Le chiffrement par clé publique

1 Concept

Dans le cas des systèmes symétriques, on utilise une même clé pour le chiffrement et le déchiffrement. Le problème repose dans la transmission de la clé : il faut une clé par destinataire. Dans le cas des systèmes asymétriques, chaque personne possède 2 clés distinctes (une privée, une publique) avec impossibilité de déduire la clé privée à partir de la clé publique. De ce fait, il est possible de distribuer librement cette dernière.

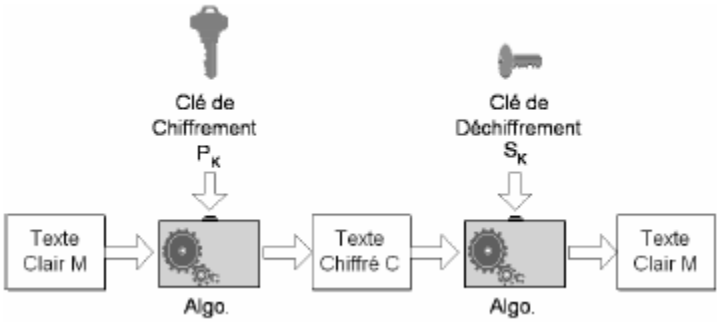


Fig. 1 – Chiffrement à clé publique

On peut classer l'utilisation des algorithmes à clé publique en 3 catégories :
– Chiffrement/déchiffrement : cela fournit le secret.
– Signatures numériques : cela fournit l'authentification.
– Échange de clés (ou des clefs de session).
Quelques algorithmes conviennent pour tous les usages, d'autres sont spécifiques à un d'eux.
Le concept date officiellement de 1976 de Diffie et Hellman. Officieusement, les bases existent depuis 1969 par Ellis. La première implémentation a lieu en 1978 par Rivest, Shamir et Adleman sous la forme de l'algorithme RSA bien que, là aussi, les fondements de ce système datent de 1973, par Cocks.

La sécurité de tels systèmes repose sur des problèmes calculatoires :
– RSA : factorisation de grands entiers : NP
– ElGamal : logarithme discret : NP
– Chor-Rivest, Diffie-Hellman : problème du sac à dos (knapsacks) : NPC
– ...
La recherche des clés par force brute est toujours théoriquement possible mais les clefs utilisées sont trop grandes (> 512bits). La sécurité se fonde sur une assez grande différence en termes de difficulté entre les problèmes faciles (déchiffrement) et difficiles (cryptanalyse) :
– généralement le problème difficile est connu, mais il est trop complexe à résoudre en pratique,
– La génération des clés exige l'utilisation de très grands nombres.
En conséquence, ce type de chiffrement est lent si on le compare aux chiffrements symétriques.

2 Merkle-Hellman

2.1 Définition du problème

Soit un havresac de capacité T et un ensemble S d'objets occupant les espaces $S = \{a_1, a_2, a_3, ..., a_n\}$. Il faut trouver un vecteur de sélection $V = \{v_1, v_2, v_3, ..., v_n\}$ satisfaisant la relation $\Sigma(a_i * v_i) = T$. Ce problème n'a pas toujours une solution. Aussi, si T et S sont très grands, il est beaucoup plus difficile de trouver le vecteur associé.

Exemple : Soit $S = \{17, 38, 73, 4, 11, 1\}$ et $T = 53 = 38 + 4 + 11$. Donc $V = \{0, 1, 0, 1, 1, 0\}$.
Pour T = 45, il n'y a pas de solution.

2.2 Idée de base

Un bloc de texte clair de longueur égale au nombre d'objets d'un tas sélectionnerait des objets. Les bits du texte clair correspondraient aux valeurs des v_i : un 1 signifierait que l'objet est présent et 0 un objet absent. Et le texte chiffré serait la somme résultante.

