# Cryptosystèmes: Sécurité et attaques

Anca Nitulescu
anca.nitulescu@ens.fr

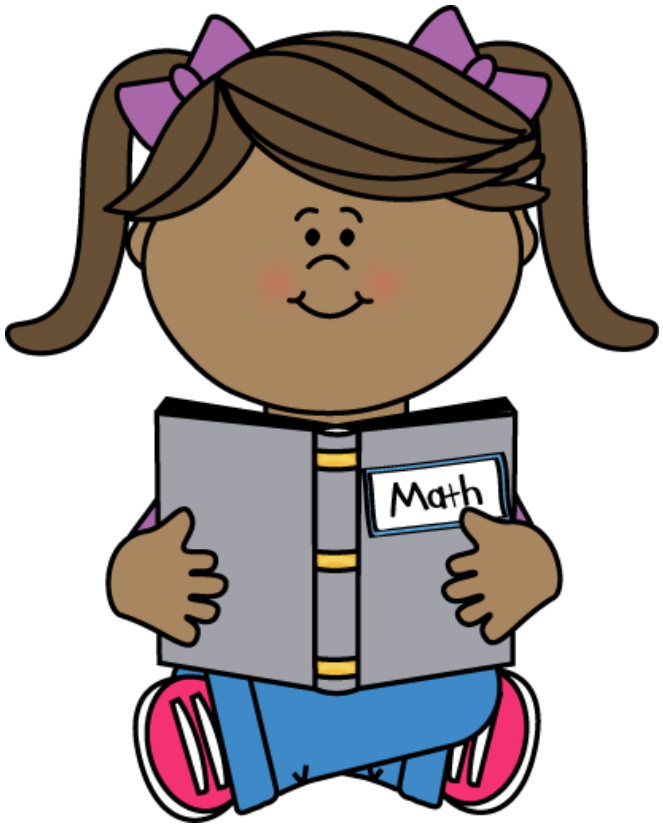
Université Paris 13 Villetaneuse

Cours 6
08/02/2016

# Rappels mathématiques

## Outils mathématiques

- Algorithme d'Euclide étendu
- Nombres premiers grands
- Exponentiation modulaire
- Inversion modulaire
- Calcul des restes chinois



# Arithmétique modulaire

- $(\mathbb{Z}_n, +)$ forme un **groupe additif commutatif d'ordre** $n$.
- $(\mathbb{Z}_n, +, \cdot)$ forme un **anneau commutatif**.
- **Inverse modulaire** de $a$ dans $\mathbb{Z}_n$ : entier $b = a^{-1}$ tel que

$$a \times b = 1 \mod n$$

- $\mathbb{Z}_n^\star =$ l'ensemble des éléments inversibles modulo $n$.
- $(\mathbb{Z}_n^\star, \cdot)$ forme un **groupe multiplicatif**.
- $(\mathbb{Z}_p, +, \cdot)$ forme un **corps commutatif**.

**Attention !**

$\mathbb{Z}_n^\star \neq \mathbb{Z}_n \setminus \{0\}$ pour $n$ **composé**
$\mathbb{Z}_p^\star = \mathbb{Z}_p \setminus \{0\}$ pour $p$ **prime**

# Critère d'inversibilité

## Les entiers inversibles modulo $n$

$x \in \mathbb{Z}_n^\star$ est inversibles modulo $n$ si et seulement si $\text{pgcd}(x, n) = 1$.
*Preuve* : T. Bézout.

## Calcul de l'inverse modulaire

Trouver $x^{-1} \mod n$

- Il existe $u$ et $v$ tels que $xu + nv = \text{pgcd}(x, n) = 1$
- Trouver l'inverse d'un élément revient à calculer $u$.
- L'algorithme d'Euclid étendu calcule des coeficients $(u, v)$

# Fonction d'Euler

## Définition

- $\varphi(n)$ est le nombre d'entiers de $[1, n]$ qui sont premiers avec $n$.
- $\varphi(n)$ désigne l'ordre du groupe multiplicatif $\mathbb{Z}_n^\star$

## Propriétés

si $p$ est premier et $q$ premier :

- $\varphi(p) = p - 1$
- $\varphi(pq) = \varphi(p)\varphi(q)$

# Exponentiation modulaire

## Théorème de Lagrange

Si $\mathbb{G}$ est un groupe multiplicatif d'ordre $n$, alors :

$$\forall g \in \mathbb{G} \quad g^n = e$$

## Théorème d'Euler

Pour tout entier $n$ et tout $a \in \mathbb{Z}_n^\star$, on a

$$a^{\varphi(n)} = 1 \mod n$$

## Petit théorème de Fermat

Pour $p$ premier et tout entier $a$ on a

$$a^p = a \mod p$$

# Exponentiation modulaire

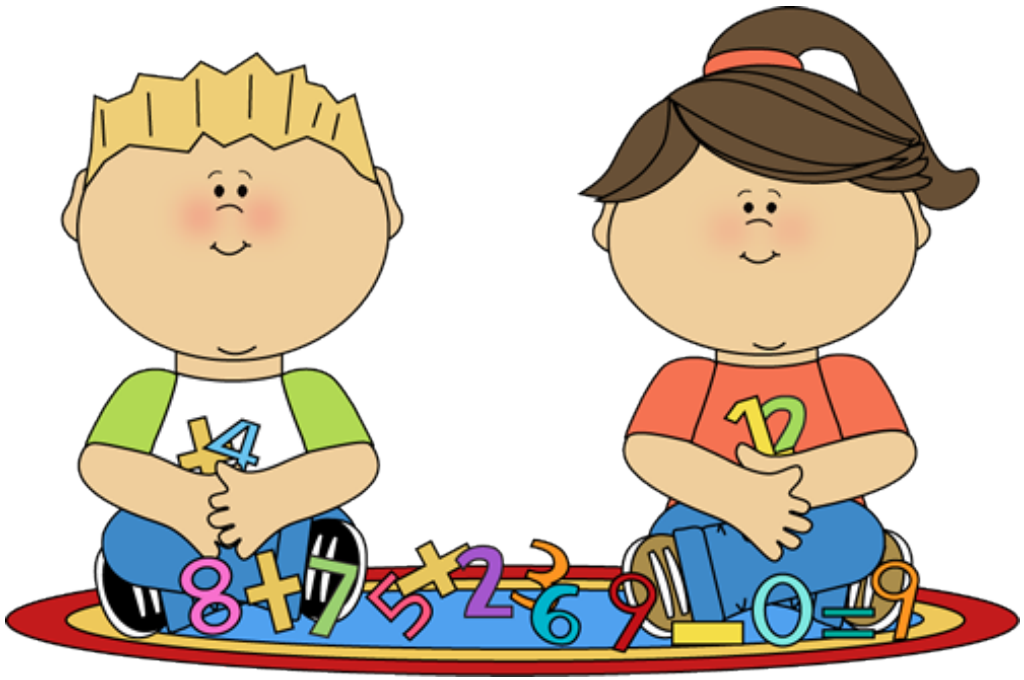
## Ordre du groupe $\mathbb{Z}_n^\star$

- ordre $|\mathbb{Z}_n^\star| = \varphi(n) \quad \Rightarrow \quad a^{\varphi(n)} = 1 \pmod{n}$
- ordre $|\mathbb{Z}_p^\star| = \varphi(p) = p - 1 \Rightarrow \quad a^{p-1} = 1 \pmod{n}$

## Règles

- Dans une exponentiation modulaire (modulo un entier $M$), les exposants doivent être pris modulo $\varphi(M)$.
- Effectuer les réduction modulaires au fur et à mesure.

