"Certificats simplifiés"

GS15 - A19 - Projet Informatique

Sujet présenté en cours le 15/10
Rapport et codes à rendre au plus tard le 05/01
Soutenance entre le 06/01 et le 10/01

---

1 Description du projet à réaliser

Le but de ce projet informatique est de vous faire créer un outil de communication sécurisé reposant sur l'ensemble des outils cryptographiques vus en cours, à savoir les certificats pour l'authentification, la signature (et donc le hashage) ainsi que le chiffrement symétrique.

Notons en préambule que l'algorithme le plus détaillé est celui qui n'est pas vu en cours, le chiffrement symétrique *Camellia* modifié par les soins de votre professeur pervers et sadique.
Les autres algorithmes sont décrits de façon très succincte pour deux raisons : (1) car ils ont été (ou seront) étudiés en cours et (2) car **dans ce projet, vous avez beaucoup de liberté quant à l'implémentation pratique des algorithmes.** En effet vous constaterez qu'il vous ait demandé de suivre certaines "règles" à partir desquelles vous pouvez faire le minimum ou en faire beaucoup plus ; des suggestions vous seront données dans les différents points.

Il vous est demandé de faire un menu permettant de tester individuellement les algorithmes implémentés; votre programme doit donc, typiquement, afficher lors de l'exécution le menu suivant:

```
Bonjour ô maître ! Que souhaitez vous faire aujourd'hui ?
->1<- Générer des couples de clés publiques / privées.
->2<- Générer un certificat.
->3<- Vérifier la validité d'un certificat.
->4<- Partager une clé secrète.
->5<- Chiffrer un message.
->6<- Signer un message.
->7<- Vérifier une signature.
->8<- I WANT IT ALL !! I WANT IT NOW !!
```

Les algorithmes que vous sont demandés pour chacun des choix possibles sont décrits (en commençant par ceux que vous pouvez le plus implémenter le plus tôt dans le semestre) ci-dessous respectivement dans les sections 2 pour le chiffrement symétrique, 3 pour le hashage, 4 pour le partage de clé et 5 pour la vérification de la clé publique (certificat).
Il est conseillé de réutiliser les fonctions données / écrites pour les devoirs, notamment pour la lecture et l'écriture des fichiers, ainsi que les fonctions arithmétiques (tests de Rabin Miller).
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 6 du présent document.

Il est obligatoire cette année un rapport répondant uniquement aux exigences de la section 6 !!).
Date limite de restitution : **Dimanche 6 janvier à 23h59** (au-delà, un point sera enlevé par minute de retard).
Une soutenance est prévue la semaine précédant les examens finaux, vous devrez vous inscrire pour "réserver" un horaire pour votre présentation.

---

2 Chiffrement Symétrique *Camellia*

Concernant le chiffrement symétrique / à clé privée , il vous ait demandé d'implémenter le chiffrement par bloc *Camelia* qui repose sur une construction itérative de type Tournée de Feistel. Attention, il y a toutefois quelques modifications mineures (notamment dans façon dont sont faites les itérations et dans l'introduction d'étapes particulières lors des itérations 6 et 12.
Vous pouvez implémenter uniquement la version de *Camelia* utilisant des clés de 128 bits.

Vous pouvez trouver pas mal de description de l'algorithme de Camléia à l'adresse suivante:
RFC 3713 de l'IETF

Une seule et unique modification de *Camelia* vous est imposée ... les SBOX (dans la fonction F de Feistel) doivent être les suivantes :
On utilisera le polynôme irréductible:

$$X^8 + X^5 + X^3 + X^2 + 1$$

et les SBOX seront remplacées par les suivantes (les blocs de 8bits seront considérés comme des éléments du corps de Galois $G\mathbb{F}_{2^8}$):
SBOX1 : $t \leftarrow t \times X^6 + X^4 + X^3 + X$
SBOX2 : $t \leftarrow t^{-1}$
SBOX3 : $t \leftarrow t + X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$
SBOX4 : $t \leftarrow (t + X^6 + X^4 + X^2 + 1) \times X^7 + X^5 + X^3 + X$
SBOX5 : $t \leftarrow t \times X^5 + X^2 + 1$
SBOX6 : $t \leftarrow (t \times X^3) + X^3 + X^2 + X + 1)^{-1}$
SBOX7 : $t \leftarrow t + X^7 + X^6 + X^5 + X^4 + X^3 + X^2 + X + 1$
SBOX8 : $t \leftarrow (t + X^7 + X^5 + X^3 + X) \times X^6 + X^4 + X^2 + 1$

