Cryptographie à clef publique
Cours 5

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC et CSSD

25/02/2021

Aujourd'hui

Plan

Le cryptosystème ElGamal dans un groupe générique

Paramètres du système : un groupe cyclique $\mathbb{G}$ d'ordre fini n, et un générateur g de $\mathbb{G}$.

ELGAMAL : GÉNÉRATION DE CLEFS

1. Choisir aléatoirement $a \in \mathbb{Z}/n\mathbb{Z}$.
2. Calculer $\alpha = g^a$
3. La **clé publique** est α, la **clé privée** est a.

ELGAMAL : CHIFFREMENT

L'espace des **clairs** est $\mathbb{G}$. Si l'on souhaite chiffrer un message $m \in \mathbb{G}$, alors

1. Tirer $k \in \mathbb{Z}/n\mathbb{Z}$ aléatoirement.
2. Calculer $b_1 = g^k$ et $b_2 = m\alpha^k$.
3. Le chiffré est (b_1, b_2).

ELGAMAL : DÉCHIFFREMENT

Pour **déchiffrer** (b_1, b_2) :

1. Avec la clé secrète, on calcule $\ell = b_1^a$.
2. On retourne b_2/ℓ.

Sécurité d'ElGamal : fuite de la résiduosité

La sécurité du schéma d'ElGamal réside sur le **problème du logarithme discret**.

Ici, on se place dans le groupe multiplicatif d'un corps fini $\mathbb{F}_q^\times = \langle g \rangle$, et on s'intéresse à la **sécurité sémantique** d'ElGamal. Pour cela, on doit estimer si l'on peut distinguer $y = g^\ell$ d'un élément aléatoire.

Pour l'exemple, **supposons que** $q = p$ **est premier et** $p \equiv 3 \mod 4$**.** Montrons d'abord qu'on peut facilement retrouver le premier bit ℓ_0 de $\ell = \log_g(y)$.

L'ensemble des résidus quadratiques est (**exercice**)

$$\mathrm{QR}_p^\times = \{g^{2k} \mid k = 0, \ldots, (p-3)/2\}.$$

Le logarithme d'un résidu quadratique est donc **pair**. Autrement dit, si $\ell = \log_g(y)$ avec $y \in \mathrm{QR}_p^\times$, alors $\ell_0 = 0$.

Enfin, on peut facilement tester si y est un résidu quadratique. Le critère d'Euler nous dit que $y \in \mathrm{QR}_p^\times \iff y^{(p-1)/2} = 1$. En général, le symbole de Legendre nous donne cette information.

Conclusion. On possède un algorithme L_0 qui calcule ℓ_0 pour n'importe quel $y = g^{\ell_0 + 2\ell'}$.

Sécurité d'ElGamal : fuite de la résiduosité

Conclusion. On possède un algorithme L_0 qui calcule ℓ_0 pour n'importe quel $y = g^{\ell_0+2\ell'}$.

Question. Peut-on généraliser à n'importe quel bit ℓ_i du logarithme ?

Toujours dans le cas $p \equiv 3 \mod 4$, on montre que si l'on connaît un algorithme L_1 qui calcule ℓ_1, alors on peut résoudre de logarithme discret.

Conséquence. Le bit ℓ_1 est aussi « difficile » à trouver que le logarithme discret.

Preuve. Soit $x = g^{\ell_0+2\ell_1+4\ell'}$. On connaît ℓ_0 et on a alors $x/g^{\ell_0} = (g^{\ell_1+2\ell'})^2$.

But : calculer exactement $g^{\ell_1+2\ell'}$ (ainsi, on pourra itérer le procédé est calculer ℓ_2, ℓ_3, etc).

On note que $(x/g^{\ell_0})^{(p+1)/4}$ vaut $\pm g^{\ell_1+2\ell'}$.

$\rightarrow$ Pour savoir si on doit multiplier par $+1$ ou -1, on utilise l'algorithme L_1 :

Si

$$L_0\big((x/g^{\ell_0})^{(p+1)/4}\big) = L_1\big(x/g^{\ell_0}\big)$$

alors on sait que $(x/g^{\ell_0})^{(p+1)/4} = g^{\ell_1+2\ell'}$. Sinon, c'est $-g^{\ell_1+2\ell'}$.

Sécurité d'ElGamal : fuite de la résiduosité

Conséquence. Pour espérer obtenir une **sécurité sémantique** pour ElGamal, il faut donc se placer dircetement dans le groupe $\mathbb{G}$ des résidus quadratiques.

