Cours: Sécurité Informatique
2022-2023

Chapitre 02 : Initiation à la cryptographie

03 - Principes des crypto-systèmes Asymétriques : Algorithme RSA

03 - Principes des crypto-systèmes Asymétriques : Algorithme RSA

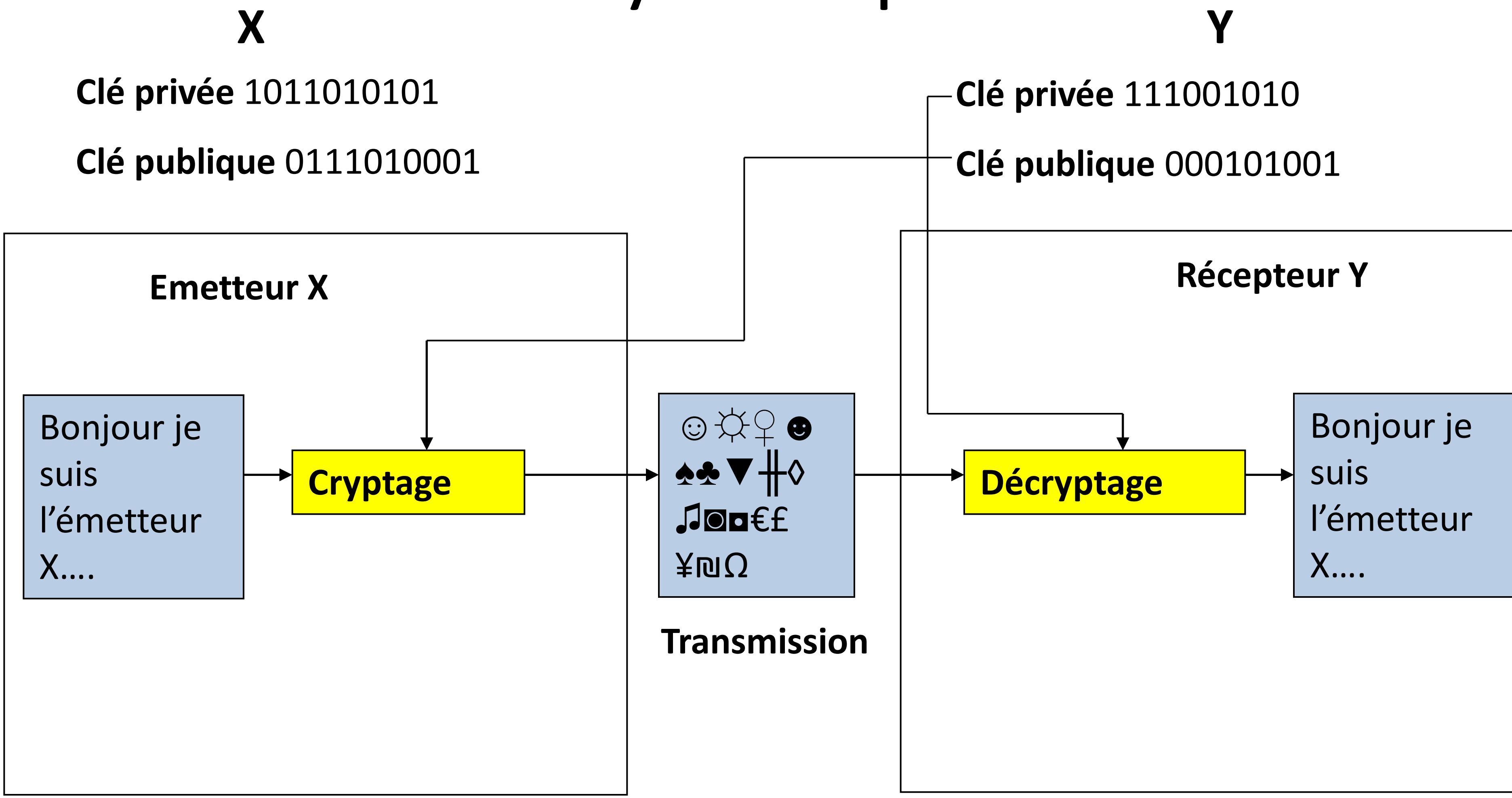
Introduction

- Les algorithmes de cryptographie **symétrique** utilisent une **seule clé** pour le chiffrement et le déchiffrement, la clé doit être connue **des deux cotées**, Emetteur et Récepteur.
- La transmission sécurisée **de la clé** reste donc un problème et une source de **vulnérabilités.**
- La connaissance de la clé par un pirate lui permet de prendre **l'identité** de l'émetteur réel (absence de contrôle d'identité).
- Pour faire face à ce genre de problèmes, la cryptographie **Asymétrique (à clé publique)** propose la notion de clé privée et de clé publique. Elle est considérée comme le plus grand **avancement** durant 3000 ans d'histoire de la cryptologie.

Introduction

- Dans ce type de chiffrement, deux types de clés sont utilisées :
  - Une **clé publique** connue **par tout le monde**, utilisé pour **crypter** ou pour **vérifier** la signature.
  - Une **clé privée** connue **seulement par sont propriétaire**, utilisé pour **décrypté** ou pour **signé** un message.
- Le principe de la cryptographie Asymétrique peut être exploité pour réaliser deux tâches : le chiffrement des messages et la signature/Authentification des messages.
- Le chiffrement requiert beaucoup d'opérations et n'est pas recommandé pour de grandes quantités d'informations:
  - EX: RSA est 1500 plus lent que DES.

Schéma général de **chiffrement** asymétrique



X envoi un message crypté à Y

Principes de base

- Tout algorithme à clé publique est basé sur l'existence d'une **fonction à trappe** (Trapdoor).

Définition : Fonction à sens unique

- Soit S , S' deux ensembles. Une fonction à sens unique $f : S \rightarrow S'$ satisfait:
