

Authentification et Intégrité : Signature numérique et Hachage

Anca Nitulescu
anca.nitulescu@ens.fr

Université Paris 13 Villetaneuse

Cours 8
29/02/2016

Rappels



Modèle de sécurité

Définir la sécurité d'un cryptosystème et la prouver :

- 1 Sécurité calculatoire :
 - le niveau de sécurité qu'on veut assurer
 - la puissance de l'adversaire à considérer
- 2 L'adversaire :
 - spécifier les buts de l'attaquant
 - spécifier ses moyens : calculs, accès aux ressources
- 3 Examiner quelles sont les chances pour un attaquant d'atteindre son but avec les moyens spécifiés
 - probabilité de succès
 - "preuve" de sécurité
- 4 Conclure (pertinence du modèle, choix des paramètres, . . .)

Modèle de sécurité



Définir la sécurité : Buts – Moyens

- Quels sont les buts de l'adversaire ?
- Quels sont les moyens de l'attaquant ?

Buts

Buts de l'attaquant

- 1 **BRK – Cassage total** : retrouver la clé de déchiffrement (équivalent à déchiffrer tout message)
- 2 **OW – Sens unique** : déchiffrer un message "challenge"
- 3 **IND – Indistinguabilité** : distinguer entre deux chiffrés
 - **Sécurité sémantique** : à partir d'un chiffré, extraire d'information sur le message correspondant
- 4 **NM – Non malléabilité** : modifier un chiffré pour obtenir un autre chiffré tel que les messages correspondants soient reliés

Moyens

Oracles

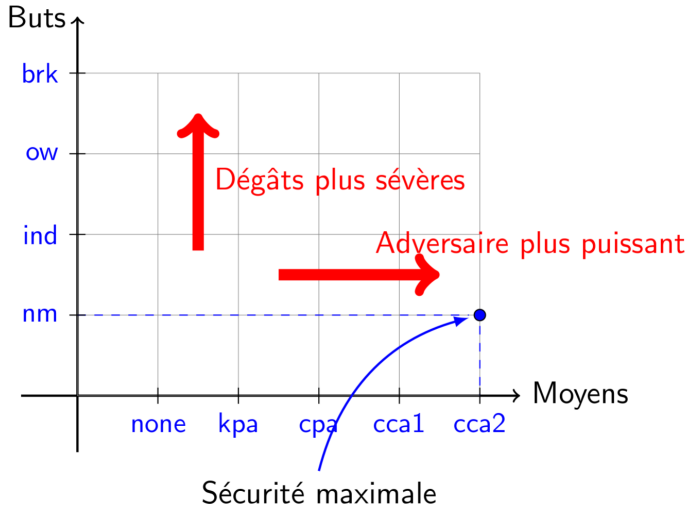
L'adversaire a accès à des oracles :

- **chiffrement** de tous les messages de son choix
- **déchiffrement** de tous les messages de son choix

Moyens de l'attaquant

- 1 **KPA – Clairs connus** : il voit des couples clairs/chiffrés
- 2 **CPA – Clairs choisis** : il chiffre des messages de son choix
- 3 **CCA1 – Chiffrés choisis** : il peut faire déchiffrer des messages arbitraires avant recevoir le "challenge"
- 4 **CCA2 – Attaque adaptative** : il peut faire déchiffrer des messages après avoir reçu le "challenge"
(restriction de ne pas faire déchiffrer ce challenge)

Les axes d'analyse



Protocole RSA

RSA - Génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

- Soit $n = p \cdot q$ (p et q premiers)
- L'ordre du groupe multiplicatif $\mathbb{Z}_n^* = \varphi(n) = (p-1)(q-1)$
- Soit e un entier premier avec $\varphi(n) = (p-1)(q-1)$
- Soit d un entier qui satisfait $d \cdot e = 1 \pmod{\varphi(n)}$

$$d \cdot e + u\varphi(n) = 1 \quad (\text{Bézout})$$

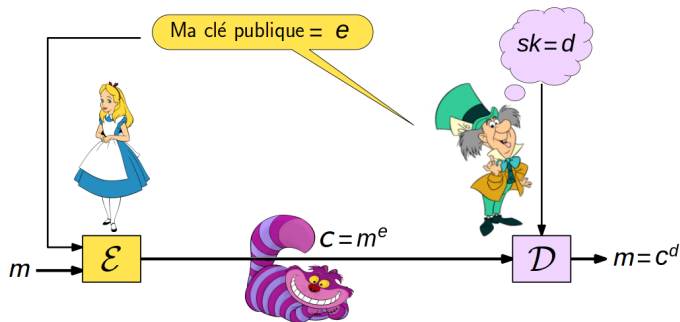
clé publique

- $n = pq$: module public
- e : exposant public

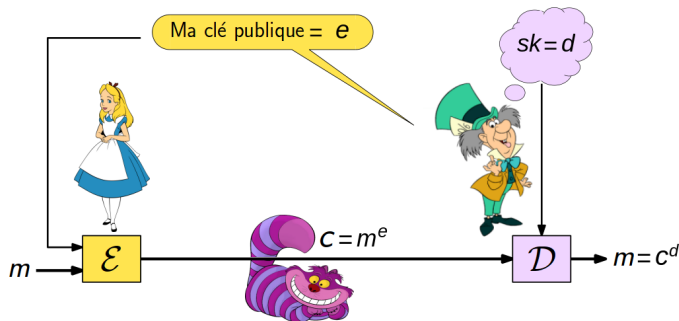
clé secrète

- $d = e^{-1} \pmod{\varphi(n)}$
- les premiers p et q

Protocole RSA



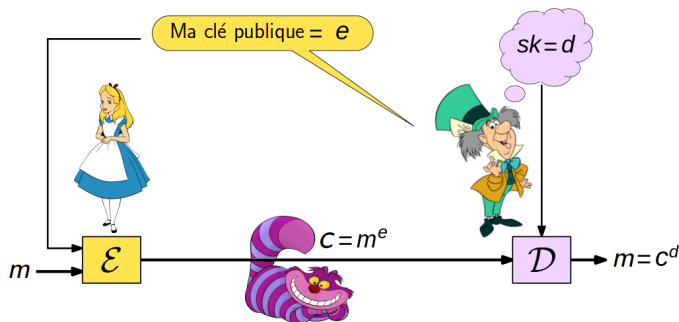
Protocole RSA



RSA - Chiffrement

$$\mathcal{E}(\text{pk} = (e, n), m) = m^e \pmod{n}$$

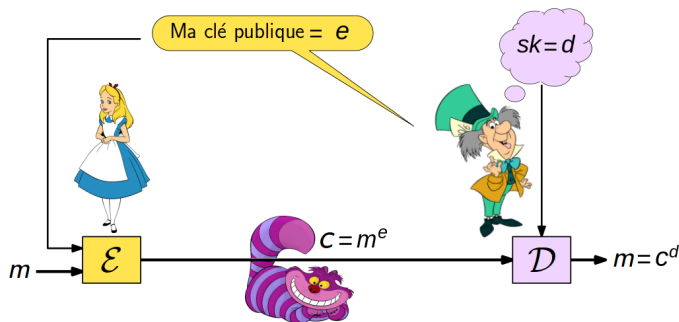
Protocole RSA



RSA - Déchiffrement

$$\mathcal{D}(sk = d, C) = C^d \pmod{n}$$

Protocole RSA



Vérification

$$(m^e)^d = m^{ed} = m^{1-u\varphi(n)} = m \cdot 1 = m \pmod{n}$$

Protocole RSA

Propriétés

- ① Chiffrement déterministe :