# Problèmes difficiles



## Factorisation

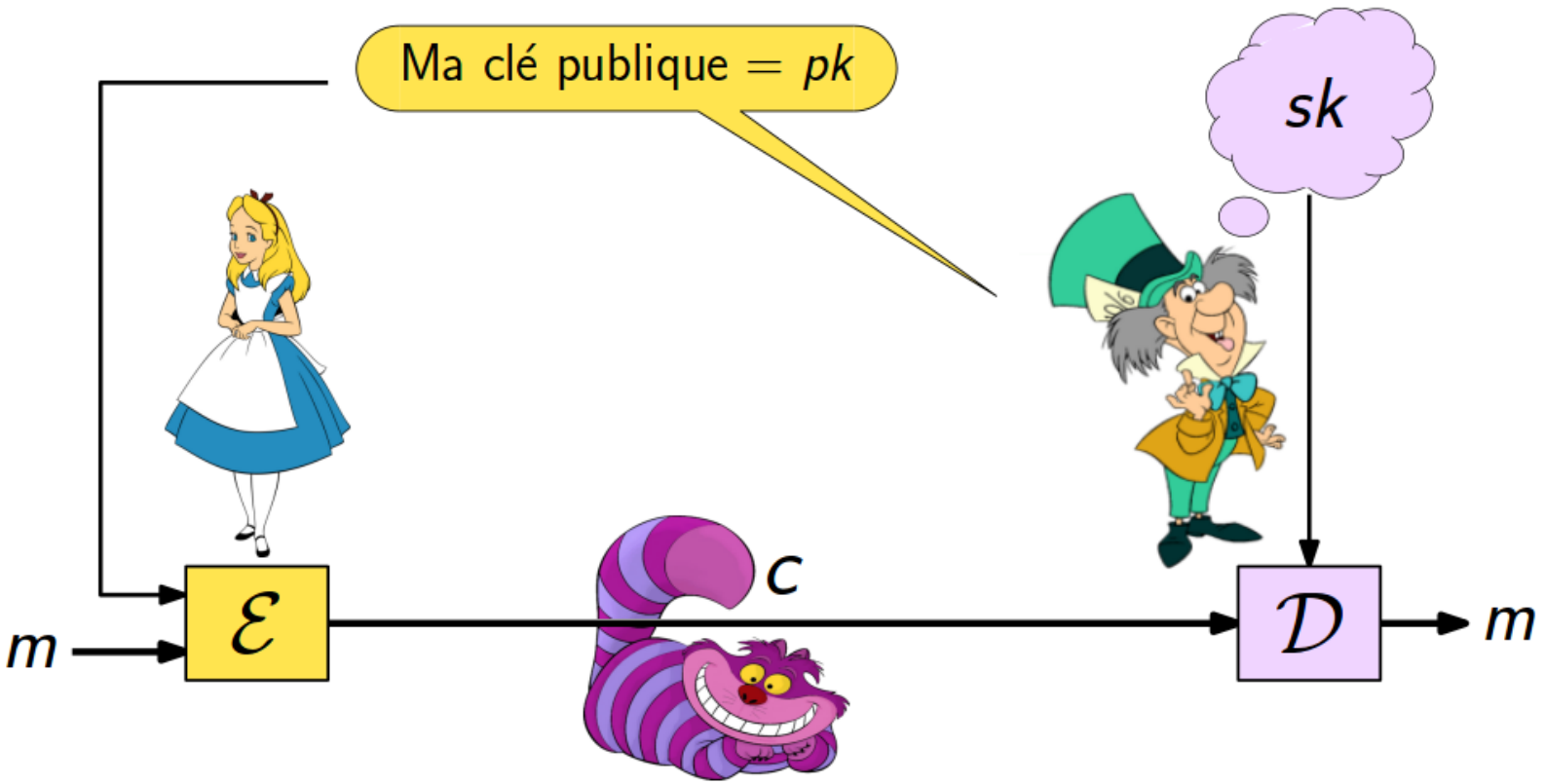
- $(p, q) \to p \cdot q$ facile
- $n = p \cdot q \to (p, q)$ difficile

# Méthodes connues = exponentielles

## Algorithmes de factorisation

- **Divisions successives** $\rightarrow \mathcal{O}(\sqrt{n})$
- **Algorithme de Fermat** : trouver $n = a^2 - b^2 \rightarrow \mathcal{O}(n^{1/3})$
  *cas facil* : $p - q$ petit
- **Méthode de Gauss** : trouver des résidus quadratiques mod $n$
  *cas facil* : $\varphi(n)$ connu
- **Algorithme $p-1$ de Pollard** $\rightarrow \mathcal{O}(p \log(p))$
  *cas facil* : $p-1$ et $q-1$ ont des petits facteurs premiers
- **Algorithme Williams**
  *cas facil* : $p+1$ et $q+1$ ont des petits facteurs premiers
- **Algorithmes sous-exponentiels**
  crible quadratique, courbes elliptiques, crible algébrique

# Chiffrement à clé publique



# Chiffrement à clé publique

## Protocole

• **Algorithme de génération des clés** $\mathcal{KG}(\ell) = (\mathsf{pk}, \mathsf{sk})$
à partir d'un paramètre de sécurité, il produit une paire de clés

• **Algorithme de chiffrement** $\mathcal{E}(\mathsf{pk}, m) = c$
produit le chiffré d'un message $m$, par la clé publique

• **Algorithme de déchiffrement** $\mathcal{D}(\mathsf{sk}, c) = m$
utilise la clé sécrete/privée sk pour retrouver $m$ à partir de $c$

# Protocole RSA

## RSA - Génération des clés

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

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

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

## clé publique

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

## clé secrète

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

# Protocole RSA

## RSA - Chiffrement

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

# Protocole RSA

## RSA - Chiffrement

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

## RSA - Déchiffrement

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

# Protocole RSA

## RSA - Chiffrement

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

## RSA - Déchiffrement

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

## Vérification

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

(Théorème d'Euler)

# La construction de RSA

## Construire RSA

Pour construire et utiliser un code RSA on a besoin de savoir faire rapidement (en temps polynomial) les opérations suivantes :

- construire des grands nombres premiers
- calculer des PGCDs
- faire de l'arithmétique modulaire (addition, multiplication, exponentiation)

# Sélection des paramètres RSA

## RSA - Choix des paramètres

Comment choisir les premiers $p$ et $q$ pour le module RSA :

- Choisir $p$ et $q$ de la même taille
- Choisir $p$ et $q$ au hasard pour éviter que $p - q$ ne soit trop petit (Algorithme de Fermat)
- Choisir $p$ et $q$ des nombres premiers forts :
  - $p - 1$ et $q - 1$ ont un grand facteur premier (Algorithme $p - 1$ de Pollard)
  - $p + 1$ et $q + 1$ ont un grand facteur premier (Algorithme Williams)

*$grand$=200 bits au moins

# Efficacité de RSA

## RSA - coût

Le coût est celui d'une exponentiation modulaire :

Chiffrement : $\mathcal{E}(\mathrm{pk} = (e, n), M) = M^e \pmod{n}$

- $3|e|/2$ multiplications
- si $|n| = |e|$ coût total $1.5 \log^3 n$

# Efficacité de RSA

## RSA - coût

Le coût est celui d'une exponentiation modulaire :

Chiffrement : $\mathcal{E}(\mathsf{pk} = (e, n), M) = M^e \pmod{\mathrm{n}}$

- $3|e|/2$ multiplications
- si $|n| = |e|$ coût total $1.5 \log^3 n$

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

- $3|p|/2$ multiplications mod $p$ $+ 3|q|/2$ multiplications mod $q$
- $\approx 3|p|$ multiplications mod $p$
- $\approx 3|n|/8$ multiplications mod $p$

# Propriétés du chiffrement

## RSA = Homomorphisme

Le chiffré d'un produit $M_1 M_2$ est égal au produit des chiffrés

$$C_1 = M_1^e \pmod{n}$$

$$C_2 = M_2^e \pmod{n}$$

$$C_1 C_2 = M_1^e M_2^e = (M_1 M_2)^e$$

Intéressant pour certains scénarios
Mais aussi nuisible à la sécurité ...

