CHAPITRE 3

CRYPTOGRAPHIE MODERNE

3.1 Introduction

Les techniques de cryptographie classique ont apporté une grande contribution à la sécurité des messages transmis durant des siècles. Elles Permettent seulement le chiffrement de données textuelles. Ces dernières ne garantissent pas une sécurité forte, car avec de simples algorithmes, un ordinateur peut les casser en une fraction de seconde.

3.2 Cryptographie moderne

Les techniques de cryptographie classique ne sont plus d'actualité, dès l'apparition des premiers ordinateurs. En effet, La cryptographie entre dans une nouvelle ère (cryptographie moderne) avec l'utilisation intensive des ordinateurs, qui permettent d'exploiter des algorithmes bien plus complexes. Comme les données traitées par les ordinateurs sont uniquement sous forme binaire, les procédés de substitution et de transposition sont toujours utilisés mais seulement sur deux éléments (0 et 1). On distingue deux techniques de cryptographie moderne à savoir **la cryptographie symétrique** et **la cryptographie asymétrique**.

3.3 Cryptographie symétrique

La cryptographie symétrique, également dite cryptographie à clé secrète. Elle repose sur un principe de **clé unique** pour chiffrer et déchiffrer. La clé secrète doit rester confidentielle et résistante à une attaque par force brute sous peine de rendre le système inefficient. Quant aux fonctions de chiffrement et de déchiffrement sont publiques. La cryptographie symétrique hérite des méthodes cryptographiques classiques, autrement dit, elles se basent sur des opérations mathématiques simples (substitutions, permutations).

3.3.1 Le partage de clés

Il arrive parfois que la clé soit connue par plusieurs personnes ou soit présente sur plusieurs serveurs du propriétaire. Le transfert de la clé doit absolument être **sécurisé**, Elle doit être échangée par un canal confidentiel pour que le chiffrement ne soit pas compromis. Plusieurs stratégies existent et une des plus

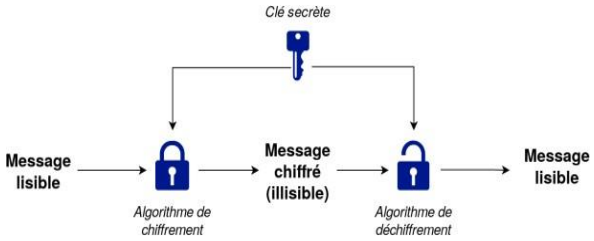


Figure 3.1: Schéma de cryptographie symétrique.

efficaces est d'utiliser un autre type de chiffrement pour chiffrer la clé secrète et d'envoyer ce message au destinataire qui pourra récupérer la clé secrète en toute sécurité. De plus, Un grands nombre de clés lors de partages deux à deux entre de nombreuses personnes est généré par la formule suivante : $Nombredecl\acute{e}s = N * (N-1)/2$ ou N représente le nombre de personnes.

Exemple : Pour 6 personnes le nombre de clés à générer est de 15 clés.

Il existe deux catégories de la cryptographie symétrique

3.3.2 Chiffrement par flots

Le chiffrement par flot (**stream cipher**), également appelé **chiffrement par flux** est une des deux grandes catégories de chiffrements modernes. Un chiffrement par flot permet de traiter des données de longueur arbitraire sans avoir à les découper (i.e., il s'agit d'un chiffrement d'une suite de caractères un à la fois, à l'aide d'une transformation qui varie au fur et à mesure du texte). Dans les algorithmes de chiffrement par flot, une suite d'octets/ bits r_i est produite à partir de la clé. Cette suite est combinée aux octets/bits du clair m_i pour donner les octets ou les bits du chiffré c_i, suivant la formule $c_i = m_i \oplus r_i$ (Généralement un XOR entre un générateur pseudo aléatoire et le message).

L'avantage du chiffrement par flot est qu'il s'implante mieux en hardware, et qu'il ne nécessite pas de zone tampon (buffer).

3.3.2.1 Exemple de chiffrement par flots

Si on souhaite chiffrer le message "Bonjour" avec la clé K, un algorithme de chiffrement effectue les opérations suivantes : la génération du flot de clés : à partir de K, on construit une clé $k_1 \ldots k_n$ de même longueur que le message. Pour chaque caractère m_i du message, on calcule le caractère chiffré correspondant $c_i = E(m_i, k_i)$ où E est une fonction qui prend en entrée un caractère m et une clé k et retourne un caractère chiffré $E(m,k)$. Le message chiffré est $c_1 \ldots c_n$.