2.1 Exigences

Il vous est demandé d'implémenter :

1. le chiffrement en mode ECB, CBC et PCBC ;
2. le chiffrement d'un fichier ainsi que son déchiffrement (le chiffré lui aussi étant écrit dans un fichier);
3. le fichier doit être d'une taille adéquate, au moins quelques kilo-octets;

2.2 Recommandations

Pour aller plus loin, vous pouvez, si vous voulez, regarder les éléments suivants :

1. implémenter le chiffrement avec des clés de 192 / 256 bits ;
2. comprendre et implémenter les modes de chiffrements Counter et GCM (Galois' Counter Mode);
3. le fichier doit être d'une taille adéquate, au moins quelques kilo-octets;

---

3 Hashage: Fonction éponge

En ce qui concerne la fonction de hashage, vous pouvez implémenter n'importe quelle fonction de hashage (e.g, MD5, Whirlpool, SHA-256, etc. ...), mais il est impératif de modifier cette fonction afin de l'implémenter dans le cadre d'une "fonction éponge" telle qu'utilisée dans SHA-3 et illustrée dans la Figure 1 ci-dessous.
Si vous souhaitez reprendre une fonction de hashage "standard", les modifications à faire pour que cette dernière soit utilisable dans le cadre des fonctions éponges sont à votre discrétion. Sinon vous pouvez aussi vous "inspirer" des fonctions de hashage classique pour inventer la vôtre.

Une seule contrainte est demandée, votre fonction de hashage doit appliquer, au moins, 2 fois la fonction afin d'obtenir le hash (votre schéma utilisant la fonction éponge ne doit pas "seulement" ressortir le résultat de la dernière itération utilisant les données du fichier.
Vous pourrez également choisir arbitrairement d'utiliser N fois la fonction de hashage en fin de processus "d'absorption".

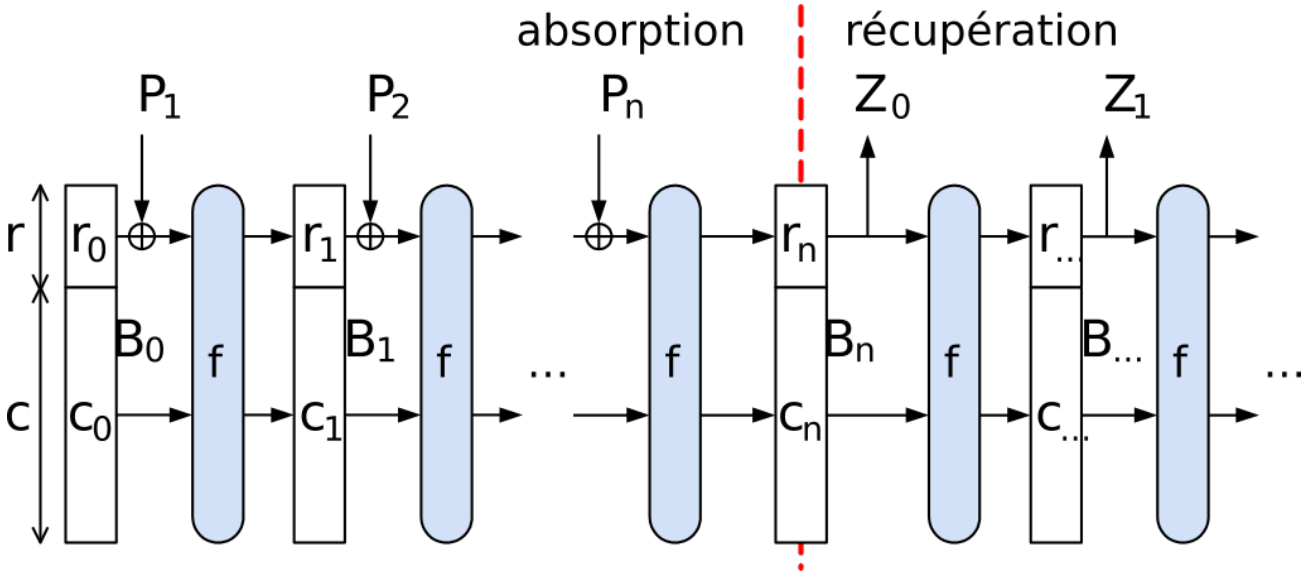

Figure 1: Rappel du principe des fonctions "éponges" pour le hashage.