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 clefs 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 ».

Contexte et motivations

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

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

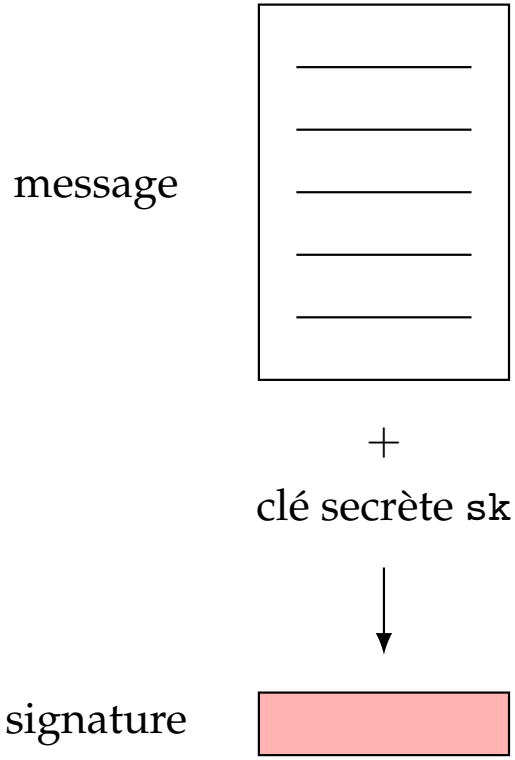
La signature va s'**apposer** au message, et devra dépendre explicitement de lui.

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

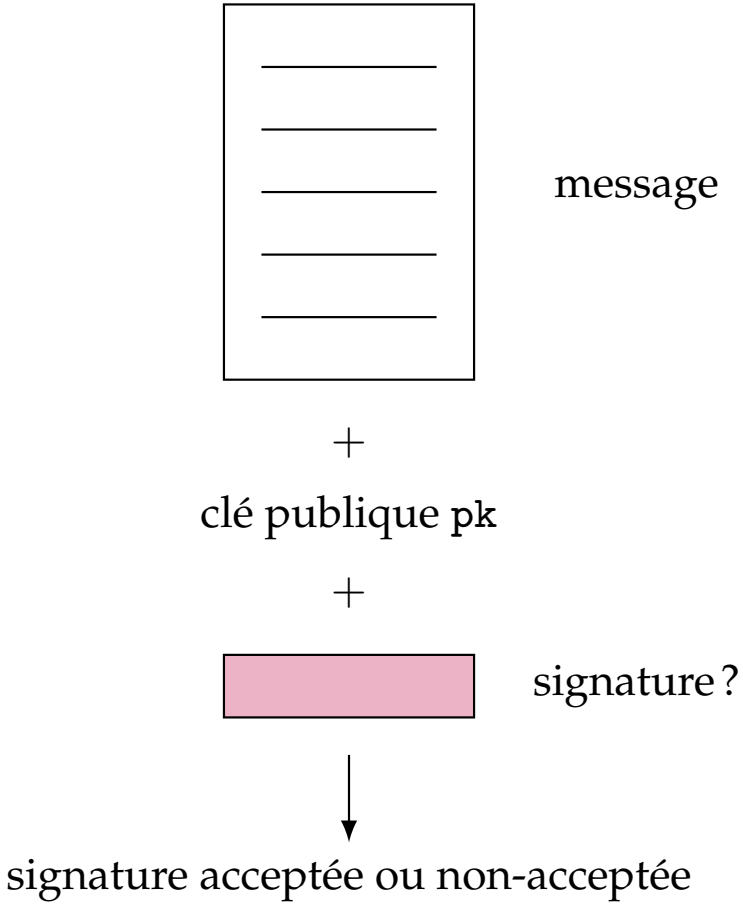
- des signatures courtes,
- des algorithmes de signature et vérification rapides,
- des clés courtes.

Un dessin récapitulatif

Phase de signature
$\rightarrow$ opérée par Alice



Phase de vérification
$\rightarrow$ opérée par quiconque



Définition

Définition. Un schéma de **signature numérique** (à clef publique) est constitué de deux ensembles et trois algorithmes :

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

Le schéma de signature est **valide** si

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

Modèle d'attaquant (moyens)

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

On distingue les moyens suivants :

1. **Attaque à clef seule** (*key-only attack*) : l'attaquant ne dispose que de la clef publique
2. **Attaque à message connu** (*known-message attack*) : l'attaquant dispose d'une liste de messages déjà signés $(m_1, s_1), \ldots, (m_\ell, s_\ell)$. Les signatures sont valides et réalisées avec la même clef.
3. **Attaque à message choisi** (*chosen-message attack*) : l'attaquant choisit des messages $m_1, \ldots, m_\ell$ et demande les signatures $s_1, \ldots, s_\ell$ associées. Les signatures sont valides et réalisées avec la même clef.