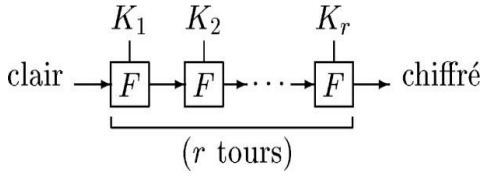
Message en clair "**Bonjour**"	01010011 01000001 01001100 01010101 01010100
Clé (générée aléatoirement)	01110111 01110111 00100100 00011111 00011010
Message Chiffré "**$6jJM**"	00100100 00110110 01101000 01001010 01001110

3.3.2.2 Exemple de cryptosystème par flot

- RC4, le plus répandu, conçu en 1987 par Ronald Rivest, Officiellement nommé Rivest Cipher 4, l'acronyme RC est aussi surnommé Ron's Code comme dans le cas de RC2, RC5 et RC6. Il a été utilisé notamment par le protocole WEP du WiFi (Wired Equivalent Privacy, de la norme 802.11), WPA (Wi-Fi Protected Access), ainsi que TLS (Transport Layer Security). Les raisons de son succès sont liées à sa grande simplicité et à sa vitesse de chiffrement. Les implémentations matérielles ou logicielles sont faciles à mettre en œuvre.

3.3.3 Chiffrement par blocs

Dans un système de chiffrement par blocs, chaque texte clair est découpé en blocs de même longueur et chiffré bloc par bloc. Le principe général d'un chiffrement itératif par blocs est le suivant : pour chaque bloc, on itère r fois une fonction interne F ; à chacun des r tours, la fonction F est paramétrée par une clé K_i ($1 \leq ou$ i $\leq$ r), et la fonction du tour i peut être notée F_{K_i} Comme on veut que le chiffrement soit inversible (pour pouvoir déchiffrer), il faut que les fonctions F_{K_i} soient bijectives.
Dans un algorithme de chiffrement par bloc, chaque message clair est découpé en blocs de taille fixe de même longueur et chiffré à l'aide d'une clé unique. Ces algorithmes sont en général construits sur un modèle itératif. Il utilise une fonction F qui prend une clé secrète k et un message M de n bits. La fonction F est itérée un certain nombre de fois (nombre de tours). Lors de chaque tour, la clé k est différente et on chiffre le message qui vient d'être obtenu de l'itération précédente. Les différentes clés k(i) qui sont utilisées sont déduites de la clé secrète k. Les algorithmes les plus connus des systèmes cryptographiques symétriques sont : le DES et l'AES.



3.3.3.1 Système de chiffrement DES

DES (Data encryption Standard) est l'ancien standard d'algorithme symétrique adoptée par NSA en 1967. Il consiste à faire des combinaisons, des substitutions et des permutations entre le texte à chiffrer et la clé. La clé du DES est une chaîne de 64 bits (succession de 0 et de 1), mais en fait seuls 56 bits servent réellement à définir la clé.

Les bits 8,16,24,32,40,48,56,64 sont des bits de parité (bits de détection d'erreur) Le 8ème bit est fait en sorte que sur les 8 premiers bits, il y ait un nombre impair de 1. Par exemple, si les 7 premiers bits sont 1010001, le 8ème bit est 0. Ceci permet d'éviter les erreurs de transmission. Il y a donc pour le DES, 2 à la puissance 56 clés possibles, soit environ 72 millions de milliards possibilités. Quant aux blocs sont de taille 64 bits. Les grandes lignes de l'algorithme sont :

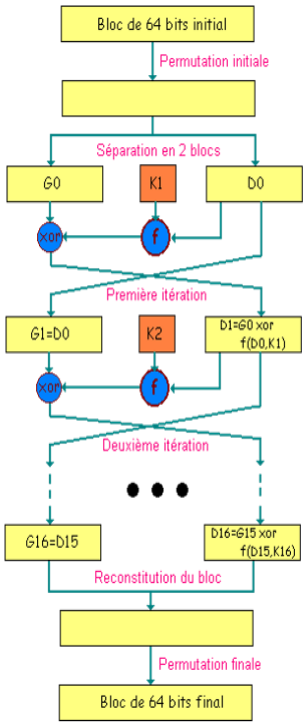
- Phase 1 : Préparation -Diversification de la clé Le texte est découpé en blocs de 64 bits. On diversifie aussi la clé K, c'est-à-dire qu'on fabrique à partir de K 16 sous-clés K_1, ..., K_{16} à 48 bits. Les K_i sont composés de 48 bits de K, pris dans un certain ordre.