  1) Pour $x \in S$ le calcul de f(x) se fait facilement (au sens de la complexité de calcul).
  2) Pour $y \in S'$ il est impossible (au sens de la complexité des calculs) de trouver x tel que $f(x) = y$.
- On dira qu'un calcul se fait facilement, s'il se fait en temps polynomial.

Principes de base

Définition : Fonction à trappe

- Soit S, S’ deux ensembles. Une fonction à trappe $f_k : S \rightarrow S'$ associée à une donnée k est telle que :
  1) **Sans la connaissance** de k la fonction f_k est une fonction à **sens unique**.
  2) **Avec la connaissance** de k la fonction f_k^{-1} **se calcule facilement** (i.e. se fait en temps polynomial).
- Le paramètre k est appelé Trappe (**Trapdoor**) de la fonction f.
- Problèmes mathématiques réputés difficiles :
  1) Factoriser un nombre composé de deux grands nombres premiers : **RSA**.
  2) Extraire le logarithme discret dans certains groupes : **ElGamal, Diffie-Hellman**.

Principes de base

- Un chiffrement à clé publique se compose de trois algorithmes:

Algorithme de génération des clés

$\mathcal{KG}(\ell) = (pk, sk)$: à partir d'un paramètre de sécurité, il produit un couple (clé publique, clé privée)

Algorithme de chiffrement

$\mathcal{E}(m, pk) = c$: utilise la clé publique pour chiffrer un message m

Algorithme de déchiffrement

$\mathcal{D}(c, sk) = m$: utilise la clé privée pour remonter à m

Algorithmes de chiffrement

- Plusieurs algorithmes de chiffrement asymétriques existent, chacun repose sur un **problème mathématique** spéciale avec une fonction **à trappe** particulière :
  - **RSA** : problème de factorisation en facteurs premiers.
  - **El-Gamal** : Problème de logarithme discret.
  - **Diffie-Hellman** (échange de clés): Problème de logarithme discret.
- Chacun de ces algorithmes permet de réaliser le schéma de communication à clé publique/privé.
- Dans ce qui suit, on essayera d'exposer les principes de base de chacun des trois algorithmes.

