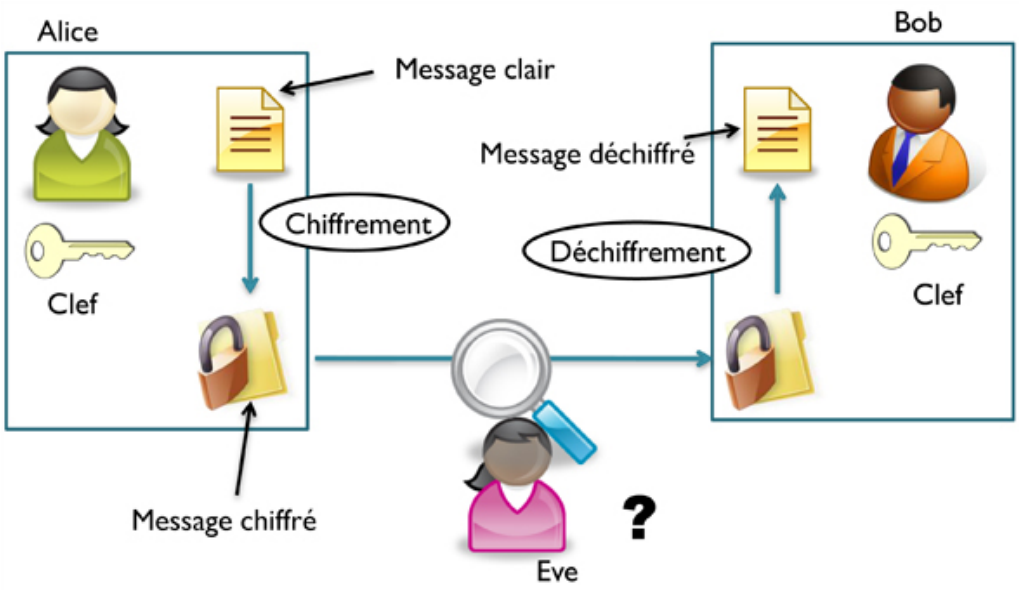
Introduction au langage C (NF05)
— projet 2020-2021 —

Taha Arbaoui, Florian Blachère & Rémi Cogranne (auteur de ce sujet)
Université de technologie de Troyes

Initiation à la cryptographie symétrique



Consignes

— Travail en monôme ou en binôme.
— Rapport (en pdf) + code + fichier exécutable sont à envoyer, en un seul fichier zip, à la personne responsable du groupe de TD dans lequel vos ferez votre soutenance ; Si vous êtes en binôme, pensez à indiquer les deux noms.
— La présentation (power point + exécution) se fera pendant les séances de TD après les vacances de Noël 2020.
— Le code devra être commenté en entier. Pour cela, il faut utiliser l'outil Doxygen [1] pour générer la documentation.
— Recommandations : le langage C est très renseigné sur internet, faites une recherche avant de contacter un ami ou un prof.
— Il est interdit de reprendre un code de quelqu'un autre au risque d'une sanction.
— Il est crucial de citer vos références.
— Code + rapport + doc sont à rendre au plus tard le 04 janvier 2017, avant minuit.

Rapport

Le rapport doit inclure :

1. une introduction qui énonce clairement le sujet, ainsi que le plan du document ;
2. une partie qui décrit les algorithmes utilisés (le fonctionnement et non pas le code) ;
3. les problèmes rencontrés et les solutions que vous avez trouvées.
4. un mode d'emploi du programme ;
5. une conclusion et des perspectives pour améliorer votre programme ;
6. une annexe qui comprend le code commenté.

Il est également demandé d'estimer la durée de réalisation des différentes parties du code.

Travail demandé

La cryptographie permet de chiffrer un message afin d'en cacher le sens, de le rendre inintelligible. Évidemment seule la personne à qui le message est destiné sera en mesure de "décoder" le message chiffré afin d'en extraire le message d'origine. Le message original est en général appelé "message clair" et le résultat du chiffrement est appelé "message chiffré" (voir la figure de présentation sur la page de garde).

1. www.doxygen.org

Le but de ce projet est de votre faire écrire un programme qui implémente une technique de cryptographie symétrique [2] simple mais reposant sur des techniques usuelles.

Nous décrivons ci-dessous l'algorithme que vous devez implémenter ainsi que les contraintes "techniques" qu'il est demandé de respecter.

La méthode de chiffrement fonctionne de façon itérative, c'est-à-dire que le même groupe d'opération sera utilisé N fois.