$$\text{message } m \Rightarrow \text{unique } C = \mathcal{E}_{pk}(m)$$

- ② Chiffrement homomorphique :

$$\mathcal{E}_{pk}(m_1 \cdot m_2) = \mathcal{E}_{pk}(m_1) \cdot \mathcal{E}_{pk}(m_2)$$

- ③ Chiffrement commutative :

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

Problèmes difficiles



Factorisation

- $(p, q) \rightarrow p \cdot q$ facile
- $n = p \cdot q \rightarrow (p, q)$ difficile

Meilleur algorithme (crible algébrique) : $\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$



Fonction RSA

Extraction de racine e -ième

- $x \rightarrow x^e \bmod n$ facile
- $y = x^e \rightarrow x \bmod n$ difficile

avec la trappe $d = e^{-1} \pmod{\varphi(n)}$: $x = y^d = x^{ed} = x \bmod n$

Analyse de sécurité pour RSA

On suppose RSA bien utilisé



Sécurité

- **BRK-CPA** : clé secrète $\rightarrow$ factorisation
- **OW-CPA** : extraction de $m = c^d$ à partir de c
racine e-ième, soit **problème RSA**



Vulnérabilités

- **IND-CPA** : il y a (au moins) un bit d'information qui fuit
($m = 1$)
- **OW-CCA** : étant donné $c = m^e$, l'attaquant $\mathcal{A}$ demande à déchiffrer $c' = 2^e \cdot c$ et obtient $2m$

Protocole ElGamal

ElGamal - Génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

- Soit un premier p et le groupe cyclique $\mathbb{Z}_p^*$
- Soit $g \in \mathbb{Z}_p^*$ un élément d'ordre $q \mid (p - 1)$.
- Soit une clé secrète $\text{sk} = x$.
- Soit $\text{pk} = g^x \pmod{p}$.

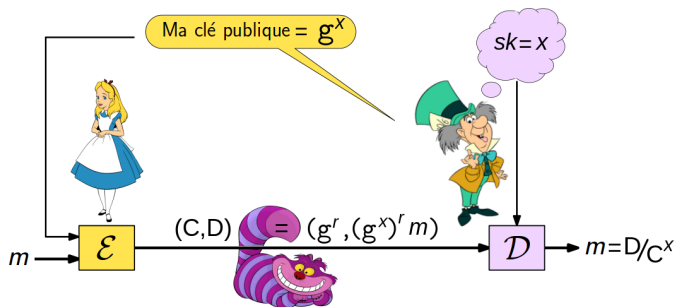
clé publique

- p et g : paramètres publics
- $\text{pk} = g^x$: clé publique

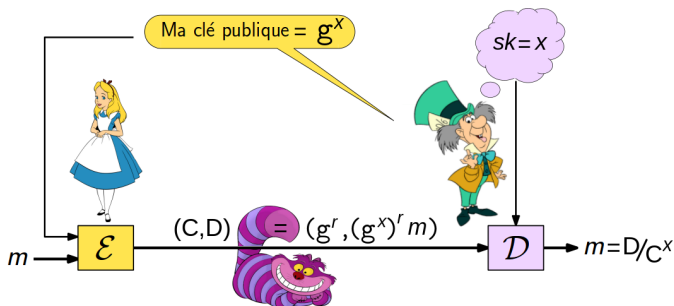
clé secrète

- $\text{sk} = x$
exposant secret

Protocole ElGamal



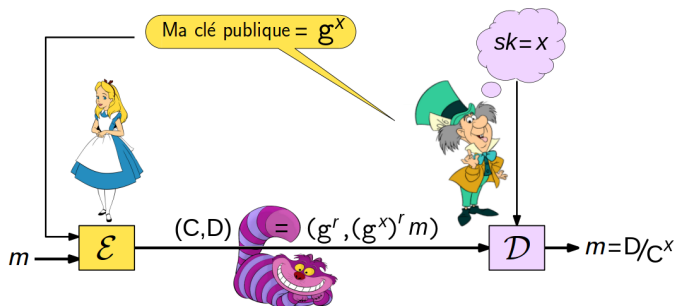
Protocole ElGamal



ElGamal - Chiffrement

$$\mathcal{E}(\text{pk} = y, m) = (C, D) = (g^r, (g^x)^r m)$$

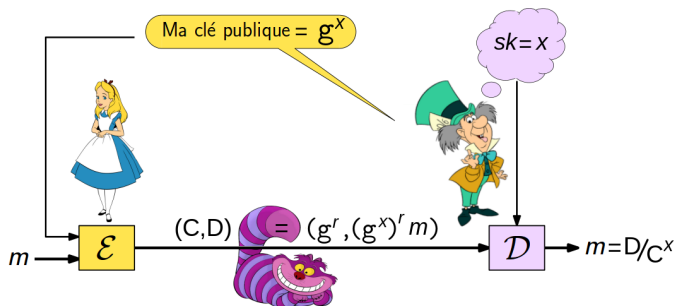
Protocole ElGamal



ElGamal - Déchiffrement

$$\mathcal{D}(sk = x, (C, D)) = D / C^x \pmod{p}$$

Protocole ElGamal



Vérification

$$D/C^x = (g^x)^r m / (g^r)^x = m \pmod{p}$$

Problèmes difficiles

Soit $\mathbb{G}$ un groupe multiplicatif cyclique, $\mathbb{G} = \langle g \rangle$:



Logarithme discret (DLOG)

Etant donnés $g \in \mathbb{G}$ et $X = g^x$,
Calculer $\log_g(X) = x$



Calculer Diffie-Hellman (CDH)

Etant donnés g , $A = g^a$ et $B = g^b$,
Calculer $C = CDH(A, B) = g^{ab}$



Décider si Diffie-Hellman (DDH)