- Phase 2 : Permutation initiale Pour chaque bloc de 64 bits **x** du texte, on calcule une permutation finie **y=P(x)**. y est représenté sous la forme $y = G_0D_0$, G_0 étant les 32 bits à gauche de y, D_0 les 32 bits à droite

- Phase 3 : Itération On applique 16 rondes d'une même fonction. A partir de $G_{i-1}D_{i-1}$ (pour i de 1 à 16), on calcule $G_i\, D_i$ en posant :

  - $G_i = D_{i-1}$.
  - $D_i = G_{i-1} XOR\, f\,(D_{i-1}, K_i)$.

- Phase 4 : Permutation finale. On applique à $G_{16}D_{16}$ l'inverse de la permutation initiale. $Z = P^1(G_{16}D_{16})$ est le bloc de 64 bits chiffré à partir de x.



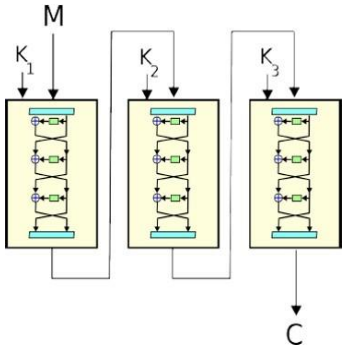
3.3.3.2 Système de chiffrement 3DES

Trois chiffrements DES sont chaînés à l'aide de deux clés de 56 bits (ce qui équivaut à une clé de 112 bits). Ce procédé est appelé Triple DES, noté TDES (parfois 3DES ou 3-DES)

Le TDES permet d'augmenter significativement la sécurité du DES, toutefois il a l'inconvénient majeur de demander également plus de ressources pour les étapes de chiffrement et de déchiffrement. On distingue habituellement plusieurs types de chiffrement triple DES :

- DES-EEE3 : 3 chiffrements DES avec 3 clés différentes

- DES-EDE3 : une clé différente pour chacune des 3 opérations DES (chiffrement, déchiffrement, chiffrement)
- DES-EEE2 et DES-EDE2 : une clé différente pour la seconde opération (déchiffrement)



En 1997 le NIST (National Institute of Standards and Technology) lança un nouvel appel à projet pour élaborer l'**AES (Advanced Encryption Standard)**, un algorithme de chiffrement destiné à remplacer le DES

3.3.3.3 Cryptographie symétrique : Avantages

- Rapidité des opérations de chiffrement
- Facilité de mise en œuvre Support des chiffrements de gros volumes de données

3.3.3.4 Cryptographie symétrique : Inconvénients

- Problématique de l'échange de la clé de chiffrement qui doit être secrète, ce qui est difficile à réaliser à grande échelle
- Établissement préalable d'un canal sûr pour la transmission de la clé.
- Le nombre de clés utilisées augmente très rapidement en fonction du nombre total d'utilisateurs
- La confidentialité des données transmises est la seule propriété de sécurité assurée, l'intégrité et l'authencité des données ne peuvent être confirmées

3.4 CRYPTOGRAPHIE asymétrique

Afin de surmonter les inconvénients de partage des clés secrètes, Whiteld Die, Martin Hellman et Ralph Merkle avaient une proposition révolutionnaire basée sur l'idée suivante :

Il n'est pas nécessaire que la clé possédée par la personne A qui crypte le message soit secrète, et qu'elle soit la même que celle du récepteur B.

Afin de réaliser un tel système, l'entité **B** publie une **clé publique** "**kpub**" connue de tous. **B** possède également une **clé privée** "**kpr**" correspondante, qui est utilisée pour le déchiffrement.