Rappels mathématiques

- Si a et n sont **premiers entre eux** (**pgcd(a,n)=1**) alors $x=a^{-1}$ existe.
  - x est l'inverse de a modulo n qui vérifié $a.x \equiv 1 \bmod n$
- Une méthode récursive permet de calculer l'inverse :

```
Function inverse (a, b, c, d : integer) : integer;
Begin
    If b=1 retourner c
    else retourner inverse(b, a mod b, d-c(a div b), c);
End;
```

- L'appel se fait au début par inverse(m,a,1,0)

Rappels mathématiques

- Le nombre des entiers a<n qui vérifient pgcd(a,n)=1 est donné par la **fonction d'Euler**:

$$\varphi(n)=|\{a\in Z_n : pgcd(a,n)=1 \}|$$

Exemple :

$\varphi(8)$ = 4 car parmi les nombres de 1 à 8, seuls les quatre nombres 1, 3, 5 et 7 sont premiers avec 8,

- Si p est premier alors : $\varphi(p)=p-1$
- Si pgcd(p,q)=1 alors $\boldsymbol{\varphi(p.q) = \varphi(p).\varphi(q)}$

Rappels mathématiques

le petit théorème de Fermat :

- Si p est un nombre premier et si a est un entier non divisible par p (n'est pas multiple de p) alors:

$$a^p \equiv a \bmod p \text{ ou bien } a^{p-1} \equiv 1 \bmod p$$

Exemples :

- $5^2 \equiv 1 \bmod 3$
- $7^1 \equiv 1 \bmod 2$
- $2^4 \equiv 1 \bmod 5$

- Nous allons voir une version améliorée de ce théorème dans le cas qui nous intéresse…

Rappels mathématiques

le petit théorème de Fermat amélioré :

- Soient p et q deux **nombres premiers distincts** et soit $n = p.q$
- Pour tout $a \in Z$ tel que **pgcd(a, n) = 1** alors :

$$a^{(p-1)(q-1)} \equiv 1 \bmod n$$

- D'après le **théorème d'Euler :**

$$a^{\varphi(n)} \equiv 1 \bmod n.$$

Rappels mathématiques

- Pour chaque « e » premier avec $\varphi(n)$, il existe un nombre « d » tel que $e.d \equiv 1 \pmod{\varphi(n)}$ ($d=e^{-1}$: modulo $\varphi(n)$), donc, $\exists\, k \in Z : e.d = 1 + k.\varphi(n)$.

Conséquence :

$\forall m \in Z_n^*$: $\mathbf{(m^e)^d\,[n]} = m^{e.d}\,[n] = m^{1+k.\varphi(n)}[n] = m.m^{k.\varphi(n)}\,[n] =$

$= m[n].\ m^{k.\varphi(n)}[n] = m[n].\ (m^{\varphi(n)})^k[n] = m[n].\ (1)^k[n] = \mathbf{m[n]} = \mathbf{m}$

Rappels mathématiques

L'exponentiation rapide:

- Nous aurons besoin de calculer rapidement des puissances modulo n. Pour cela il existe une méthode beaucoup plus efficace que de calculer d'abord a^k puis de le réduire modulo n.
- Il faut garder à l'esprit que les entiers que l'on va manipuler ont des dizaines voir des centaines de chiffres.
- Pour calculer a^k (mod n):
  - $a^k \pmod{n} \equiv (a^2)^{k/2} \pmod{n}$ si k est pair.
  - $a^k \pmod{n} \equiv a.(a^2)^{(k-1)/2} \pmod{n}$ si k est impair.