Etant donnés g , $A = g^a$, $B = g^b$ et $C = g^c$ dans $\mathbb{G}$
Décider si $C = g^{ab}$

Analyse de sécurité pour ElGamal



Sécurité

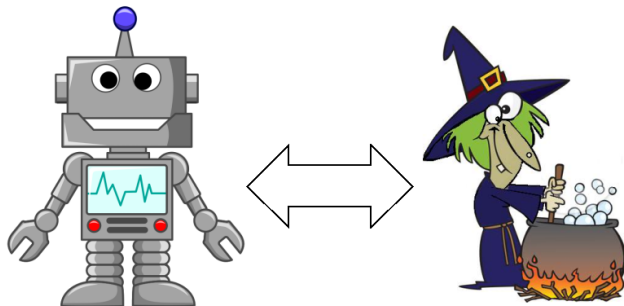
- **BRK-CPA** : clé secrète $\rightarrow$ **DLOG**
- **OW-CPA** : extraction de $m = D \cdot C^{-x}$ à partir de C
problème calculatoire Diffie-Hellman **CDH**
- **IND-CPA** : distinguer entre m_0 et m_1
problème décisionnel Diffie-Hellman **DDH**



Vulnérabilités

- **OW-CCA** : étant donné (C, D) , l'attaquant $\mathcal{A}$ demande à déchiffrer $(C, 2D)$ et obtient $2m$

Preuves de sécurité



Principe de la preuve

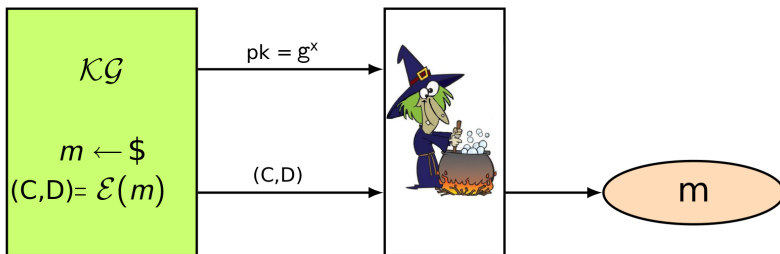
Hypothèse : Un problème P difficile

Réduction : Si $\mathcal{A}$, un adversaire (polynomial), casse le schéma de chiffrement, alors $\mathcal{A}$ peut être utilisé pour construire un algorithme **ROBOT** qui résout P

Rétrouver le message chiffré

OW - CPA

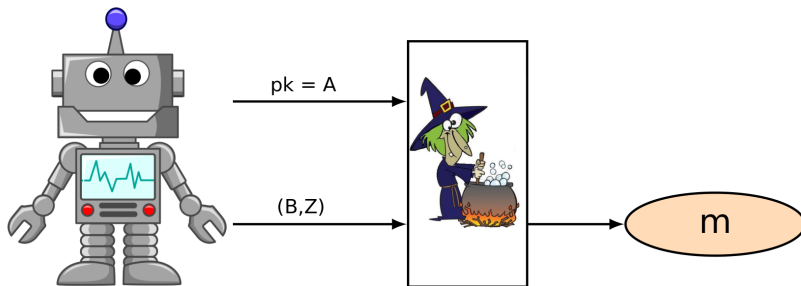
Déchiffrer un message arbitraire



$$C = g^r \quad \text{et} \quad D = (g^x)^r \cdot m \Rightarrow \mathcal{A} \text{ calcule } m = D / (C^x)$$

Réduction OW - CPA $\Rightarrow$ CDH

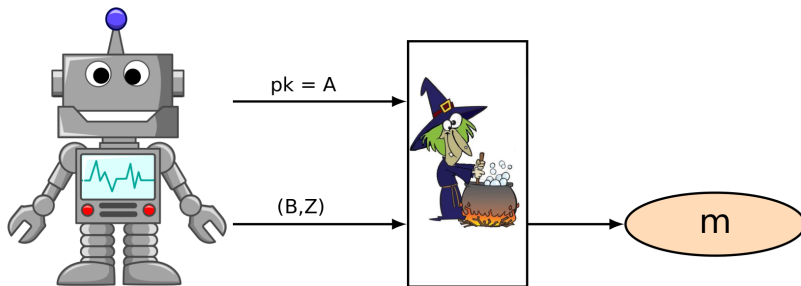
input $(A = g^a, B = g^b)$



But : $C = g^{ab} = B^a$

Réduction OW - CPA $\Rightarrow$ CDH

input $(A = g^a, B = g^b)$

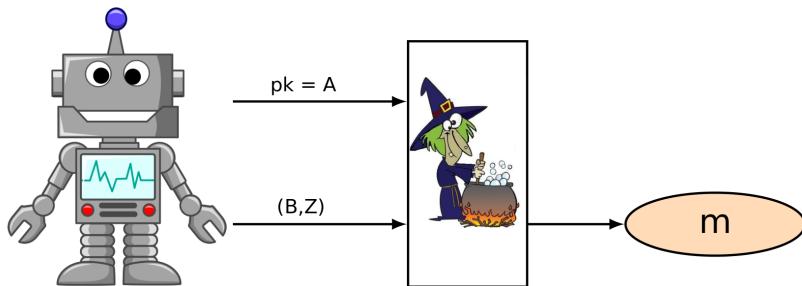


But : $C = g^{ab} = B^a$

$\mathcal{A}$ calcule $m = Z / (B^a)$

Réduction OW - CPA $\Rightarrow$ CDH

input $(A = g^a, B = g^b)$



But : $C = g^{ab} = B^a$

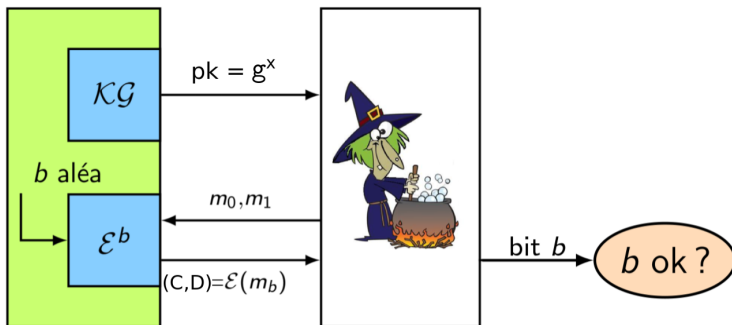
output $Z/m = B^a$

$\mathcal{A}$ calcule $m = Z/(B^a)$

Distinguer entre deux chiffrés

IND - CPA

$\mathcal{A}$ choisit deux messages m_0 et m_1 :