Exemple :

M :	1	1	1	0	0	1	0	1	0	1	1	0
Tas :	1	5	6	11	14	20	1	5	6	11	14	20
C :	1+5+6+20 = 32						5+11+14 = 30					

2.3 Les empilements

Il y a 2 problèmes d'empilement :
– 1 soluble en temps linéaire
– 1 soluble en temps exponentiel
L'empilement facile peut être transformé pour créer un empilement difficile. Pour la clé publique, on utilisera un empilement difficile qui servira à chiffrer. La clé privée quant à elle, utilisera un empilement facile, qui donne un moyen simple de déchiffrer les messages. Bien sûr, ceux qui ne connaissent pas la clé privée sont obligés de résoudre le problème d'empilement difficile, ce qui est infaisable en pratique.

2.3.1 Empilement facile

Si la liste des poids est super-croissante, on utilise un algorithme (appelé glouton) de la manière suivante :
1. Prendre le poids total et le comparer avec le plus grand nombre de la suite.
   - Si le poids total est inférieur à ce nombre, alors celui-ci n'est pas dans le tas. On recommence l'opération avec le nombre suivant dans le tas (qui, par définition de la suite, sera plus petit)
   - Si le poids total est supérieur à ce nombre, alors celui-ci est dans le tas
2. Réduire le poids du tas à créer de ce nombre et passer au plus grand nombre suivant de la suite.
3. Répéter jusqu'à ce que ce soit terminé.
4. Si le poids total a pu être ramené à 0 : il y a une solution.

2.3.2 Empilement difficile

Dans le cas présent, on ne connaît pas d'algorithme rapide. Il faut tester méthodiquement toutes les solutions possibles, ce qui, si la suite des poids est suffisamment longue, est impraticable. Ces algorithmes sont exponentiels.
Le cryptosystème de Merkle-Hellman exploite cette propriété. La clé privée est une suite de poids super-croissante. A partir de celle-ci, on calcule la clé publique. Ce calcul consiste à prendre la suite super-croissante, et à la multiplier par (n modulo m), avec m supérieur à la somme de tous les termes de la suite, et n ne devant avoir aucun facteur commun avec m.

2.4 Algorithme

Le chiffrement consiste à additionner les termes où un 1 apparaît. Pour le déchiffrement, on calcule n^{-1} tel que

$$n * n^{-1} \equiv 1 \bmod m$$

Ensuite, on multiplie chaque valeur du texte chiffré par n^{-1} mod m.
En pratique, les sacs contiennent environ 250 éléments. Et chaque terme a une longueur de 200 à 400 bits. Le module a une longueur de 100 à 200 bits.

Exemple de calcul de la clé publique Soit S, une séquence super-croissante de h entiers : par exemple S = {1, 2, 4, 9}.
Choisissons un multiplicateur n et un module m : soit n = 15 et m = 17
– 1 * 15 mod 17 ⇒ 15
– 2 * 15 mod 17 ⇒ 13
– 4 * 15 mod 17 ⇒ 9
– 9 * 15 mod 17 ⇒ 16
Le havresac difficile est donc H = {15, 13, 9, 16}, et représente la clé publique.
Le message est ainsi traité comme une séquence de bits :

$$P = [p_1, p_2, p_3, ..., p_k]$$

On le divise en blocs de h bits :

$$P_0 = [p_1, p_2, p_3, ..., p_h], P_1 = [p_{h+1}, p_{h+2}, p_{h+3}, ..., p_{2*h}], ...$$

On utilise chaque bloc comme vecteur V du problème de havresac.

Exemple de chiffrement Soit P = 0100101110100101 ⇒ 0100 1011 1010 0101
– [0, 1, 0, 0] * [15, 13, 9, 16] ⇒ 13
– [1, 0, 1, 1] * [15, 13, 9, 16] ⇒ 40

– [1, 0, 1, 0] * [15, 13, 9, 16] ⇒ 24
– [0, 1, 0, 1] * [15, 13, 9, 16] ⇒ 29
Le message chiffré est donc {13, 40, 24, 29} en utilisant le havresac public (la clef publique) H = [15, 13, 9, 16].