Rappels mathématiques

L'exponentiation rapide:

- Par exemple pour calculer 5^{11} (mod 14) :

$$5^{11} \equiv 5 \times (5^2)^5 \pmod{14}$$

$$5^{11} \equiv 5 \times 11^5 \pmod{14}$$

$$5^{11} \equiv 5 \times 11 \times (11^2)^2 \pmod{14}$$

$$5^{11} \equiv 5 \times 11 \times (9)^2 \pmod{14}$$

$$5^{11} \equiv 5 \times 11 \times 11 \pmod{14}$$

$$5^{11} \equiv 3 \pmod{14}$$

Nous obtenons donc un calcul de 5^{11} (mod 14) en 5 opérations au lieu de 10 si on avait fait $5 \times 5 \times 5 \times \ldots$.

Algorithmes de chiffrement : LE RSA

- Le **RSA** est l'algorithme de cryptage à **clé publique** le plus connue. Initialement proposé par **R**ivest, **S**hamir et **A**dlmen en 1977 à l'université MIT.
- Il utilise des **grands** nombres (sur 1024 bit!)
- La sécurité dépend de la **difficulté de factoriser** un grand nombre en facteurs premiers.
- La factorisation d'un grand nombre entier ne peut être effectuée pratiquement dans un **temps raisonnable**.

Algorithmes de chiffrement : LE RSA

- Pour crypter un message, on commence par le transformer en un –ou plusieurs– nombre. Les processus de chiffrement et déchiffrement font appel à plusieurs notions :
  - On choisit deux **nombres premiers** p et q que l'on garde secrets et on pose $n = p \times q$. Le principe étant que même connaissant n il est très difficile de retrouver p et q (qui sont des nombres ayant des centaines de chiffres).
  - La clé secrète et la clé publique se calculent à l'aide de l'**algorithme d'Euclide** et des **coefficients de Bézout**.
  - Les calculs de cryptage se feront **modulo** n.
  - Le déchiffrement fonctionne grâce à une variante du **petit théorème de Fermat**.

Algorithmes de chiffrement : LE RSA

- Les étapes de la méthode sont comme suite :

1) Génération des clés (publique et privé):

On choisie deux nombres **premiers** p et q. On pose N=p.q et φ(N)=(p-1).(q-1).

- On choisie un nombre e < N et **premier avec** φ(N). La **clé publique** est composée du couple **(N,e)**.
- Soit « d » le nombre qui vérifie **e.d≡1 (mod φ(N))** c.-à-d : $d=e^{-1}$ modulo φ(N).
- « d » donc peut être calculé en utilisent le théorème de Bézout : résoudre l'équation : e.d+u.φ(N)= 1
- **La clé privée** est donnée par **d**.

Algorithmes de chiffrement : LE RSA

2) Procédure de chiffrement :

Soit un message m à codé, représenté sous forme d'un nombre <N. Le message chiffré sera **$C = m^e \bmod N$**

- Chaque message est codé en nombre entier, le cryptage ne nécessite que la connaissance de N et e (**la clé publique**).

3) Procédure de déchiffrement :

Pour un message crypté C, le message original est calculé par: **$m=C^d \bmod N$**.

Preuve : $C^d \bmod N = (m^e)^d \bmod N = m^{e.d} \bmod N$

Puisque $e.d \equiv 1 (\bmod \varphi(N)) \Rightarrow e.d=1+k.\varphi(N)$ Donc :

$$C^d[N]=m^{1+k.\varphi(N)}[N]=m.(m^{\varphi(N)})^k[N]=m.(1)^k[N]=m[N]=m$$

$(m<N)$

Exemple : **LE RSA**

- Dans cet exemple, un émetteur qui veut envoyer un message secret au récepteur. Le processus se décompose ainsi :
  1) Le récepteur prépare une clé publique et une clé privée,
  2) L'émetteur utilise la clé publique du récepteur pour crypter son message,
  3) Le récepteur reçoit le message crypté et le déchiffre grâce à sa clé privée.

