7 – Cryptographie à clef publique – II

Sac-à-dos et cryptographie asymétrique

- le problème du « Sac-à-dos » est un problème d'optimisation combinatoire *(Knapsack problem en anglais)*

  On considère une collection d'objets possédant chacun une valeur et un poids. Comment remplir le sac de sorte à maximiser la valeur qu'il contient sans excéder sa capacité en poids ?

- un problème est calculatoirement difficile s'il n'existe pas d'algorithme de résolution de ce problème en temps polynomial pour un modèle de calcul déterministe
- les problèmes NP-complets sont donc de bons candidats pour construire des fonctions à sens unique (pourvu qu'on puisse trouver une trappe pour déchiffrer)
- Hellman et son étudiant Merkle ont proposé leur chiffre en 1978, un an après RSA
- *a priori* plus simple que RSA, il a été montré vulnérable par Shamir en 1982
- le chiffrement et le déchiffrement sont plus rapides que pour les cryptosystèmes basés sur le problème du log discret
- à ce jour, tous les cryptosystèmes asymétriques basés sur les problèmes de sac-à-dos ont été cryptanalysés.

Subset sum problem (pb de la somme des sous-ensembles)

Pour leur cryptosystème, Merkle et Hellman ont choisi le problème (dérivé du sac-à-dos) suivant :

- *Donnée*
  un n-uplet $A = (a_1, \ldots, a_n)$ d'entiers distincts et un entier k
- *Problème*
  Existe-t-il un sous-ensemble de A dont la somme des éléments est égale à k ?

  (le même problème s'énonce sur les entiers relatifs en prenant $k = 0$)

```
def subsetSum(L, k) :
    if k == 0 : return True
    for i in range(len(L)) :
        if subsetSum (L[:i] + L[i+1:], k - L[i]) :
            print(L[i]) ; return True
    return False
```

Exemple

Soit $A = (236, 2022, 1834, 571, 27, 1286, 335, 991)$ et $k = 3611$. On trouve que :

$k = 236 + 2022 + 27 + 335 + 991$.

Suites super-croissantes

- en toute généralité, la résolution d'un tel problème requiert l'énumération de la somme des éléments de chaque sous-ensemble
- en posant $n = |A|$, la complexité d'un tel algorithme est : $\mathcal{O}(2^n)$

  pour $n = 52$ avec 10^6 opér./s, il faudrait environ 1.4271 siècle ...

- c'est justement l'argument de Merkle & Hellman concernant la sûreté de leur système
- en outre, ce problème est facile pour une certaine variété de suites : ceci va permettre d'envisager une trappe
- une suite d'entiers naturels $A = (a_i)_{1 \leq i \leq n}$ est super-croissante si chaque entier est supérieur ou égal à la somme de ses prédécesseurs :

$$\forall\, 1 < i \leq n \quad \left(\sum_{j=1}^{i-1} a_j \right) \leq a_i$$

- ainsi, le problème de la somme des sous-ensembles se résout en complexité polynomiale pour ces suites.

Résolution

Exemple

- soit la suite super-croissante suivante L, existe-t-il un sous-ensemble de L tel que la somme de ses éléments égale 1111 ?

$$L = (5, 7, 22, 37, 77, 158, 309, 676, 1598)$$

- algorithme linéaire limité aux suites super-croissantes :

```
def linearSubsetSum (L, k) :   #  que si L est super-croissante !
    res = []
    s = k
    for i in range(len(L)-1,-1,-1) :
        if s >= L[i] :
            res.append(L[i])
            s = s - L[i]
    if s == 0 :
        return (res,k)
    else :
        return (False,k)
```

- il en résulte : $([676, 309, 77, 37, 7, 5], 1111)$
  tandis qu'avec 1234, il n'y aurait pas eu de solution : $(False, 1234)$