La cryptographie **à clé publique (asymétrique)** consiste en l'existence d'une paire de clés de chaque côté (émetteur et récepteur) liées mathématiquement. Chaque paire est composée d'une clé privée (et différente pour chaque utilisateur qui doit être gardée secrète), et d'une clé publique connu par tous les utilisateurs. La cryptographie asymétrique se base sur des fonctions à sens unique. Cela veut dire que les données cryptées avec la clé publique ne peuvent être décryptées que s'il on possède la clé secrète.

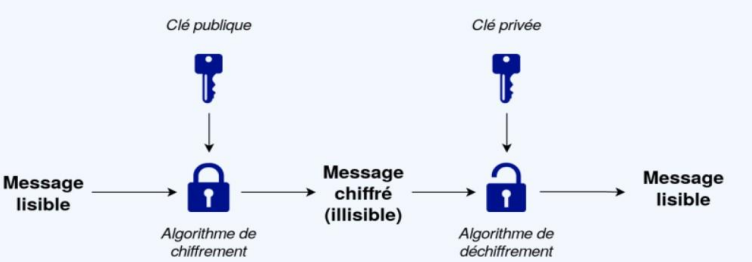


Figure 3.2: Schéma de cryptographie asymétrique.

3.4.1 Protocole de Diffie-Hellman

Parallèlement à leur découverte de l'aspect asymétrique de la cryptographie, Diffie et Hellman ont proposé en 1976 un protocole d'échange de clés totalement sécurisé. Le problème est le suivant. Deux entités A et B veulent s'échanger un message crypté en utilisant un algorithme nécessitant une clé K. Ils veulent s'échanger cette clé K, mais ils ne disposent pas de canal sécurisé pour cela. Le protocole d'échange de clés de Diffie et Hellman répond à ce problème lorsque K est un nombre entier. Il repose sur l'arithmétique modulaire.

3.4.1.1 Principe du protocole

- Etant donné des entiers g,n,x, avec n premier et $1 <= g <= n-1$
- Il est facile de calculer l'entier $Y = g^x mod n$
- Si on connait $Y = g^x mod n$, g et n, il est très difficile de retrouver x, pourvu que n soit assez grand.
- Retrouver x connaissant g^x $(mod n)$, g et n s'appelle résoudre le problème du logarithme discret. Comme pour la factorisation d'entiers, c'est un problème pour lequel on ne dispose pas d'algorithme efficace.

3.4.1.2 Fonctionnement du protocole

Supposant que les entités A et B veulent s'échanger une clé avec le protocole Diffie Hellman, les étapes du protocole sont comme suit :

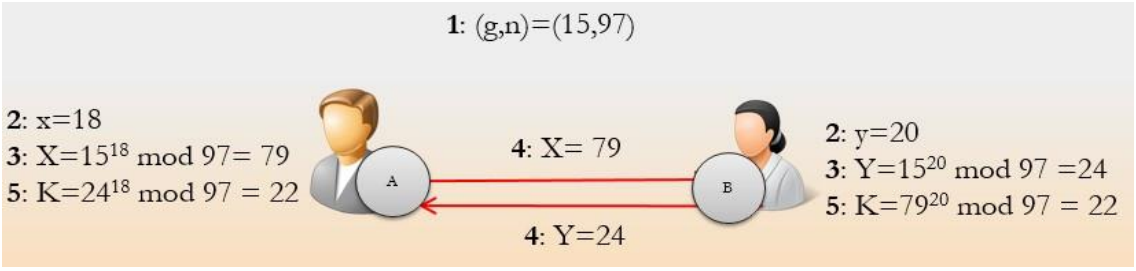
- Etape 1 : A et B choisissent ensemble un grand nombre premier n et un entier g tel que $1 \leq g$ *ou* $\leq n-1$. Cet échange n'a pas besoin d'être sécurisé.
- Etape 2 : A et B choisissent secrètement un entier chacun qui sont respectivement x et y.
- Etape 3 : A calcule $X = g^x mod n$; B calcule $Y = g^y mod n$
- Etape 4 : A et B s'échangent les valeurs de X et Y. Cet échange n'a pas besoin d'être sécurisé

- Etape 5 : A Calcule $Y^x = (g^y)^x mod n = g^{yx} mod n$, et appelle ce nombre K , qui est la clé partagée avec B. De même B calcule $X^y = (g^y)^x mod n = g^{yx} mod n$, et appelle ce nombre K , qui représente la clé partagée avec A.