## Attaque à chiffré choisi

1. Soit $C = M^e$ un message chiffré
2. On fabrique $C' = A^e M^e$ le chiffré du message $A \cdot M$
3. Le déchiffrement de $C'$ fournit celui de $C$

# Propriétés du chiffrement

## RSA = Déterministe

**Chiffrement déterministe** = si l'on chiffre plusieurs fois le même message, on obtient le même chiffré

Méthodes pour éviter le chiffrement par substitution :

- couper le message en grands blocs
- modifier la taille de blocs à chaque fois
- randomiser le chiffrement RSA

# Sécurité de RSA

## RSA - Sécurité

RSA est considéré sécure :

- impossible à déchiffrer sans connaitre l'exposant $d$ de la clé secrète
- trouver la clé secrète $d$ est equivalent à factoriser $n = pq$
- pas d'algorithme polynomial en temps en fonction de la taille des données (la taille des nombres $n$ et $e$) pour factoriser $n$
- meilleur algorithme connu est sous-exponentiel (crible algébrique) :

$$\mathcal{O}\left(e^{1.92(\ln n)^{1/3}(\ln \ln n)^{2/3}}\right)$$

# Conseils d'utilisation du RSA

## RSA - Précautions

Il y a de nombreuses manières de **mal utiliser** RSA et d'ouvrir des failles de sécurité !

- Ne jamais utiliser de valeur $n$ trop petite
- Ne jamais utiliser d'exposant $e$ trop petit
- N'utiliser que des clés fortes
  ($p-1$ et $q-1$ ont un grand facteur premier)
- Ne pas chiffrer de blocs trop courts
- Ne pas utiliser de $n$ communs à plusieurs clés

# Attaques RSA

## RSA - Attaques mathématiques

- factoriser $n = pq$ et par conséquent trouver $\varphi$(n)

et puis $d$

- déterminer $\varphi$(n) directement et trouver $d$
- trouver $d$ directement (si petit)
- attaques "broadcast"
- attaques sur modulo $n$ commun
- attaques de synchronisation

(sur le fonctionnement du déchiffrement)

# Sécurité de RSA

## RSA - Sécurité

Pour évaluer et tester la sécurité du RSA :

- évaluer la rapidité des algorithmes de factorisations de grands nombres entiers
- démontrer que la clef secrète de déchiffrement $d$ ne peut pas être obtenue sans factoriser $n = pq$
- montrer que on ne peut pas déchiffrer un message sans la clé secrète $d$

# Problèmes difficiles

## Factorisation

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

- $x \to x^e \mod n$ facile
- $y = x^e \to x \mod n$ difficile

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

# Problèmes difficiles

## Factorisation

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

- $x \to x^e \mod n$ facile
- $y = x^e \to x \mod n$ difficile

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

# Difficulté de RSA

## Réduction

Si on connaît la factorisation, on casse RSA :

**RSA se réduit à la factorisation !**

Le contraire est peut-être faux !

- calculer des racines $e$-ièmes sans factoriser ? ? ?

# Difficulté de RSA

## Réduction

Si on connaît la factorisation, on casse RSA :

**RSA se réduit à la factorisation !**

Le contraire est peut-être faux !

- calculer des racines $e$-ièmes sans factoriser ? ? ?

## En pratique

La factorisation est la seule méthode **connue** pour casser RSA.

# Protocole ElGamal

## ElGamal - Génération des clés

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

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

## clé publique

- $p$ et $g$ : paramètres publiques
- $\mathsf{pk} = y = g^x$ : clé publique

## clé secrète

- $\mathsf{sk} = x$
  exposant sécret

# Protocole ElGamal

## ElGamal - Chiffrement

$\mathcal{E}(\text{pk} = y, M) = (C, D)$

Pour un aléa $r$ on calcule une paire $(C, D)$ (le chiffré de $M$)

$$C = g^r \pmod{\text{p}}$$

$$D = M \cdot y^r \pmod{\text{p}}$$

## ElGamal - Déchiffrement

$\mathcal{D}(\text{sk} = x, (C, D)) = D \cdot C^{-x} \pmod{\text{p}}$.

# Protocole ElGamal

## ElGamal - Chiffrement

$\mathcal{E}(\text{pk} = y, M) = (C, D)$

Pour un aléa $r$ on calcule une paire $(C, D)$ (le chiffré de $M$)

$$C = g^r \pmod{\text{p}}$$

$$D = M \cdot y^r \pmod{\text{p}}$$

## ElGamal - Déchiffrement

$\mathcal{D}(\text{sk} = x, (C, D)) = D \cdot C^{-x} \pmod{\text{p}}$.

## Vérification

$$D \cdot C^{-x} = M\ y^r\,(g^r)^{-x} = M\,(g^x)^r\,(g^r)^{-x} = M \pmod{p}$$

# Emploi de ElGamal

## Avantages

- Tous les utilisateurs peuvent utiliser le même groupe $\mathbb{Z}_p^\star$ et le même $g$
- Possibilité de techniques d'exponentiation rapide
- Chiffrement randomisé nativement
- Meilleure sécurité basique

## Inconvénients

- Augmentation de la taille du chiffré
- Deux fois plus lent que RSA

# Efficacité de ElGamal

## ElGamal - coût

Le coût est celui de deux exponentiations modulaires :

- El Gamal est 2 fois plus lent que RSA.
- La taille des données chiffrées représente 2 fois celle des données en clair.

# Propriétés multiplicatives

## ElGamal -Attaque à chiffré choisi

Si $(C, D)$ est un chiffré de $M$, pour tout $A$, le couple $(C, A \cdot D)$ chiffre $A \cdot M$ :

$$C = g^r \pmod{p}$$
$$A \cdot D = A \cdot M \cdot y^r \pmod{p}$$

# Attaques ElGamal

## ElGamal avec le même aléa

**Ne jamais utiliser deux fois le même aléa !**

Si $M_1$ et $M_2$ sont chiffrés avec le même aléa $r$, alors :

$$(C_1, D_1) = (g^r, y^r M_1) \pmod{p}$$
$$(C_2, D_2) = (g^r, y^r M_2) \pmod{p}$$

d'où :

$$\frac{M_1}{M_2} = \frac{D_1}{D_2}$$

# Problème difficile

## Logarithme discret (DLOG)

**Définition** Soit $\mathbb{G}$ un groupe multiplicatif, $g \in \mathbb{G}$ et $y \in \langle g \rangle$ :

$$\log_g(y) = x \quad \text{où} \quad g^x = y$$

# Difficulté de ElGamal

## Réduction

La recherche de la clé privée $x$ à partir de la clé publique $y = g^x$ est équivalente au problème du logarithme discret (DLOG).

**ElGamal se réduit au logarithme discret !**

# Réduction de ElGamal

## En pratique

- Si le problème du logarithme est résolu polynomialement, alors El Gamal sera cassé.
- Le contraire est peut-être faux !
- Rien ne prouve qu'il n'est pas cassable par un autre moyen.

Le log discret est la seule méthode **connue** pour casser ElGamal.