Chiffre de Merkle-Hellman

- préparation des clefs :
  1. le destinataire Bob choisit une suite *strictement* super-croissante $B = (B_i)_{1 \leq i \leq n}$ et un nombre $m \gg \sum_{i=1}^{n} B_i$
  2. il choisit un entier e vérifiant $0 < e < m$ premier avec m (*i.e.* $\text{pgcd}(e,m) = 1$)
  3. sa clef privée est composée de (B, d, m) avec $d \equiv e^{-1} \mod m$
  4. il calcule sa clef publique $A = (A_i)_{1 \leq i \leq n}$ (A n'est plus super-croissante) et la publie :
     $$\forall 1 \leq i \leq n \quad A_i = (e \,.\, B_i) \mod m$$
- chiffrement du message M :
  Alice chiffre son message binaire $M = M_1 \ldots M_n$ (de longueur imposée n) avec la clef publique $A = (A_i)_{1 \leq i \leq n}$ de Bob et lui envoie le chiffré C suivant :
  $$C = E(M) = \sum_{i=1}^{n} M_i \,.\, A_i$$
- déchiffrement de C :
  Bob calcule à partir de sa clef privée (B, d, m) et du chiffré C envoyé par Alice :
  $$S = d \,.\, C \mod m$$
  la résolution du problème de la somme des sous-ensembles avec la somme S et la suite strictement super-croissante B va lui permettre de reconstituer le message $M = D(C)$ car :
  $$S = \sum_{i=1}^{n} M_i \,.\, B_i$$

Exemple

- préparation des clefs :
  1. le destinataire Bob choisit B *strictement* super-croissante (fixant n) et le module m :
     $$B = (3, 5, 10, 25, 57, 119, 243, 496, 1002)$$
     $$n = 9$$
     $$m = 2020$$
  2. il choisit e tel que $\text{pgcd}(e,m) = 1$:
     $$e = 777$$
  3. il calcule l'inverse d de e modulo m :
     `bezout(m,e)=(1,-5,13)`
     $$d = 13$$
  4. il calcule sa clef publique A et la publie :
     $$A = (311, 1865, 1710, 1245, 1869, 1563, 951, 1592, 854)$$
- chiffrement du message d'Alice $M = 110101101$:
  $$C = 311 + 1865 + 1245 + 1563 + 951 + 854 = 6789$$
- Bob calcule S avant de déchiffrer C :
  $$S = (13 \,.\, 6789) \mod 2020 = 1397$$
  `linearSubsetSum(B,1397)= ([1002, 243, 119, 25, 5, 3], 1397)`
  $$M = 110101101$$

Chiffrement El Gamal

- chiffre proposé par T. El Gamal en 1984, utilisé entre autres dans GNU PRIVACY GUARD et aussi dans les versions récentes de PGP
- il est basé sur le problème du logarithme discret (*cf.* Cours 6)
- il s'agit d'un chiffrement aléatoire dans la mesure où un nombre aléatoire intervient pour chiffrer :
  - 2 messages identiques ne seront pas forcément chiffrés pareils
  - ce caractère aléatoire est un plus pour la sûreté du chiffre
- la sécurité est garantie sur des entiers plus petits que pour RSA et il est plus rapide
- un inconvénient : la taille du chiffré est 2 fois plus grande que celle du clair ...
- ce chiffre est transposable pour toute sorte de groupes où le problème du logarithme discret est encore plus difficile : c'est notamment le cas pour le groupe des points d'une courbe elliptique (*e.c.*)
- chiffrement à ne pas confondre avec l'algorithme de signature El Gamal devenu le standard DSA (*cf.* Cours 9 à venir).

Chiffrement El Gamal (mise en œuvre)

- préparation des clefs :
  1. le destinataire Bob choisit un grand entier premier p et un générateur α du groupe multiplicatif $\mathbb{Z}_p^*$
  2. il choisit pour clef privée un entier k vérifiant $1 < k < p$
  3. il calcule $\beta = \alpha^k \mod p$ et publie sa clef publique (p, α, β)
- chiffrement du message M :

  pour son message $M \in \mathbb{Z}_p^*$, Alice choisit un entier aléatoire r, calcule $c_1 = \alpha^r \mod p$ et $c_2 = M.\beta^r \mod p$ et envoie à Bob le chiffré $C = (c_1, c_2)$
- déchiffrement de C :

  Bob déchiffre $C = (c_1, c_2)$ en calculant :

$$M = (c_2 \,.\, c_1^{-k}) \mod p$$

  en effet :

$$(c_2 \,.\, c_1^{-k}) \mod p = (M \,.\, \beta^r \,.\, \alpha^{-rk}) \mod p = (M \,.\, \beta^r \,.\, \beta^{-r}) \mod p = M$$

