Cryptographie à clé publique
Cours 5

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

04/03/2026

Aujourd'hui

1. Schémas de signature
   - Rappels succincts
   - Signature ElGamal
   - DSA et ECDSA

2. Applications
   - Certification
   - Chiffrement à clef publique authentifié

Plan

Plan

RSA-FDH : *full domain hash*

RSA : KeyGen

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

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

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

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 $s = h^d \mod n$.

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

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.

Plan

Description de la signature ElGamal (1)

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

Description de la signature ElGamal (1)

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

On va décrire une signature « brute », sans fonction de hachage :

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

Une version sûre et efficace sera donnée via la signature (EC)DSA, un peu plus tard.

Description de la signature ElGamal (1)

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

On va décrire une signature « brute », sans fonction de hachage :

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

Une version sûre et efficace sera donnée via la signature (EC)DSA, un peu plus tard.

La génération des clés est **identique** à celle du chiffrement ElGamal :

ElGamal : KeyGen

1. Choisir aléatoirement $a \in \mathbb{Z}/(p-1)\mathbb{Z}$.
2. Calculer $\alpha = g^a \mod p$.
3. La clé publique est $\mathtt{pk} = \alpha$, la clé privée est $\mathtt{sk} = a$.

Description de la signature ElGamal (2)

Rappel : message $m \in \mathbb{F}_p^\times$.

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

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

Remarque. L'élément b est d'abord vu comme un élément du groupe $\mathbb{F}_p^\times$, puis comme un entier modulo $p-1$ (donc, un "exposant").
Aussi, observons qu'il ne dépend pas du message.

Description de la signature ElGamal (2)

Rappel : message $m \in \mathbb{F}_p^\times$.

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

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

Remarque. L'élément b est d'abord vu comme un élément du groupe $\mathbb{F}_p^\times$, puis comme un entier modulo $p-1$ (donc, un "exposant").
Aussi, observons qu'il ne dépend pas du message.

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

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.

Validité

Résumé (signature ElGamal dans $\mathbb{F}_p^\times$).

Clés : $\mathtt{pk} = \alpha = g^a$, $\mathtt{sk} = a$,

Signature : $s = (b, c)$ où

$b = g^k \mod p$

$c = (m - ab)k^{-1} \mod (p-1)$

Vérification : $\alpha^b \cdot b^c \stackrel{?}{\equiv} g^m \mod p$

Validité

Résumé (signature ElGamal dans $\mathbb{F}_p^\times$).

Clés : $\mathtt{pk} = \alpha = g^a$, $\mathtt{sk} = a$,

Signature : $s = (b, c)$ où

$b = g^k \mod p$

$c = (m - ab)k^{-1} \mod (p-1)$

Vérification : $\alpha^b \cdot b^c \stackrel{?}{\equiv} g^m \mod p$

Validité. On vérifie que

$$\alpha^b \cdot b^c = g^{ab+k(m-ab)k^{-1} \mod (p-1)} \equiv g^m \mod p$$

Validité

Résumé (signature ElGamal dans $\mathbb{F}_p^\times$).

Clés : $\mathtt{pk} = \alpha = g^a$, $\mathtt{sk} = a$,

Signature : $s = (b, c)$ où

$b = g^k \mod p$

$c = (m - ab)k^{-1} \mod (p-1)$

Vérification : $\alpha^b \cdot b^c \stackrel{?}{\equiv} g^m \mod p$

Validité. On vérifie que

$$\alpha^b \cdot b^c = g^{ab + k(m-ab)k^{-1} \bmod (p-1)} \equiv g^m \mod p$$

Remarque. On peut facilement définir une variante de cette signature dans le sous-groupe des résidus quadratiques.

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Génération de clés. Alice engendre la clé privée $a = 127$; la clé publique est donc $\alpha = g^a = 2^{127} \mod 467 \equiv 132$.

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Génération de clés. Alice engendre la clé privée $a = 127$; la clé publique est donc $\alpha = g^a = 2^{127} \mod 467 \equiv 132$.

Signature. Supposons qu'Alice veuille signer le message $m = 100$.

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Génération de clés. Alice engendre la clé privée $a = 127$; la clé publique est donc $\alpha = g^a = 2^{127} \mod 467 \equiv 132$.

Signature. Supposons qu'Alice veuille signer le message $m = 100$.