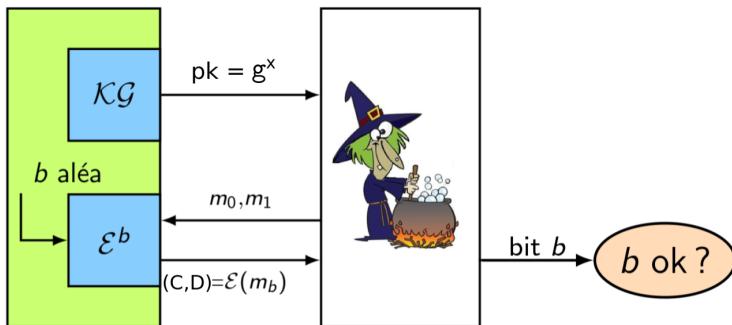


$C = g^r$ et $D = (g^x)^r \cdot m_b \Rightarrow \mathcal{A}$ reconnaît le chiffré de m_b

Distinguer entre deux chiffrés

IND - CPA

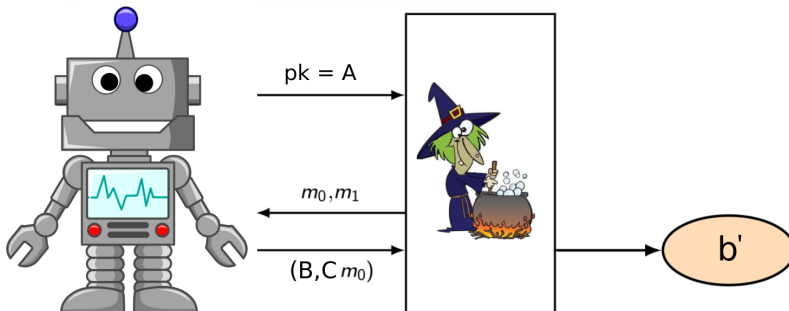
$\mathcal{A}$ choisit deux messages m_0 et m_1 :



$C = g^r$ et $D = (g^x)^r \cdot m_b \Rightarrow \mathcal{A}$ reconnaît le chiffré de m_b

Réduction IND - CPA $\Rightarrow$ DDH

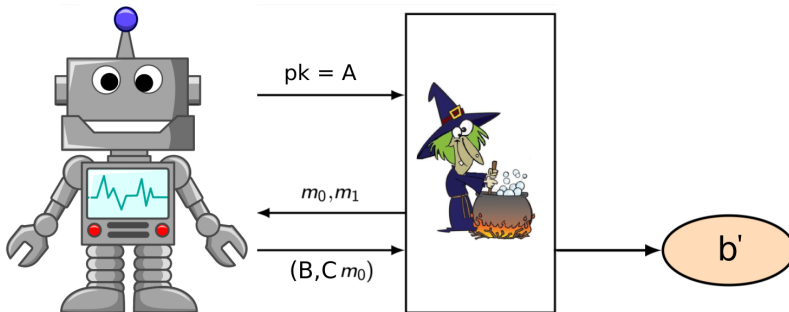
input $(A = g^a, B = g^b, C = g^c)$



But : $C \stackrel{?}{=} g^{ab}$

Réduction IND - CPA $\Rightarrow$ DDH

input $(A = g^a, B = g^b, C = g^c)$

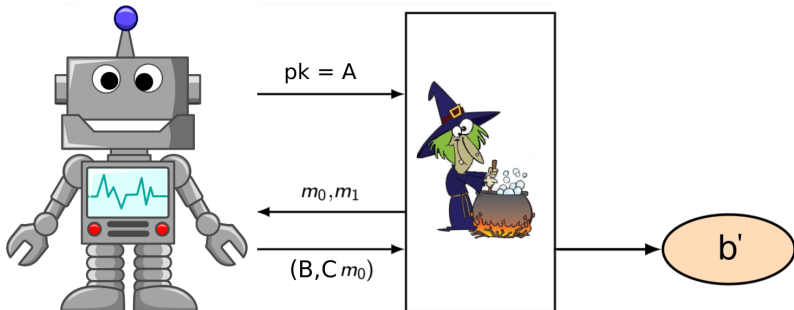


But : $C \stackrel{?}{=} g^{ab}$

Si $b' = 0$ **output** oui

Réduction IND - CPA $\Rightarrow$ DDH

input $(A = g^a, B = g^b, C = g^c)$



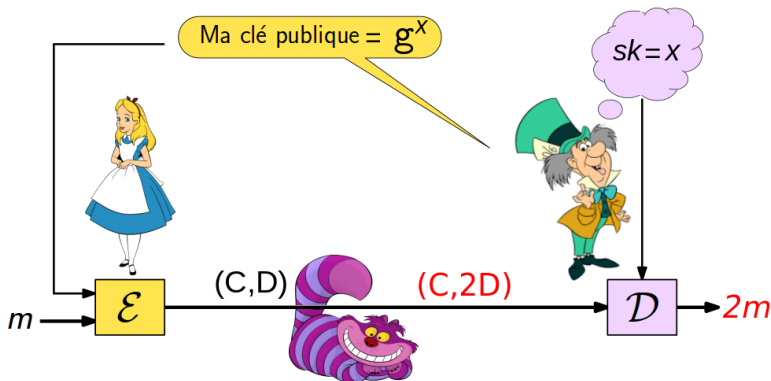
But : $C \stackrel{?}{=} g^{ab}$

Si $b' = 0$ **output** oui

Si $C = g^{ab} : (B, C m_0) = (g^b, g^{ab} m_0) = \mathcal{E}(m_0)$

Si $C = g^c : (B, C m_0) = (g^b, g^c m_0)$ est un aléa

Attack IND - CCA



Solution : Authentification

Authentification - Motivations



Usurpation d'identité

La confidentialité de la communication n'est pas toujours suffisante !

Motivations



Usurpation d'identité

- Bob crédite le compte d'Alice sur Internet
- L'adversaire intercepte le message et remplace les données bancaires de Alice avec les siennes
- La banque crédite le compte de l'adversaire !

Motivations



Usurpation d'identité

Pourquoi faut-il authentifier les clés publiques ?

- Alice souhaite envoyer un message secret à Bob.
- Alice utilise la clé publique de Bob pk_B .
- Mais comment Alice a-t-elle obtenu pk_B ?
- Oscar lui fait croire que la clé publique de Bob est pk_O (clé publique de Oscar)
- Alice envoie à Bob le secret chiffré avec la clé de Oscar
- Oscar décrypte le message et apprend le secret !