Dans ce protocole, la clé publique est (g,n), la clé privée de A est (x) et la clé privée de B est (y).

3.4.1.3 Exemple

Exécution du protocole Diffie Hellman pour (g, n) = (15,97), x = 18 et y = 20.



3.4.1.4 Partage de clé entre plus entités

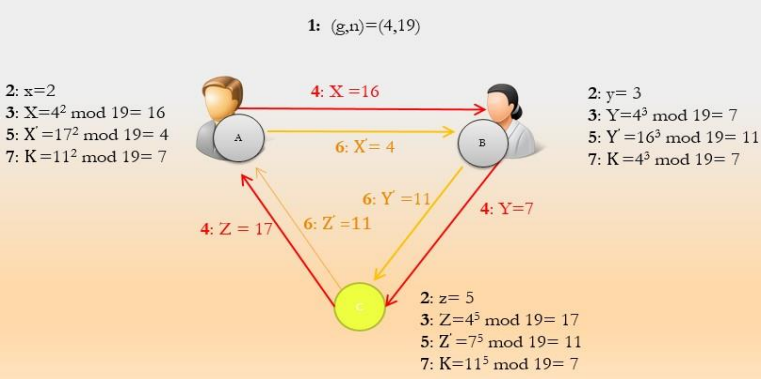
L'avantage du protocole Diffie Hellman est qu'il est applicable pour le partage de clé entre plus de deux entités. Si on l'applique pour trois entités, les étapes sont comme suit :

1. A, B et C choisissent ensemble un grand nombre premier n et un entier g tel que $1 \leq g\ ou \leq n-1$ Cet échange n'a pas besoin d'être sécurisé.
2. A, B et C choisissent secrètement un entier chacun qui sont respectivement x, y et z.
3. **A** calcule $X = g^x$ mod n, **B** calcule $Y = g^y mod n$ et **C** calcule $Z = g^z mod n$
4. **A** envoie X à **B**, **B** envoie Y à **C**, et **C** envoi Z à **A**. Cet échange n'a pas besoin d'être sécurisé.
5. **A** Calcule $X' = Z^x = (g^z)^x mod n = g^{zx} mod n$. **B** calcule $Y' = X^y = (g^x)^y mod n = g^{xy} mod n$ et **C** calcule $Z' = Y^z = (g^y)^z mod n = g^{yz} mod n$
6. **A** envoie X' à **B**, **B** envoie Y' à **C**, et **C** envoie Z' à **A**. Cet échange n'a pas besoin d'être sécurisé.
7. **A** Calcule $K = Z'^x = ((g^y)^z)^x mod n = g^{yzx} mod n$, **B** calcule $K = X'^y = ((g^z)^x)^y mod n = g^{zxy} mod n$ et **C** calcule $K = Y'^z = ((g^x)^y)^z mod n = g^{xyz} mod n$

Dans ce protocole, la clé publique est (g, n), la clé privée de A est x, la clé privée de B est y et la clé privée de C est z.

3.4.1.5 Exemple

Exécution du protocole diffie hellman pour (g, n) = (4, 19), x= 2 et y = 3, z= 5.



Cette découverte de Diffie et Hellman est une vraie révolution dans l'histoire de la cryptographie. Le problème de l'échange des clés est en effet résolu. Ce protocole est cependant vulnérable à l'attaque de l'homme du milieu, qui implique un attaquant capable de lire et de modifier tous les messages échangés.

3.4.2 Le cryptosystème RSA

L'algorithme à clé publique RSA (Rivest, Shamir et Adleman) a été inventé en 1977 par Ronald Rivest, Adi Shamir et Leonard Adleman. La sécurité de l'algorithme RSA provient du caractère complexe de la factorisation de grands entiers produits de deux grands nombres entiers. Il est facile de multiplier ces nombres, mais la détermination des nombres entiers d'origine, la « **factorisation** ». Le principe de l'algorithme RSA est le suivant :

- Choisir deux grands nombres premiers distincts **p** et **q** de même ordre
- Calculer leur produit **n= p*q** est (représente le module du chiffrement public).
- Calculer $\Phi(n) = (p-1) * (q-1)$
- Choisir également un nombre naturel **e** tel que **e** et $\Phi(n)$ soient premiers entre eux (PGCD(e, $\varphi(n)$)=1).
- Calculer **d** de telle sorte : $(e * d \ mod \varphi(n) = 1)$.
- Pour chiffrer un message **M** l'émetteur calcule : $C = M^e mod n$ et envoie C,
- Le récepteur déchiffre le message à l'aide de la clé privée d en calculant : $M = C^d \ mod n$

L'expéditeur et le destinataire doivent connaître la valeur de n. L'expéditeur connaît la valeur de e, et seul le récepteur connaît la valeur de d. Ainsi, la clé publique est définie comme **(e, n)** et la clé privée est définie comme **(d, n)**.