Pour le déchiffrement, le destinataire légitime connaît le havresac simple S et les valeurs de n et de m. Il peut donc déterminer n^{-1}.
Exemple de déchiffrement Avec n = 15 et m = 17, n^{-1} vaut 8 car 15 * 8 = 120 = 7 * 17 + 1. On a alors, par l'algorithme glouton :
– 13 * 8 mod 17 = 104 mod 17 = 2 = [1, 2, 4, 9] * [0100]
– 40 * 8 mod 17 = 320 mod 17 = 14 = [1, 2, 4, 9] * [1011]
– 24 * 8 mod 17 = 192 mod 17 = 5 = [1, 2, 4, 9] * [1010]
– 29 * 8 mod 17 = 232 mod 17 = 11 = [1, 2, 4, 9] * [0101]
et le texte en clair est 0100 1011 1010 0101 ⇒ 0100101110100101. On a donc bien retrouvé le texte original.

2.5 Sécurité
Plusieurs failles ont été découvertes, elles ont rendu l'algorithme obsolète.
Herlestam a démontré en 1978 qu'un bit de texte clair pouvait souvent être retrouvé. Malgré les modifications apportées à l'algorithme à la suite de ces découvertes, des travaux de Shamir (1982-84) et Brickell (1985) ont fait sortir des standards de chiffrement l'algorithme de Merkle-Hellman.

3 RSA : Rivest - Shamir - Adleman
Il est basé sur le calcul exponentiel. Sa sécurité repose sur la fonction unidirectionnelle suivante : le calcul du produit de 2 nombres premiers est aisé. La factorisation d'un nombre en ses deux facteurs premiers est beaucoup plus complexe.
Il s'agit du système le plus connu et le plus largement répandu, basé sur l'élévation à une puissance dans un champ fini sur des nombres entiers modulo un nombre premier. Le nombre d'exponentiation prend environ $O((\log n)^3)$ opérations ce qui est rapide et facile. Il emploie de grands nombres entiers (par exemple représentés sur 1024 bits).
Ce cryptosystème utilise deux clés d et e, interchangeables. Le chiffrement se fait selon

$$C = M^e \bmod n$$

et le déchiffrement par

$$M = C^d \bmod n.$$

La sécurité repose sur le coût nécessaire pour factoriser de grands nombres. Le nombre de factorisation prend environ $O(e^{\log n \log(\log n)})$ opérations ce qui demande un temps de calcul trop important pour les machines actuelles, dans un cadre privé. On l'utilise pour la confidentialité, l'authentification, ou encore une combinaison des 2.

3.1 Principes
On possède une paire de clés, l'une publique (e,n) et une privée (d,n). La première étape revient à choisir n. Il doit s'agir d'une valeur assez élevée, produit de 2 nombres premiers très grands p et q. En pratique, si p et q ont 100 chiffres décimaux, n possèdera 200 chiffres. Selon le niveau de sécurité souhaité, la taille de n peut varier : 512 bits, 768, 1024 ou 2048[3].
Dans un second temps, on choisira un très grand entier e, relativement premier à (p-1)*(q-1). La clé publique sera formée par (e,n). On choisira ensuite un d tel que

$$e * d \equiv 1 \bmod (\Phi(n)).$$

La clé privée sera donnée par (d,n).
Dernière phase : on jette p et q. Le cryptanalyste devant retrouver ces valeurs, il faut les détruire pour éviter les fuites.

3.1.1 Justification de l'inversibilité
Par les théorèmes d'Euler et de Fermat, on sait que $a^{\Phi(n)} \equiv 1 \bmod n$

et

$a^{\Phi(n)} \bmod n = 1$ où (a,n)=1.

Dans le RSA, on a n = p * q. De plus, Φ(n) donne le nombre d'entiers positifs plus petits que n et relativement premiers à n (si p est premier, Φ(p) = p − 1). Si n = p * q, avec p et q premiers, il vient

$$\Phi(n) = \Phi(p) * \Phi(q) = (p - 1) * (q - 1)$$

De par la façon de choisir e et d,

$$e * d \equiv 1 \bmod \Phi(n) = k * \Phi(n) + 1$$

pour un certain k.