Modèle d'attaquant (objectifs)

Les modèles d'attaque 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 (m, s) où s est une signature valide de m. Ce message n'aura pas été signé précédemment.

Sécurité d'une signature

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 $\begin{cases} \text{la clé publique } \mathtt{pk}, \\ \text{une liste de messages } m_1, \ldots, m_\ell \text{ qu'il a choisi,} \\ \text{et les signatures associées } s_1, \ldots, s_\ell, \end{cases}$

a une probabilité négligeable de retourner un message $m' \notin \{m_i\}$ et une signature s' tels que $\mathsf{Verif}(m', s', \mathtt{sk}) = \mathtt{true}$.

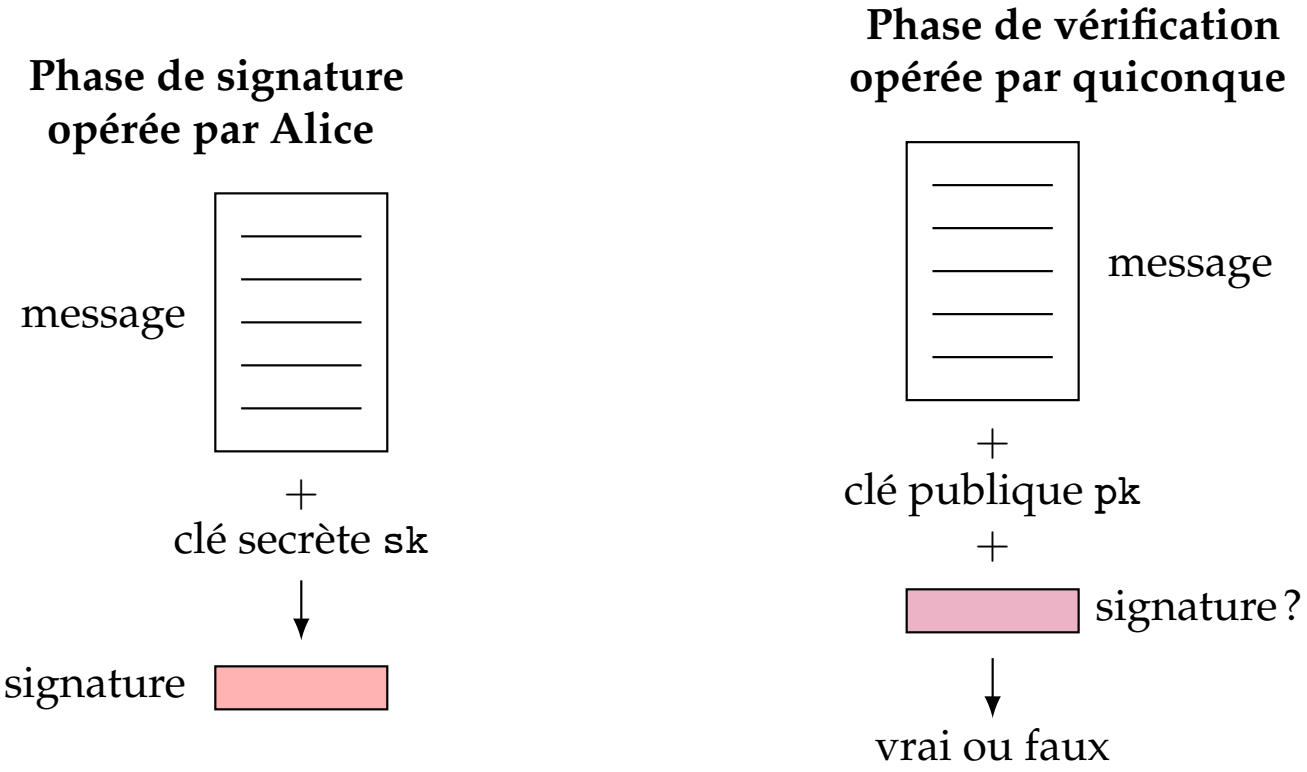
$\Longrightarrow$ EUF-CMA est le standard de sécurité usuellement requis.

Remarque. Comme l'attaquant a accès à la procédure de vérification (publique), il est impossible d'obtenir un schéma de signature à sécurité inconditionnelle.