La sécurité de l'algorithme RSA repose sur la difficulté de factoriser les grands nombres : factoriser un grand nombre n en deux nombres premiers p et q et retrouver la clé privée d à partir de n et e.

3.4.2.1 Exponentiation modulaire rapide

L'exponentiation modulaire est un type de calcul de puissance (exponentiation) suivie d'un calcul de modulo, qui est souvent utilise dans la cryptographie. L'objectif de l'exponentiation modulaire rapide est de calculer $c = a^e mod n$ en réduisant le nombre de multiplications. En effet, lorsque l'on souhaite calculer $a^e mod n$, on peut naïvement effectuer (e) multiplications en effectuant le produit : $a^e = a * a * \ldots\ldots a$ Par contre, l'algorithme d'exponentiation modulaire rapide se contente au maximum de

$2 * log\,2(e)$ multiplications.

Pour calculer $c = B^A mod\ N$ avec l'algorithme d'exponentiation rapide, on utilise la fonction CExpMod (P, J, R) tel que P = B, J = A et R = 1

Algorithm 1 CExpMod()

Require: P : Grand nombre, J : Grand nombre, R : Grand nombre
Ensure: Result : Grand nombre;

```
while J > 0 do
    if J == 1 then
        Result = R * P mod N
    end if
    if J mod 2 == 0 then
        P ← P^2 mod N
        J ← J/2
        R ← R mod N
    end if
    if J mod 2 == 1 then
        R ← P * R mod N
        P ← P^2 mod N
        J ← (J-1)/2
    end if
end while
```

Pour illustrer l'algorithme d'une manière plus simple :

- On commence par convertir le A en binaire, tel que : $A = a_0 * 2^0 + a_1 * 2^1 + ... + a_K * 2^k$. La longueur de A est de k bits. a_i peut prendre la valeur 0 ou 1 pour tout i.
- Commençant par le poids le plus faible, on calcule $(B mod N)$, ensuite, pour chaque a_i tel que $i = 0$, on calcule $résultat\,(a_i) = (résultat\,(a_i - 1))^2 mod\ N$.
- Enfin, $C = \prod_{i=0}^{k-1} résultat(a_i = 1)$, Autrement dit, pour tous les $a_i = 0$, leur résultat n'est pas comptabilisé dans C, pour les $a_i = 1$, on multiplie leurs résultats mod N, ce qui va donner C.

3.4.2.2 Exemple d'exponentiation modulaire rapide

Nous calculons $C = 150^{233} mod\ 437$
$(233)_2 = 11101001$

1	150
0	$150^2 mod\ 437 = 213$
0	$213^2 mod\ 437 = 358$
1	$358^2 mod\ 437 = 123$
0	$123^2 mod\ 437 = 271$
1	$271^2 mod\ 437 = 25$
1	$25^2 mod\ 437 = 188$
1	$188^2 mod\ 437 = 384$

C= (150*123*25*188*384)mod 437=351

D'ou $C = 150^{233} mod\, 437 = 351$

3.4.2.3 Exemple du protocole RSA

Pour p= 19, q= 23 et e=17
Calculer la clé privée correspondante à e=17 puis chiffrer le message M=351.
- n = 19 × 23 = 437
- Φ(n) = (19 1) × (23 1) = 396
- 17 × d mod 396 = 1 → d=233 d'où la clé privée (d,n)=(233,437)
- C = 351^{17} mod 437 = 150

3.4.3 Chiffrement de Rabin

L'algorithme de chiffrement de *Rabin* tire sa sécurité du problème de factorisation des grands nombres entiers. L'algorithme se déroule en trois étapes :

1. **Création de clés :** On choisit deux nombres premiers p et q, tels que
   p mod 4 = 3 et
   q mod 4 = 3.
   La clé publique est $n = p \times q$ et la clé privée est (p, q).

