Introduction à la cryptographie - TP 3

1 Notion de partage de secret

Avant que ne soit créé l'algorithme **RSA**, la communauté cryptographique se doutait de l'existence de la possibilité de pouvoir échanger des messages sur un réseau hostile sans que les deux partis, Alice et Bob, n'aient, au préalable, communiqué sur un domaine sécurisé.
L'analogie était la suivante : Alice veut envoyer un message à Bob. Alice enferme son message dans une boite qu'elle verrouillera à l'aide de son cadena. Elle envoie cette boite à Bob, qui va à son tour ajouter son propre cadena à la boite. La boite est donc protégée par deux cadenas : celui d'Alice et celui de Bob. Bob renvoie le colis à Alice qui retirera son cadena avant de le retourner à Bob. Bob enlève sa protection (son cadena), ouvre la boite et lit le message. Ce procédé de communication était la "preuve" que l'on pouvait converser dans un environnement inhospitalier. Mais encore fallait-il trouver la façon de le faire « efficacement ».
Si $\mathbf{m}$ est le message qu'Alice souhaite envoyer à Bob, $\mathbf{K_A}$ le cadena d'Alice (que l'on appellera ... clef privée), $\mathbf{K_B}$ le cadena de Bob et $\mathbf{F}$ la fonction de chiffrement alors le protocole décrit précédemment peut se formuler de la manière suivante :

1. Alice ajoute son cadena à $\mathbf{m}$: $\mathbf{F}(\mathbf{m}, \mathbf{K_A})$
2. Bob ajoute son cadena : $\mathbf{F}(\mathbf{F}(\mathbf{m}, \mathbf{K_A}), \mathbf{K_B})$
3. Alice retire son cadena : $\mathbf{F}(\mathbf{m}, \mathbf{K_B})$
4. Bob retire son cadena et obtient $\mathbf{m}$

Pour que cela puisse fonctionner, il faut que les opérations de chiffrements puissent commuter entre elles. Autrement dit que

$$\mathbf{F}(\mathbf{F}(\mathbf{m}, \mathbf{K_A}), \mathbf{K_B}) = \mathbf{F}(\mathbf{F}(\mathbf{m}, \mathbf{K_B}), \mathbf{K_A})$$

Certes, les chiffrements de type **César** ou les chiffrements **linéaires** ($c = a \times m$) offrent cette propriété mais sont bien trop vulnérables pour être adoptés dans la cryptographie d'aujourd'hui.

Il fallait donc trouver un moyen de raffiner le processus précédent afin de lui fournir une vraie sécurité. L'idée fut alors d'introduire la notion de « partage de secret ». Alice et Bob se mettront d'accord une information secrète K_{AB} sans qu'ils puissent en décider la véritable teneur.
La métaphore souvent usitée pour expliquer la technique est celle du mélange de peinture. Alice possède son propre tube de gouache d'une certaine couleur. Elle garde cette information secrète. De même pour Bob, qui ne révèlera pas

la couleur de sa peinture. Au milieu d'eux, deux pots de peinture de couleur verte. Alice va prendre l'un d'eux et ajoutera un peu de sa peinture (la peinture ajoutée est une quantité fixe) . Bob fera de même dans le deuxième pot encore inutilisé. Il y a donc deux mélanges, ces mélanges sont publics. Bob prendra le pot d'Alice et vis versa. Bob et Alice verseront dans les récipients leurs couleurs respectives. Ils devraient individuellement obtenir la même **nouvelle** couleur, qui est le **secret commun** qu'ils partagent tous les deux. Une personne qui aurait eu accès aux premiers mélanges d'Alice et Bob ne pourra pas parvenir à extraire **facilement** les couleurs choisies par les deux compères.

