Cryptographie à clef publique
Cours 6

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC et CSSD

18/03/2022

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 $\mathtt{pk} = (e, n)$, la clé privée est $\mathtt{sk} = (d, \phi(n))$.

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

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

On suppose que $\boldsymbol{m} \in \{0,1\}^*$ et que H est à valeurs dans $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

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

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.

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 $\mathtt{pk} = \alpha$, la clé privée est $\mathtt{sk} = a$.

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$(m, \mathtt{sk})$

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$(m, s, \mathtt{pk})$

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.

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.

- il faut garder un groupe $\mathbb{F}_p^\times$ de taille 2048 bits pour que le problème du logarithme reste difficile,
- **mais** on peut avoir un nombre d'exposants possibles 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 $\mathtt{pk} = \alpha$, la clé privée est $\mathtt{sk} = a$.

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 $b = (g^k \mod p) \mod q$.
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 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. Si le problème du logarithme discret est difficile, et sous certaines propriétés idéales pour la fonction de hachage (oracle aléatoire) et sur le générateur d'aléa, 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 $\mathtt{pk} = A$, la clé privée est $\mathtt{sk} = a$.

.

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 $Q = h(c^{-1} \bmod q)P \oplus (bc^{-1} \bmod q)A$, 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).

Plan

D'autres signatures

Beaucoup d'**autres signatures** existent.

- Signature de **Schnorr** (voir séance suivante, 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).

Signature aveugle

Une **signature aveugle** permet d'apposer une signature sur un document dont le signataire ne connaît pas le contenu. Autrement dit, auteur $\neq$ signataire.

Application : vote électronique (l'autorité organisant le suffrage va signer les votes sans les voir), porte-monnaie électronique.

Exemple de schéma de signature aveugle avec RSA-FDH. Alice veut faire signer à Bob un message $\boldsymbol{m}$ sans lui réveler cette valeur. On note (n, e) la clé publique RSA de Bob et d sa clé privée.

1. « Aveuglement » :
   - Alice engendre aléatoirement $r \in (\mathbb{Z}/n\mathbb{Z})^\times$
   - Alice calcule $m' = r^e H(\boldsymbol{m}) \mod n$ et envoie m' à Bob
2. Signature : Bob signe $s' = m'^d \mod n$ et renvoie s' à Alice
3. « Révélation » : Alice calcule et émet $s = s' r^{-1} \mod n$.

On observe que s est une signature valide de $\boldsymbol{m}$, car la vérification

$$s^e \equiv (s')^e r^{-e} \equiv (m')^{ed} r^{-e} \equiv m' r^{-e} \equiv H(\boldsymbol{m}) \mod n$$

est satisfaite.

Exercice. Construire un schéma de signature aveugle avec la signature DSA.

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 $\mathtt{ID}(\mathrm{Alice})$ représente l'identité d'Alice :

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

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

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

Plan

Une première proposition de chiffrement authentifié

Objectif. Alice veut envoyer un message $\boldsymbol{m}$ chiffré et authentifié à Bob.

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

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

1. Bob déchiffre $\mathsf{Dec}_B(c) = \boldsymbol{m} \parallel s$, puis chiffre $\boldsymbol{m} \parallel s$ avec l'algorithme de chiffrement Enc_C public associé à Charlie, une autre personne.
2. Charlie croit que Alice lui a envoyé un message $\mathsf{Enc}_C(\boldsymbol{m} \parallel s)$, 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 $\mathtt{ID}_A$ et $\mathtt{ID}_B$.

PROTOCOLE « SIGN-THEN-ENCRYPT »

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

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

Pour déchiffrer et authentifier le message reçu c, Bob :

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

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 $\mathtt{ID}_A$ et $\mathtt{ID}_B$.

PROTOCOLE « ENCRYPT-THEN-SIGN »

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

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

Pour déchiffrer et authentifier le message reçu, Bob :

1. vérifie que $\mathsf{Verif}_A(c \parallel \mathtt{ID}_B, s)$,
2. déchiffre c et obtient $\boldsymbol{m}$ et l'identité $\mathtt{ID}_A$,
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 ?