“VCES” & “true-RSA”

GS15 - A15 - Projet Informatique

Sujet présenté en cours le 25/10

Rapport à rendre avant le 09/01 - Soutenance entre le 09/01 et le 13/01

---

1 Description du projet à réaliser

Le but de ce projet informatique est de vous faire créer un outil offrant la possibilité d'utiliser deux algorithmes différents de chiffrement : l'un à symétrique, l'autre à clé publique et un algorithme de hashge. Le choix de l'algorithme utilisé est laissé à l'utilisateur, qui pourra par exemple l'indiquer en entrant un nombre spécifique au clavier avec une interface du type :

```
Selectionner votre fonction de chiffrement
->1<- Chiffrement symétrique ThreeFish (3 ><(((> )
->2<- Chiffrement de Cramer-Shoup
->3<- Hashage d'un message
->4<- Déciffrement symétrique ThreeFish (3 ><(((> )
->5<- déchiffrement de Cramer-Shoup
->6<- Vérification d'un Hash
```

L'utilisateur entre son choix (1, 2, 3, 4 ou 5) ... et le programme doit ensuite le guider, lui demander de choisir une clé, un fichier, etc. . . .

Les trois algorithmes qui vous sont demandés sont décrits ci-dessous, respectivement dans les sections 2 et 3.

Il est conseillé de ré-utilisé les fonctions données en TP pour la lecture et l'écriture des fichiers ainsi que les fonctions que vous avez pu écrire durant les séances de TP.

Enfin, le choix du langage de programmation vous appartient, néanmoins votre enseignant n'étant pas omniscient, un soutient n'est assuré que pour les langages Matlab/GMPint et C/GMP. La seule contrainte **obligatoire** est seulement de respecter les consignes données dans la section 5 du présent document.

Il n'est pas obligatoire cette année de rendre un rapport !!).

Date limite de restitution : **Dimanche 7 janvier à 23h59** (au delà, un point sera enlevé par minute de retard).

Une soutenance est prévue la semaine précédent les examens finaux, vous devrez vous inscrire pour “réserver” un horaire pour votre présentation.

---

2 Chiffrement Symétrique ThreeFish

Dans cette partie, le but du projet est de proposer une méthode de chiffrement appelé Threefish. Cette méthode accepte en entrée des blocs de 256, 512 ou 1024 bits avec des clés de tailles équivalentes.
Elle repose sur des **mots** de 64 bits et utilisent en plus deux "tweaks" de 64 bits également (faisant parti de la clé).
Enfin, pour simplifier un peu les choses, on utilisera toujours 76 tournées et un nombre de clé de tournées de $76/4 + 1 = 20$, chacune des clés de tournées ayant la taille de la clé principale, égale donc à celle des blocs en entrés.

Génération des clés des tournées :

La clé originale K est découpe en $N = \{4, 8, 16\}$ (en fonction de la taille des blocs et de la clé) mots de 64 bits $K = (k_0, k_1, \ldots, k_{N-1})$. À cette clé est ajouté un $N + 1$-ième mot défini par :

$$k_{N+1} = k_0 \oplus k_1 \oplus \ldots \oplus k_{N-1} \oplus C,$$

où $\oplus$ représente le ou exclusif et la constante C est donnée par :

$$\begin{aligned} C &= 0x1bd11bdaa9fc1a22_{16} \\ &= 0001101111010001000110111101101010101001111111000001101000100010_2. \end{aligned}$$

En outre, à partir des deux tweaks initiaux t_0 et t_1 on définit le troisième élément :

$$t_2 = t_0 \oplus t_1.$$

La clé utilisée pour chaque tournée, de N mots, est définie de la façon suivante : à la tournée $i^{\in}\{0, 19\}$ la clé $K^{(i)} = (k_0^{(i)}, k_1^{(i)}, \ldots, k_{N-1}^{(i)})$ est constituée des éléments suivants :

$$\begin{cases} k_n^{(i)} = k_{i+n \mod N+1} & \text{pour le bloc } n \in \{0, N-4\} \\ k_n^{(i)} = k_{i+n \mod N+1} \boxplus t_{i \mod 3} & \text{pour le bloc } n = N-3 \\ k_n^{(i)} = k_{i+n \mod N+1} \boxplus t_{i+1 \mod 3} & \text{pour le bloc } n = N-2 \\ k_n^{(i)} = k_{i+n \mod N+1} \boxplus i & \text{pour le bloc } n = N-1 \end{cases}$$

où l'opération $\boxplus$ représente l'addition module 2^64 (i.e sans retenue).

Vous pouvez calculer ces clés avant chaque tournées ou, plus simplement, calculer toutes ces clés au début ...

Chiffrement avec ThreeFish :

Le principe générale du chiffrement de Threefish repose sur le schéma de substituion / permutation décrit dans AES.
Encore une fois, 76 tournées seront réalisées, l'ajout avec la clé ne fait que toutes les 4 tournées ainsi que tout au début (initialisation) et tout à la fin (étape finale), il y a donc bien $76/4 + 1$ clé de tournées nécessaires. Les deux fonctions de substitution et de permutation sont décrites ci-dessous.