Si nous tentons de formaliser la démarche, nous obtenons la procédure qui suit :

1. Alice ajoute sa peinture $\mathbf{P_A}$ au pot vert $\mathbf{V}$: $\mathbf{V} + \mathbf{P_A}$.
2. Bob ajoute sa peinture $\mathbf{P_B}$ au pot vert $\mathbf{V}$: $\mathbf{V} + \mathbf{P_B}$.
3. Alice ajoute sa peinture $\mathbf{P_A}$ au pot de Bob : $(\mathbf{V} + \mathbf{P_B}) + \mathbf{P_A}$
4. Bob ajoute sa peinture $\mathbf{P_B}$ au pot d'Alice : $(\mathbf{V} + \mathbf{P_A}) + \mathbf{P_B}$

Évidemment, il faut que $(V + P_B) + P_A = (V + P_A) + P_B$ mais aussi qu'il soit **difficile**, connaissant $\mathbf{V} + \mathbf{P_A}$ ou $\mathbf{V} + \mathbf{P_B}$, de remonter à P_A ou P_B. La labeur consiste maintenant à trouver une telle fonction.

2 Logarithme Discret

Sauriez-vous trouver aisément l'entier n tel que $5^n = 100 \bmod 127$?
Ou encore $524^n = 319730 \bmod 524287$?
Avant même que vous ne commenciez à chercher, la réponse est non. Même s'il existe des astuces évitant une recherche exhaustive sur n, il n'existe, en revanche, pas de procédures efficaces pour faire face à cette besogne. Ce problème se nomme « discrete logarithm problem » (problème du logarithme discret). L'exemple ci-dessus a été donné dans les entiers mais tout **groupe** peut être employé. Cela dit, tous les groupes n'offrent pas les propriétés satisfaisantes qui rendent le **DLP** difficile. Les courbes elliptiques, très en vogues aujourd'hui, puisent leur résistance dans ce problème **réputé** rude.
L'échange précédent entre Alice et Bob peut être réalisé à l'aide d'une fonction du type $f(a) = a^n \bmod m$ (avouez que c'est quand même plus pratique que des pots de peinture) et l'écueil de l'attaquant reste, à l'évidence, celui du **DLP**.

1. Choisir un élément g du groupe $\mathbf{G}$. Cet élément est public
2. Alice ajoute sa peinture $\mathbf{P_A}$ au pot vert $\mathbf{V}$: $\mathbf{g^{P_A}}$.
3. Bob ajoute sa peinture $\mathbf{P_B}$ au pot vert $\mathbf{V}$: $\mathbf{g^{P_B}}$.

4. Alice ajoute sa peinture $\mathbf{P_A}$ au pot de Bob : $(\mathbf{g}^{\mathbf{P_A}})^{\mathbf{P_B}}$
5. Bob ajoute sa peinture $\mathbf{P_B}$ au pot d'Alice : $(\mathbf{g}^{\mathbf{P_B}})^{\mathbf{P_A}}$

On vérifie aisément que $(\mathbf{g}^{\mathbf{P_B}})^{\mathbf{P_A}} = (\mathbf{g}^{\mathbf{P_A}})^{\mathbf{P_B}}$.

3 Casser le Logarithme Discret ? Un pas de géant !

Si l est l'ordre d'un groupe $\mathbf{G}$ et si $h = g^n$ (n un entier, g un générateur du groupe $\mathbf{G}$) alors n peut s'écrire comme

$$n = u + sv$$

où $s = \lceil l \rceil + 1$ et $u < s$ et $v < s$.

Nous allons ici décrire un algorithme (**Baby Step, Giant Step**) permettant de trouver n en $O(\sqrt{l})$ (la recherche exhaustive est évidemment d'une complexité de $O(l)$).
Notons les égalités suivantes :

1. $h = g^n = g^{u+sv}$
2. $h = g^u \times (g^s)^v$
3. $h \times (g^{-1})^u = (g^s)^v$

L'Algorithme :

1. Dans un premier temps (Baby steps), il faut calculer et stocker l'ensemble $\mathbb{E}$ des $h \times (g^{-1})^u$ pour tous les $u < s$.
2. Ensuite, (Giant steps) pour chaque $v < s$, calculer $A = (g^s)^v$.
3. Vérifiez si $A \in \mathbb{E}$. Si oui, rendre les u et v correspondants.

Bien sur, il faut que l'opération « $A \in \mathbb{E}$ » soit rapide. On peut imaginer, par exemple, une recherche dichotomique.

L'un des principaux soucis est qu'il faut **énormément** de mémoire jouissant d'un accès rapide (typiquement, de la RAM) pour stocker $\mathbb{E}$. Pour $l \approx 2^{80}$, il faudrait environ ... 20000 Go ...