Signature numérique : idée générale

Objectifs

Reproduire les caractéristiques d'une signature manuscrite :

- Lier un document à son auteur
- Rendre la signature difficilement imitable
- Responsabilité de l'auteur (juridique)



Signature numérique

Propriétés

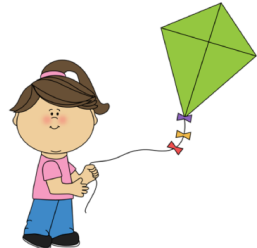
La signature numérique dépend du signataire et du document :

- La signature appartient à un seul document :
 - impossible à découper une signature sur un message et la recoller sur un autre.
 - le document signé ne peut être modifié.
- La signature ne peut être falsifiée.
- La signature ne peut pas être reniée.

Signature numérique



Signataire



Vérifieur

Signature σ —————→ OK/NO

m —————→ m

Les différents acteurs

Le Signataire

Doit pouvoir facilement émettre des signatures en son nom

Les différents acteurs

Le Signataire

Doit pouvoir facilement émettre des signatures en son nom

Le vérifieur

Doit pouvoir vérifier la signature sans avoir d'information confidentielle

Potentiellement tout le monde doit pouvoir vérifier.

Les différents acteurs

Le Signataire

Doit pouvoir facilement émettre des signatures en son nom

Le vérifieur

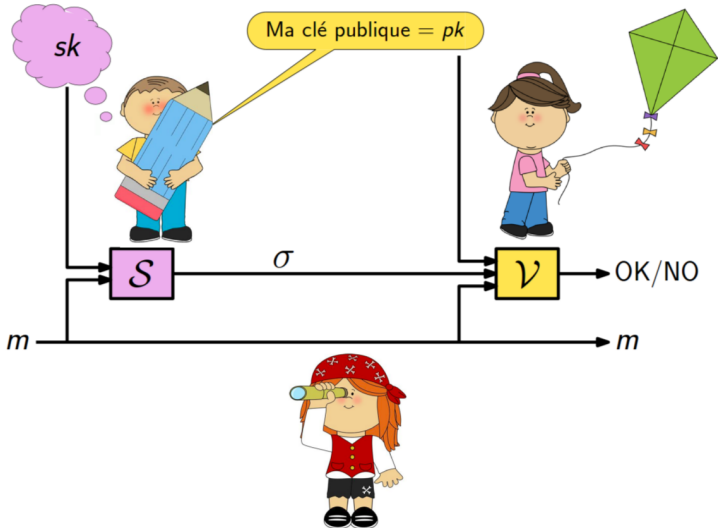
Doit pouvoir vérifier la signature sans avoir d'information confidentielle

Potentiellement tout le monde doit pouvoir vérifier.

L'ennemi

Il cherche à générer une fausse signature

Fonctionnement



Signatures numériques

Algorithme de génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

à partir d'un paramètre de sécurité, il produit un couple de clés : clé publique, clé secrète

Signatures numériques

Algorithme de génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

à partir d'un paramètre de sécurité, il produit un couple de clés : clé publique, clé secrète

Algorithme de signature

$$\mathcal{S}(\text{sk}, m) = \sigma$$

utilise la clé secrète sk pour signer le message

Signatures numériques

Algorithme de génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

à partir d'un paramètre de sécurité, il produit un couple de clés : clé publique, clé secrète

Algorithme de signature

$$\mathcal{S}(\text{sk}, m) = \sigma$$

utilise la clé secrète sk pour signer le message

Algorithme de vérification

$$\mathcal{V}(\text{pk}, m, \sigma) = \text{yes/no}$$

utilise la clé publique pour vérifier le message et la signature reçus

Signature RSA

Génération des clés

$$\mathcal{KG}(\ell) = (pk, sk)$$

- Soit $n = p \cdot q$ (p et q premiers)
- L'ordre du groupe multiplicatif $\mathbb{Z}_n^* = \varphi(n) = (p - 1)(q - 1)$
- Soit e un entier premier avec $\varphi(n) = (p - 1)(q - 1)$
- Soit d un entier qui satisfait $d \cdot e = 1 \pmod{\varphi(n)}$

$$d \cdot e + u\varphi(n) = 1 \quad (\text{Bézout})$$

clé publique

- $n = pq$: module public
- e : clé de vérification

clé secrète

- $d = e^{-1} \pmod{\varphi(n)}$
- les premiers p et q

Chiffrement RSA

RSA - Génération des clés

- $pk = e$
- $sk = d$

RSA - Chiffrement

$$\mathcal{E}(pk = (e, n), m) = C$$
$$C = m^e \pmod{n}$$

RSA - Déchiffrement

$$\mathcal{D}(sk = d, C) = m$$
$$m = C^d \pmod{n}$$

Signature RSA

σ RSA - Génération des clés

- $pk = e$
- $sk = d$

σ RSA - Signer

$$\mathcal{S}(sk = d, m) = \sigma$$
$$\sigma = m^d \pmod{n}$$

σ RSA - Vérifier

$$\mathcal{V}(pk, m, \sigma) = \text{yes/no}$$
$$m \stackrel{?}{=} \sigma^e \pmod{n}$$

Signature RSA



Avantages

- La signature assure l'authentification
- La vérification est publique



Inconvénients

- Signature déterministe (on peut la reutiliser pour envoyer de nouveau le même message signé).
- À partir d'une signature σ on peut fabriquer un message m pour laquelle σ est validée : $m = \sigma^e$.
- Malléabilité : Si on a deux messages signés (m_1, σ_1) et (m_2, σ_2) on crée un nouveau message signé $(m_1 m_2, \sigma_1 \sigma_2)$.

Signature ElGamal

ElGamal - Génération des clés

$$\mathcal{KG}(\ell) = (\text{pk}, \text{sk})$$

- Soit un premier p et le groupe cyclique $\mathbb{Z}_p^*$
- Soit $g \in \mathbb{Z}_p^*$ un élément d'ordre $q \mid (p - 1)$.
- Soit une clé secrète $\text{sk} = x$.
- Soit $\text{pk} = g^x \pmod{p}$.

clé publique

- p et g : paramètres publics
- $\text{pk} = g^x$: clé de vérification

clé secrète

- $\text{sk} = x$
exposant secret

Chiffrement ElGamal

Signature ElGamal

ElGamal - Clés

- $pk = g^x$
- $sk = x$

σ ElGamal - Clés

- $pk = g^x$
- $sk = x$

ElGamal - Chiffrement

$$\mathcal{E}(pk = g^x, m) = (C, D)$$
$$(C, D) = (g^r, (g^x)^r m)$$

σ ElGamal - Signer

$$\mathcal{S}(sk = x, m) = \sigma$$
$$\sigma = (C, D) = (g^r, (m - xg^r)/r)$$

ElGamal- Déchiffrement

$$\mathcal{D}(sk = x, (C, D)) = m$$
$$m = D/C^x \pmod{p}$$

σ ElGamal - Vérifier

$$\mathcal{V}(pk = g^x, m, \sigma) = \text{yes/no}$$
$$g^m \stackrel{?}{=} C^D (g^x)^C \pmod{p}$$

Signature ElGamal



Avantages

- La signature est très efficace (une seule exponentiation modulaire)
- Signature randomisée (chaque fois qu'on signe le même message on obtient une nouvelle signature)



Inconvénients

- La vérification est couteuse (3 exponentiations)
- La signature est longue : deux fois la taille de p

Généralités - Signatures



Problématiques

- La malléabilité : $\sigma(m_1), \sigma(m_2) \Rightarrow \sigma(m_1 m_2)$
- La construction d'un message valide à partir d'une signature.
- Signer des messages arbitrairement longs avec un schéma
 - Signature d'un document de 15 Mo avec RSA 1024 bits ??



