\widetilde{\mathcal{P}} \leftrightarrow \mathcal{V}']$ avec un vérifieur $\mathcal{V}'$, la probabilité que le vérifieur $\mathcal{V}'$ soit convaincu est non-négligeable.

Selon l'activité de l'attaquant dans l'**étape d'apprentissage**, on différencie deux moyens d'attaque :

- les attaques **passives** sont celles où l'attaquant ne fait qu'observer les échanges entre le prouveur et un vérifieur externe,
- les attaques **actives** sont celles où l'attaquant est également autorisé à jouer le rôle d'un vérifieur (il va donc « conduire » les exécutions du protocole de vérification).

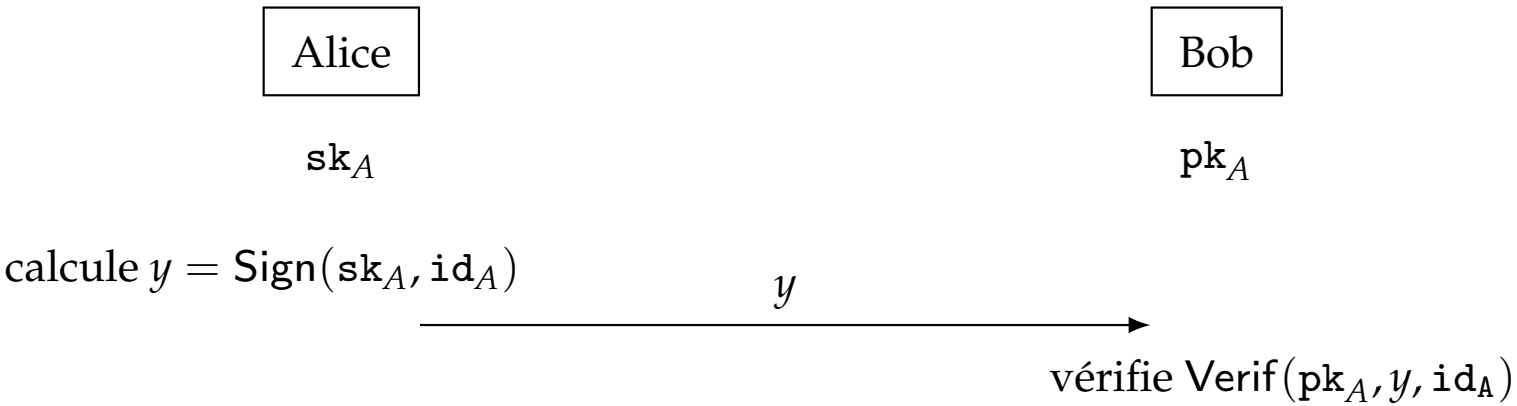
Une tentative de construction

Théorème (informel). Il est possible de construire un schéma d'identification sûr contre des adversaires actifs, **à partir** d'une signature numérique EUF-CMA.

Essayons !

Mise en place. On suppose que Alice est munie d'une paire de clefs $(\mathtt{pk}_A, \mathtt{sk}_A)$ certifiée, pour un schéma de signature numérique quelconque.

Une première idée :



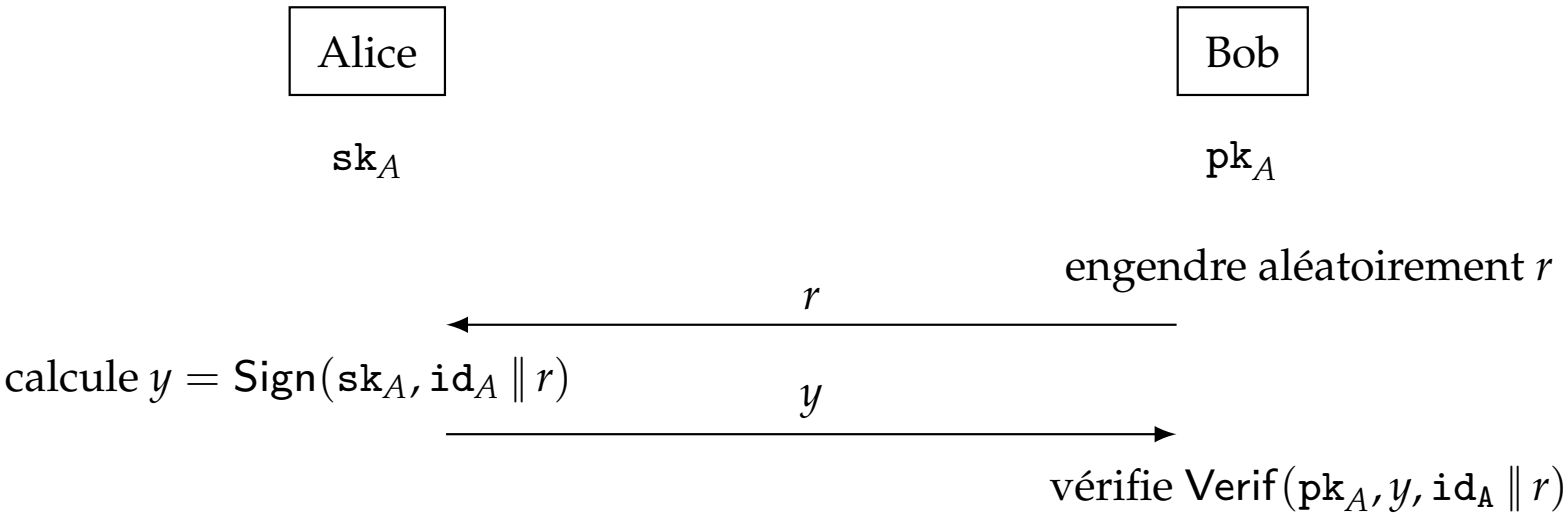
Problème. Attaque par rejeu.