De plus, par définition et propriétés des opérations modulo n, on a :

$$D_k(E_k(M)) = ((M)^e \bmod n)^d \bmod n = (M^e)^d \bmod n = M^{e*d} \bmod n$$

Et donc :

$$M^{e*d} = M^{k*\Phi(n)+1} = M^{k\Phi(n)}.M \bmod n = 1.M \bmod n = M \bmod n$$

3.2 Résumé

1. Génération de 2 nombres premiers p et q
2. Calcul de n = p*q
3. Déterminer e tel que $3 < e < \Phi(n)$ et $(e, \Phi(n)) = 1$
4. Calculer d tel que $e * d \equiv 1 \bmod \Phi(n)$
5. Clé publique : (e,n)
6. Clé privée : (d,n)
7. p et q doivent rester secrets, voire supprimés
8. $C = M^e \bmod n$ et $M = C^d \bmod n$

Exemple :

Soient p = 31, q = 53 c'est-à-dire n=1643. $\Phi(n) = 1560$ (nombre d'éléments relativement premiers à n et < n).
Soit e = 11 (par exemple, et on a bien $(e,\Phi(n))=1$).
On détermine que d = 851 (inverse modulaire de e sur $Z_{\Phi(n)}$).
La clé publique est donc (11,1643) et la clé privée est (851,1643).
Soit le codage par la position dans l'alphabet du mot «ANEMONE». Il vient

01 14 05 13 15 14 05

On procède selon deux conditions :

1. Découpage en morceaux de même longueur, ce qui empêche la simple substitution :

011 405 131 514 05_

On ajoute un padding initial si nécessaire.

001 140 513 151 405

Cela provoque la perte des patterns (« NE »).

2. Découpage en morceaux de valeur inférieure à n, car opération modulo n.

Lors du chiffrement, on a

$001^{11} \bmod 1643 =$	1
$140^{11} \bmod 1643 =$	109
$513^{11} \bmod 1643 =$	890
$151^{11} \bmod 1643 =$	1453
$405^{11} \bmod 1643 =$	374

Fig. 2 – Opération de chiffrement

et pour le déchiffrement,

$1^{851} \bmod 1643 =$	1
$109^{851} \bmod 1643 =$	140
$890^{851} \bmod 1643 =$	513
$1453^{851} \bmod 1643 =$	151
$374^{851} \bmod 1643 =$	405

Fig. 3 – Déchiffrement

Lors du déchiffrement, sachant qu'il faut obtenir des blocs de 2 éléments (grâce au codage particulier de l'exemple), on a bien

01	14	05	13	15	14	05
A	N	E	M	O	N	E

3.2.1 Remarques

Il n'est pas très astucieux de choisir d'aussi petites valeurs car on peut retrouver d très facilement. En pratique, il faut prendre de très grandes valeurs de p et q. Pour retrouver ces grandes valeurs, il faudra alors utiliser le Jacobien et le test de Solovay-Strassen par exemple.

3.3 Sécurité

3.3.1 Attaques

Il existe trois approches pour attaquer le RSA :

- recherche par force brute de la clé (impossible étant donné la taille des données),
- attaques mathématiques (basées sur la difficulté de calculer Φ(n), la factorisation du module n) :
  - factoriser n=p*q et par conséquent trouver Φ(n) et puis d,
  - déterminer Φ(n) directement et trouver d,
  - trouver d directement.
- attaques de synchronisation (sur le fonctionnement du déchiffrement).

A l'heure actuelle, la factorisation connait de lentes améliorations au cours des années. La meilleure amélioration possible reste l'optimisation des algorithmes. Excepté un changement dramatique, le RSA-1024 restera sûr pour les prochaines années. D'après les projections, une clé de 2048 bits est sensée tenir jusque 2079 si on tient compte de la loi de Moore. Mais ces valeurs sont correctes uniquement si on respecte les propriétés de e, d, p et q.

3.3.2 Attaque de synchronisation (timing attack)

Développé dans le milieu des années 90, il s'agit d'exploiter les variations de temps pris pour effectuer certaines opérations (par exemple la multiplication par un petit ou un grand nombre). Plusieurs contre-mesures existent telles que l'emploi de temps constants d'élévation à une puissance, l'ajout de délais aléatoires, ou le fait de rendre non visibles les valeurs utilisées dans les calculs. Dans ce dernier cas, cela reviendrait à calculer :

$$(r^e * m^e)d \bmod n$$

3.4 Conseils d'utilisation du RSA

Pour garantir une bonne sécurité, il faut respecter certains règles telles que :

- Ne jamais utiliser de valeur n trop petite,
- 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,
- Si (d,n) est compromise ne plus utiliser n.