Exemple : LE RSA

1) Génération des clés (publique et privé):

– $p = 5$ et $q = 17$
– $n = p \times q = 85$
– $\varphi(n) = (p - 1)\times(q - 1) = 64$

- Vous noterez que le calcul de $\varphi(n)$ n'est possible que si la décomposition de n sous la forme $p \times q$ est connu. D'où le caractère secret de $\varphi(n)$ même si n est connu de tous.

Choix d'un exposant et calcul de son inverse:

– Le récepteur choisit par exemple $e = 5$ et on a bien $\mathrm{pgcd}(e, \varphi(n)) = \mathrm{pgcd}(5,64) = 1$,
– Le récepteur calcule d, l'inverse de e modulo $\varphi(n)$, alors $d = 13$.

Clé publique: $n = 85$ et $e = 5$

Clé privée: $d = 13$

Exemple : LE RSA

2) Chiffrement du message: L'émetteur veut envoyer un message secret au récepteur. Il se débrouille pour que son message soit un entier (découper son texte en bloc et à transformer chaque bloc en un entier).

Message : Le message est un entier m, tel que $0 < m < n$.

Soit $m = 10$.

Message chiffré: L'émetteur calcule à l'aide de l'algorithme d'exponentiation rapide, le message chiffré :

$$c \equiv m^e \pmod{n}$$

$$c \equiv 10^5 \pmod{85}$$

- On peut ici faire les calculs à la main :

$c \equiv 10^5 \equiv 10 \times 10^4 \equiv 10\times(10^2)^2 \equiv 10\times15^2 \equiv 40 \pmod{85}$

Le message chiffré est donc $c = 40$.

Exemple : LE RSA

3) Déchiffrement du message:

- Le récepteur reçoit le message c chiffré par L'émetteur, il le décrypte à l'aide de sa clé privée d, par l'opération:

$$m \equiv c^d \pmod{n}$$

$c^d \equiv (40)^{13} \pmod{85}$.

Calculons à la main $40^{13} \pmod{85}$:

$40^{13} \equiv 40 \times (40^2)^6 \pmod{85}$

$40^{13} \equiv 40 \times 70^6 \pmod{85}$

$40^{13} \equiv \equiv 40 \times (70^2)^3 \pmod{85}$

$40^{13} \equiv 40 \times 55^3 \equiv 40 \times 55 \times 55^2 \equiv 10 \pmod{85}$

Donc :

$c^d \equiv 40^{13} \equiv 10 \pmod{85}$ qui est bien le message m de L'émetteur.

Algorithmes de chiffrement : **LE RSA**

- La sécurité du RSA repose sur deux conjectures :

1) « Casser » RSA nécessite la **factorisation** du nombre N,
2) La factorisation est un problème difficile, car il n'existe pas d'algorithme suffisamment rapide. Les mathématiciens affirment qu'il n'existe pas d'algorithme de **complexité polynomiale** en temps qui donne les facteurs premiers d'un nombre quelconque.

- Taille de la clé RSA ≥ 768 bits pour un usage privé et ≥ 1024 à 2048 bits pour un usage sensible.
- Factorisé une clé de taille < à 256 bits est possible en quelques heures sur un PC.

Algorithmes de chiffrement : **LE RSA**

- La factorisation de N permet de déterminer p et q;
- La connaissance de p et q permet de déduire la clé privée à partir de la clé publique car e.d=1+k.(p-1)(q-1).
- La complexité de la factorisation croît exponentiellement avec la valeur du chiffre à factorisé.
- Ainsi les deux nombres premiers p et q choisis doivent être très grand.

Exemple :

p=1113954325148827987925490175477024844070922844843

q=1917481702524504439375786268230862180696934189293

p.q=21359870359209100823950227049996287970510953418264174064425241650085839577464450884050094308659999

Echange de clés : Diffie-Hellman

- L'**échange de clés Diffie-Hellman**, du nom de ses auteurs **Whitfield Diffie** et **Martin Hellman**, est une méthode par laquelle deux personnes émetteur et récepteur peuvent se **mettre d'accord** sur un nombre (qu'ils peuvent utiliser comme **clé** pour chiffrer la conversation) sans qu'une troisième personne C puisse **découvrir** le nombre, même en ayant écouté **tous leurs échanges**.
- Ce protocole est généralement utilisé pour **l'échange des clés** de chiffrement utilisés par des algorithmes de chiffrement **symétrique**.
- Il est basé sur le principe mathématique du **logarithme discret** dans un groupe cyclique (Z_p^* avec p premier).