Il vous ait demandé de pouvoir chiffrer des fichiers quelconques, vous pouvez tester votre code avec des textes (car cela est visualisable) mais vous ne devez pas vous restreindre à ce type de fichier (vous devez par exemple prendre en compte les fichiers .doc et les images .jpg par exemple).

Votre programme doit lire n'importe quel fichier par octets. La méthode de chiffrement que vous devez implémenter fonctionne sur des "blocs" de 4 octets. Il s'agit d'une méthode itérative, utilisant N itérations, chacune étant constituée des cinq fonctions décrites ci-dessous. Nous définirons dans ce qui suit une unique itération, la première, en notant O_0, O_1, O_2, O_3 le bloc de 4 octets initial (en clair). Au fil des opérations ces octets seront notés, $O_i \to U_i \to V_i \to X_i \to Y_i \to Z_i$, avec $i \in \{0, \ldots, 3\}$.

1. L'octet O_i est dans un premier temps modifié suivant une table de permutation. On rappellera qu'une permutation est une fonction bijective (inversible) $p : \{0, 1, \ldots, K-1, K\} \mapsto \{0, 1, \ldots, K-1, K\}$. Autrement dit, un octet est modifié en un autre octet de sorte que la sortie obtenue, pour deux octets de valeurs distinctes, ne peuvent être identiques.
   On notera cette modification par $U_i = p(O_i)$; cette fonction ne pouvant prendre que 256 entrées distinctes il est possible de la tabulée de sorte que l'image est donnée par l'élément d'indice O_i.
   C'est à vous de créer une fonction qui permet de construire une telle permutation qui doit dépendre de la clé "d'itération" K_1 (elle même étant générée à partir de la clé de chiffrement).
2. Chaque octet x est modifié par une permutation 2-à-2 ; on entend par "permutation 2-à-2" une fonction qui modifie x en y mais inversement y en x.
   Cette permutation, contrairement à la précédente, sera toujours la même et devra être renseignée dans un fichier.
   On notera cette modification par $V_i = q(O_i)$
3. Un octet est ensuite décomposé en 8 bits, représenté sous la forme d'un vecteur (colonne) ; ce vecteur de bits est modifié par la fonction (matricielle) affine suivante

2. On parle de *cryptographie symétrique* lorsque les algorithmes utilisés pour chiffrer et déchiffrer sont très similaires et, surtout, lorsque les clés de déchiffrement et de déchiffrement sont identiques.

