Fiche 004 Niveau : 0 version 2 (1/05/2010) **Gris**

Calculer une puissance modulo n

1 Algorithme square and multiply

Il s'agit de calculer **dans un groupe** a^n (ou na si on note l'opération additivement). Ceci rentre exactement dans le cadre de l'étude faite dans la fiche *fichecrypto_002_V2* sur l'algorithme de Hörner. Notons $n_k n_{k-1} \cdots n_1 n_0$ la **décomposition binaire** de l'exposant n. Alors en suivant l'algorithme de Hörner pour calculer n on voit que lorsqu'on multiplie par 2 la valeur courante de l'exposant on élève au carré la valeur courante de la puissance, et quand on ajoute 1 à la valeur courante de l'exposant on multiplie la valeur courante de la puissance par a. L'algorithme, qui suit exactement l'algorithme de Hörner de reconstitution de l'exposant est donc le suivant :

Hörner
entrées : un tableau N de $K+1$ bits représentant n en binaire
sortie : le nombre $n = \sum_{i=0}^{K} N[i]2^i$

début
  $n \leftarrow 0$;
  $i \leftarrow K$;
  Tant que $i \geq 0$ **faire**
    $n \leftarrow 2 * n$;
    si $N[i] == 1$
      alors $n \leftarrow n + 1$;
    finsi
    $i \leftarrow i - 1$;
  fintq
  retourner n ;
fin

Square and multiply
entrées : un tableau N de $K+1$ bits représentant n en binaire
sortie : le nombre $P = a^n$

début
  $P \leftarrow 1$;
  $i \leftarrow K$;
  tant que $i \geq 0$ **faire**
    $P \leftarrow P^2$;
    si $N[i] == 1$
      alors $P \leftarrow P * a$;
    finsi
    $i \leftarrow i - 1$;
  fintq
  retourner P ;
fin

Remarque : Si l'élévation à la puissance n se fait dans un groupe $\mathbb{Z}/p\mathbb{Z}$ il ne faut pas oublier de prendre le modulo après chaque opération multiplicative sous peine d'avoir des débordements de mémoire. Il faudra donc calculer $P^2 \mod p$ et $P * a \mod p$ dans l'algorithme précédent.

2 Aspect récursif de l'algorithme

L'algorithme square and multiply peut être compris sous l'aspect récursif suivant, même si évidemment il faut le programmer sous sa forme itérative :

```
fonction puissance(a, n)

si n vaut 0
   alors retourner 1
   sinon si n est pair
      alors retourner (puissance(a, n/2))^2
      sinon retourner a * (puissance(a, (n - 1)/2))^2
   finsi
finsi
```

3 Complexité

La comparaison entre l'algorithme de Hörner et l'algorithme square and multiply montre que leurs nombres d'itérations sont identiques. On voit sur la description de l'algorithme que le nombre de boucles est donné par la taille de n et à chaque tour de boucle on fait une ou deux multiplications (une élévation au carré plus éventuellement une multiplication par a). Si on note $t = \lfloor \ln(n) + 1 \rfloor$ la taille de n, Le nombre de multiplications est donc majoré par $2t$. On a donc ici un algorithme qui utilise $O(t)$ opérations simples.

4 Considération des bits de bas en haut

Dans l'algorithme donné précédemment, on a traité les bits de la puissance en commençant par ceux de poids forts. On peut décrire un algorithme fondé sur un principe analogue qui traite d'abord les bits de poids faible ainsi qu'on l'a décrit dans la fiche *fichecrypto_002_V2* :

```
entrées : Un entier n
sortie : S = a^n.

début
   A ← a ;
   S ← e ;
   N ← n ;
   tant que N ≥ 1 faire
      si N pair alors
         A ← A.A ;
         N ← N/2 ;
      sinon
         S ← S.A ;
         N ← N − 1 ;
      finsi ;
   fintq ;
   retourner S ;
fin ;
```

FIG. 1 – Algorithme de bas en haut

Bien entendu, là encore les multiplications se font dans le groupe considéré. En particulier si le groupe est $\mathbb{Z}/p\mathbb{Z}$ il ne faut pas oublier de multiplier modulo p sous peine de débordement mémoire.