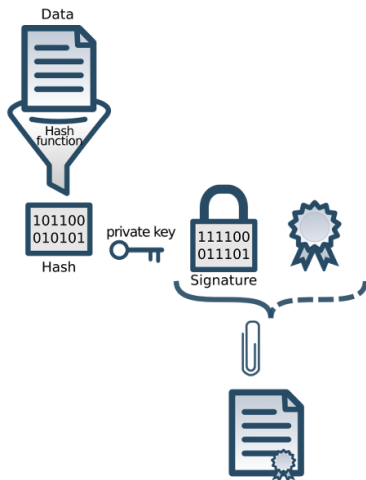
Solution : Hash & Sign

Utilisation de fonctions de hachage

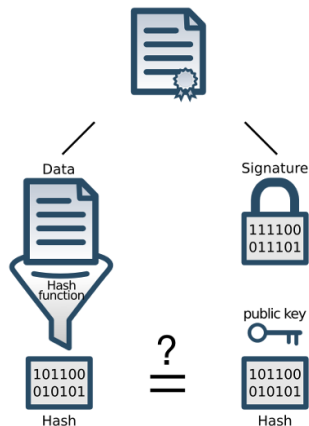
- Taille de signature indépendante de la taille du message
- Sécurité conservée : les falsifications restent difficiles

Signature et hachage

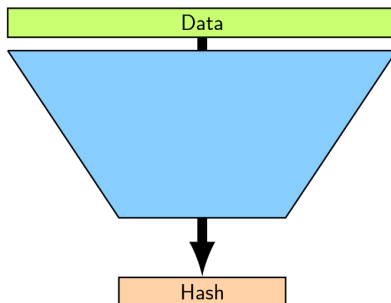
Signature



Verification



Idée générale



Fonction de hachage

Une fonction qui, à partir d'une donnée arbitraire, calcule une empreinte servant à condenser et identifier rapidement la donnée initiale

But

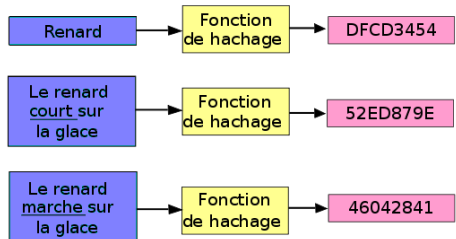
L'intégrité des données

💣 Vérifier que les infos transmises n'ont pas subi d'altérations.

💣 Produire une empreinte condensée qui soit facilement vérifiable.

Entrée

Empreinte



Fonctions de hachage

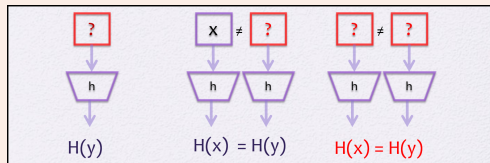
Propriétés

Une bonne fonction de hachage doit être :

- publique (pas de notion de secret)
- facile et rapide à calculer
- à sortie de taille fixe (quelque soit l'entrée)
- bien répartie en sortie ("mélange" bien)

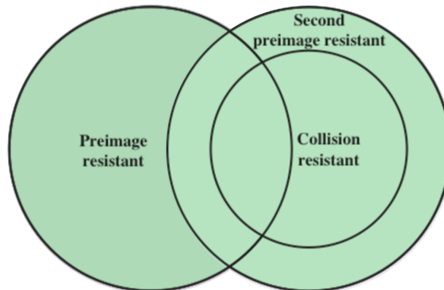
Fonctions de hachage

• Propriétés



- **Résistance à la préimage :**
étant donné h , il est difficile de trouver y tel que $h = H(y)$
- **Résistance à la seconde préimage :**
étant donné x , il est difficile de trouver $y \neq x$ tel que $H(x) = H(y)$
- **Résistance à la collision :**
il est difficile de trouver x et y tels que $H(x) = H(y)$

Fonctions de hachage



Types des fonctions

- **Fonction de Hachage à Sens Unique**
résistance à la préimage et à la seconde préimage
- **Fonction de Hachage résistante aux collisions**
résistance à la seconde préimage et à la collision

Paradoxe des anniversaires

Problème

Calculer le nombre de personnes que l'on doit réunir pour avoir une chance sur deux que deux personnes de ce groupe aient leur anniversaire le même jour.



Paradoxe des anniversaires



Résultat

Une classe de 23 des élèves est suffisante, ce qui choque l'intuition, car il y a 365 possibilités pour les dates d'anniversaire !

Solution

Calculer $\bar{P}$

$P = \text{Prob}$ (il y a une paire de personnes nées le même jour)

$\bar{P} = \text{Prob}$ (aucune paire de personnes nées le même jour)

n personnes, 365 jours possibles $\Rightarrow 365^n$ possibilités.

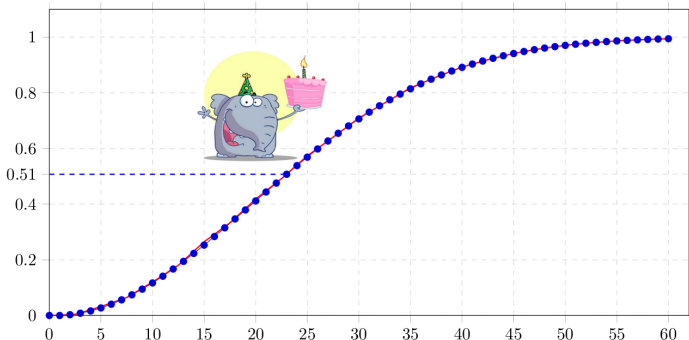
Si on veut des jours différents, nous obtenons un arrangement de n parmi 365 :

- personne 1 : 365 choix,
- personne 2 : 364 choix,
- ...

$$A_{365}^n = (365 - 0)(365 - 1) \dots (365 - n + 1) = \frac{365!}{(365 - n)!}$$

$$\bar{P} = \frac{1}{365^n} \frac{365!}{(365 - n)!} \Rightarrow P = 1 - \bar{P}$$

Paradoxe des anniversaires



Résultat

À partir de 57 personnes, la probabilité est supérieure à 99%.