3.1 Exigences

Il vous est demandé d'implémenter :

1. le hashage en utilisant les fonctions éponge ;
2. Utiliser une fonction de hashage non standard (ou modifier une fonction standard pour qu'elle soit utilisable dans le cadre des fonctions éponges);

---

4 Partage de clé : Diffie Hellman

Pour le partage d'une clé privée utilisée pour le chiffrement symétrique *Camelia* vous devrez implémenter la méthode d'échange de clé de Diffie-Hellman.
Pour cela il est imposé de chercher un entier premier p de au moins 512 bits.
Par ailleurs, il est tout à possible (voire intelligent, mais cela n'est pas une qualité obligatoire pour réussir GS15) de générer les entiers premiers et les clés pour la génération de signature en utilisant les mêmes fonctions / méthodes.

---

5 Générer et vérifier une signature (avec hashage)

La génération des signatures (des "certificats simplifiés" ou de clés pour le chiffrement asymétrique) nécessite dans un premier temps de générer de grands nombres premiers p. Plus difficile encore, pour des raisons de sécurité et pour des raisons de recherche rapide d'un élément α générateur de $\mathbb{Z}_p$, il est nécessaire de trouver un entier p premier de sorte que $q = 2p - 1$ soit également.
Cette recherche de tels nombres dits "fortement premiers" est commune avec l'échange de clé de Diffie-Hellman et nécessite l'implémentation du test de primalité de Rabin-Miller.

Concernant la partie signature, il vous est demandé d'implémenter un système simple de signature (proche dans la conception du standard X509):
Bien que cela ne soit pas vu en cours, le principe est extrêmement simple. Nous considérerons que nous avons 3 protagonistes; Visiteur, Site, et Certificateur; le Visiteur visite Site et peut vérifier auprès de Certificateur son certificat.
Step0: Une phase préalable est nécessaire, avant toutes choses, Certificateur doit générée une paire de clés privée / publique (notée Pub_C et $Priv_C$) Step1: lors de la création de ton site d'achat en ligne, Site doit générer une paire de clés privée / publique (notée Pub_S et $Priv_S$) et demander à Certificateur de signer sa clé publique ; Certificateur doit donc retourner à Site la signature (avec la clé $Priv_C$) de la clé publique de Site Pub_S, ce qui est noté $CERTIFICAT = S_{Priv_C}(Pub_S)$.
Step2: Visiteur souhaite faire un achat sur le Site, mais avant cela il veut pouvoir vérifier qu'il est réellement en train de communiquer avec le vrai vendeur .... Il va donc demander au Certificateur de lui donner sa clé publique Pub_C avec de vérifier la signature notée$S_{Priv_C}(Pub_S)$.

5.1 Exigences

Il vous est demandé d'implémenter :

1. Utiliser de grands entiers premiers (au moins 512 bits) ;
2. Utiliser la signature DSA ou bien El Gamal;
3. Les certificats (et les clés) doivent être écrits dans un fichier !.

6 Questions pratiques et autres détails

Il est impératif que ce projet soit réalisé en binôme. Tout trinôme obtiendra une note divisée 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 informatiques du monde, l'aide pour la programmation ne sera assuré que pour Matlab/GMPint et C/GMP (et un peu python, mais pas trop quand même).
Par ailleurs, votre code devra être commenté (succinctement, de façon à comprendre les étapes de calculs, pas plus).
De même les soutenances se font dans mon bureau (H109). Si vous avez codés votre projet sans utiliser de bibliothèques très spécifiques (en python2/3) vous pourrez normalement faire la démo sur mon PC. Dans tous les cas, amener si possible votre machine afin d'assurer de pouvoir exécuter votre code durant la présentation.
Votre code doit être a minima capable de prendre en entrée un texte (vous pouvez aussi vous amuser à 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 très court est demandé. Par de formalisme excessif, il est simplement attendu que vous indiquiez les difficultés rencontrées, les solutions mises en oeuvre et, si des choix particuliers ont été faits (par exemple quelle fonction de signature) les justifier brièvement. Faites un rapport très court (1 à 2 pages) ce sera mieux pour moi comme pour vous. Le rapport est à envoyer avec les codes sources.

La présentation est très informelle, c'est en fait plutôt une discussion autour des choix d'implémentation que vous avez faits 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 présenté dans ce projet ... cela sera bienvenu, mais assurez-vous de faire *a minima* ce qui demandé, ce sera déjà très bien.

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