Plan

Plan

Première idée



Idée informelle : dans le chiffrement RSA, Alice est la seule à savoir inverser la fonction à sens-unique $f : x \mapsto x^e \mod n$. Mais tout le monde sait calculer f.

Donc, pour un message m :

- la signature est $s = f^{-1}(m)$
- on vérifie publiquement que $f(s) = m$.

Signature RSA

Signature RSA : KeyGen

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

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

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

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

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

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

Sécurité

Résumé (signature RSA).

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

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

Proposition. Il existe une attaque de falsification existentielle sur la signature RSA « brute » avec la clé publique seule.

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

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

Conséquence. La signature RSA brute n'est donc pas EUF-KOA (*key-only attack*).

Proposition. Il existe une attaque de falsification universelle sur la signature RSA « brute », par message choisi. Autrement dit, RSA n'est pas UUF-CMA (UUF $\rightarrow$ *universal unforgeability*).

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

Inconvénients

Inconvénients :

1. La sécurité (voir slide précédente).
2. On ne peut pas signer un fichier de taille quelconque ($\mathcal{M} = \mathbb{Z}/n\mathbb{Z}$).

Solution : fonction de hachage !

Plan

Rappels sur les fonctions de hachage

Définition. Une famille de **fonctions de hachage cryptographiques** est un ensemble de fonctions $H_\kappa : \{0,1\}^* \to \mathcal{H}$, où $\mathcal{H}$ est un ensemble de taille fixe, et κ est un paramètre de la famille. L'élément $H_\kappa(\boldsymbol{m})$ est appelé **haché** de $\boldsymbol{m}$.

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

$$\mathbb{P}\left[\boldsymbol{h} \neq \boldsymbol{h} \text{ et } H_\kappa(\boldsymbol{m}) = H_\kappa(\boldsymbol{m}') \mid \mathcal{A}(t) = (\boldsymbol{h}, \boldsymbol{h}')\right]$$

est négligeable devant un paramètre de sécurité donné.

Exemple : les fonctions SHA-3 (*secure hash algorithm*), dont les sorties sont de taille 224, 256, 384 ou 512 bits (au choix).

Important. Les « anciennes » fonctions MD5 et SHA-1 ne sont pas résistantes aux collisions, mais restent utilisées pour des applications non-cryptographiques (à éviter néanmoins).

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

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 clef est identique : $\texttt{pk} = (n, e)$, $\texttt{sk} = d$.

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

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

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

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

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

Sécurité

Théorème. Dans le modèle de l'oracle aléatoire, si le problème RSA est difficile, alors la signature RSA-FDH est EUF-CMA.

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 que le résultat de sécurité ci-dessus soit utilisable, il faut que l'espace de définition de la fonction RSA soit le même que l'espace des hachés : « *full domain hash* ».

Problème : pour RSA, il faut choisir n de 2048 bits minimum.

Solutions : il y en a plusieurs.

- *padding*
- concaténation de hachés successifs $H^{(i)}(\boldsymbol{m})$
- concaténation de hachés avec incrément $H^{(i)}(\boldsymbol{m} \parallel \mathsf{ctr})$.

Exemple :

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

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

Performance

RSA-FDH est une brique de base pour le standard RSA PKCS#1 v2.1.

Performances :

- Calcul efficace : un haché + $O(1)$ exponentiations modulaires (les clefs sont de taille indépendante de celle du fichier).
- Taille de clefs :

  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

Plan

Description

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

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

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 $s = (b, c)$.

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

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 $\mathbb{F}_p$).

- Clés : $\texttt{pk} = \alpha = g^a$, $\texttt{sk} = a$,
- Signature : $s = (b, c)$ où
$$\begin{aligned} b &= g^k \mod p \\ c &= (m - ab)k^{-1} \mod (p-1) \end{aligned}$$
- Vérification : $\alpha^b \cdot b^c \overset{?}{\equiv} g^m \mod p$

Remarque. Le premier élément b de la signature s ne dépend pas du message.

Validité. On vérifie que

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

Exemple

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

Génération de clefs. Alice engendre la clef privée $a = 127$; la clef 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

$\mathrm{pgcd}(k, p-1) = \mathrm{pgcd}(213, 466) = 1$, 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 n'est pas EUF-KOA.

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

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

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

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 :

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

Remarque. Par l'utilisation d'une fonction de hachage, ces menaces peuvent être levées (voir cours suivant).

Fin du cours

Questions ?