Recommended key sizes for RSA

Old standard:	
Individual users	~~512 bits (155 decimal digits)~~
New standard:	
Individual users	**768 bits** (231 decimal digits)
Organizations (short term)	**1024 bits** (308 decimal digits)
Organizations (long term)	**2048 bits** (616 decimal digits)

Fig. 6.4 – Taille des clés pour une utilisation sûre.

3.5 utiliser le théorème du reste chinois (CRT) lors du déchiffrement

Algorithme

On reçoit y et on doit calculer $x = y^d \bmod n$. On calcule

$dp = d \bmod (p - 1)$;

$dq = d \bmod (q - 1)$

puis

$u_p = q * (q^{-1} \bmod p) \bmod n$

$u_q = p * (p^{-1} \bmod q) \bmod n$

$x_p = y^{dp} \bmod p$

$x_q = y^{dq} \bmod q$

et finalement $x = x_p u_p + x_q u_q \bmod n$

4 El Gamal

C'est un algorithme à clef publique présent à la base de la norme U.S. de signature électronique. Il fut inventé par Taher ElGamal en 1984. Il est basé sur la difficulté de calculer des logarithmes discrets. Le problème du logarithme discret consiste à retrouver un entier λ tel que

$$h = g^{\lambda} \bmod p.$$

4.1 Principe du chiffrement

Soit un entier premier p très grand et $p - 1$ doit avoir un grand facteur premier. On produit :

- une clé secrète s, telle que $s \in (1...p - 2)$,
- une clé publique reposant sur l'entier p, un entier a premier avec p, et l'entier P tel que

$$P = a^s \bmod p$$

Le nombre a est pris tel que $a \in (0...p - 1)$ et $\forall\, k \in (1...p - 2)$:

$$a^k \neq 1 \bmod p$$

Soit un message M, avec $M < p$. On détermine un nombre aléatoire k qui n'est connu que de celui qui chiffre et différent à chaque message. On calcule alors

$$C_1 = a^k \bmod p$$
$$C_2 = M.P^k \bmod p$$

On obtient alors le message chiffré $C = (C_1,C_2)$. Le message chiffré est alors deux fois plus long que le message original.

4.2 Principe du déchiffrement

A la réception, on calcule

$$\begin{aligned} R_1 &= (C_1)^s \bmod p \\ &= a^{sk} \bmod p \\ &= P^k \bmod p \end{aligned}$$

Le destinataire possède la clé privée (s). Ayant P^k, on divise C_2 par cette valeur :

$$\begin{aligned} D_K(C) &= C_2/(R_1) \\ &= M.P^k \bmod p / P^k \bmod p \\ &= M \end{aligned}$$

P^k est donc considéré comme un masque appliqué sous forme multiplicative à M.

Pour décrypter le message, il faudra soit trouver directement un masque jetable, soit trouver la clé privée s, solution de $P = a^s \bmod p$ (et donc trouver le logarithme discret).

4.3 L'utilisation des courbes elliptiques

Il s'agit d'un concept proposé en 1985 par deux chercheurs Miller et Klobitz. Ce type de cryptographie, toujours basé sur le modèle asymétrique permet aussi bien de chiffrer que de signer. On utilise souvent l'abréviation ECC, pour Elliptic Curve Cryptography. Les clés utilisées sont plus courtes pour une sécurité égale ou supérieure. La théorie sous-jacente, ainsi que l'implémentation sont plus complexes, ce qui explique le fait que cette technologie soit moins répandue. Toutefois, de par la nécessité de traiter plus rapidement l'information, de gérer des quantités de données importantes et de miniaturiser au maximum, les avantages de cette technique sont intéressants.