Rappels mathématiques

- Soit p un entier **premier**. On dit qu'un entier g est un **générateur** modulo p si pour tout entier b $\in Z_p^*$ il existe un entier « a » tel que b = g^a (mod p).
- Pour tout entier premier p il existe au moins un générateur 0 < g < p.
- On appelle **Logarithme Discret (LD)** en base g de b, l'entier «a» qui vérifie b=g^a (mod p), noté souvent a=$\log_g$(b).
- **Problème (Logarithme discret):**

  Soit p un entier premier et g un générateur modulo p. Résoudre le problème du logarithme discret en base g d'un entier b, consiste à trouver« a » : **b = g^a (mod p)**

Echange de clés : Diffie-Hellman

Exemple :

p=97, donc on travaille dans Z_{97}^* ={1,….,96}, 5 est un générateur.

5^{32}(mod 97)=35 $\Rightarrow$ $\log_5 35$=32 dans Z_{97}^*

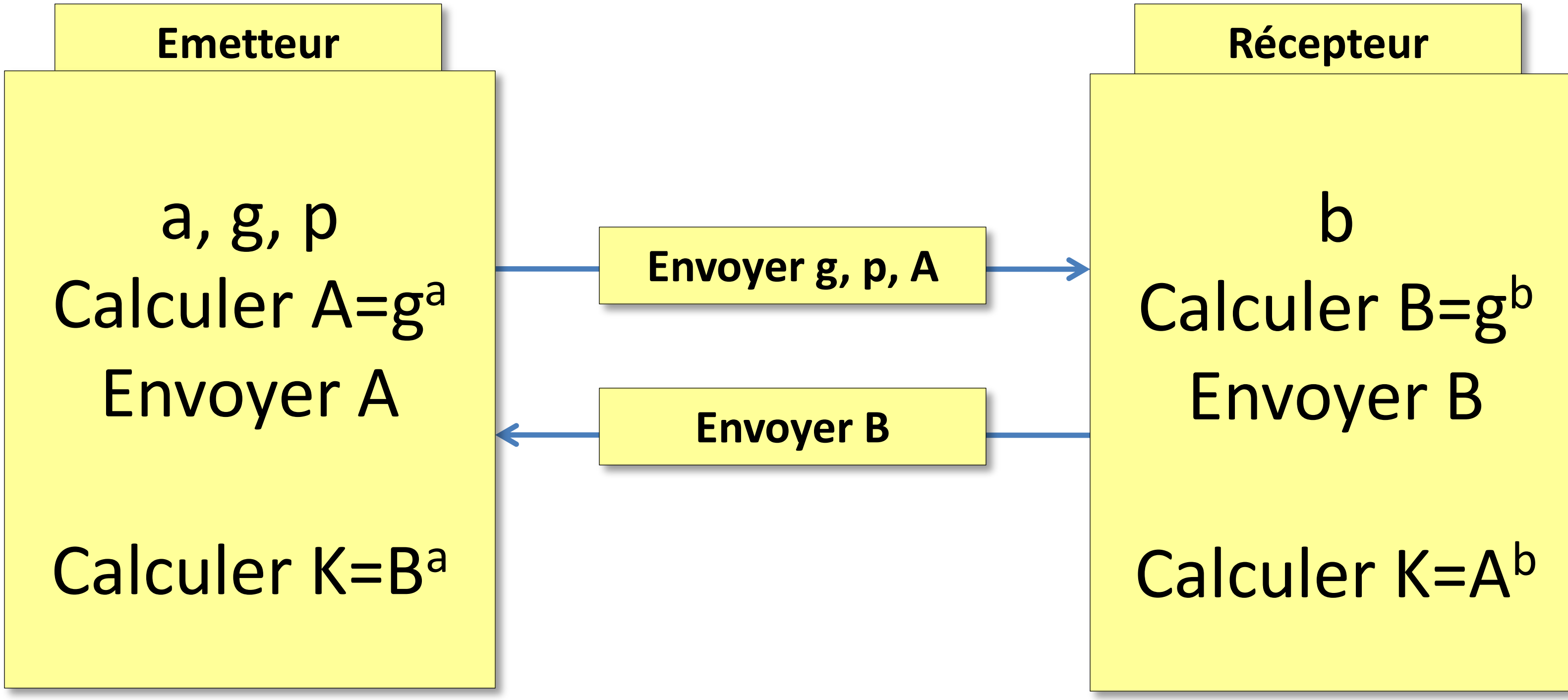
- La fonction f: $Z_p^* \rightarrow Z_p^*$