Attaques des anniversaires

Trouver des collisions

Si la fonction de hachage a N valeurs possibles, $\sqrt{N}$ calculs suffisent en moyenne pour obtenir une collision :

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

Attaques des anniversaires

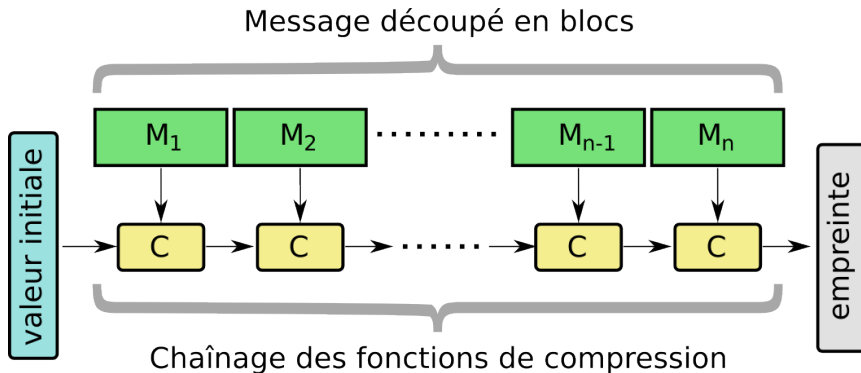
Trouver des collisions

Si la fonction de hachage a N valeurs possibles, $\sqrt{N}$ calculs suffisent en moyenne pour obtenir une collision :

$$H : \{0, 1\}^* \rightarrow \{0, 1\}^n$$

La recherche brutale de collisions a plus d'une chance sur 2 d'aboutir après seulement $\mathcal{O}(2^{n/2})$ essais !

Hachage - Principe de fonctionnement



Fonctions de hachage classiques

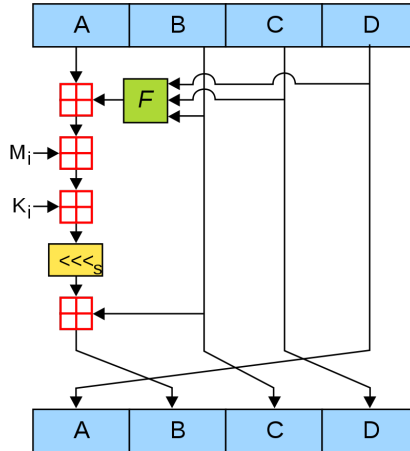
Fonction MD

- Proposé par Rivest en 1991
- MD2, MD4, MD5 : condensés de 128 bits
- MD2 : calcul (théorique) de pré-images en $\mathcal{O}(2^{104})$
- MD4 : peu utilisée, collisions "faciles" (2004)

Fonction MD5

- MD5 : blocs de 512 bits, sortie de 128 bits
- 64 rondes sur 4 sous-blocs de 32 bits A, B, C, D
- Sécurité de MD5 face aux collisions
 - Force brute : $\mathcal{O}(2^{64})$ (attaque anniversaires)
 - Collisions en $\mathcal{O}(2^{42})$ exhibées en 2004 [Wang04]
 - Collisions en $\mathcal{O}(2^{30})$ exhibées en 2005 [Sasaki05]

La fonction de compression de MD5



Fonctions de hachage classiques

Fonction SHA (Secure Hash Algorithm)

- Norme NIST : Blocs de 512 bits, empreinte de 160 bits
- 80 rondes sur 5 sous-blocs de 32 bits A, B, C, D, E

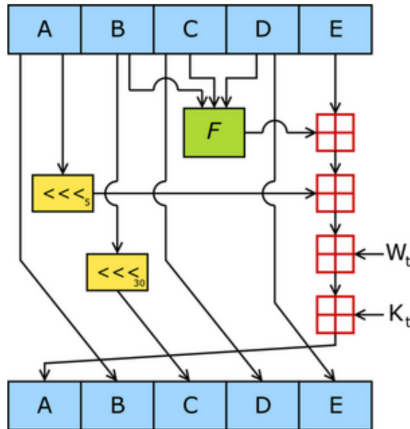
Fonction SHA-1 (1994)

- Considéré comme plus sûr que MD5 face aux collisions
 - Force brute : $\mathcal{O}(2^{80})$ (attaque anniversaires)
 - Attaque théorique en $\mathcal{O}(2^{63})$ [Wang05]
 - Encore largement utilisé (signatures)

Fonction SHA-2 (2000)

- Blocs de 512 bits, empreinte de 256 bits
 - Force brute : $\mathcal{O}(2^{128})$ (attaque anniversaires)
 - Pas d'attaque connues pour le moment

La fonction de compression de SHA-1



Application : Stockage de mot de passe



Problémématique

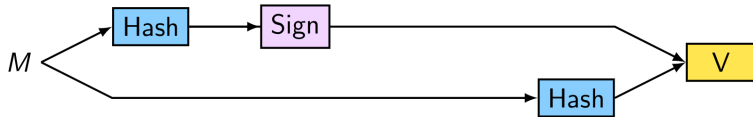
- Connection et identification sur un site : login + mot de passe
- Logins et mots de passe enregistrés dans une base de données
- Risque : acces à la base de données $\Rightarrow$ usurpation d'identité !



Solution : Utilisation de fonctions de hachage

- Pour pallier à cette faille, **les empreintes** des mots de passe sont stockés dans la base de données
- À chaque identification, vous produirez votre mot de passe dont l'empreinte sera calculée et comparée à celle stockée dans la base de données.
- La résistance à la préimage assure la difficulté de produire un mot de passe définissant la même empreinte !

Application : Signatures



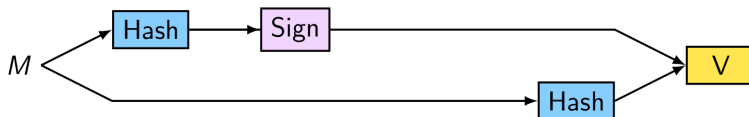
Idée : Hash & Sign

Au lieu de signer le message très long, on signe un condensé du message

Fonctionnement

- La signature fonctionne sur le condensé de taille fixe
- Chaque condensé doit être unique ! ?
Impossible !

Application : Signatures



Conséquences

Attaque en second antécédent :

- À partir d'un message signé, on en trouve un autre avec le même condensé $\Rightarrow$ la signature reste valide

Attaque en collision :

- À partir d'une collision $H(m_1) = H(m_2)$, on demande la signature du message $m_1 \Rightarrow$ c'est aussi valide pour m_2
- On répudie la signature de m_1 et prétendre avoir signé m_2

Analyse de sécurité

Que souhaite-t-on ?