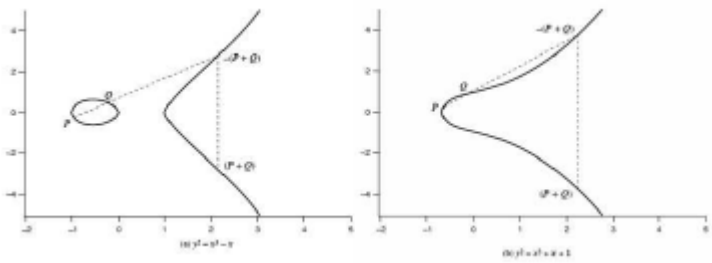
D'une manière générale, sur R, les courbes elliptiques seront considérées comme l'ensemble des couples (x,y) tels que :

$$y^2 = x^3 + ax + b$$

dont le discriminant $-(4a^3 + 27b^2)$ est non nul.

Pour la dessiner, pour a et b fixés, on calcule y tel que :

$$y = \sqrt{x^3 + ax + b}$$



Remarque : Contrairement à ce que l'on peut croire à première vue, il ne s'agit pas de travailler à partir d'ellipses.

4.3.1 Définition géométrique

Soit une opération (l'addition, +) pour l'ensemble E(a, b) tel que a et b répondent à la condition du discriminant.
Si 3 points sur une EC sont alignés, leur somme vaut O (point à l'infini).
1. O est l'identité pour l'addition : O = −O.
2. Pour n'importe quel point P + O = P.
3. L'opposé d'un point P(x, y) est P(x,−y)
4. Pour additionner 2 points P et Q, on trace la droite les reliant. Cela nous donne un point d'intersection R. On définit l'addition telle que P + Q = −R. En conséquence, on définit P + Q comme étant l'opposé de ce point R.

4.3.2 Les EC sur Z

Les variables et coefficients prennent des valeurs dans l'ensemble [0, p − 1] pour un certain nombre premier p, et où toutes les opérations sont calculées modulo p. L'équation devient

$$y^2 \bmod p = (x^3 + ax + b) \bmod p$$

Cette équation est par exemple satisfaite pour a = 1, b = 1, x = 9, y = 7 et p = 23.

$$7^2 \bmod 23 = (9^3 + 9 + 1) \bmod 23$$
$$49 \bmod 23 = 739 \bmod 23$$
$$3 = 3$$

On note Ep(a, b) l'ensemble des couples d'entiers (x, y) qui satisfont cette équation. On parle de groupe elliptique.
Exemple : Soient p = 23 et la courbe elliptique $y^2 = x^3 + x + 1$. On est donc dans $E_{23}(1, 1)$. Comme nous travaillons dans Zp, les couples (x, y) répondant à l'équation sont donnés à la figure suivante

(0, 1)	(6, 4)	(12, 19)
(0, 22)	(6, 19)	(13, 7)
(1, 7)	(7, 11)	(13, 16)
(1, 16)	(7, 12)	(17, 3)
(3, 10)	(9, 7)	(17, 20)
(3, 13)	(9, 16)	(18, 3)
(4, 0)	(11, 3)	(18, 20)
(5, 4)	(11, 20)	(19, 5)
(5, 19)	(12, 4)	(19, 18)

Pour tous points P,Q ∈ Ep(a, b) :
1. P + O = P
2. Si P = (x_P , y_P), alors P + (x_P ,$-y_P$) = 0 et (x_P ,$-y_P$) = −P.
Retour à l'exemple : Dans $E_{23}(1, 1)$, pour P = (13, 7),−P = (13,−7) = (13, 16)
3. Si P = (x_P , y_P) et Q = (x_Q, y_Q) avec P ≠ −Q, alors on détermine R = P + Q = (x_R, y_R) comme suit :

$$x_R = (\lambda^2 - x_P - x_Q) \bmod p$$
$$y_R = (\lambda\,(x_P - x_R) - y_P\,) \bmod p$$

où

$$\lambda = \begin{cases} \left(\frac{y_Q - y_P}{x_Q - x_P}\right) \bmod p \text{ si } P \neq Q \\ \left(\frac{3x_P^2 + a}{2y_P}\right) \bmod p \text{ si } P = Q \end{cases}$$