La fonction de substitution prend deux mots de 64 bits (et est appliquée sur chaque paire de mots, soit 2, 4 ou 8 fois suivant la taille du bloc). Notons ces deux mots m_1 et m_2, ainsi que $m_1{}'$ et $m_2{}'$ le resultat de la fonction de mélange : $(m_1{}', m_2{}') = \text{Mix}(m_1, m_2)$..
La fonction mix procède en deux étapes de la façon suivante :

$$\begin{cases} m_1{}' = m_1 \boxplus m_2 \\ m_2{}' = m_1{}' \oplus (m_2 <<< R) \end{cases}$$

où $m_2 <<< R$ signifie la permutation circulaire (vers la gauche) de m_2 d'un décalage de R bits ; R est ici laissé à votre choix (par exemple 49) et peut même varier pour chacune des paires de mots ...

La fonction de permutation est laissé à votre guise. Vous pouvez par exemple, inverser l'ordre des mots et conserver chacun des N mots inchangés, ou bien permuter les bits de chaque mots suivant une permutation de $\{1 \dots 64\} \to \{1 \dots 64\}$, voire mélanger les mots par paires ou bien être encore plus tordu que votre prof de GS15 (si cela est possible) ...

Enfin, comme indiqué en début de sous-section, toutes les 4 tournées, la clé de tournées est ajoutée (somme sans retenue ou XOR, à vous de choisir) au bloc en cours de chiffrement.

Le déchiffrement de chaque bloc se fait en inversant les opérations une à une, autrement, en commençant avec la dernière clé de tounées que l'on soustrait au bloc chiffré, puis en inversant la fonction de permutation puis la fonction de mélange ... et toutes les quatres itération on soustrait à nouveau la clé de la tournée précédente, etc ....

La Figure 1 illustre le fonction d'une tournée de Threefish (e.g la première tournée) avec l'ajout initial de la clé, puis les 4 itération de mélange (substitution), **par bloc de 2 mots** suivi de la permutation, et enfin application à nouveau de l'ajout de la clé.

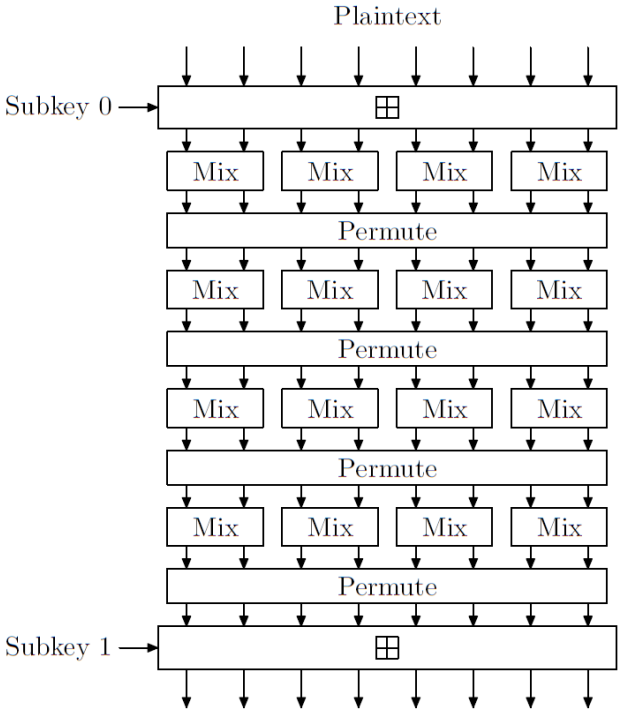

Figure 1: Représentation graphique du système du chiffrement de flux A5/1 basé sur 3 LFSR.

2.1 Recommandations / Éxigences

Autant que faire se peut, il vous est demandé d'implémenter :

1. le chiffrement en mode ECB et CBC ;
2. de proposer le choix de la taille de bloc à l'utilisateur (en dur ou à choisir en ligne).
3. le chiffrement d'un fichier ainsi que son déchiffrement (le chiffré lui aussi étant écrit dans un fichier);
4. le fichier doit être d'une taille adéquate, au moins quelques dizaine de bloc de 1024 bits ;

3 Chiffrement à clé publique de Cramer-Shoup

Dans cette partie il vous ait demandé d'implémenter le systéme de chiffrement de Cramer-Shoup décrit ci-dessous:

(Attention : Bien que cela ne soit pas précisé, il est évident que tous les calculs ne font dans $\mathbb{Z}_p$!)

Étape de Génération des clés :

1. Alice choisit un entier premier (grand) p ainsi que deux générateurs, notés α_1 et α_2 de $\mathbb{Z}_p$.
2. Elle choisit ensuite aléatoirement 5 entiers de $\mathbb{Z}_p$ notés, x_1, x_2, y_1, y_2 et w.
3. Elle calcule enfin $X = {\alpha_1}^{x_1}{\alpha_2}^{x_2}$, $Y = {\alpha_1}^{y_1}{\alpha_2}^{y_2}$ et $W = {\alpha_1}^{w}$.
4. Alice peut désormais sa clé publique constituer des éléments $(p, \alpha_1, \alpha_2, X, Y, W)$ et les choses sérieuses peuvent commencer ...

