Cryptographie à clé publique
Cours 4

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

18/02/2026

Aujourd'hui

Plan

Contexte et motivations

On souhaite imiter (voire améliorer) certaines propriétés des signatures manuscrites.

Exemples de ce qu'on souhaite signer **numériquement** :

- des emails
- du code (mise à jour de logiciels)
- des transactions bancaires (ecommerce)
- de la communication publique (sites web)
- des clés de chiffrement, des certificats

Contexte et motivations

On souhaite imiter (voire améliorer) certaines propriétés des signatures manuscrites.

Exemples de ce qu'on souhaite signer **numériquement** :

- des emails
- du code (mise à jour de logiciels)
- des transactions bancaires (ecommerce)
- de la communication publique (sites web)
- des clés de chiffrement, des certificats

Objectifs :

- **Intégrité** : on peut vérifier si le message a été modifié ou non.
- **Authenticité** : on peut associer un message à un émetteur.
- **Non-répudiation** : on ne peut pas nier avoir émis une signature valide.
- **Infalsifiabilité** : une autre personne ne peut pas prétendre avoir émis la signature.
- **Non-réutilisation :** on ne peut pas utiliser une même signature sur deux messages différents.

Contexte et motivations

On souhaite imiter (voire améliorer) certaines propriétés des signatures manuscrites.

Exemples de ce qu'on souhaite signer **numériquement** :

- des emails
- du code (mise à jour de logiciels)
- des transactions bancaires (ecommerce)
- de la communication publique (sites web)
- des clés de chiffrement, des certificats

Objectifs :

- **Intégrité** : on peut vérifier si le message a été modifié ou non.
- **Authenticité** : on peut associer un message à un émetteur.
- **Non-répudiation** : on ne peut pas nier avoir émis une signature valide.
- **Infalsifiabilité** : une autre personne ne peut pas prétendre avoir émis la signature.
- **Non-réutilisation :** on ne peut pas utiliser une même signature sur deux messages différents.

Remarque. Ces propriétés ne sont pas toutes vérifiées par la signature « physique ».

Remarque. Une signature d'un message ***m*** n'est pas :

1. du chiffrement (on ne cache pas la valeur du message),
2. « l'inverse du chiffrement asymétrique » (même si parfois ça y ressemble),
3. un MAC (*message authentication code*) : les signatures sont **publiquement** vérifiables.

Contexte et motivations

Remarque. Une signature d'un message ***m*** n'est pas :

1. du chiffrement (on ne cache pas la valeur du message),
2. « l'inverse du chiffrement asymétrique » (même si parfois ça y ressemble),
3. un MAC (*message authentication code*) : les signatures sont **publiquement** vérifiables.

La signature va s'**apposer** au message, et devra dépendre explicitement de lui.

Contexte et motivations

Remarque. Une signature d'un message ***m*** n'est pas :

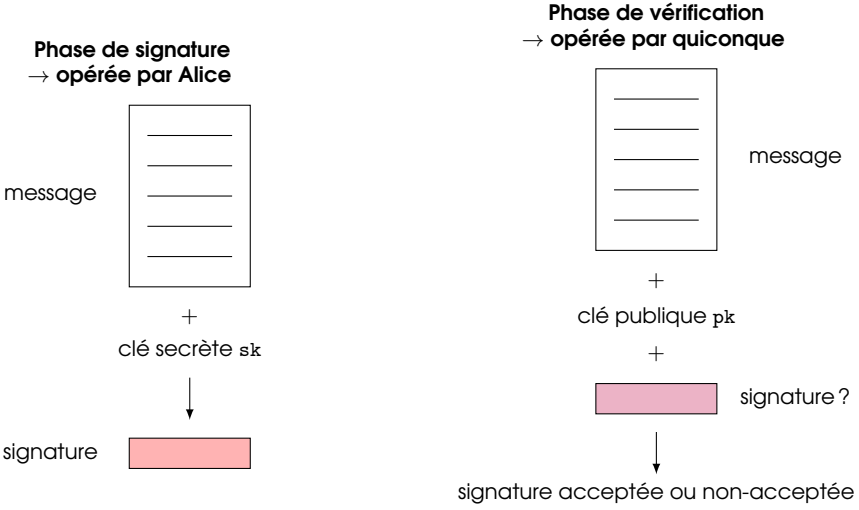
1. du chiffrement (on ne cache pas la valeur du message),
2. « l'inverse du chiffrement asymétrique » (même si parfois ça y ressemble),
3. un MAC (*message authentication code*) : les signatures sont **publiquement** vérifiables.

La signature va s'**apposer** au message, et devra dépendre explicitement de lui.

Par conséquent, les propriétés **additionnelles** désirables d'une signature sont :

- des signatures **courtes**,
- des algorithmes de signature et vérification **rapides**,
- des clés **courtes**.

Un dessin récapitulatif



Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

et de trois algorithmes :

Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

et de trois algorithmes :

1. KeyGen l'**algorithme de génération de clés**. Il renvoie un couple $(\mathtt{pk}, \mathtt{sk})$ de clé publique/clé privée.

Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

et de trois algorithmes :

1. KeyGen l'**algorithme de génération de clés**. Il renvoie un couple $(\mathtt{pk}, \mathtt{sk})$ de clé publique/clé privée.
2. Sign un **algorithme de signature**, qui prend en entrée un message $\boldsymbol{m} \in \mathcal{M}$, la clé privée $\mathtt{sk}$, et retourne une signature $\boldsymbol{s} = \mathrm{Sign}(\boldsymbol{m}, \mathtt{sk}) \in \mathcal{S}$.

Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

et de trois algorithmes :

1. KeyGen l'**algorithme de génération de clés**. Il renvoie un couple $(\mathtt{pk}, \mathtt{sk})$ de clé publique/clé privée.
2. Sign un **algorithme de signature**, qui prend en entrée un message $\boldsymbol{m} \in \mathcal{M}$, la clé privée $\mathtt{sk}$, et retourne une signature $\boldsymbol{s} = \mathrm{Sign}(\boldsymbol{m}, \mathtt{sk}) \in \mathcal{S}$.
3. Verif un **algorithme de vérification**, qui renvoie une valeur booléenne true/false. L'algorithme Verif prend en entrée la clé publique $\mathtt{pk}$, le message $\boldsymbol{m}$ et la signature $\boldsymbol{s}$.

Définition

Définition. Un schéma de **signature numérique** (à clé publique) est constitué de deux ensembles

1. $\mathcal{M}$ un **ensemble de messages**,
2. $\mathcal{S}$ un **ensemble de signatures**,

et de trois algorithmes :

1. KeyGen l'**algorithme de génération de clés**. Il renvoie un couple $(\mathtt{pk}, \mathtt{sk})$ de clé publique/clé privée.
2. Sign un **algorithme de signature**, qui prend en entrée un message $\boldsymbol{m} \in \mathcal{M}$, la clé privée $\mathtt{sk}$, et retourne une signature $\boldsymbol{s} = \mathrm{Sign}(\boldsymbol{m}, \mathtt{sk}) \in \mathcal{S}$.
3. Verif un **algorithme de vérification**, qui renvoie une valeur booléenne true/false. L'algorithme Verif prend en entrée la clé publique $\mathtt{pk}$, le message $\boldsymbol{m}$ et la signature $\boldsymbol{s}$.

Le schéma de signature est **valide** si

$$\forall (\boldsymbol{m}, \boldsymbol{s}) \in \mathcal{M} \times \mathcal{S}, \quad \mathrm{Verif}(\boldsymbol{m}, \boldsymbol{s}, \mathtt{pk}) = \mathrm{true} \iff \boldsymbol{s} = \mathrm{Sign}(\boldsymbol{m}, \mathtt{sk})$$

Modèle d'attaquant (moyens)

Pour la **sécurité** du schéma, il faut définir un modèle d'attaquant (moyens) et un modèle d'attaque (objectifs).

Modèle d'attaquant (moyens)

Pour la **sécurité** du schéma, il faut définir un modèle d'attaquant (moyens) et un modèle d'attaque (objectifs).

On distingue différents **moyens d'attaques**, notamment :

Modèle d'attaquant (moyens)

Pour la **sécurité** du schéma, il faut définir un modèle d'attaquant (moyens) et un modèle d'attaque (objectifs).

On distingue différents **moyens d'attaques**, notamment :

1. **Attaque à clé seule** (*key-only attack*, KOA) : l'attaquant ne dispose que de la clé publique

Modèle d'attaquant (moyens)

Pour la **sécurité** du schéma, il faut définir un modèle d'attaquant (moyens) et un modèle d'attaque (objectifs).

On distingue différents **moyens d'attaques**, notamment :

1. **Attaque à clé seule** (*key-only attack*, KOA) : l'attaquant ne dispose que de la clé publique
2. **Attaque à message connu** (*known-message attack*, KMA) : l'attaquant dispose d'une liste de messages déjà signés $(\boldsymbol{m}_1, \boldsymbol{s}_1), \ldots, (\boldsymbol{m}_\ell, \boldsymbol{s}_\ell)$. Les signatures sont valides et réalisées avec la même clé.

Modèle d'attaquant (moyens)

Pour la **sécurité** du schéma, il faut définir un modèle d'attaquant (moyens) et un modèle d'attaque (objectifs).

On distingue différents **moyens d'attaques**, notamment :

1. **Attaque à clé seule** (*key-only attack*, KOA) : l'attaquant ne dispose que de la clé publique
2. **Attaque à message connu** (*known-message attack*, KMA) : l'attaquant dispose d'une liste de messages déjà signés $(\boldsymbol{m}_1, \boldsymbol{s}_1), \ldots, (\boldsymbol{m}_\ell, \boldsymbol{s}_\ell)$. Les signatures sont valides et réalisées avec la même clé.
3. **Attaque à message choisi** (*chosen-message attack*, CMA) : l'attaquant choisit des messages $\boldsymbol{m}_1, \ldots, \boldsymbol{m}_\ell$ et demande les signatures $\boldsymbol{s}_1, \ldots, \boldsymbol{s}_\ell$ associées. Les signatures sont valides et réalisées avec la même clé.