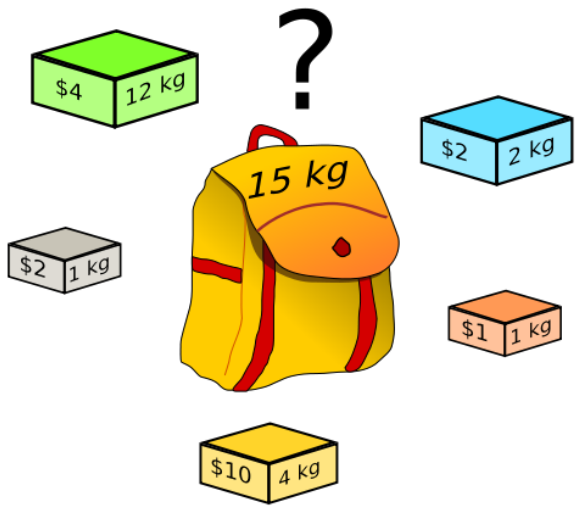
# Autres problèmes difficiles

## Problème du sac à dos (KP)

**Knapsack problem** Il modélise une situation analogue au remplissage d'un sac à dos :

- poids du sac fixé
- ensemble d'objets donné
- chacun a un poids et une valeur

Les objets mis dans le sac à dos doivent maximiser la valeur totale, sans dépasser le poids maximum.



# Cryptosystème Merkle-Hellman

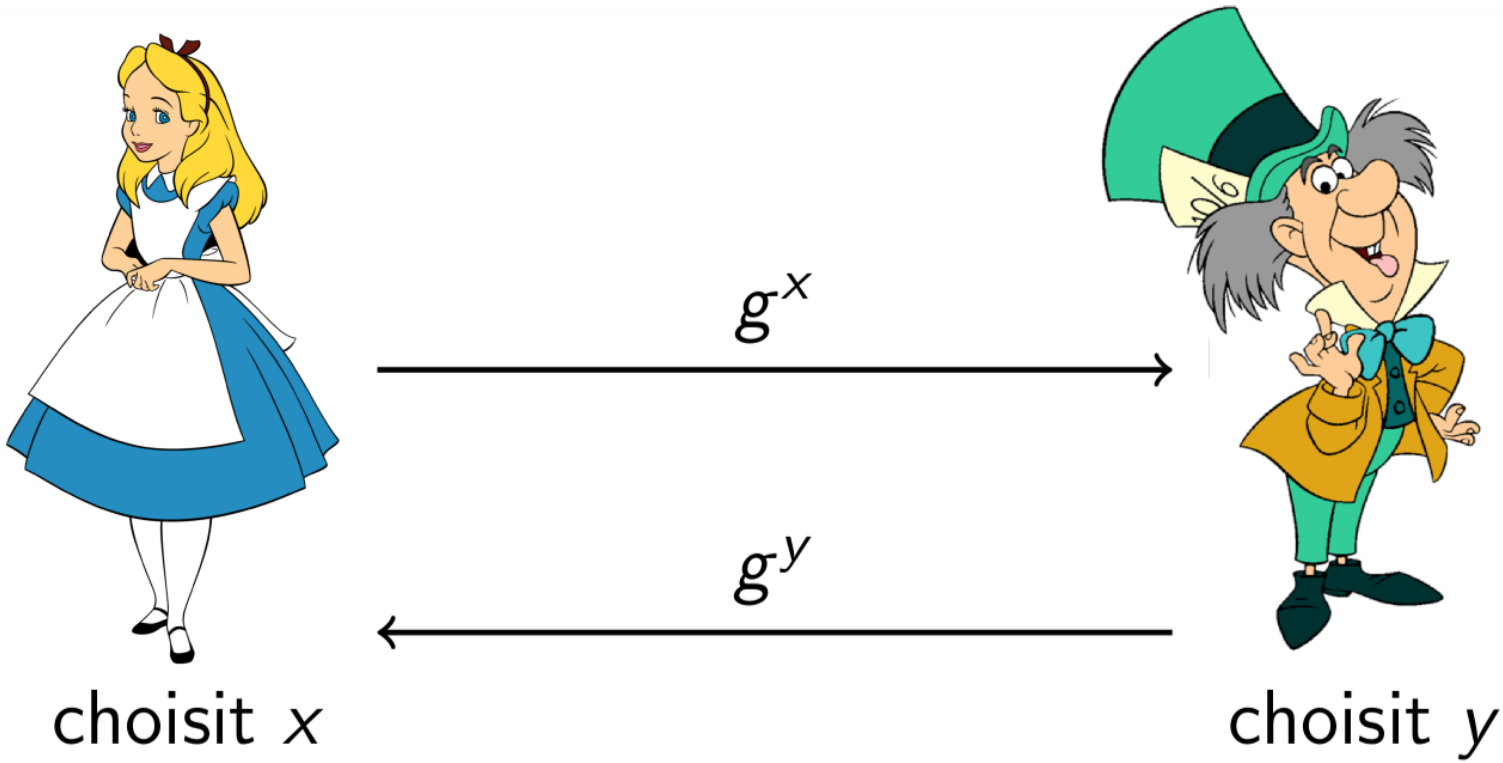
Premier algorithme à clé publique (1978)

## Principe

- un très grand nombre d'objets, chacun de poids différents
- la liste de tous les objets est publique
- le message est chiffré en choisissant secrètement un certain nombre d'objets qui sont mis dans le sac à dos
- le poids total du sac à dos est public
- le contenu du sac à dos est secret

L'algorithme à clé publique est basé sur le fait qu'il est difficile de trouver la listes des objets à partir de leur poids total

# Echange de clé




**clé commune** $g^{xy}$

# Problèmes difficiles

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

**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 $\mathbb{G}$

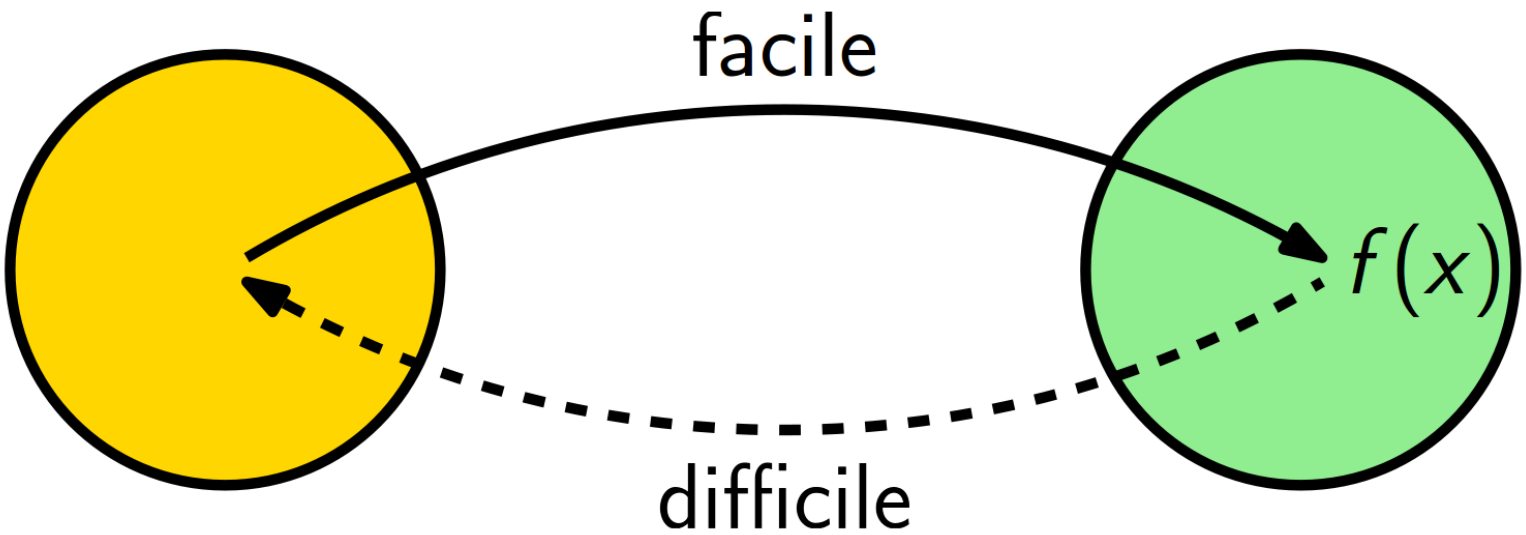
Décider si $C = g^{ab}$

# Réduction de Diffie-Hellman

## Attaquer Diffie-Hellman

- Si le problème **CDH** est résolu, alors l'attaquant peut calculer une clé Diffie-Hellman
- Si le problème **DDH** est résolu, alors l'attaquant peut distinguer entre une clé valide et une clé fausse

