info607 : mathématiques pour l'informatique

TD 4 : cryptographie 1 (DES, Diffie-Hellman)

Pierre Hyvernat
Laboratoire de mathématiques de l'université de Savoie
bâtiment Chablais, bureau 22
téléphone : 04 79 75 94 22
email : Pierre.Hyvernat@univ-savoie.fr
www : http://www.lama.univ-savoie.fr/~hyvernat/

Exercice 1 : DES simplifié

Nous allons utiliser une version simplifiée de DES : "simplified DES". Une description succincte suit, et les détails seront donnés à l'oral.

Une clé secrète partagée de 10 bits permet de générer deux sous clés de 8 bits :
- on applique la permutation $(3,5,2,7,4,10,1,9,8,6)$ sur la clé
- le résultat est divisé en deux parties de 5 bits, et chaque partie est "décalée" (permutation circulaire) vers la gauche. On obtient (A_1, A_2)
- on applique la fonction $P = (6,3,7,4,8,5,10,9)$ sur (A_1, A_2) pour obtenir la première sous-clé : K_1
- on décale A_1 et A_2 de deux bits à gauches pour obtenir (B_1, B_2) et on applique la fonction P pour obtenir la deuxième sous-clé K_2

Pour l'encodage proprement dit, on utilise des blocs de 8 bits. On procède comme suit :
- on commence par permuter le bloc avec $(2,6,3,1,4,8,5,7)$
- le bloc est divisé en deux : (L_0, R_0) et on calcule $\big(L_1 = L_0 \oplus F(R_0, K_1), R_1 = R_0\big)$
- on inverse les blocs $(L'_1 = R_1, R'_1 = L_1)$
- on calcule $\big(L_2 = L'_1 \oplus F(R'_1, K_2), R_2 = R'_1\big)$
- on termine en utilisant la permutation inverse de la première étape : $(4,1,3,5,7,2,8,6)$

Il reste maintenant à définir la fonction F : elle prend en entrée un bloc de 4 bits et une sous clé K_i. Ensuite,
- elle l'étend en un bloc de 8 bits avec la fonction $(4,1,2,3,2,3,4,1)$
- elle ajoute ($\oplus$) la sous-clé K_i
- les 4 premiers bits sont passés dans S_1 et les 4 derniers dans S_2 (voir plus bas)
- le résultat (4 bits) est finalement permuté avec $(2,4,3,1)$; c'est le résultat.

Les matrices S_1 et S_2 sont définies par

$$S_1 = \begin{pmatrix} 1 & 0 & 3 & 2 \\ 3 & 2 & 1 & 0 \\ 0 & 2 & 1 & 3 \\ 3 & 1 & 3 & 2 \end{pmatrix} \qquad \text{et} \qquad S_2 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 0 & 1 & 0 \\ 2 & 1 & 0 & 3 \end{pmatrix}$$

et sont utilisées comme suit : étant donnés 4 bits (b_1, b_2, b_3, b_4)
- on utilise les bits (b_1, b_4) pour former un entier en base 2, qui nous donne la ligne
- les bits (b_2, b_3) forment un entier en base 2 qui donne la colonne
- on renvoie les deux bits constituant l'entrée correspondante dans la matrice.

Le véritable DES est très similaire, mais utilise
- des clés de 58 bits
- des blocs de 64 bits
- 16 itérations au lieu de 2.

Question 1 : On veut encoder, avec la clé (0110011100), le texte "`AB`" qui correspond, en ASCII, à la chaîne de bits (1000001 1000010). Comment l'encodez-vous ?

Question 2 : on veut décrypter, avec la clé (1101111001) la chaîne : (00011101 11010111). Quel est le texte clair ?

Question 3 : programmez (chez vous) le DES simplifié dans le langage de votre choix.

Question 4 : le DES double n'est pas considéré comme sûr. La raison est que le temps nécessaire pour casser DES double n'est pas très différent du le temps pour casser DES. Le seul problème est qu'il faut disposer d'une quantité de mémoire énorme...

Supposons qu'Alice envoie un message à Bob DES double : $c = \text{DES}\big(\text{DES}(m, k_1), k_2\big)$. Si un observateur malveillant (Eve) peut essayer de casser le code par une attaque "meet in the middle". Pour faire ça, Eve doit disposer d'au moins un message clair m et de son code c ; Eve peut ensuite calculer $\text{DES}(m, k)$ pour toutes les clés k possible, et décoder c avec $\text{DES}^{-1}(c, k)$ pour toutes les clés possibles k.

Comment peut-on se servir de ces résultats pour casser l'encryption de DES double ? Est-ce que c'est facilement faisable en pratique ?

Exercice 2 : calcul d'une puissance modulo

Étant donné deux entiers a et n, et un nombre premier p, le calcul de "$a^n \bmod p$" est une opération fondamentale en cryptographie.

Question 1 : écrivez un programme qui prend en arguments trois entiers quelconques a, n et p et calcule "$a^n \bmod p$". (Vous avez droit aux opérations arithmétiques usuelles...)

Question 2 : évaluez la complexité de votre algorithme en fonction de n. Que se passe-t'il si n comporte une centaine de chiffres ?

Question 3 : nous avons les égalités suivantes

$$a^{2n} = a^n * a^n$$
$$a^{2n+1} = a * a^n * a^n$$

Déduisez en un algorithme récursif plus efficace pour calculer "$a^n \bmod p$". Évaluez la complexité de cet algorithme en fonction de n. Qu'en pensez-vous ? (Cette méthode de calcul est appelée "méthode chinoise".)

Question 3-bis : pouvez-vous trouver un algorithme itératif qui fait le même calcul ?

Question 4 : en cryptographie, les nombre manipulés ont facilement plusieurs centaines de chiffres. En supposant que le type "`int`" utilise 32 bits, quel est l'entier maximal que l'on peut stocker ? Combien de chiffres possède-t'il ? Même question si on utilise 64 bits.

Question 5 : écrivez, sans trop de détails, une petite librairie (dans votre langage favori) pour manipuler des entiers de plusieurs centaines de chiffres. Il faudra faire en particulier l'addition, la multiplication et la division...

Exercice 3 : échange de clé de Diffie-Hellman

Question 1 : faites tourner l'algorithme d'échange de clé de Diffie-Hellman avec les valeurs suivantes

- $p = 11$ comme nombre premier
- $g = 2$ comme générateur de $\mathbf{Z}/p\mathbf{Z}$
- $a = 4$ comme nombre secret choisi par Alice
- $b = 8$ comme nombre secret pour Bob

Détaillez les calculs en mettant en avant les messages échangés par Alice et Bob. Quelle est la clé ainsi obtenue ?

Question 2 : vérifiez que 2 est bien un élément générique de $\mathbf{Z}/p\mathbf{Z}$. Est-ce que 3 est générique ? Que se passe-t'il si g n'est pas générique ?

Question 3 : que se passe-t'il si le canal de communication est compromis et qu'un observateur malveillant (Eve) écoute les communications ?

Question 4 : pouvez-vous généraliser le protocole d'échange pour partager une clé entre trois personnes ? Entre quatre ?