Elle choisit la valeur aléatoire $k = 213$. On vérifie bien que

$\operatorname{pgcd}(k, p-1) = \operatorname{pgcd}(213, 466) = 1$, et

$k^{-1} \mod (p-1)$ vaut alors 431.

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Génération de clés. Alice engendre la clé privée $a = 127$; la clé publique est donc $\alpha = g^a = 2^{127} \mod 467 \equiv 132$.

Signature. Supposons qu'Alice veuille signer le message $m = 100$.

Elle choisit la valeur aléatoire $k = 213$. On vérifie bien que

$\operatorname{pgcd}(k, p-1) = \operatorname{pgcd}(213, 466) = 1$, et

$k^{-1} \mod (p-1)$ vaut alors 431.

Alice calcule alors

$$b = g^k = 2^{213} \equiv 29 \mod 467$$

$$c = (m - ab)k^{-1} = (100 - 127 \times 29) \times 431 \equiv 51 \mod 466$$

Exemple

Paramètres. On prend de petites tailles pour l'exemple : $p = 467$, $g = 2$.

Génération de clés. Alice engendre la clé privée $a = 127$; la clé publique est donc $\alpha = g^a = 2^{127} \mod 467 \equiv 132$.

Signature. Supposons qu'Alice veuille signer le message $m = 100$.

Elle choisit la valeur aléatoire $k = 213$. On vérifie bien que

$\operatorname{pgcd}(k, p-1) = \operatorname{pgcd}(213, 466) = 1$, et

$k^{-1} \mod (p-1)$ vaut alors 431.

Alice calcule alors

$$b = g^k = 2^{213} \equiv 29 \mod 467$$

$$c = (m - ab)k^{-1} = (100 - 127 \times 29) \times 431 \equiv 51 \mod 466$$

Vérification. On peut alors publiquement vérifier la signature $(29, 51)$ d'Alice pour le message $m = 100$:

d'une part, $\alpha^b \cdot b^c = 132^{29} \cdot 29^{51} \equiv 189 \mod 467$,

d'autre part, $g^m = 2^{100} \equiv 189 \mod 467$.

Sécurité

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

Sécurité

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\text{Verif}(m, s, \texttt{pk}) = \texttt{true}$.

Sécurité

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\mathrm{Verif}(m, s, \mathtt{pk}) = \mathtt{true}$.

Idée : on va écrire $b = g^i \alpha^j$ pour $i, j \in \{0, \dots, p-2\}$. Cette écriture **n'est pas unique**.

Sécurité

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\text{Verif}(m, s, \texttt{pk}) = \texttt{true}$.

Idée : on va écrire $b = g^i \alpha^j$ pour $i, j \in \{0, \dots, p-2\}$. Cette écriture **n'est pas unique**.
Dans ce cas, la condition de vérification est :

$$\alpha^b (g^i \alpha^j)^c = g^m \mod p \iff \alpha^{b+jc} = g^{m-ic} \mod p$$

Sécurité

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\text{Verif}(m, s, \texttt{pk}) = \texttt{true}$.

Idée : on va écrire $b = g^i \alpha^j$ pour $i, j \in \{0, \dots, p-2\}$. Cette écriture **n'est pas unique**.
Dans ce cas, la condition de vérification est :

$$\alpha^b (g^i \alpha^j)^c = g^m \mod p \iff \alpha^{b+jc} = g^{m-ic} \mod p$$

Cette condition est vérifiée **en particulier** si on a :

$$b + jc \equiv 0 \mod (p-1) \quad \text{et} \quad m - ic \equiv 0 \mod (p-1)$$

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\text{Verif}(m, s, \texttt{pk}) = \texttt{true}$.

Idée : on va écrire $b = g^i \alpha^j$ pour $i, j \in \{0, \dots, p-2\}$. Cette écriture **n'est pas unique**.
Dans ce cas, la condition de vérification est :

$$\alpha^b (g^i \alpha^j)^c = g^m \mod p \quad \Longleftrightarrow \quad \alpha^{b+jc} = g^{m-ic} \mod p$$

Cette condition est vérifiée **en particulier** si on a :

$$b + jc \equiv 0 \mod (p-1) \quad \text{et} \quad m - ic \equiv 0 \mod (p-1)$$

L'idée est alors de construire d'abord i quelconque et j inversible modulo $p-1$ quelconque, puis de définir b, c et m en fonction :

$$\begin{cases} b &= g^i \alpha^j & \mod p \\ c &= -bj^{-1} & \mod (p-1) \\ m &= ic & \mod (p-1) \end{cases}$$

On va montrer que la signature ElGamal "brute" **n'est pas EUF-KOA**.