# Fonctions à sens unique



# 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 $B = g^b$,

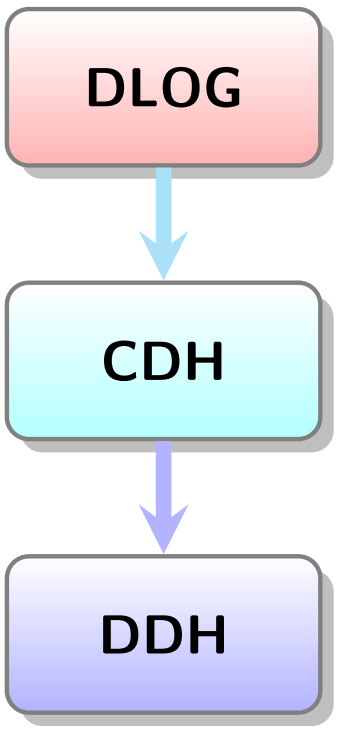
Calculer $C = CDH(A, B) = g^{ab}$

## Décider si Diffie-Hellman (DDH)

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

Décider si $C = g^{ab}$

# Hiérarchie




## CDH < DLOG

Etant donnés $g$, $A = g^a$ et $B = g^b$,

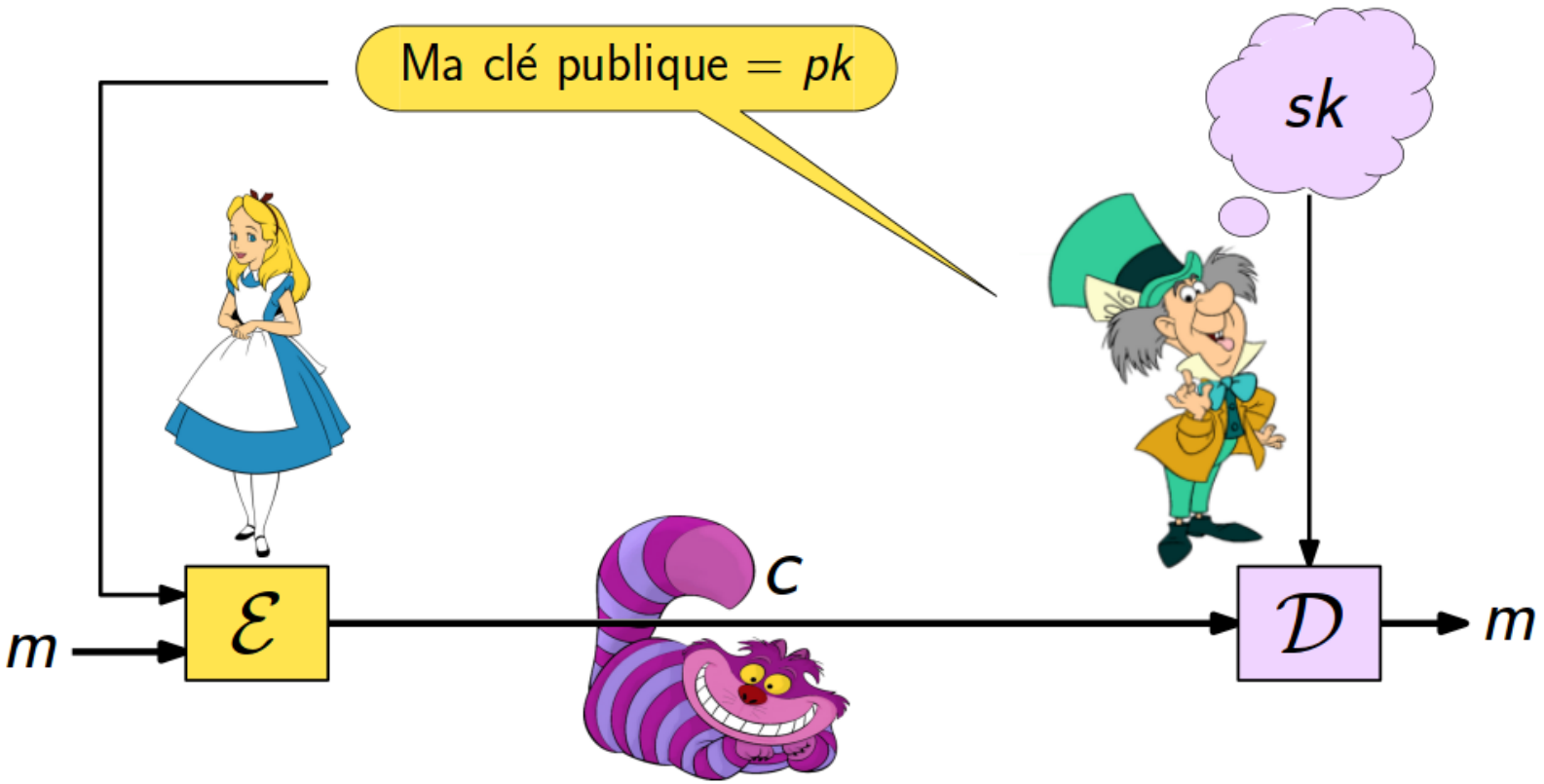
- on calcule $b$ =DLOG($B$)
- on trouve $C = A^b = g^{ab}$

## DDH < CDH

Etant donnés $g$, $A = g^a$, $B = g^b$ et $C = g^c$

- on calcule $CDH(A, B) = g^{ab}$
- on compare avec $C$

# Analise de sécurité



# Analise de sécurité


**Sécurité PARFAITE ?**

Au sens de Shannon (théorie de l'information), la sécurité parfaite est impossible en clé publique !

## Pourquoi ?

La connaissance de la clé publique et du chiffré biaise la distribution de probabilité du message clair

## Quel espoir ?

Un attaquant avec une capacité de calcul polynomiale peut-il exploiter ce biais et accéder à une information ?

# Prouver la sécurité

## Modèle de sécurité

Pour analyser la sécurité d'un cryptosystème :

1. spécifier les buts de l'attaquant
2. 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, . . .)

# L'adversaire

## L'attaquant

- le plus **intelligent** possible
  - il peut faire toutes les opérations qu'il souhaite
- il dispose d'un **temps limité**
  - on ne considère pas les attaques faisables en $2^{60}$ ans
  - force brute : énumérer toutes les clés – temps $2^{\text{taille(clés)}}$

# Modéliser l'adversaire

## L'algorithme de l'attaquant

Adversaire modélisé par une **machine de Turing** :

- **probabiliste :**
  - il génére des clés
  - il tire au sort certaines étapes de son comportement
- **polynomiale** en la taille des clés :
  - il a un temps *raisonnable* d'exécution

# Représenter l'attaquant

**L'attaquant $\mathcal{A}$** est un algorithme (machine de Turing) probabiliste et polynomiale

## Cryptographie

" $\forall \mathcal{A}$ securité"
- l'attaquant est une boîte noire
(on ignore son code)

## Cryptanalyse

"$\exists \mathcal{A}$ victorieux"
- on cherche à exhiber un attaquant
(on construit son code)

# Preuves de sécurité

## Preuve par réduction

### Principe

1. **Hypothèse** :
   Un problème $P$ diffcile = il n'y a pas d'algorithme polynomial
   **P= RSA, DLOG, DDH, CDH...**
2. **Réduction** :
   - si $\mathcal{A}$ un adversaire (polynomial) casse le schéma de chiffrement,
   - alors $\mathcal{A}$ peut être utilisé pour résoudre $P$ en temps polynomial (ce qui est considéré impossible)
3. **Résultat de sécurité** : il n'existe pas d'adversaire polynomial