Étape de Chiffrement :

Pour chiffrer un (bloc du) message $m \in \mathbb{Z}_p$ Bob choisit aléatoirement un entier $b \in \mathbb{Z}_p$ puis calcule:

1. $B_1 = {\alpha_1}^{b}$ et $B_2 = {\alpha_2}^{b}$.
2. $c = W^b \times m$.
3. $\beta = H(B_1, B_2, c)$.
   Avec, ici, $H(\cdot)$ une fonction de hashge cryptographiquement sécurisée et $H(B_1, B_2, c)$ est appliquée d'une façon qu'il vous reste à définir (par exemple hashage de la concaténation des trois éléments, ou bien concaténation des trois hash pris séparément, etc ...
4. Enfin, Bob calcule la "vérification" $v = X^b \times Y^{b\beta}$ et retourne à Alice le chiffré constitué de (B_1, B_2, c, v).

Étape de Déchiffrement (et de vérification) :

Pour déchiffrer un (bloc du) message reçu (B_1', B_2', c', v') Alice procède en deux étape:

Étape de vérification : Alice calcule $\beta' = H(B_1', B_2', c')$ puis vérifie que $v' = {B_1'}^{x_1}{B_2'}^{x_2}\left({B_1'}^{y_1}{B_2'}^{y_2}\right)^{\beta'}$.

Le déchiffrement n'est pas la vérification précéde est correcte, c'est à dire si l'égalité est vérifiée. Dans le cas contraire, le message est considéré comme non-valide et ignorer.

Étape de déchiffrement : Si la vérification est correcte, Alice peut déchiffrer le message avec une opération similaire à celle utilisée dans El Gamal : $m = \left({B_1}^{w}\right)^{-1} \times c$.

3.1 Recommandations / Éxigences

Autant que faire se peut, il vous est demandé d'implémenter :

1. La génération des clés (ce qui inclut la génération d'un grand nombre premier non friable p et la recherche des deux générateurs α_1 et α_2) ;
2. Un systéme pour écrire les clés publiques et privés dans un fichier ;
3. Le code permettant de chiffrer un message en utilisant le fichier contenant la clé publique ;
4. Le code permettant de déchiffrer en utilisant le fichier de clé privée ;

Naturellement, il est ici impératif d'utiliser des grands entiers (au moins quelques centaines de bits). Enfin, le choix de la fonction de hashage est laissé à votre discretion ; vous pourrez (par exemple) inventer une fonction "un peu tordue quand méme, mais pas trop compliqué à implémenter" ou bien redévelopper une fonction de hashage "classique" type MD5, SHA-x ...

4 Hashage

Cette partie ne vous demande quasiment aucun travail puisque la fonction de hashage qu'il vous ait demandé d'utiliser est celle déjà implémenter dans le système de chiffrement à clé publique de Cramer-Shoup. Il vous suffit donc d'ajouter (1) la lecture d'un fichier pour en calculer son empreinte/hash (2) l'écriture de la signature dans un fichier, et, (3) l'étape de vérification de la signature.

5 Questions pratiques et autres détails

Il est impératif que ce projet soit réalisé en binéme. Tout triném obtiendra une note divisé en conséquence (par $3/2$, soit une note maximale de $13,5$).
Encore une fois votre enseignant n'étant pas omniscient et ne connaissant pas tous les langages informatique du monde, l'aide pour la programmation ne sera assuré que pour Matlab/GMPint et C/GMP. Par ailleurs, votre code devra étre commenté (succinctement, de faéon é comprendre les étapes de calculs, pas plus).
Votre code doit étre a minima capable de prendre en entré un texte (vous pouvez aussi vous amuser é assurer la prise en charge d'image pgm comme en TP, de fichiers binaires, etc .... mais la prise en charge des textes est le minimum souhaité).
Un rapport n'est plus attendu cette année. Attendez vous tout de même à présenter en soutenance les principales difficultés rencontrées et à répondre sur la façon dont vous avez implémenté telle ou telle fonction.
La présentation est trés informelle, c'est en fait plutét une discussion autour des choix d'implémentation que vous avez fait avec démonstration du fonctionnement de votre programme.
Vous avez, bien sûr, le droit de chercher des solutions sur le net dans des livres (ou, en fait, où vous voulez), par contre, essayez autant que possible de comprendre les éléments techniques trouvés pour pouvoir les

présenter en soutenance, par exemple comment trouver un entier premier sécurisé, comment trouver un générateur, etc ...

Enfin, vous pouvez vous amuser à faire plus que ce qui est présenter dans ce projet ; par exemple proposer un protocole d'échange de clé basé sur votre chiffrement de Cramer-Shoup pour votre threeefish, inventer une signature de Cramer-Shoup ...

Je réponds volontiers aux questions (surtout en cours / TD) mais ne ferais pas le projet à votre place ... bon courage !