$$a \rightarrow f(a) = b = g^a \pmod{p}$$

Est une fonction **a sens unique**: f^{-1} est très difficile à calculer.

- L'échange de clé Diffie-Hellman utilise cette fonction pour **partager** une clé secrète entre un émetteur et un récepteur.
- L'émetteur et le récepteur se mettent d'accord sur le **corps utilisé** (la valeur de nombre premier **p** et le générateur **g**). Ces valeurs peuvent être **publiques** sans affecter la sécurité de l'échange.

Echange de clés : Diffie-Hellman



- La clé partagé sera commune entre les deux :

  $K = A^b = B^a = (g^a)^b = (g^b)^a = g^{ab}$
- L'émetteur ne connaît pas la valeur de «b» et le récepteur ne connaît pas la valeur de «a».
- Une personne non autorisée qui écoute peut connaître p, g, A et B, est ce qu'elle peut déduire K?

Echange de clés : Diffie-Hellman

- La clé K est donnée par :$A^b = B^a = g^{ab}$. Pour trouver K, la personne non autorisée doit calculer K à partir de A, B, p et g, c'est-à-dire :

1) Trouver «a» à partir de A : $a = Log_g(A)$
2) Calculer $K = B^a \pmod{p}$

- Trouver «a» revient à résoudre le problème du logarithme discret.
- La sécurité de l'échange Diffie-Hellman repose sur la difficulté de résoudre le problème du logarithme discret (pour de grandes valeurs de p et de g).
- La seule vulnérabilité de ce système est l'attaque de l'**Homme au milieu**.

Echange de clés : Diffie-Hellman

Exemple:

1) Alice et Bob choisissent un nombre premier p et une base g. Dans notre exemple, p=23 et g=3
2) Alice choisit un nombre secret a=6
3) Elle envoie à Bob la valeur $A = g^a$ [mod p] = 3^6 [23] = 16
4) Bob choisit à son tour un nombre secret b=15
5) Bob envoie à Alice la valeur $B = g^b$ [mod p] = 3^{15} [23] = 12
6) Alice peut maintenant calculer la clé secrète : $(B)^a$ [mod p] = 12^6 [23] = 9
7) Bob fait de même et obtient la même clé qu'Alice : $(A)^b$ [mod p] = 16^{15} [23] = 9

Conclusion

- Plusieurs autres algorithmes de chiffrement asymétrique existent, basés sur d'autres problèmes mathématiques (problème du Sac à dos (Merkle-Hellman), congruences quadratiques (Rabin)……).
- Des problèmes techniques d'implémentation sont généralement présents pour réaliser des systèmes pratiques: exponentiation modulaire, inversion modulaire, génération de grands nombres premiers fiables …..
- L'inconvénient majeur des algorithmes asymétrique est la lenteur du chiffrement/déchiffrement, la solution est d'utiliser des approches hybrides Symétrique/ Asymétrique.
- L'application la plus importante de ces algorithmes : la **signature numérique**.