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

1. **Cassage total** : l'attaquant détermine une clé privée équivalente à celle d'Alice.

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

1. **Cassage total** : l'attaquant détermine une clé privée équivalente à celle d'Alice.
2. **Falsification universelle** : avec probabilité non-négligeable, l'attaquant peut falsifier une signature d'un message choisi par quelqu'un d'autre. Ce message n'aura pas été signé précédemment.

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

1. **Cassage total** : l'attaquant détermine une clé privée équivalente à celle d'Alice.
2. **Falsification universelle** : avec probabilité non-négligeable, l'attaquant peut falsifier une signature d'un message choisi par quelqu'un d'autre. Ce message n'aura pas été signé précédemment.
3. **Falsification existentielle** : avec probabilité non-négligeable, l'attaquant peut créer un couple $(\boldsymbol{m}, \boldsymbol{s})$ où $\boldsymbol{s}$ est une signature valide de $\boldsymbol{m}$. Ce message n'aura pas été signé précédemment.

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

1. **Cassage total** : l'attaquant détermine une clé privée équivalente à celle d'Alice.
2. **Falsification universelle** : avec probabilité non-négligeable, l'attaquant peut falsifier une signature d'un message choisi par quelqu'un d'autre. Ce message n'aura pas été signé précédemment.
3. **Falsification existentielle** : avec probabilité non-négligeable, l'attaquant peut créer un couple $(\boldsymbol{m}, \boldsymbol{s})$ où $\boldsymbol{s}$ est une signature valide de $\boldsymbol{m}$. Ce message n'aura pas été signé précédemment.

Le fait qu'un schéma de signature ne permette pas d'effectuer une falsification **existentielle** sera noté **EUF** (*Existential UnForgeability*).

Modèle d'attaquant (objectifs)

Les **modèles d'attaque** principaux sont les suivants :

1. **Cassage total** : l'attaquant détermine une clé privée équivalente à celle d'Alice.
2. **Falsification universelle** : avec probabilité non-négligeable, l'attaquant peut falsifier une signature d'un message choisi par quelqu'un d'autre. Ce message n'aura pas été signé précédemment.
3. **Falsification existentielle** : avec probabilité non-négligeable, l'attaquant peut créer un couple $(\boldsymbol{m}, \boldsymbol{s})$ où $\boldsymbol{s}$ est une signature valide de $\boldsymbol{m}$. Ce message n'aura pas été signé précédemment.

Le fait qu'un schéma de signature ne permette pas d'effectuer une falsification **existentielle** sera noté **EUF** (*Existential UnForgeability*).

De même, le fait qu'un schéma de signature ne permette pas d'effectuer une falsification **universelle** sera noté **UUF** (*Universal UnForgeability*).

Sécurité d'une signature

Comme pour le chiffrement, on **combine** les définitions précédentes pour définir la sécurité d'un schéma de signature.

Par exemple :

Définition (exemple). On dit qu'un schéma satisfait la propriété d'**infalsification existentielle sous une attaque à message choisi** (EUF-CMA, *Existential UnForgeability under Chosen Message Attack*) si :

tout attaquant ayant accès à $\begin{cases} \text{la clé publique } \mathrm{pk}, \\ \text{une liste de messages } \boldsymbol{m}_1, \dots, \boldsymbol{m}_\ell \text{ qu'il a choisis}, \\ \text{et les signatures associées } \boldsymbol{s}_1, \dots, \boldsymbol{s}_\ell, \end{cases}$

a une probabilité négligeable de retourner un message $\boldsymbol{m}' \notin \{\boldsymbol{m}_i\}$ et une signature s' tels que

$$\mathrm{Verif}(\boldsymbol{m}', \boldsymbol{s}', \mathrm{pk}) = \mathrm{true}$$

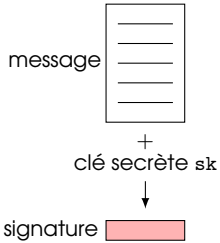
Remarque. EUF-CMA est le **standard de sécurité** usuellement requis.