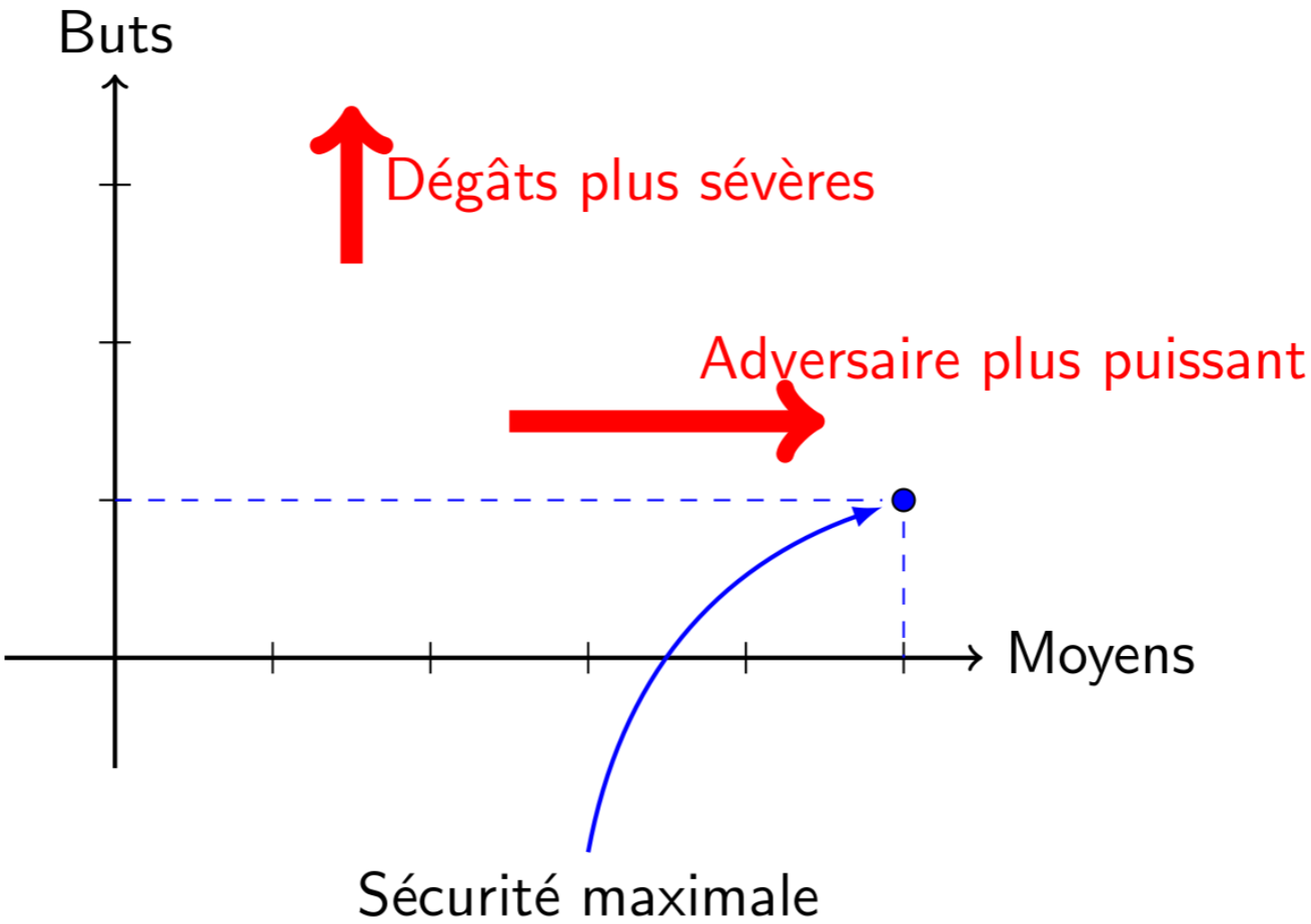
# Modèle de sécurité

## Deux axes d'analyse : Buts – Moyens

Définir la sécurité d'un algorithme de chiffrement :

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

# Deux axes d'analyse



# Buts

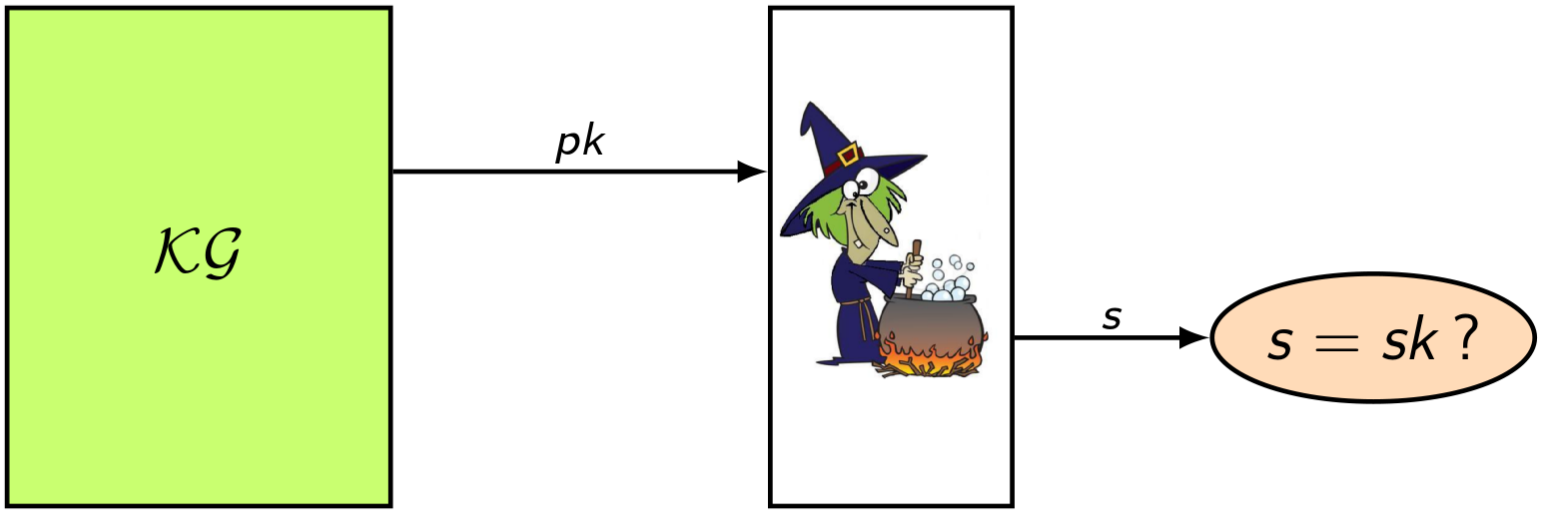
## Buts de l'attaquant

1. **BRK – Cassage total** : retrouver la clé de déchiffrement (equivalent 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