Exemple

- préparation des clefs :
  1. le destinataire Bob choisit un entier premier $p = 17$ et un générateur $\alpha = 3$ du groupe multiplicatif $\mathbb{Z}_p^*$
  2. il choisit pour clef privée un entier $k = 6$ vérifiant $1 < k < p$
  3. il calcule $\beta = 15$ et publie sa clef publique $(p, \alpha, \beta) = (17, 3, 15)$

Exemple

- préparation des clefs :
  1. le destinataire Bob choisit un entier premier $p = 17$ et un générateur $\alpha = 3$ du groupe multiplicatif $\mathbb{Z}_p^*$
  2. il choisit pour clef privée un entier $k = 6$ vérifiant $1 < k < p$
  3. il calcule $\beta = 15$ et publie sa clef publique $(p, \alpha, \beta) = (17, 3, 15)$
- chiffrement du message M :

  pour son message $M = 8$ avec $0 < 8 < p$, Alice choisit un entier aléatoire $r = 11$, calcule $c_1 = 7$ et $c_2 = 4$ et envoie à Bob le chiffré $C = (7, 4)$

Exemple

- préparation des clefs :
  1. le destinataire Bob choisit un entier premier $p = 17$ et un générateur $\alpha = 3$ du groupe multiplicatif $\mathbb{Z}_p^*$
  2. il choisit pour clef privée un entier $k = 6$ vérifiant $1 < k < p$
  3. il calcule $\beta = 15$ et publie sa clef publique $(p, \alpha, \beta) = (17, 3, 15)$
- chiffrement du message M :

  pour son message $M = 8$ avec $0 < 8 < p$, Alice choisit un entier aléatoire $r = 11$, calcule $c_1 = 7$ et $c_2 = 4$ et envoie à Bob le chiffré $C = (7, 4)$
- déchiffrement de C :

  Bob reçoit $C = (7, 4)$ et le déchiffre en calculant d'abord l'inverse de ${c_1}^k \mod p$:

$${c_1}^k \mod p = 7^6 \mod 17 = 9$$

$$\texttt{bezout}\ (c_1^k \mod p,\ p) = \texttt{bezout}\ (9,\ 17) = (1, 2, -1)$$

$$M = c_2 \,.c_1^{-k} \mod p = 4 \,.2 \mod 17 = 8$$

La cryptographie asymétrique est incontournable

- la cryptographie asymétrique est lente et coûteuse, cela dit, elle est très utilisée
- son usage le plus fréquent est comme préambule à la cryptographie symétrique, juste le temps de s'échanger une clef secrète

Exemples de protocoles qui l'utilisent :

- HTTPS
- PGP, GPG
- Internet Key Exchange
- ZRTP (protocole VoIP)
- Secure Socket Layer
- SILC
- SSH
- Secure Electronic Transaction (SET)
- Bitcoin
- ...

Mise en gage

- imaginons qu'Alice veuille prouver qu'elle détient une information ou un choix
- le but est d'en convaincre Bob ou Victor sans la lui révéler, ou seulement *a posteriori*
- mais que Bob ou Victor puissent vérifier ce choix ...

- c'est un problème qui s'apparente à de la cryptographie moderne (*cf.* TD n° 5)
- ce n'est pas de la cryptographie asymétrique
- ce sont des protocoles basées sur des preuves à divulgation nulle de connaissance (Zéro-Knowledge proof)
- on peut le résoudre en se servant du problème du logarithme discret

les fonctions de hachage permettent aussi la mise en gage (*cf.* TD n° 6 à venir)

Protocole de Schnorr (1989)

Paramètres publics

- on dispose d'un groupe dans lequel le problème du logarihme discret est difficile, par exemple $\mathbb{Z}_p^*$ avec p un grand premier
- soit g un générateur du groupe

Secret

- Alice choisit une information x dans $\mathbb{Z}_p^*$ et calcule $X = g^x \mod p$
- elle transmet seulement X à Bob et Victor

Engagement

- Alice tire au hasard r dans $\mathbb{Z}_p^*$ et calcule $R = g^r \mod p$
- elle transmet R à Bob et Victor

Défi

- Bob choisit un entier y de $\mathbb{Z}_p^*$ et l'envoie à Alice

Ouverture

- Alice calcule $Y = r - xy$ et elle envoie le gage Y
- il est accepté si $R = g^Y.X^y$

A suivre ...