Plan

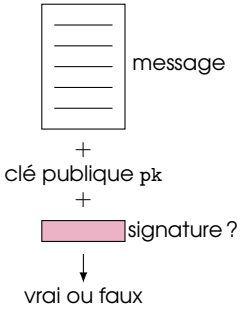
Plan

Première idée

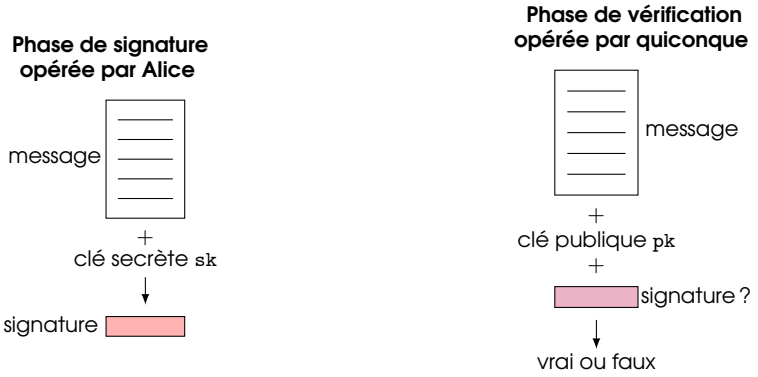
Phase de signature opérée par Alice



Phase de vérification opérée par quiconque

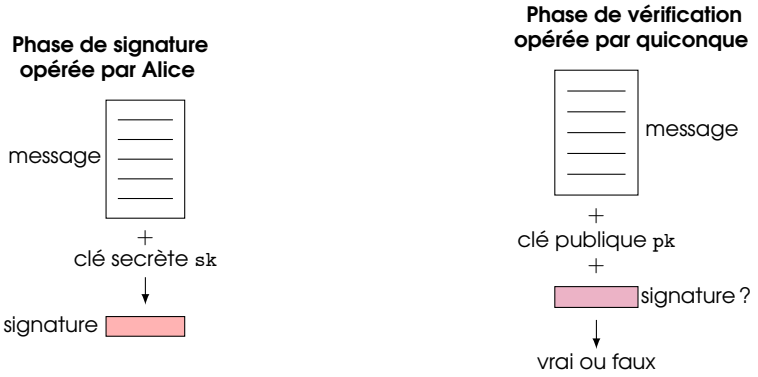


Première idée



Idée informelle : admettons que l'on ait à disposition une **fonction à sens unique** f, dont Alice connaît une trappe (par exemple, dans RSA, $f : x \mapsto x^e \mod n$). Tout le monde sait calculer f, seule Alice sait l'inverser.

Première idée



Idée informelle : admettons que l'on ait à disposition une **fonction à sens unique** f, dont Alice connaît une trappe (par exemple, dans RSA, $f : x \mapsto x^e \mod n$). Tout le monde sait calculer f, seule Alice sait l'inverser.

Alors, pour signer un message $\boldsymbol{m}$:

- Alice serait la seule à pouvoir émettre la signature $\boldsymbol{s} = f^{-1}(\boldsymbol{m})$
- tout le monde pourrait vérifier que $f(\boldsymbol{s}) = \boldsymbol{m}$.

Signature RSA

RSA : génération de clés KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où p 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 $\mathtt{sk} = d$.

Signature RSA

RSA : génération de clés KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où p 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 $\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$ et celui des signatures est $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

Signature RSA

RSA : génération de clés KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où p 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 $\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$ et celui des signatures est $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

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

1. Calculer $s = m^d \mod n$.
2. Retourner s.

Signature RSA

RSA : génération de clés KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où p 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 $\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$ et celui des signatures est $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

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

1. Calculer $s = m^d \mod n$.
2. Retourner s.

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

1. Calculer $m' = s^e \mod n$.
2. Faire le test $m' = m$ et retourner le booléen associé.

Signature RSA

RSA : génération de clés KeyGen

1. Calculer $n = pq$ et $\phi(n) = (p-1)(q-1)$, où p 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 $\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$ et celui des signatures est $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$.

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

1. Calculer $s = m^d \mod n$.
2. Retourner s.

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

1. Calculer $m' = s^e \mod n$.
2. Faire le test $m' = m$ et retourner le booléen associé.

Validité. $m' \equiv s^e \equiv m^{ed} \equiv m \mod n$.

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \stackrel{?}{\equiv} m \mod n$

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \stackrel{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \overset{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Proposition. La signature RSA « brute » **n'est pas EUF-KOA**. Autrement dit, il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \overset{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Proposition. La signature RSA « brute » **n'est pas EUF-KOA**. Autrement dit, il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

Preuve. À partir de la clé publique $\mathtt{pk} = (n, e)$, on peut forger un message m' et une signature s' valide pour la clé privée $\mathtt{sk} = d$ d'Alice.

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \overset{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Proposition. La signature RSA « brute » **n'est pas EUF-KOA**. Autrement dit, il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

Preuve. À partir de la clé publique $\mathtt{pk} = (n, e)$, on peut forger un message m' et une signature s' valide pour la clé privée $\mathtt{sk} = d$ d'Alice.

1. Choisir s' aléatoirement dans $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$,
2. Calculer $m' = (s')^e \mod n$.

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \overset{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Proposition. La signature RSA « brute » **n'est pas EUF-KOA**. Autrement dit, il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

Preuve. À partir de la clé publique $\mathtt{pk} = (n, e)$, on peut forger un message m' et une signature s' valide pour la clé privée $\mathtt{sk} = d$ d'Alice.

1. Choisir s' aléatoirement dans $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$,
2. Calculer $m' = (s')^e \mod n$.

Proposition. Il existe également une attaque de falsification **sélective** sur la signature RSA « brute », avec des messages choisis.

Résumé (signature RSA).

Clés : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$,
Signature : $s = m^d \mod n$

Vérification : $s^e \mod n \overset{?}{\equiv} m \mod n$

Que dire de la **sécurité** de ce schéma ?