# Rétrouver la clé secrète

## BRK – Break the system

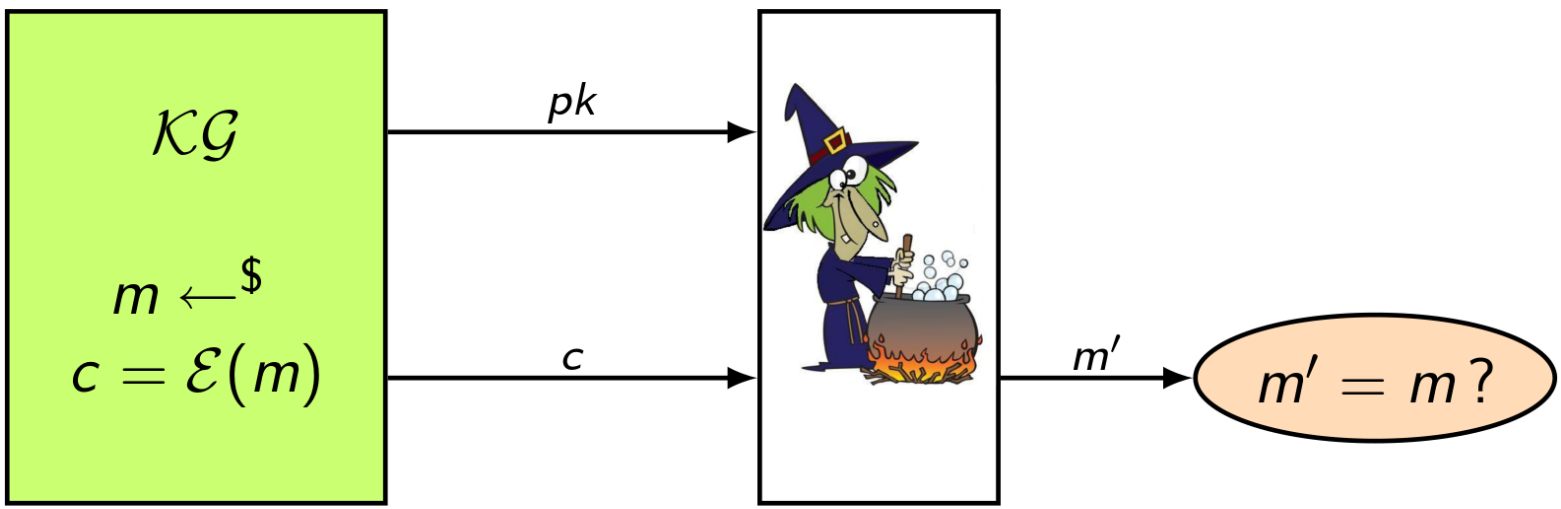
« Cassage total »



# Inverser le chiffrement

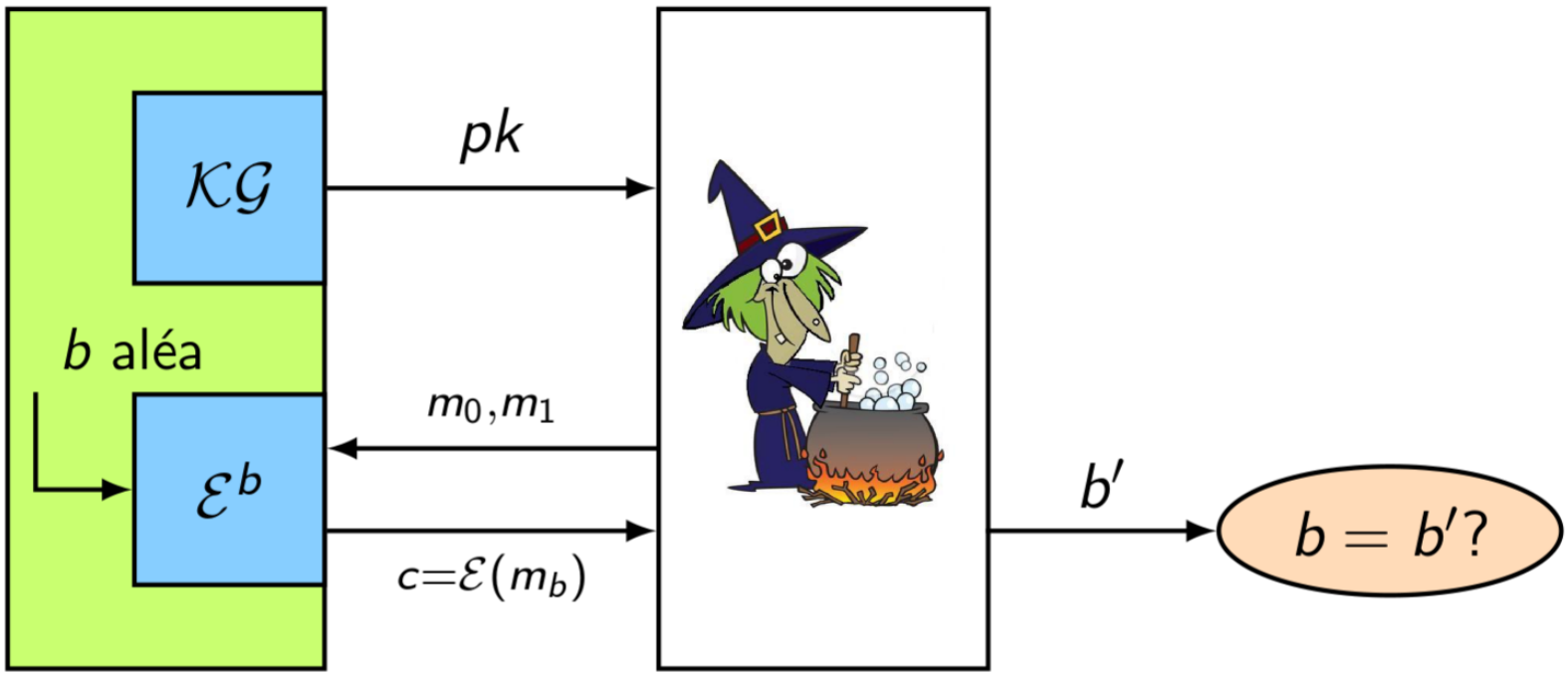
## OW – One-Wayness

Déchiffrer un message arbitraire



# Obtenir un bit d'information

## IND – Indistinguishability

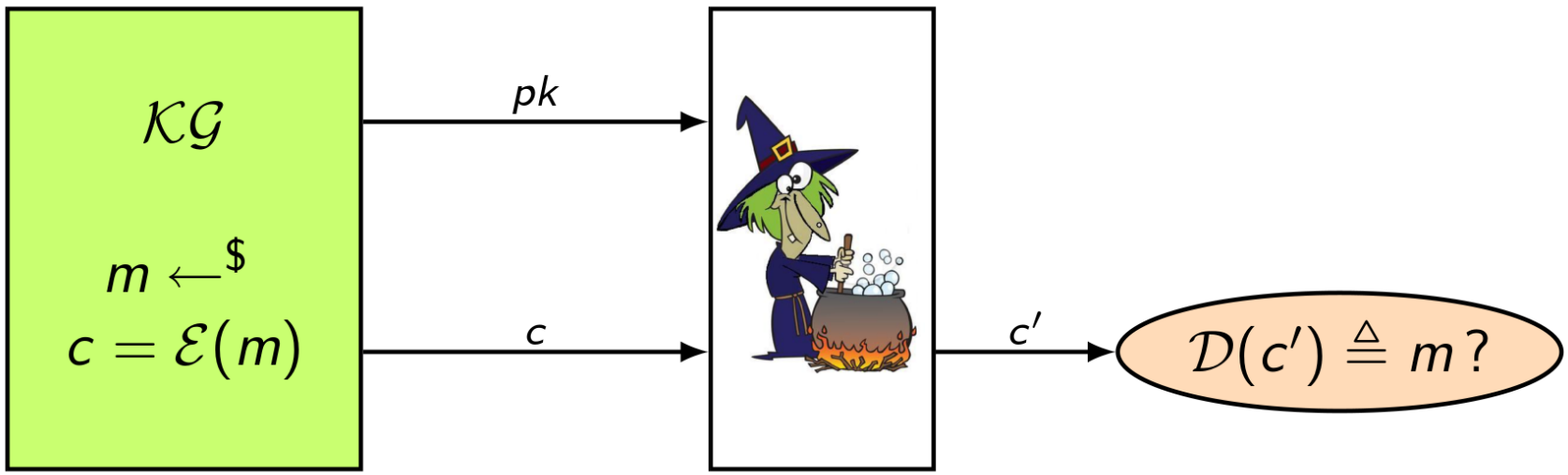



C'est $\mathcal{A}$ qui choisit les messages $m_0$ et $m_1$ !

