Signatures Numériques

Authentification, RSA, DSA et point d'étape

Yann Rotella

UVSQ - Université Paris-Saclay

9 avril 2026

Plan du Cours

SIGNATURES

Lettre de Recommandation

Pour Mr/Mme [...]

Je soussigné Yann Rotella, maître de Conférences à l'UVSQ, atteste du niveau et des compétences de l'étudiant.e [...].

En effet, [...]

Versailles, le 9 avril
Y. Rotella

Quelles propriétés voulons-nous ?

1. Attester l'authenticité d'un document
2. Que l'« on » puisse vérifier ceci
3. Si le document est modifié **après** la signature, « on » le remarque (intégrité)

AVEC DES MAC ?

- ✍ Rappeler ce qu'est un MAC
- ✍ Pourquoi ça ne fonctionne pas ?

On va avoir besoin de cryptographie asymétrique

SIGNATURE NUMÉRIQUE - DÉFINITION FORMELLE

DÉFINITION (SIGNATURE NUMÉRIQUE)

Un schéma de signature numérique sur $\mathcal{M}$ est la donnée de trois algorithmes $\texttt{KeyGen}(\lambda)$, $\texttt{Sign}(sk, m)$ *et* $\texttt{Verify}(vk, m, \sigma)$ *:*

- $\texttt{KeyGen}(\lambda) : \mathbb{N} \to \mathcal{S}_k \times \mathcal{V}_k$, *qui à partir d'un paramètre de sécurité* λ *produit (de manière probabiliste) un couple* (sk, vk) *où* sk *est une clef secrète et* vk *est la clef de vérification.*
- $\texttt{Sign} : \mathcal{S}_k \times \mathcal{M} \to S$ *qui, à partir d'une clef secrète et d'un message* $m \in \mathcal{M}$ *renvoie une signature* $\sigma \in S$*, l'espace des signatures.*
- $\texttt{Verify} : \mathcal{V}_k \times \mathcal{M} \times S \to \{\texttt{Vrai}, \texttt{Faux}\}$*, qui prend en entrée la clef de vérification* vk*, un élément* $x \in \mathcal{M}$ *et un élément de* $s \in S$ *et renvoie si* s *est une signature de* x *avec la clef* sk ***associée*** *à* pk*.*

Parenthèse sur le paramètre de sécurité

- On veut une sécurité **calculatoire**
- La puissance des ordinateurs augmente
- On voit (en TD) que la complexité des attaques dépend de la taille des paramètres.
- Tous les algorithmes pour des utilisateurs honnêtes (suivant les protocoles) sont polynomiaux en la taille des paramètres.
- Ce n'est pas le cas pour les attaques
- A priori, augmenter λ permet de se prémunir de futures améliorations de performance
- Permet de faire des réductions

SIGNATURE NUMÉRIQUE CORRECTE

- $\texttt{KeyGen}(\lambda) : \mathbb{N} \to \mathcal{S}_k \times \mathcal{V}_k$
- $\texttt{Sign} : \mathcal{S}_k \times \mathcal{M} \to S$
- $\texttt{Verify} : \mathcal{V}_k \times \mathcal{M} \times S \to \{\texttt{Vrai}, \texttt{Faux}\}$

On remarque que vk joue un rôle similaire à pk dans un contexte de chiffrement asymétrique.

DÉFINITION (SIGNATURE NUMÉRIQUE CORRECTE)

Un schéma de signature numérique sur $\mathcal{M}$ est ***correct*** *si pour toute paire (sk, vk) générée par* `KeyGen` *et pour tout message $m \in \mathcal{M}$,*

$$\texttt{Verify}(vk, m, \texttt{Sign}(sk, m)) = \texttt{Vrai}$$

Signature numérique sécurisée

- $\texttt{KeyGen}(\lambda) : \mathbb{N} \rightarrow \mathcal{S}_k \times \mathcal{V}_k$
- $\texttt{Sign} : \mathcal{S}_k \times \mathcal{M} \rightarrow S$
- $\texttt{Verify} : \mathcal{V}_k \times \mathcal{M} \times S \rightarrow \{\texttt{Vrai}, \texttt{Faux}\}$

Définition (Signature Numérique Sécurisée)

*Un schéma de signature numérique sur $\mathcal{M}$ est **sécurisé** si pour tout message $m \in \mathcal{M}$ et pour toute clef de vérification vk générée par* `KeyGen` *aucun individu **qui ne possède pas** sk ne peut générer $s \in S$ tel que*

$$\texttt{Verify}(vk, m, s)) = \texttt{Vrai}$$

✍ Expliquer pourquoi cette propriété n'est jamais satisfaite.