But. À partir de la clé publique uniquement, calculer m et $s = (b, c)$ tel que $\text{Verif}(m, s, \texttt{pk}) = \texttt{true}$.

Idée : on va écrire $b = g^i \alpha^j$ pour $i, j \in \{0, \dots, p-2\}$. Cette écriture **n'est pas unique**.
Dans ce cas, la condition de vérification est :

$$\alpha^b (g^i \alpha^j)^c = g^m \mod p \quad \Longleftrightarrow \quad \alpha^{b+jc} = g^{m-ic} \mod p$$

Cette condition est vérifiée **en particulier** si on a :

$$b + jc \equiv 0 \mod (p-1) \quad \text{et} \quad m - ic \equiv 0 \mod (p-1)$$

L'idée est alors de construire d'abord i quelconque et j inversible modulo $p-1$ quelconque, puis de définir b, c et m en fonction :

$$\begin{cases} b &= g^i \alpha^j & \mod p \\ c &= -bj^{-1} & \mod (p-1) \\ m &= ic & \mod (p-1) \end{cases}$$

Important. Par l'utilisation d'une fonction de hachage, ces menaces peuvent être levées.

Plan

Description de DSA

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

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 $\boldsymbol{m}$ qui intervient dans l'exposant.

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 $\boldsymbol{m}$ 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)

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 : obtenir des 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 $\mathbb{F}_p^\times$.

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 (par exemple).

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 (par exemple).

Signature DSA : KeyGen

1. Choisir a aléatoirement dans $\{0,\dots,q-1\}$.
2. Calculer $\alpha = g^a \mod p$.
3. La clé publique est $\mathtt{pk} = \alpha \in \mathbb{F}_p^\times$, la clé privée est $\mathtt{sk} = a \in \{0,\dots,q-1\}$.

Description de DSA

- L'espace des messages est $\mathcal{M} = \{0,1\}^*$.
- L'espace des signatures est $\mathcal{S} = \{1,\dots,q-1\}^2$.

Description de DSA

- L'espace des messages est $\mathcal{M} = \{0,1\}^*$.
- L'espace des signatures est $\mathcal{S} = \{1,\dots,q-1\}^2$.

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

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

Description de DSA

- L'espace des messages est $\mathcal{M} = \{0,1\}^*$.
- L'espace des signatures est $\mathcal{S} = \{1,\dots,q-1\}^2$.

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

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

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

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

Validité, sécurité et performance de DSA

Validité.

Validité. On a :

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

Par définition, on a donc

$$x \equiv b \mod q$$

Validité. On a :

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

Par définition, on a donc

$$x \equiv b \bmod q$$

Théorème. Sous les hypothèses suivantes :

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

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

Validité, sécurité et performance de DSA

Validité. On a :

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

Par définition, on a donc

$$x \equiv b \mod q$$

Théorème. Sous les hypothèses suivantes :
- le problème du logarithme discret est difficile,
- on se place dans le modèle de l'oracle aléatoire (fonctions 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 !

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 (par exemple).

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 (par exemple).

Signature ECDSA : KeyGen

1. Choisir a aléatoirement dans $\{0, \ldots, q-1\}$.
2. Calculer le point de la courbe $A = aP$.
3. La clé publique est $\mathtt{pk} = A$, la clé privée est $\mathtt{sk} = a$.

Description de ECDSA

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

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

Description de ECDSA

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

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir $k \in \{1, \ldots, q-1\}$ aléatoirement.
3. Calculer le point $M = kP$, puis $b = x_M \mod q$.
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 : 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.

Description de ECDSA

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

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir $k \in \{1, \ldots, q-1\}$ aléatoirement.
3. Calculer le point $M = kP$, puis $b = x_M \mod q$.
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 : 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.

Description de ECDSA

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

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir $k \in \{1, \ldots, q-1\}$ aléatoirement.
3. Calculer le point $M = kP$, puis $b = x_M \mod q$.
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 : 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 $\rightarrow$ 224 bits.