4. La multiplication est définie comme une répétition d'additions (ex : 3P = P + P + P)
Retour à l'exemple : Soient P = (3, 10) et Q = (9, 7) dans $E_{23}(1, 1)$. Il vient

$$\lambda = (7 - 10)/(9 - 3\,) \bmod 23 = (-3/6\,) \bmod 23 = (-1/2\,) \bmod 23 = 11$$
$$x_R = (11^2 - 3 - 9) \bmod 23 = 109 \bmod 23 = 17$$
$$y_R = (11(3 - 17) - 10) \bmod 23 = -164 \bmod 23 = 20$$

Et ainsi, P + Q = (17, 20). Pour trouver 2P, on a

$$\lambda = (3(3^2) + 1)/(2 * 10\,) \bmod 23 = (5/20) \bmod 23 = (1/4) \bmod 23$$
$$x^R = (6^2 - 3 - 3) \bmod 23 = 30 \bmod 23 = 17$$
$$y_R = (6(3 - 7) - 10) \bmod 23 = -34 \bmod 23 = 12$$

et donc 2P = (7, 12).

4.3.3 La cryptographie sur courbes elliptiques (ECC)

Pour utiliser les courbes elliptiques en cryptographie, il faut trouver un problème difficile (tel que la factorisation d'un produit en ses facteurs premiers dans le cas du RSA).

Considérons l'équation $Q = kP$

où $Q, P \in Ep(a, b)$ et $k < p$.

Il est facile de calculer Q connaissant k et P, mais il est difficile de déterminer k si on connait Q et P. Il s'agit du problème du logarithme discret pour les courbes elliptiques : $\log_P (Q)$.

Dans une utilisation réelle, le k est très grand, rendant l'attaque par force brute inutilisable (rappelons qu'a priori, l'attaque par force brute est toujours possible. . .).

4.3.3.1 ECC pour l'échange de clés

Soit un grand entier premier q et les paramètres a et b satisfaisant l'équation $y^2 \bmod q = (x^3 + ax + b) \bmod q$. Cela nous permet de définir Eq(a, b).

Prenons ensuite un point de départ $G(x_1, y_1)$ dans Eq(a, b) dont l'ordre n est élevé. L'ordre n d'un point sur une EC est le plus petit entier positif tel que $nG = O$.

Eq(a, b) et G sont rendu publiques.

L'échange d'une clé par ECC entre deux entités A et B se déroule comme suit :

- A choisit un n_A inférieur à n qui sera sa clé privée. A génère alors sa clé publique $P_A = n_A \times G$.
- B choisit un n_B inférieur à n qui sera sa clé privée. B génère alors sa clé publique $P_B = n_B \times G$.
- A génère la clé secrète $K = n_A \times P_B$ et B génère la clé secrète $K = n_B \times P_A$.

Exemple:

– Soient p = 211, $Ep(0,-4)(\Rightarrow y^2 = x^3 - 4)$ et G = (2, 2). On calcule que 240G = O et donc n = 240.

– A choisit $n^A = 121$, ce qui lui donne $P_A = 121(2, 2) = (115, 48)$.

– B choisit $n_B = 203$, ce qui lui donne $P_B = 203(2, 2) = (130, 203)$.

– La clé secrète K générée est $121(130, 203) = 203(115, 48) = (161, 69)$.

4.3.3.2 ECC pour chiffrer des données

Même si la cryptographie par courbes elliptiques est souvent employée pour l'échange d'une clé symétrique, elle est aussi utilisée pour chiffrer directement les données. Voici un exemple de cryptosystème les utilisant. Il faudra ici encoder le texte clair m comme un point Pm de coordonnées x et y. C'est ce point qui sera chiffré. Il faut ici aussi rendre publique un point G et un groupe elliptique Eq(a, b). Les utilisateurs doivent également choisir une clé privée et générer la clé publique correspondante.

Pour chiffrer le message, A détermine aléatoirement un nombre entier positif k et produit C_m comme un couple de points tel que $C_m = \{kG, P_m + kP_B\}$

On remarquera l'utilisation de la clé publique de B. Pour déchiffrer, B devra multiplier le premier point par sa clé privée, et soustraire le résultat au second point reçu :

$$P_m + kP_B - n_B(kG) = P_m + k(n_BG) - n_B(kG) = P_m$$