Signature numérique sécurisée calculatoirement

- $\texttt{KeyGen}(\lambda) : \mathbb{N} \to \mathcal{S}_k \times \mathcal{V}_k$
- $\texttt{Sign} : \mathcal{S}_k \times \mathcal{M} \to S$
- $\texttt{Verify} : \mathcal{V}_k \times \mathcal{M} \times S \to \{\texttt{Vrai}, \texttt{Faux}\}$

Définition (Signature numérique t-sécurisée)

*Un schéma de signature numérique sur $\mathcal{M}$ est t**-sécurisé** si pour tout message $m \in \mathcal{M}$ et pour toute clef de vérification vk générée par* `KeyGen` *aucun **algorithme tournant en temps au plus** t qui ne possède pas sk ne peut générer $s \in S$ tel que*

$$\texttt{Verify}(vk, m, s)) = \texttt{Vrai}$$

✍ Expliquer pourquoi cette propriété n'est jamais satisfaite.

Signature numérique sécurisée calculatoirement avec probabilité négligeable

- $\texttt{KeyGen}(\lambda) : \mathbb{N} \to \mathcal{S}_k \times \mathcal{V}_k$
- $\texttt{Sign} : \mathcal{S}_k \times \mathcal{M} \to S$
- $\texttt{Verify} : \mathcal{V}_k \times \mathcal{M} \times S \to \{\texttt{Vrai}, \texttt{Faux}\}$

Définition (Signature Numérique (t, ε, q)-Sécurisée)

*Un schéma de signature numérique sur $\mathcal{M}$ est (t, ε, q)**-sécurisé** si pour tout message $m \in \mathcal{M}$ et pour toute clef de vérification vk générée par* `KeyGen`*, **pour tout algorithme** $\mathcal{A}$, qui s'éxécute en temps au plus t où $\mathcal{A}(vk, (m_1, \sigma_1), (m_2, \sigma_2), \ldots, (m_q, \sigma_q)) \mapsto S$ qui ne possède pas sk ne peut générer $(m, s) \in \mathcal{M} \times S \backslash \{(m_i, \sigma_i)_{1 \leq i \leq q}\}$ tel que*

$$\texttt{Verify}(vk, m, s)) = \texttt{Vrai}$$

Signature RSA - génération de clefs

- $\texttt{KeyGen}(\lambda) \mapsto (d, N, e)$
- Il faut savoir générer de manière pseudo-aléatoire des nombres premiers.
- On ne sait faire beaucoup mieux que
  1. Tirer un nombre au hasard (avec un PRG et une graine par exemple)
  2. Tester si ce nombre est premier
  3. Et ce de manière probabiliste, plusieurs a aléatoires et regarder si ($a^{(p-1)} = 1$ si p premier)
  4. Recommencer jusqu'à en trouver un

Signature RSA

- $\texttt{KeyGen}(\lambda) \mapsto (d, N, e)$ où N de taille $c\lambda$ pour c une constante. $ed = 1 \mod \varphi(N)$.

$$\mathcal{M} = \mathbb{Z}_N, S = \mathbb{Z}_N$$

- Signer :

$$\begin{aligned} \texttt{Sign} : \mathbb{N} \times \mathbb{Z}_N &\to \mathbb{Z}_N \\ (d, m) &\mapsto \sigma = m^d \mod N \end{aligned}$$

- Vérifier :

$$\begin{aligned} \texttt{Verify} : \mathbb{N} \times \mathbb{Z}_N &\to \{\texttt{Vrai}, \texttt{Faux}\} \\ (e, s) &\mapsto (s^e == m \mod N) \end{aligned}$$

Problème

Ce schéma de signature n'est pas sécurisé.

✍ Expliquer pourquoi.

- On veut aussi signer des messages de n'importe quelle taille.