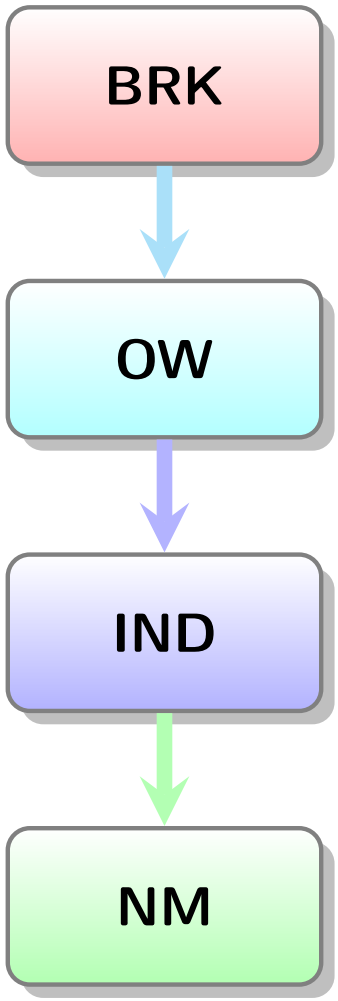
# Modiffier un chiffré

## NM – Non-Malleability

Fabriquer deux chiffrés « liés » (*plaintext-dependent*)



# Hiérarchie de sécurité



# Moyens

## Oracles

L'adversaire a accès à des oracles :

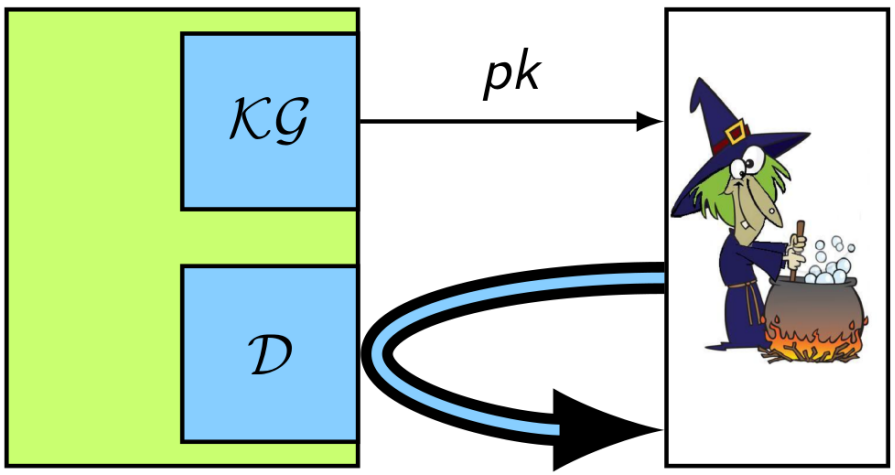
- **chiffrement** de tous les messages de son choix
- **déchiffrement** de tous les messages de son choix

## 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)

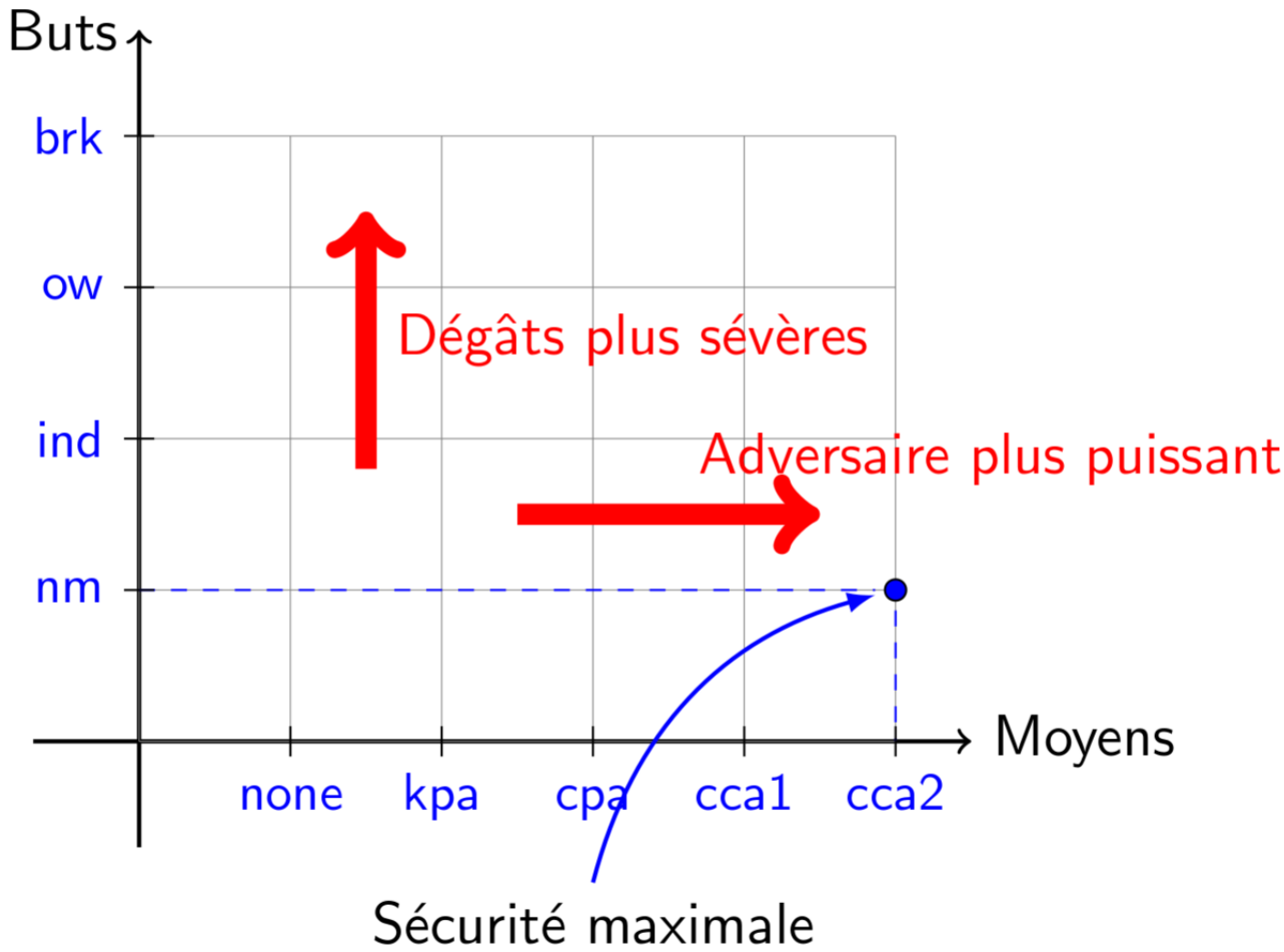
# CCA – Attaques à chiffrés choisis

Attaques les plus sévères (*Chosen-Ciphertext Attacks, CCA*)




$\mathcal{A}$ observe des déchiffrements et espère obtenir de l'information

# Les axes d'analyse



# Modèle de sécurité

## Attaques possibles

**Un modèle d'attaque = un but + des moyens**

Exemple :

- Sécurité **OW-CPA** : chiffrement à sens unique sous une attaque à clair choisi

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

# Analyse de sécurité pour RSA

**On suppose RSA bien utilisé**

## Sécurité

- **BRK-CPA** : clé secrète —> **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 $2m$

# Analyse de sécurité pour ElGamal

## Sécurité

- **BRK-CPA** : clé secrète —> **DLOG**
- **OW-CPA** : extraction de $M = D \cdot C^{-x}$ à partir de $(C, D)$ probléme 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 $2M$