- Un adversaire ne doit pas pouvoir signer à la place du signataire :
 - un message quelconque ("67fc 3a90 b245")
 - un message de son choix ("Dette = 256,000 \$")
- Un adversaire ne doit pas pouvoir retrouver la clé privée
 - à partir de la clé publique
 - à partir de signatures interceptées
 - ... ou qu'il aurait "choisies"

Analyse de sécurité

Remarque



La sécurité inconditionnelle n'existe pas en signature !

- il est toujours possible de tester toutes les signatures possibles jusqu'à obtenir une vérification correcte
- cela vient du fait que l'algorithme de vérification est public
- la sécurité sera toujours calculatoire

Prouver la sécurité

Goldwasser, Micali and Rivest



Modèle de sécurité

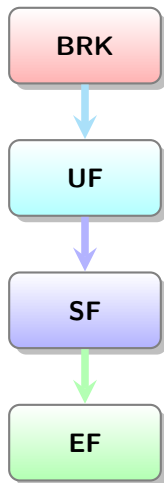
- 1 **les buts de l'attaquant**
ce qu'il cherche à obtenir $\Rightarrow$ le niveau de sécurité souhaité
- 2 **les moyens de l'attaquant**
ce que peut faire l'adversaire $\Rightarrow$ puissance de l'attaque

Paramètres du modèle

Buts de l'attaquant

- ① BRK – Cassage total : retrouver la clé secrète de signature
- ② UF – Forge universelle : signer n'importe quel message
- ③ SF – Forge sélective : signer un message choisi
- ④ EF – Forge existentielle : signer un message quelconque

Hierarchie de sécurité

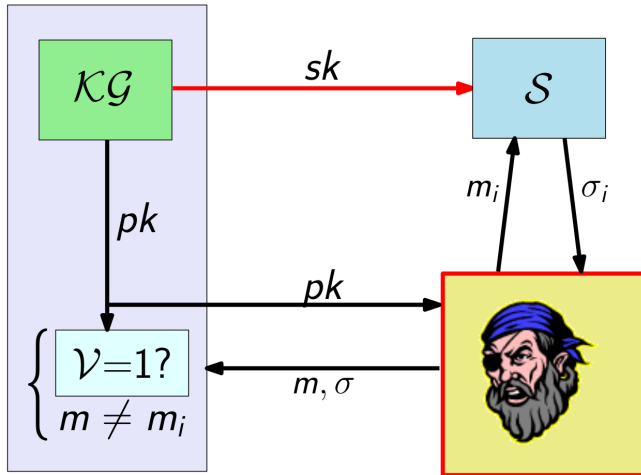


Paramètres du modèle

Moyens de l'attaquant

- ① **KO – Key-Only** : il ne voit aucun message
- ② **KMA – Messages connus** : il intercepte une liste de messages signés
- ③ **CMA – Messages choisis** : il peut faire signer des messages choisis

Sécurité EF-CMA



Modèle de sécurité

Attaques possibles

Modèle de sécurité = **un but** + **des moyens**

Exemple

- Sécurité **EF-CMA** (Existential Forgery, Chosen Message Attack)

Absence de forge existentielle sous une attaque à messages choisis adaptative

Prendre le cas de la signature RSA : quel niveau peut-on espérer atteindre ?

Analyse de sécurité pour σ RSA

Signature σ RSA

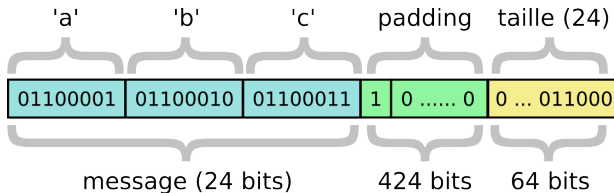
- Clés : $pk = e$ $sk = d$
- Signer : $\sigma = m^d \pmod{n}$
- Vérifier : $m \stackrel{?}{=} \sigma^e \pmod{n}$



Vulnérabilités

- **EF-KO** – Existentiellement falsifiable par attaque sans message :
 - étant donné σ , l'attaquant calcule $m = \sigma^e$
- **UF-CMA** – Universellement falsifiable par attaque à messages choisis :
 - l'attaquant fait signer $m/2$ et 2 et en déduit une signature pour m

Solution : Padding



L'encodage (bourrage, encapsulation)

Permet de briser la propriété de multiplicativité

- les attaques basiques sont rendues inopérantes
- le message est encapsulé dans un certain format
- chaîne de bits fixe accolée au message
- codage d'une propriété du message (checksum, longueur, etc.)
- chaîne de bits variable mais avec redondance

Signature RSA avec Hachage Plein

FDH : Full-Domain Hash

Hash-and-Sign

Cette méthode suppose que l'on dispose d'une fonction de hachage produisant comme empreintes des entiers modulo n .

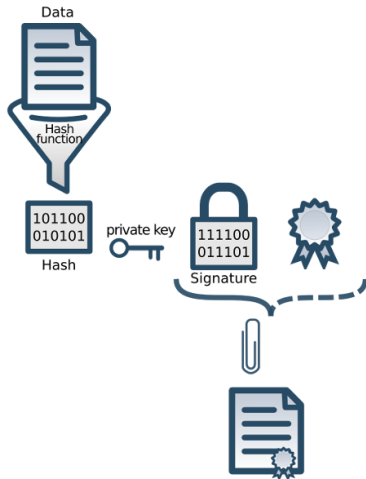
- on calcule le condensé du message avant de signer
- le hachage joue le rôle de l'encodage

Pour signer un message m , on calcule

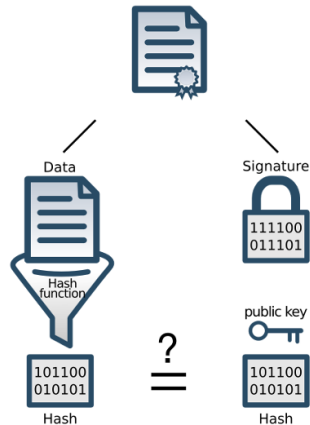
$$\sigma = H(m)^d$$

Signature et hachage

Signature



Verification



Signature RSA-FDH



Sécurité

Le schéma a été prouvé sûr par Bellare et Rogaway en 1996

- sécurité EF-CMA
- sous l'hypothèse de difficulté du problème RSA
- dans le modèle de l'oracle aléatoire



Inconvénients

- La réduction proposée donne une borne mauvaise (i.e., on est obligé de prendre des grands paramètres)
- Pour une probabilité de forge de 2^{-80} avec 2^{60} calculs de hachage, la taille du module RSA doit être de 4096 bits.
- Preuve améliorée en 2000 par Coron : un module de 2048 bits est suffisant