Le paradigme Hash-and-Sign

Soit $H : \{0,1\}^* \to \mathcal{C}$ une fonction de hachage cryptographique où $\mathcal{C}$ est un ensemble de chiffrés d'un chiffrement asymétrique.
Soit $\mathrm{Enc}(pk, \cdot)$ et $\mathrm{Dec}(sk, \cdot)$ les procédures de chiffrement et de déchiffrement asymétrique associées.
On définit alors le schéma de signature suivant :

- $\mathtt{Sign}(sk, m) = \sigma = \mathrm{Dec}(sk, H(m))$
- $\mathtt{Verify}(pk, x, s) = (\mathrm{Enc}(pk, H(x)) == s)$

✍ Montrer que si le chiffrement asymétrique est correct, ce schéma de signature l'est aussi.

✍ **Sécurité :** regardons ensemble ce qu'un attaquant pourrait faire.

✍ Et on peut signer des messages arbitrairement longs !

Application à RSA

- $\texttt{KeyGen}(\lambda) \mapsto (d, N, e)$ où N de taille $c\lambda$ pour c une constante. $ed = 1 \mod \varphi(N)$.

$$\mathcal{M} = \mathbb{Z}_N, S = \mathbb{Z}_N$$

$$H : \{0,1\}^* \to \mathbb{Z}_N$$

- Signer :

$$\begin{aligned} \texttt{Sign} : \mathbb{N} \times \mathbb{Z}_N &\to \mathbb{Z}_N \\ (d, m) &\mapsto \sigma = (H(m))^d \mod N \end{aligned}$$

- Vérifier :

$$\begin{aligned} \texttt{Verify} : \mathbb{N} \times \mathbb{Z}_N &\to \{\texttt{Vrai}, \texttt{Faux}\} \\ (e, s) &\mapsto (s^e == H(m) \mod N) \end{aligned}$$

DSA - DIGITAL SIGNATURE ALGORITHM

$$H : \{0,1\}^* \to \{0,1\}^{256}$$

- $\texttt{KeyGen} : \mathbb{N} \to \mathbb{N} \times \mathbb{N}^4$
  1. p et q suffisamment grands, premiers, tels que $p-1 = qz$
  2. Choisir h, avec $1 < h < p-1$ et calculer $g = h^z \mod p$

     ✍ Quel est l'ordre multiplicatif de g ?
  3. Tirer sk aléatoirement dans $\{1,\ldots,q-1\}$
  4. $vk = (p,q,g,y)$
- $\texttt{Sign} : \{0,1\}^* \times \mathbb{Z}_q \to \mathbb{Z}_q \times \mathbb{Z}_q$
  1. $r \leftarrow_\$ \{2,\ldots,q-1\}$
  2. $\sigma_1 \leftarrow g^r \mod p \mod q$
  3. $\sigma_2 \leftarrow (H(m) + \sigma_1 sk) r^{-1} \mod q$
  4. Renvoyer (σ_1, σ_2).

✍ Comment vérifier ?

✍ Est-ce que ce schéma est sécurisé ?

✍ Pouvez-vous identifier des valeurs problématiques ?

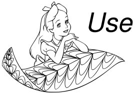
Point d'étape

- Chiffrements symétriques : rapides mais nécessitent un secret partagé.
- Chiffrements authentifiés : permettent de détecter la modification d'un message lors de la transmission.
- Fonctions de hachage cryptographiques.
- Chiffrements asymétriques : envoyer un message chiffré c à la personne qui connaît la clef secrète sk associée à une clef publique pk.
- Échanges de clefs (symétriques).
- Signatures numériques : authentifie que « la personne qui a la clef secrète sk associée à la clef vk a signé ce message m ».
- On a réussi a construire de tels algorithmes, IND-CPA ou « unforgeable » sous des hypothèses différentes (log discret, DDH, CDH, le pb RSA, fonctions de hachages).

Problème

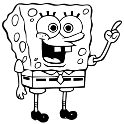
Le logarithme discret et la factorisation sont « cassés » en temps polynomial par un ordinateur quantique.

Mais à qui on parle ?



Hi, I'm Alice !
Use pk_A please

Hi, I'm Alice !
Use pk_E please



sk_A, pk_A

Nice cipher !

$c_1 = \mathcal{E}_{pk_E}(m)$

c_1

c_2

$\mathcal{E}_{sk_A}(c_2) = m$

$m = \mathcal{D}_{sk_E}(c_1)$

$c_2 = \mathcal{E}_{pk_A}(m)$