2. **Chiffrement :** Le message à chiffrer doit être strictement inferieur à n. Pour chiffrer un message M, on calcule C =M2 mod n.

3. **Déchiffrement :** Pour d´echiffrer le message, on calcule tout d'abord mp, mq, yp, et yq tels que

   mp = C(p+1)/4 mod p,
   mq = C(q+1)/4 mod q,
   et $p \times yp + q \times yq = 1$.
   Ensuite, on calcule R et S, tels que
   R = ($p \times mq \times yp + q \times mp \times yq$) mod n,
   et S = ($p \times mq \times yp - q \times mp \times yq$) mod n.
   L'une des solutions {R, S, n−R, n−S} est le message M.

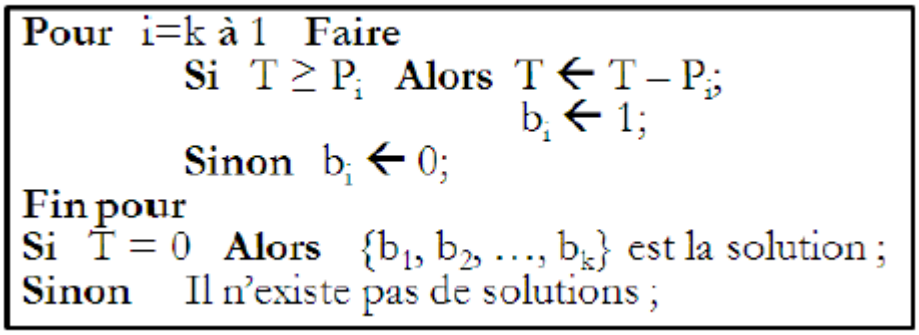
3.4.3.1 Exemple

p = 11, q = 23) n = 253.
Pour M= 158, C = 1582 mod 253 = 170.
mp = 170(11+1)/4 mod 11 = 4.
mq = 170(23+1)/4 mod 23 = 3.
$11yp + 23yp = 1$) $yp = -2$ et $yq = 1$.
R = (11 × 3 × (−2) + 23 × 4 × 1) mod 253 = 26.
S = (11 × 3 × (−2) − 23 × 4 × 1) mod 253 = 95.
Les solutions possibles sont : {26, 95, (253 − 26) = 227, (253 − 95) = **158**}.

3.4.4 Chiffrement de Merkel Hellman

L'algorithme de chiffrement de *Merkel-Hellman* tire sa sécurité du problème de *sac-a-dos*.
Etant données des valeurs entières P1, P2, ..., Pk et un poids T. Est-il possible de déterminer des valeurs binaires $b1$, $b2$, ..., bk de telle sorte que T = b1P1 + b2P2 +... + bkPk ? Si la suite des poids Pi est super-

croissante (chaque poids P*i* est strictement supérieur `a la somme de tous les poids précédents), alors il existe un algorithme polynomial permettant de déterminer les valeurs de *bi* :



On peut vérifier qu'avec la suite super-croissante {2, 3, 6, 12} pour T = 15 on obtient la solution {0, 1, 0, 1}.

La résolution d'une suite non super-croissante est un problème NP-complet. La sécurité du système de chiffrement *Merkle-Hellman* est base sur cette difficulté. Il se déroule en trois étapes :

1. **Création de clés :** On choisit une suite super-croissante A = {P1, P2, ..., Pk}. Ensuite, on choisit deux nombres *n* et *m* tel que *m* est supérieur `a la somme de tous les P*i*, et *n* n'a de facteur commun avec aucun nombre de la suite. On calcule, ensuite, la suite non super-croissante B = {P'1, P'2, ..., P'*k*} tel que :
   P'*i* = *n* × P*i* mod *m*.

   La clé publique est B, et la clé privée est (A, *n*,*m*).

2. **Chiffrement :** Le message à chiffrer doit être une suite binaire tel que M =*b*1*b*2...*bk*. On calcule :
   C = *b*1P'1 + *b*2P'2 + ... + *bk*P'*k*.

3. **Déchiffrement** *:* Pour déchiffrer le message :
   - On calcule tout d'abord *n*−1 tel que *n* × *n*−1 mod *m* = 1.
   - Ensuite, on calcule T = *n*−1 × C mod *m*.
   - Enfin, on calcule les valeurs *bi* du message M en utilisant l'algorithme de résolution d'une suite super-croissante.

