EPFL - Information, Calcul et Communication - Partie théorique

Série 13: Solutions

1 Chiffrer et déchiffrer

a) Le fait d'utiliser un alphabet de 28 lettres plutôt qu'un alphabet de 26 lettres augmente le nombre de chiffrements possibles de 26! à 28!, donc d'un facteur multiplicatif = $28!/26! = 28 \times 27 = 756$. Le temps de déchiffrement sera donc de 756 heures.

Note: Il y des moyens plus efficaces que l'attaque par force brute pour décrypter un tel chiffrement! (notamment en analysant les fréquences d'apparition des lettres dans le message chiffré et en essayant de faire correspondre celles-ci aux fréquences communes des lettres en français)

b) Doubler la taille de la clé K de 20 à 40 bits multiplie le nombre de possibilités par un facteur $2^{20} \simeq 10^6$. Avec un ordinateur 100 fois plus puissant que le premier, le temps de déchiffrement sera donc environ $10'000$ fois plus long ($\simeq 50'000$ minutes $\simeq 833$ heures).

2 Protocole d'échange de clé de Diffie-Hellman avec 3 personnes

Le protocole pour se mettre d'accord sur une clé commune est le suivant:

- Les 3 personnes (disons Alice, Bob et Charlie) se mettent d'accord sur un grand nombre premier P et un nombre Q entre 1 et $P-1$. Ces 2 nombres sont publics. Puis Alice, Bob, Charlie choisissent respectivement N_1, N_2, N_3, des nombres compris entre 1 et $P-1$. Ces 3 nombres restent privés.

- A la suite de cela, Alice, Bob, Charlie calculent respectivement $N_4 = Q^{N_1} \pmod P$, $N_5 = Q^{N_2} \pmod P$, $N_6 = Q^{N_3} \pmod P$ et publient ces nombres.

- Lors d'un deuxième round de calcul:

Alice calcule $N_7 = N_5^{N_1} \pmod P = Q^{N_2 N_1} \pmod P$

Bob calcule $N_8 = N_6^{N_2} \pmod P = Q^{N_3 N_2} \pmod P$

Charlie calcule $N_9 = N_4^{N_3} \pmod P = Q^{N_1 N_3} \pmod P$

Ces trois nombres N_7, N_8 et N_9 sont également publiés.

- Pour arriver à une clé K commune, un dernier round est nécessaire:

Alice calcule $N_8^{N_1} \pmod P = Q^{N_3 N_2 N_1} \pmod P = K$

Bob calcule $N_9^{N_2} \pmod P = Q^{N_1 N_3 N_2} \pmod P = K$

Charlie calcule $N_7^{N_3} \pmod P = Q^{N_2 N_1 N_3} \pmod P = K$

On peut vérifier de plus qu'Eve, en possession des nombres $P, Q, N_4, N_5, N_6, N_7, N_8$ et N_9 mais pas de N_1, N_2 ou N_3 ne peut déchiffrer la clé K à moins d'être capable de résoudre le problème du logarithme discret.

3 Chiffrement d'El Gamal (1984)

Pour déchiffrer le message M à partir de $Q^R \pmod P$ et $M \cdot N_2^R \pmod P$, Bob, qui connaît N_1, calcule d'abord $(Q^R)^{N_1} \pmod P = Q^{N_1 R} \pmod P = N_2^R \pmod P$ et effectue la division:

$$\frac{M \cdot N_2^R}{N_2^R} \pmod P = M$$

Une question légitime est ici: comment effectuer concrètement une division en arithmétique modulaire? Il est bon de savoir que pour tout nombre X compris entre 1 et $P-1$, on a $X^{P-1} (\text{mod } P) = 1$ (c'est l'énoncé du petit théorème de Fermat: voir exercice 4). Donc $X^{P-2} \cdot X (\text{mod } P) = 1$, ce qui se lit encore: X^{P-2} est l'inverse de $X (\text{mod } P)$. Donc diviser un nombre Y par un nombre X revient en fait à multiplier Y par X^{P-2} en arithmétique modulaire.

Si Eve vient maintenant à intercepter le message envoyé par Alice, à savoir $Q^R (\text{mod } P)$ et $M \cdot Q^{N_1 R} (\text{mod } P)$, mais ne connaît pas N_1, elle ne pourra pas répéter la même opération que Bob. Et pour apprendre la valeur de N_1 (à partir des valeurs de P, Q et N_2), Eve devrait être capable de résoudre le problème du logarithme discret, ce qui est difficile.