Proposition. La signature RSA « brute » **n'est pas EUF-KOA**. Autrement dit, il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

Preuve. À partir de la clé publique $\mathtt{pk} = (n, e)$, on peut forger un message m' et une signature s' valide pour la clé privée $\mathtt{sk} = d$ d'Alice.

1. Choisir s' aléatoirement dans $\mathcal{S} = \mathbb{Z}/n\mathbb{Z}$,
2. Calculer $m' = (s')^e \mod n$.

Proposition. Il existe également une attaque de falsification **sélective** sur la signature RSA « brute », avec des messages choisis.

Preuve. Au prochain TD. *Indication : 2 messages préliminaires suffisent.*

Inconvénients

La signature RSA « brute » admet donc plusieurs **inconvénients :**

1. La sécurité (voir slide précédente).
2. D'un point de vue pratique, on ne peut pas signer un fichier de taille quelconque ($\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$).

Inconvénients

La signature RSA « brute » admet donc plusieurs **inconvénients :**

1. La sécurité (voir slide précédente).
2. D'un point de vue pratique, on ne peut pas signer un fichier de taille quelconque ($\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$).

Solution :

Inconvénients

La signature RSA « brute » admet donc plusieurs **inconvénients :**

1. La sécurité (voir slide précédente).
2. D'un point de vue pratique, on ne peut pas signer un fichier de taille quelconque ($\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$).

Solution : fonction de hachage !

Plan

Définition

Rappel. On note $\{0,1\}^* = \cup_{n\in\mathbb{N}} \{0,1\}^n$ l'ensemble des chaînes binaires de longueur finie (y compris 0) et $\{0,1\}^+ = \cup_{n\in\mathbb{N}^*} \{0,1\}^n$ celui des chaînes binaires de longueur ≥ 1.

Définition

Rappel. On note $\{0,1\}^* = \cup_{n\in\mathbb{N}} \{0,1\}^n$ l'ensemble des chaînes binaires de longueur finie (y compris 0) et $\{0,1\}^+ = \cup_{n\in\mathbb{N}^*} \{0,1\}^n$ celui des chaînes binaires de longueur ≥ 1.

Définition. Une **fonction de hachage cryptographique** est une application $H : \{0,1\}^* \to \{0,1\}^\ell$, où ℓ est la **taille** (ou *longueur*) de sortie.
Pour un message $\boldsymbol{m} \in \{0,1\}^*$, l'élément $H(\boldsymbol{m}) \in \{0,1\}^\ell$ est appelé le **haché** (ou *empreinte,* ou *condensat*) de $\boldsymbol{m}$.

Définition

Rappel. On note $\{0,1\}^* = \cup_{n\in\mathbb{N}} \{0,1\}^n$ l'ensemble des chaînes binaires de longueur finie (y compris 0) et $\{0,1\}^+ = \cup_{n\in\mathbb{N}^*} \{0,1\}^n$ celui des chaînes binaires de longueur ≥ 1.

Définition. Une **fonction de hachage cryptographique** est une application $H : \{0,1\}^* \to \{0,1\}^\ell$, où ℓ est la **taille** (ou *longueur*) de sortie.
Pour un message $\boldsymbol{m} \in \{0,1\}^*$, l'élément $H(\boldsymbol{m}) \in \{0,1\}^\ell$ est appelé le **haché** (ou *empreinte*, ou *condensat*) de $\boldsymbol{m}$.

L'objectif d'une fonction de hachage est de transformer un message de longueur arbitrairement grande, en une séquence de bits de taille fixe (et générale assez petite) et qui **paraît aléatoire**.

→ *Informellement, on souhaite que deux messages différents produisent deux hachés distincts et indépendants (même si les deux messages ne différent que d'un bit).*

Sécurité d'une fonction de hachage

Un peu plus formellement, une fonction de hachage doit satisfaire les propriétés suivantes.

Sécurité d'une fonction de hachage

Un peu plus formellement, une fonction de hachage doit satisfaire les propriétés suivantes.

1. **Résistance aux préimages :** étant donnée une valeur de haché, il est difficile de construire un message qui donne ce haché.

Sécurité d'une fonction de hachage

Un peu plus formellement, une fonction de hachage doit satisfaire les propriétés suivantes.

1. **Résistance aux préimages :** étant donnée une valeur de haché, il est difficile de construire un message qui donne ce haché.
2. **Résistance aux secondes préimages :** étant donné un message, il est difficile de construire un autre message qui donne le même haché que celui du premier message.

Sécurité d'une fonction de hachage

Un peu plus formellement, une fonction de hachage doit satisfaire les propriétés suivantes.

1. **Résistance aux préimages :** étant donnée une valeur de haché, il est difficile de construire un message qui donne ce haché.
2. **Résistance aux secondes préimages :** étant donné un message, il est difficile de construire un autre message qui donne le même haché que celui du premier message.
3. **Résistance aux collisions :** il soit difficile de construire deux messages distincts qui ont le même haché.

Sécurité d'une fonction de hachage

Un peu plus formellement, une fonction de hachage doit satisfaire les propriétés suivantes.