→ Oscar, qui observe le canal de transmission, peut réutiliser la signature y afin d'être identifié comme Alice.

Identification à partir d'une signature

Mise en place. On suppose que Alice est munie d'une paire de clefs $(\mathtt{pk}_A, \mathtt{sk}_A)$ certifiée, pour un schéma de signature numérique quelconque.

Une seconde idée :



Théorème (informel). Il est possible de construire un schéma d'identification sûr contre des adversaires actifs, **à partir** d'une signature numérique EUF-CMA.

Identification mutuelle

Remarque. On peut également construire un protocole d'identification **mutuelle**, où Alice et Bob sont mutuellement convaincus de leurs identités réciproques.

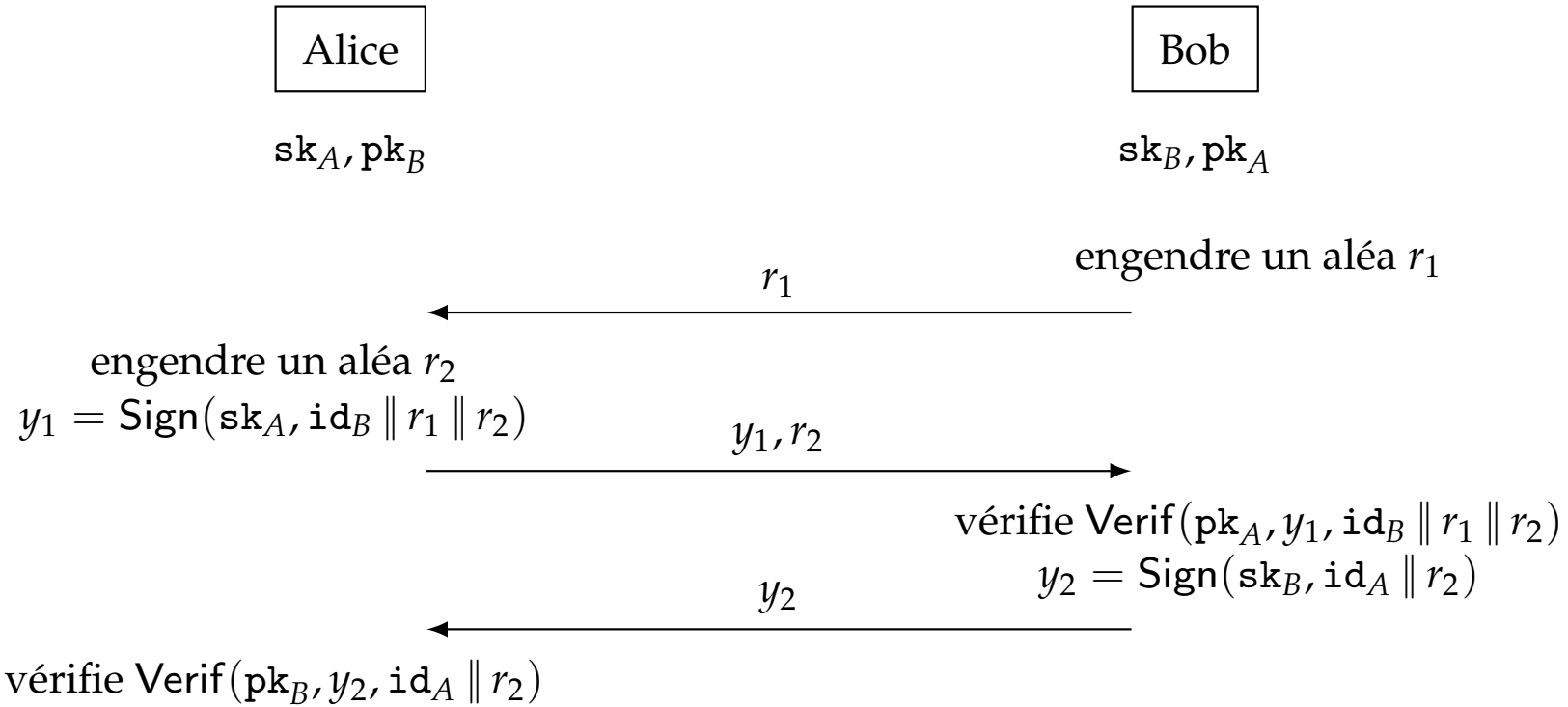


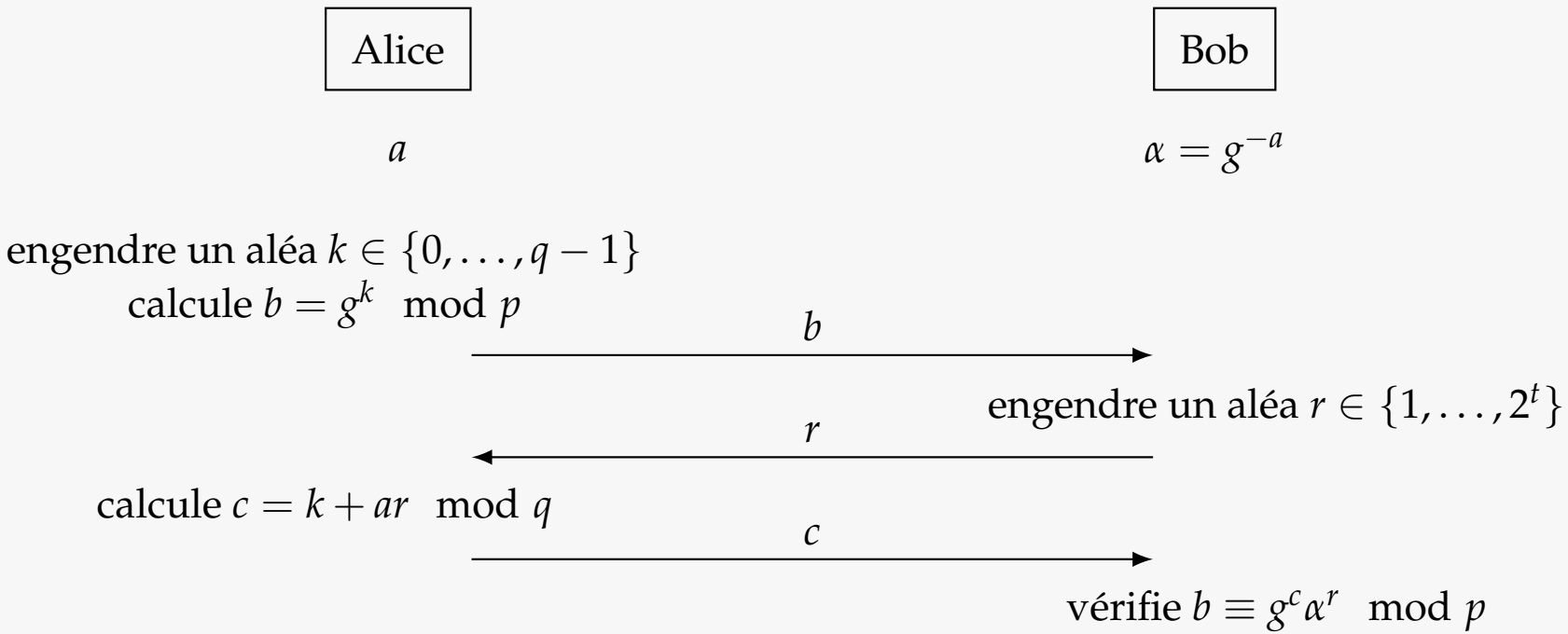
Schéma d'identification de Schnorr

But : construire un schéma d'identification sans signature préalable.

On suppose qu'Alice a émis une clé publique $\alpha = g^{-a}$ qui est **certifiée**, où

- $g \in \mathbb{F}_p$ est d'ordre premier q dans $\mathbb{F}_p^\times$
- $a \in \{0, \ldots, q-1\}$ est gardé secrètement par Alice.