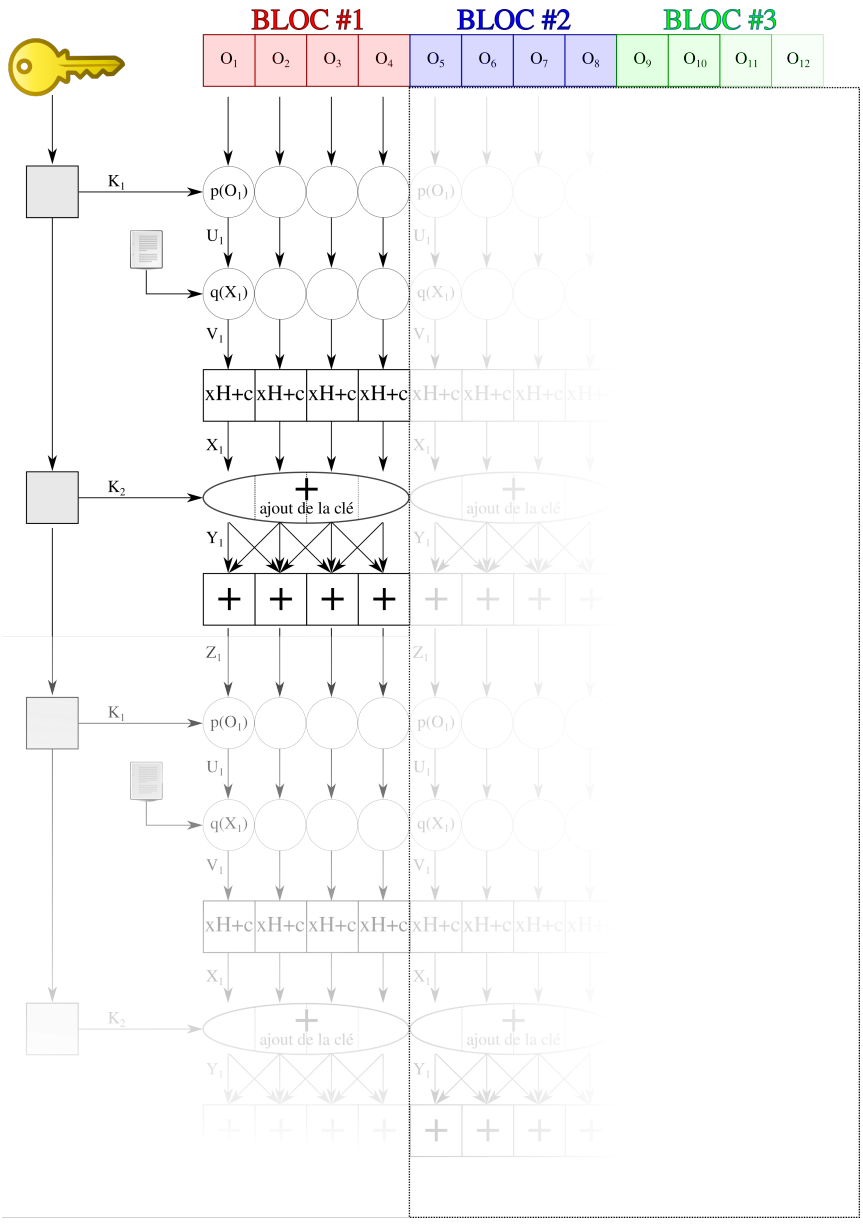


Fig. 1: Représentation graphique de la méthode de chiffrement. On peut voir le découpage de blocs de 4 octets du message clair, les itérations composées des 5 fonctions décrites dans le présent document. Dans le cas présent seulement $N = 2$ illustrations sont représentées, en pratique il en faudrait au moins 10 pour un système réaliste...

$\mathbf{X}_i = \mathbf{H}\mathbf{V}_i + \mathbf{c}$ avec :

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad \text{et} \quad \mathbf{c} = \begin{pmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{pmatrix}$$

La fonction inverse de cette fonction affine est également donnée par : $\mathbf{V}_i = \mathbf{H}'\mathbf{X}_i + \mathbf{c}'$ avec :

$$\mathbf{H}' = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{pmatrix} \quad \text{et} \quad \mathbf{c}' = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

ATTENTION : l'opération sur les bits doit se faire dans le corps $\mathbb{Z}_2$, c'est à dire avec uniquement les $\{0, 1\}$; un bit ne peut effectivement prendre que l'une de ces deux valeurs. Dans la pratique les opérations sont simplement réalisé "modulo 2" (que vous pouvez faire en toute fin de calcul).

4. L'octet Y_i décomposé en 8 bits est modifié par une opération de "XOR". On a donc $Y_i = X_i \oplus K_2$ avec $\oplus$ l'opération de "XOR" appliqué bit par bit.
   Il est demandé que r soit une clé "d'itération" ; c'est-à-dire que vous devez créer, à partir de la clé de chiffrement, N "sous-clés", pour chacune des itérations, et générer r de façon pseudo-aléatoire en fonction de la "sous-clé" pour chacune des itérations.

5. Enfin, la dernière opération de l'itération nécessaire de concaténer 4 octets notés x_i , $i \in \{0, \ldots, 3\}$. C'est 4 octets sont modifiés conjointement, ensemble, de la façon suivante :
   $$\begin{cases} Z[0] = Y[0] + Y[1]; \\ Z[1] = Y[0] + Y[1] + Y[2]; \\ Z[2] = Y[1] + Y[2] + Y[3]; \\ Z[3] = Y[2] + Y[3]; \end{cases}$$
   .
   avec y_i , $\{0, \ldots, 3\}$ les octets modifiés.
   ATTENTION : Les précédents calculs sont à faire avec des octets donc dans

l'ensemble $\mathbb{Z}_{256}$ (qui n'est pas un corps ...), c'est à dire modulo 256. C'est à vous de trouver la fonction inversant cette opération (c'est assez facile à résoudre comme système d'équations linéaires, non ?).

Attention les calculs avec des octets sont tous à faire modulo 255 ...

Bonus (ou pas)

Votre programme doit permettre de choisir le nombre d'itérations N .
Dans votre programme, chaque fonction doit utiliser un pointeur en entrée et en sortie.

Toute amélioration possible est la bienvenue !
Vous pouvez par exemple regarder les *modes de chiffrement* CBC, et CTR, générer des clés d'itération en vous inspirant de ce qui est fait dans la vraie vie ...