1. **Résistance aux préimages :** étant donnée une valeur de haché, il est difficile de construire un message qui donne ce haché.
2. **Résistance aux secondes préimages :** étant donné un message, il est difficile de construire un autre message qui donne le même haché que celui du premier message.
3. **Résistance aux collisions :** il soit difficile de construire deux messages distincts qui ont le même haché.

Cela se formalise mathématiquement, par exemple :

Une fonction de hachage H est **résistante aux collisions** si pour tout algorithme probabiliste $\mathcal{A}$ de complexité polynomiale, la probabilité

$$\mathbb{P}\left[\boldsymbol{m} \neq \boldsymbol{m}' \text{ et } H(\boldsymbol{m}) = H(\boldsymbol{m}') \mid (\boldsymbol{m}, \boldsymbol{m}') \leftarrow \mathcal{A}\right]$$

est négligeable devant un paramètre de sécurité donné.

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire.

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) = \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right)$$

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\begin{aligned}\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) &= \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right) \\ &= \mathbb{P}(X_2 \neq X_1) \times \mathbb{P}(X_3 \notin \{X_1, X_2\} \mid |\{X_1, X_2\}| = 2) \times \dots \\ &\quad \cdots \times \mathbb{P}(X_N \notin \{X_1, \dots, X_{N-1}\} \mid |\{X_1, \dots, X_{N-1}\}| = N - 1)\end{aligned}$$

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\begin{aligned}
\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) &= \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right) \\
&= \mathbb{P}(X_2 \neq X_1) \times \mathbb{P}(X_3 \notin \{X_1, X_2\} \mid |\{X_1, X_2\}| = 2) \times \dots \\
&\qquad \cdots \times \mathbb{P}(X_N \notin \{X_1, \dots, X_{N-1}\} \mid |\{X_1, \dots, X_{N-1}\}| = N-1) \\
&= \frac{M-1}{M} \times \frac{M-2}{M} \times \cdots \times \frac{M-(N-1)}{M}
\end{aligned}$$

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\begin{aligned}
\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) &= \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right) \\
&= \mathbb{P}(X_2 \neq X_1) \times \mathbb{P}(X_3 \notin \{X_1, X_2\} \mid |\{X_1, X_2\}| = 2) \times \dots \\
&\qquad \cdots \times \mathbb{P}(X_N \notin \{X_1, \dots, X_{N-1}\} \mid |\{X_1, \dots, X_{N-1}\}| = N-1) \\
&= \frac{M-1}{M} \times \frac{M-2}{M} \times \cdots \times \frac{M-(N-1)}{M} \\
&= \prod_{i=1}^{N-1} \left(1 - \frac{i}{M}\right) = \cdots \text{(calcul classique, exercice)} \cdots \simeq
\end{aligned}$$

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\begin{aligned}
\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) &= \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right) \\
&= \mathbb{P}(X_2 \neq X_1) \times \mathbb{P}(X_3 \notin \{X_1, X_2\} \mid |\{X_1, X_2\}| = 2) \times \dots \\
&\qquad \cdots \times \mathbb{P}(X_N \notin \{X_1, \dots, X_{N-1}\} \mid |\{X_1, \dots, X_{N-1}\}| = N-1) \\
&= \frac{M-1}{M} \times \frac{M-2}{M} \times \cdots \times \frac{M-(N-1)}{M} \\
&= \prod_{i=1}^{N-1} \left(1 - \frac{i}{M}\right) = \cdots (\text{calcul classique, exercice}) \cdots \simeq e^{-\frac{N(N-1)}{2M}}.
\end{aligned}$$

Paradoxe des anniversaires

Question. Soit $H : \{0,1\}^* \to \{0,1\}^\ell$ et $t \geq 1$. Quelle est la longueur de sortie minimale ℓ nécessaire pour éviter l'obtention de collision en temps $O(2^t)$? *(par exemple pour $t = 80$)*

Un algorithme générique pour obtenir une collision peut se déduire du **paradoxe des anniversaires**.

Problème. Étant donné N éléments tirés indépendamment **avec remise** dans un ensemble de taille M (ici $M = 2^\ell$), quelle est la probabilité pour qu'au moins l'un des éléments ait été tiré plusieurs fois ?

On calcule la probabilité complémentaire. Soit X_i la valeur du i-ème tirage. Alors,

$$\begin{aligned}
\mathbb{P}(\text{tous les } X_i \text{ sont distincts}) &= \mathbb{P}\left((X_2 \neq X_1) \text{ et } (X_3 \notin \{X_1, X_2\}) \text{ et } \dots \text{ et } (X_N \notin \{X_1, \dots, X_{N-1}\})\right) \\
&= \mathbb{P}(X_2 \neq X_1) \times \mathbb{P}(X_3 \notin \{X_1, X_2\} \mid |\{X_1, X_2\}| = 2) \times \dots \\
&\qquad \cdots \times \mathbb{P}(X_N \notin \{X_1, \dots, X_{N-1}\} \mid |\{X_1, \dots, X_{N-1}\}| = N-1) \\
&= \frac{M-1}{M} \times \frac{M-2}{M} \times \cdots \times \frac{M-(N-1)}{M} \\
&= \prod_{i=1}^{N-1} \left(1 - \frac{i}{M}\right) = \cdots \text{(calcul classique, exercice)} \cdots \simeq e^{-\frac{N(N-1)}{2M}} .
\end{aligned}$$