3.4.4.1 Exemple

A = {2, 3, 6, 13, 27, 52}, *n* = 31 et *m* = 105) B = {62, 93, 81, 88, 102, 37}.
Pour M= 53 = 110101(2), C = 1×62+1×93+0×81+1×88+0×102+1×37 =280.
31 × *n*−1 mod 105 = 1) *n*−1 = 61.
T = (61×280) mod 105 = 70 = 1×2+1×3+0×6+1×13+0×27+1×52)
d'où M= 110101(2) = 53.

3.4.5 Chiffrement d'ElGamal

Le chiffrement El Gamal est un protocole de cryptographie asymétrique invente par Taher Elgamal en 1984. Ce chiffrement est basé sur le problème du logarithme discret. Son principe est le suivant :

1. **Génération de clés**

- Choisir un grand nombre premier **p** et deux nombres **a** et **g** tel que : $a, g < p$;
- Puis on calcule $A = g^a mod p$.
- La clé publique est **(A,g, p)** et la clé privée est **a**.

2. **Chiffrement**

   Pour chiffrer un message **M**, il faut procéder comme suit :

   a. Choisir un nombre aléatoire b, tel que $b < a$ et le PGCD (b,p-1) =1, ;

   b. Puis on calcule $B = g^b mod p$ et $C = M * A^b mod p$

   c. Le message chiffré est alors **(B,C)**.

3. **Déchiffrement**

   Pour déchiffrer un cryptogramme **C**:

   a. Calculer $M = C * B^{p-a-1} mod p$.

Le logarithme discret est la seule méthode connue pour casser le chiffre d'ElGamal, Néanmoins, El Gamal est 2 fois plus lent que RSA et la taille des données chiffrées représente 2 fois celle des données en clair.

3.4.5.1 Exemple

Déterminer la clé publique et la clé privée pour p = 23, g = 7, a = 6, b = 3. Puis chiffrer le message M = 7, et déchiffrer le résultat.

- Génération de clés
  - On a p est un nombre premier, on a aussi g < p; a < p. Donc les paramètres sont valides.
  - $A = 7^6 mod\, 23 = 4$
  - La clé privée est **a = 6** et la clé publique est **(A, g, p) = (4, 7, 23)**
- Chiffrement du message
  - on a le pgcd(b, 22) = 1
  - $B = 7^3 mod\, 23 = 21$;
  - $C = 7 * 4^3 mod\, 23 = 11$.
  - Le message chiffré est donc (C,B)= (11,21).
- Déchirement du message
  - $M = 11 * 21^{23-6-1} mod\, 23 = ((11 mod\, 23) * (21^{23-6-1} mod\, 23)) mod\, 23 = 7$

3.4.6 Chiffrement Asymétriques : Avantages

- Le problème n'existe plus pour communiquer la clé de déchiffrement, dans la mesure où les clés publiques peuvent être envoyées librement.
- Le chiffrement par clés publiques permet donc à des personnes d'échanger des messages chiffrés sans pour autant posséder de secret en commun, seule la clé secrète à besoin d'être conservée de manière secrète.

- Selon l'usage, une paire de clé (publique/secrète) peut être utilisée plus longtemps qu'une clé symétrique.
- La cryptographie à clé publique permet de réaliser des schémas de signature électronique assurant un service de non répudiation.

3.4.7 Chiffrement Asymétriques : Inconvénients

- Tout le challenge consiste à s'assurer que la clé publique que l'on récupère est bien celle de la personne à qui l'on souhaite faire parvenir l'information chiffrée.
- Les performances des systèmes asymétriques sont beaucoup moins bonnes que celles des systèmes symétriques car ces systèmes nécessitent de pouvoir calculer sur des grands nombres.
- La taille des clés est généralement plus grande pour ces systèmes que pour les systèmes à clé secrète.
- La cryptographie à clé publique nécessite la mise en place d'une infrastructure de gestion de clé afin d'éviter les attaques par le milieu.
- Sa lenteur par rapport au chiffrement symétrique. En effet, le chiffrement asymétrique exige une puissance de calcul bien supérieure en raison de sa complexité mathématique.