SCHÉMA D'IDENTIFICATION DE SCHNORR



Sécurité du schéma d'identification de Schnorr

Validité. Si le protocole est respecté, alors on a bien

$$g^c \alpha^r \equiv g^{k+ar} g^{-ar} \equiv g^k \equiv b \mod p\,.$$

Sécurité. On peut démontrer le résultat suivant.

Théorème (informel). S'il existe un algorithme polynomial qui réalise une imposture sur le schéma d'identification de Schnorr avec probabilité $\varepsilon > 2^{-t+1}$, alors il existe un algorithme probabiliste qui résout le logarithme discret avec complexité $O(1/\varepsilon)$.

Ce type de schéma d'identification est appelé « **identification en trois passes** ».

1. Phase d'**engagement** (*commitment*) : Alice s'engage sur une valeur aléatoire (k) qu'elle transmet de manière cachée ($g^k \mod p$).
2. Phase de **défi/challenge** : Bob construit un défi aléatoire (c) auquel Alice doit répondre. La résolution de ce défi dépend de l'engagement d'Alice.
3. Phase de **réponse** : Alice répond au défi aléatoire de Bob, qui vérifie ensuite si la réponse est correcte.

Remarque. Il existe des schémas d'identification à 5 passes, comportant deux challenges et deux réponses.

Exemple-jouet pour Schnorr

Un exemple-jouet (petites valeurs) pour le schéma d'identification de Schnorr.

Paramètres. Soit $p = 88667$ et $q = 1031$. On fixe $t = 10$ (taille des défis). Un sous-groupe de $\mathbb{F}_p^\times$ d'ordre q est engendré par $g = 70322$ par exemple.

Clés. La clé privée d'Alice est $a = 755$. La clé publique correspondante est :

$$\alpha = g^{-a} = 70322^{1031-755} \equiv 13136 \mod 88667$$

Engagement. Alice choisit $k = 543$ et envoie

$$b = g^k \equiv 70322^{543} \equiv 84109 \mod 88667$$

Défi. Bob envoie le défi $r = 1000 < 2^t$.

Réponse. Alice calcule

$$c = k + ar = 543 + 755 \times 1000 \equiv 851 \mod 1031$$

Vérification. Bob vérifie que $b \equiv g^c \alpha^r \mod p$, c'est-à-dire

$$84109 \equiv 70322^{851} 13136^{1000} \mod 88667\,.$$

Plan

Transformation de Fiat–Shamir

La **transformation** (ou **heuristique**) de **Fiat–Shamir** permet de convertir un protocole de vérification interactif en une preuve non-interactive.

Si le protocole initial est un **schéma d'identification**, alors la transformation de Fiat-Shamir permet d'obtenir un **schéma de signature numérique**.

Pour des protocoles en trois passes (engagement, défi, réponse), l'idée est la suivante :

- le prouveur s'engage sur un élément x
- plutôt que de recevoir un défi aléatoire r du vérifieur, le prouveur hache x et le message $\boldsymbol{m}$
- le prouveur calcule ensuite une réponse c au défi $r = H(x, \boldsymbol{m})$, qui constitue alors la signature de $\boldsymbol{m}$.

Si l'on instancie cette idée avec le protocole d'identification de Schnorr, on obtient la signature de Schnorr.

Remarque. Dans certains cas, la transformation de Fiat-Shamir permet également de préserver une propriété de **non-divulgation** du protocole interactif.

Signature de Schnorr

Signature de Schnorr : KeyGen

1. Choisir a aléatoirement dans $\mathbb{Z}/q\mathbb{Z}^\times$.
2. Calculer $\alpha = g^a$.
3. La clé publique est $\texttt{pk} = \alpha$, la clé privée est $\texttt{sk} = a$.

Signature de Schnorr : Sign$(m, \texttt{sk})$

1. Choisir $k \in \{1, \dots, q-1\}$ aléatoirement.
2. Calculer $r = (g^k \mod p) \mod q$.
3. Calculer $b = H(r \,\|\, m)$
4. Calculer $c = k + ab \mod q$.
5. Retourner $s = (b, c)$.

Signature de Schnorr : Verif$(m, s, \texttt{pk})$

1. Calculer $r' = g^c \alpha^{-b}$
2. Calculer $b' = H(r' \,\|\, m)$
3. Faire le test $b' \equiv b \mod q$ et retourner le booléen associé.

Performances. $\simeq$ DSA : clés courtes et adaptable sur les courbes elliptiques.

Fin du cours

Questions ?