Application numérique. Dans une classe de 23 étudiant·es, la probabilité qu'au moins deux étudiant·es partagent le même jour d'anniversaire est $\simeq 1 - e^{-23\times 22/(2\times 365)} \simeq 0.50$

Du paradoxe à la collision

En **pratique**, pour obtenir une collision sur une fonction de hachage de longueur ℓ, on adopte la stratégie suivante :

1. Initialiser un dictionnaire vide D.
2. **Répéter** :
   - Tirer un message $\boldsymbol{m}$ aléatoirement (dans un espace assez grand pour les tirages soient distincts).
   - Calculer $\boldsymbol{h} = H(\boldsymbol{m})$.
   - Si $\boldsymbol{h}$ est une clé du dictionnaire D, alors retourner la collision obtenue : le message $\boldsymbol{m}$ et le message $D[\boldsymbol{h}]$.
   - Sinon, insérer dans le dictionnaire $D[\boldsymbol{h}] = \boldsymbol{m}$.

Du paradoxe à la collision

En **pratique**, pour obtenir une collision sur une fonction de hachage de longueur ℓ, on adopte la stratégie suivante :

1. Initialiser un dictionnaire vide D.
2. **Répéter** :
   - Tirer un message $\boldsymbol{m}$ aléatoirement (dans un espace assez grand pour les tirages soient distincts).
   - Calculer $\boldsymbol{h} = H(\boldsymbol{m})$.
   - Si $\boldsymbol{h}$ est une clé du dictionnaire D, alors retourner la collision obtenue : le message $\boldsymbol{m}$ et le message $D[\boldsymbol{h}]$.
   - Sinon, insérer dans le dictionnaire $D[\boldsymbol{h}] = \boldsymbol{m}$.

D'après le paradoxe des anniversaires, le nombre d'itérations est en $O(\sqrt{2^\ell})$.

Du paradoxe à la collision

En **pratique**, pour obtenir une collision sur une fonction de hachage de longueur ℓ, on adopte la stratégie suivante :

1. Initialiser un dictionnaire vide D.
2. **Répéter** :
   - Tirer un message $\boldsymbol{m}$ aléatoirement (dans un espace assez grand pour les tirages soient distincts).
   - Calculer $\boldsymbol{h} = H(\boldsymbol{m})$.
   - Si $\boldsymbol{h}$ est une clé du dictionnaire D, alors retourner la collision obtenue : le message $\boldsymbol{m}$ et le message $D[\boldsymbol{h}]$.
   - Sinon, insérer dans le dictionnaire $D[\boldsymbol{h}] = \boldsymbol{m}$.

D'après le paradoxe des anniversaires, le nombre d'itérations est en $O(\sqrt{2^\ell})$.

Application numérique. Pour empêcher une collision en temps 2^{80}, on doit donc choisir $\ell \geq 2 \cdot 80 = 160$.

Du paradoxe à la collision

En **pratique**, pour obtenir une collision sur une fonction de hachage de longueur ℓ, on adopte la stratégie suivante :

1. Initialiser un dictionnaire vide D.
2. **Répéter** :
   - Tirer un message $\boldsymbol{m}$ aléatoirement (dans un espace assez grand pour les tirages soient distincts).
   - Calculer $\boldsymbol{h} = H(\boldsymbol{m})$.
   - Si $\boldsymbol{h}$ est une clé du dictionnaire D, alors retourner la collision obtenue : le message $\boldsymbol{m}$ et le message $D[\boldsymbol{h}]$.
   - Sinon, insérer dans le dictionnaire $D[\boldsymbol{h}] = \boldsymbol{m}$.

D'après le paradoxe des anniversaires, le nombre d'itérations est en $O(\sqrt{2^\ell})$.

Application numérique. Pour empêcher une collision en temps 2^{80}, on doit donc choisir $\ell \geq 2 \cdot 80 = 160$.

Exemples de fonctions de hachage

Standard : les fonctions de la famille **SHA-3** (pour *secure hash algorithm*, 3ème version), dont les sorties sont de taille 224, 256, 384 ou 512 bits (au choix).

Exemples de fonctions de hachage

Standard : les fonctions de la famille **SHA-3** (pour *secure hash algorithm*, 3ème version), dont les sorties sont de taille 224, 256, 384 ou 512 bits (au choix).

Important. Les « anciennes » familles de fonctions MD5 et SHA-1 **ne sont plus considérées** comme résistantes aux collisions, mais restent utilisées pour des applications non-cryptographiques.

Exemples de fonctions de hachage

Standard : les fonctions de la famille **SHA-3** (pour *secure hash algorithm*, 3ème version), dont les sorties sont de taille 224, 256, 384 ou 512 bits (au choix).