Note: On peut montrer également que pour envoyer plusieurs messages $M_1, M_2, M_3, \ldots$ à Bob en utilisant ce chiffrement, Alice doit retirer chaque fois au hasard des nombres $R_1, R_2, R_3, \ldots$ pour assurer la confidentialité de ses messages.

4 Pour le plaisir: trouver un grand nombre premier*

a) Le nombre de nombres premiers plus petits ou égaux à X étant donné (approximativement) par $\frac{X}{\log X}$, on en déduit que:

- si $X = 10^n - 1$, ce nombre vaut approximativement $\frac{10^n}{n \log(10)}$

- si $X = 10^{n-1}$, ce nombre vaut approximativement $\frac{10^{n-1}}{(n-1) \log(10)}$

Donc le nombre de nombres premiers à n chiffres vaut approximativement $\frac{10^n}{n \log(10)} - \frac{10^{n-1}}{(n-1) \log(10)} \simeq \frac{9 \cdot 10^{n-1}}{n \log(10)}$ et la proportion de ceux-ci entre 10^{n-1} et $10^n - 1$ vaut $\frac{1}{n \log(10)}$.

En moyenne, il faut donc tirer au hasard de l'ordre de n nombres à n chiffres pour avoir la chance de tomber sur un nombre premier.

b) Les nombres divisibles par 2, 3 ou 5 sont très facilement identifiables (en effet, les premiers sont pairs, les deuxièmes ont pour somme de leurs chiffres un multiple de 3 et les troisièmes finissent par 0 ou 5). Leur proportion parmi les autres nombres est de 22/30 (il suffit de compter explicitement le nombre de ceux-ci entre 1 et 30=ppcm(2,3,5)). Ceci nous permet donc d'éliminer rapidement plus des 2/3 des tirages effectués ci-dessus.

c) Si $X = 1$, on ne peut rien déduire de spécial du petit théorème de Fermat: en effet, on sait que la conclusion est vraie, mais rien ne nous dit que l'hypothèse l'est aussi. Si par contre $X \neq 1$, alors par la contraposée du petit théorème de Fermat, on sait que N n'est pas premier et qu'il faut chercher un autre nombre.

d) Par le deuxième résultat de l'énoncé, on sait que si N n'est pas premier, alors la chance qu'on obtienne $X = 1$ est inférieure à 1/2. De manière analogue, la chance qu'on obtienne $X_1 = X_2 = \ldots = X_k = 1$ est inférieure à $1/2^k$. En choisissant le nombre de tirages k suffisamment grand, on peut donc être très confiant, si $X_1 = X_2 = \ldots = X_k = 1$, que N est effectivement premier. Si par contre il existe $1 \leq j \leq k$ tel que $X_j \neq 1$, alors à nouveau, on sait que N n'est pas premier.

e*) Les opérations d'exponentation effectuées aux étapes c) et d) peuvent être exécutées efficacement; plus précisément, on peut montrer que le nombre d'opérations nécessaires pour identifier si un nombre à n chiffres est premier est un $\Theta(n^3)$, et vu qu'il faut tirer de l'ordre de n nombres au hasard pour tomber sur un nombre premier (cf partie a), la complexité totale de l'opération est un $\Theta(n^4)$. L'algorithme n'est cependant pas garanti d'aboutir 100% du temps, mais en pratique, il fonctionne très bien! Un autre algorithme (AKS) a été démontré fonctionner en temps polynomial avec des garanties de résultat à 100% (démontrant ainsi que le problème de la primalité est dans P). Cependant, le temps de calcul de ce second algorithme est $\Theta(n^6)$, ce qui le rend inutilisable en pratique.

Note: Un détail manque dans l'analyse ci-dessus. Il existe quelques (rares) nombres N, appelés nombres de Carmichael, qui ne sont pas premiers et pour lesquels $Q^{N-1} (\text{mod } N) = 1$ pour tout nombre Q entre 1 et $N-1$. Pour ces nombres, le petit théorème de Fermat ne nous est d'aucune aide; il faut donc modifier l'algorithme ci-dessus pour prendre ces nombres en compte si on veut faire bien les choses: voir le *test de Miller-Rabin*, dont la complexité est similaire à celle de l'algorithme ci-dessus.