Description de ECDSA

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

1. Calculer $h = H(\boldsymbol{m})$.
2. Choisir $k \in \{1, \ldots, q-1\}$ aléatoirement.
3. Calculer le point $M = kP$, puis $b = x_M \mod q$.
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 : 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 $\rightarrow$ 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).

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).

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.

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 :

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.

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.
- SPKI et PKI décentralisées : l'idée est de *moins* dépendre d'un seul groupe d'autorités de confiance (souvent des organismes à but commercial)

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.
- SPKI et PKI décentralisées : l'idée est de *moins* dépendre d'un seul groupe d'autorités de confiance (souvent des organismes à but commercial)
- Plus récemment, et à relativiser : *blockchains* (ex : CertCoin).

Certificats

Un **certificat** est une structure de données reliant

- l' **identité** d'une personne (ex : nom, adresse email, etc.),
- une **information** à certifier (ex : une clef publique),
- et la **signature** d'une autorité de confiance.

Certificats

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

On peut y ajouter des informations additionnelles.

Certificats

Un **certificat** est une structure de données reliant
- l' **identité** d'une personne (ex : nom, adresse email, etc.),
- une **information** à certifier (ex : une 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).

Certificats

Un **certificat** est une structure de données reliant
- l' **identité** d'une personne (ex : nom, adresse email, etc.),
- une **information** à certifier (ex : une 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 Verif_{CA} de vérification de sa signature. Elle garde secrètement l'algorithme de signature associé Sign_{CA}.

Protocole de création/vérification d'un certificat

L'**autorité de certification** (CA) émet publiquement un algorithme Verif_{CA} de vérification de sa signature. Elle garde secrètement l'algorithme de signature associé Sign_{CA}.

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

Protocole de création/vérification d'un certificat

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

Protocole de création/vérification d'un certificat

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

Protocole de création/vérification d'un certificat

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

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

Protocole de création/vérification d'un certificat

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

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

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

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

Démo !

Exemple en ligne

Plan

Une première proposition de chiffrement authentifié

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

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.

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

Une première proposition de chiffrement authentifié

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

Notation.

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

Première idée (*sign-and-encrypt*) :

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

Une première proposition de chiffrement authentifié

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

Notation.

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

Première idée (*sign-and-encrypt*) :

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

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

Une première proposition de chiffrement authentifié

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

Notation.

- Enc_B/Dec_B sont les algorithmes de chiffrement public / déchiffrement privé de Bob.
- Sign_A/Verif_A sont les algorithmes de signature privé / vérification publique d'Alice.

Première idée (*sign-and-encrypt*) :

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

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

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

Une seconde proposition

Seconde idée (*encrypt-and-sign*) :

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

Une seconde proposition

Seconde idée (*encrypt-and-sign*) :

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

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

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

Une seconde proposition

Seconde idée (*encrypt-and-sign*) :

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

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

1. Oscar observe (c, s) et remplace s par sa signature $s' = \mathrm{Sign}_O(c)$.
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 »

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 $s = \text{Sign}_A(\boldsymbol{m} \| \texttt{ID}_B)$
2. calcule un chiffré $c = \text{Enc}_B(\boldsymbol{m} \| s \| \texttt{ID}_A)$

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 $s = \text{Sign}_A(\boldsymbol{m} \| \texttt{ID}_B)$
2. calcule un chiffré $c = \text{Enc}_B(\boldsymbol{m} \| s \| \texttt{ID}_A)$

Pour déchiffrer et authentifier le message reçu c, Bob :

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

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 $s = \text{Sign}_A(\boldsymbol{m} \| \texttt{ID}_B)$
2. calcule un chiffré $c = \text{Enc}_B(\boldsymbol{m} \| s \| \texttt{ID}_A)$

Pour déchiffrer et authentifier le message reçu c, Bob :

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

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 »

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

Pour déchiffrer et authentifier le message reçu, Bob :

1. vérifie que $\mathrm{Verif}_A(c \,\|\, \texttt{ID}_B, s)$,
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.

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

Pour déchiffrer et authentifier le message reçu, Bob :

1. vérifie que $\text{Verif}_A(c \,\|\, \texttt{ID}_B, s)$,
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 $\boldsymbol{m}$.

Fin du cours

Questions ?