Important. Les « anciennes » familles de fonctions MD5 et SHA-1 **ne sont plus considérées** comme résistantes aux collisions, mais restent utilisées pour des applications non-cryptographiques.

Dans toute la suite, on prend $H : \{0,1\}^* \to \{0,1\}^\ell$ une fonction de hachage résistante aux collisions.

Plan

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

La **génération de clés** est identique : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$.

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

La **génération de clés** est identique : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$.

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

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

La **génération de clés** est identique : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$.

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

Signature RSA-FDH : 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 $s = h^d \mod n$.

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

La **génération de clés** est identique : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$.

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

Signature RSA-FDH : 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 : 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.

RSA-FDH : *full domain hash*

Idée : au lieu de signer directement le message $\boldsymbol{m}$, on signe $H(\boldsymbol{m})$ avec l'algorithme de signature RSA « brut » vu précédemment.

La **génération de clés** est identique : $\mathtt{pk} = (n, e)$, $\mathtt{sk} = d$.

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

Signature RSA-FDH : 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 : 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.

Validité. $h' \equiv s^e \equiv h^{ed} \equiv h \mod n$.

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

En pratique, on peut utiliser la fonction de hachage SHA-3, de sortie 224 ou 256 bits par exemple.

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

En pratique, on peut utiliser la fonction de hachage SHA-3, de sortie 224 ou 256 bits par exemple.

Pour s'appuyer le résultat (théorique) de sécurité ci-dessus, il faut que l'**espace de définition de la fonction RSA soit le même que l'espace des hachés** : c'est la notion de « *full domain hash* ».

Sécurité

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

En pratique, on peut utiliser la fonction de hachage SHA-3, de sortie 224 ou 256 bits par exemple.

Pour s'appuyer le résultat (théorique) de sécurité ci-dessus, il faut que l'**espace de définition de la fonction RSA soit le même que l'espace des hachés** : c'est la notion de « *full domain hash* ».

Problème : pour RSA, il faut choisir un module n de 2048 bits minimum, alors que les fonctions de hachage usuelles forment des hachés de moins de 512 bits.

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

En pratique, on peut utiliser la fonction de hachage SHA-3, de sortie 224 ou 256 bits par exemple.

Pour s'appuyer le résultat (théorique) de sécurité ci-dessus, il faut que l'**espace de définition de la fonction RSA soit le même que l'espace des hachés** : c'est la notion de « *full domain hash* ».

Problème : pour RSA, il faut choisir un module n de 2048 bits minimum, alors que les fonctions de hachage usuelles forment des hachés de moins de 512 bits.

Plusieurs solutions :

- faire du remplissage aléatoire (*padding*),
- concaténer des hachés successifs du message $H^{(i)}(\boldsymbol{m})$ ou des hachés avec incrément $H^{(i)}(\boldsymbol{m} \| \text{ctr})$.

Théorème. Dans le modèle de l'oracle aléatoire, **si** le problème RSA est difficile, **alors** la signature RSA-FDH est **EUF-CMA** (résistante à toute falsification existentielle par une attaque à message choisi) .

Remarque. Le modèle de l'**oracle aléatoire** permet d'idéaliser les fonctions de hachage. C'est une hypothèse plus forte que le **modèle standard**.

En pratique, on peut utiliser la fonction de hachage SHA-3, de sortie 224 ou 256 bits par exemple.

Pour s'appuyer le résultat (théorique) de sécurité ci-dessus, il faut que l'**espace de définition de la fonction RSA soit le même que l'espace des hachés** : c'est la notion de « *full domain hash* ».

Problème : pour RSA, il faut choisir un module n de 2048 bits minimum, alors que les fonctions de hachage usuelles forment des hachés de moins de 512 bits.

Plusieurs solutions :

- faire du remplissage aléatoire (*padding*),
- concaténer des hachés successifs du message $H^{(i)}(\boldsymbol{m})$ ou des hachés avec incrément $H^{(i)}(\boldsymbol{m} \| \mathrm{ctr})$.

Exemple :

$$FDH(\boldsymbol{m}, \mathrm{IV}) = H(\boldsymbol{m} \| n \| \mathrm{IV} + 0) \,\|\, H(\boldsymbol{m} \| n \| \mathrm{IV} + 1) \,\|\, H(\boldsymbol{m} \| n \| \mathrm{IV} + 2) \,\|\, \cdots$$

où IV est un vecteur d'initialisation public apposé au message.

Performance

La signature RSA-FDH est une brique de base du standard RSA PKCS#1 v2.1.

Performance

La signature RSA-FDH est une brique de base du standard RSA PKCS#1 v2.1.

Performances :

- **Calcul efficace ?** Un haché + $O(1)$ exponentiations modulaires (les clés sont de taille indépendante de celle du fichier).
- **Taille de clés ?**

  clé publique : $2\log_2 n \simeq 4096$ bits minimum
  clé privée : $\log_2 n \simeq 2048$ bits minimum
- **Taille de signature ?** $\log_2 n = 2048$ bits minimum

Fin du cours

Questions ?