Projet Nigma : Deuxième soutenance

CrypTeam : LAPÔTRE Guillaume (lapotr_g)
GANIVET Justin (ganive_j)
LADEVIE Stéphane (ladevi_s)
GISLAIS Sébastien (gislai_s)

Table des matières

1 Introduction

2 Cryptographie

Cette fois-ci encore, nous avons respecté ce que nous avions prévu de présenter pour cette seconde soutenance. En effet nous avons implémenté l'algorithme DES (Data Encryption Standard). Contrairement au RSA présenté en première soutenance, le DES est un algorithme de chiffrement symétrique. Cela signifie que pour chiffrer ou déchiffrer un fichier on utilise la même clé. Ce type de chiffrement a ses avantages et inconvénients par rapport au chiffrement asymétrique. Son principal avantage est sa rapidité : il est 1000 fois plus rapide que le RSA ! Son principal inconvénient est le fait de ne possèder qu'une seule clé. Ainsi le problème principal est l'échange de cette clé entre les différents protagonistes qui veulent échanger des données chiffrées. En effet pour le RSA il suffisait de donner sa clé publique aux personnes désirant chiffrer des messages, ensuite ils renvoient les messages chiffrés et on peut les déchiffrer à l'aide de notre clé privé jamais échangé avec personne. Si l'on envoie notre clé DES au n'importe qui, il y a un risque qu'un pirate puisse intercepté la clé et donc s'en servir ensuite pour déchiffrer les messages confidentiels. Cepandant il y a une astuce permettant de contourner le problème : On chiffre la clé DES avec un chiffrement RSA !

Un petit exemple : Alice veut envoyer un message confidentiel à Bob. Bob crée donc un jeu de clé RSA puis envoie la clé publique à Alice. Alice crée sa clé DES, puis chiffre son message à envoyer avec cette clé. Finalement, elle envoie à Bob sa clé DES chiffrée avec la clé publique RSA ainsi que le message chiffré avec la clés DES. Bob reçoit donc les deux fichiers, avec sa clé privé RSA il déchiffrer le fichier de clé envoyé apr Alice, puis avec la clé DES qu'il vient de déchiffrer il peut déchiffrer le message d'Alice !

Le DES a été achevé en 1977, c'est donc un algorithme de chiffrement ancien. Sa sécurité n'est plus optimal, en effet un état peut casser une clé DES en quelque minutes maintenant. Cepandant nons pensons qu'il est intéressant de se pencher sur cet algorithme qui fût un des premiers algorithmes de chiffrement symétrique défini rigoureusement.

Du fait de son ancienneté les spécifications du DES étaient prévus pour une réalisation matérielle. C'est à dire pour que ce soit des puces qui réalisent le chiffrement. Ainsi certaines partie du protocole sont faciles à réaliser sur une puce mais sont difficile à réaliser en Ocaml. Nous développerons dans nos parties respectives.

2.1 Partie de Guillaume Lapôtre

Pour implémenter le DES en Ocaml je me suis tout d'abord documenté sur cet algorithme de chiffrement complexe bien qu'utilisant que des opérations basiques (en effet le DES est optimisé pour une réalisation matérielle ...). Je me suis donc basé sur un livre qui m'a été bien utile pour la l'implémentation de RSA qui se nomme "Cryptographie Appliquée" par Bruce Schenier. Ce livre est très pratique car il explique très bien comment fonctionne un bon nombre d'algorithme de chiffrement et il y a aussi l'histoire de chaque algorithme : Comment il a été créé, par qui, suite à quels besoins etc...

J'ai donc tout d'abord commencé par la création d'un clé DES. Sa création fût bien plus simple que la création d'une paire de clé RSA. En effet, la génération des clefs RSA nécessitent de possèder au préalable de grands nombres premiers qui furent quelque peu difficile à obtenir. Une clé DES est un nombre de 64 bits. Puis le DES ne se sert que de 56 bits des 64 bits présents initialement. Les 8 bits inutilisés sont des bits de parités. Ils servent à vérifier lorsque l'on envoie la clé que la clé reçue n'est pas corompue. C'est à dire qu'aucun bit n'a changé au court du transfert. Ainsi, une clé DES à la propriété d'avoir un nombre pair de bits à 1 pour chaque octet. Ce sont les 8 bits de parité qui permettent d'assurer cette propriété.

Je me suis ensuite occupé des différentes fonctions de manipulation de la clef. Tout d'abord il y a une permutation de clef qui réarrange les bits dans un ordre prédéfini et qui ignore les bits de parité. On se retrouve donc avec une clé de 56 bits. Ensuite à chaque ronde du DES [1], on sépare la clé en deux sous-clés de 28 bits chacune et on effectue une ou deux rotations gauche suivant la ronde que l'on est en train d'effectuer. Ensuite on réassemble les deux sous clé puis on effectue une permutation compressive qui compresse la clé qui était de 56 bits en une clé de 48 bits. L'orde de lequel les bits sont réarrangés est spécifié dans le DES. Ainsi j'ai pu générer les 16 sous-clés requises pour le chiffrement d'un bloc.

Je me suis ensuite occupé d'une toute autre partie dans l'algorithme DES qui est la gestion des S-box ou tables de substitution. Ces tables sont le point-clé dans la sécurité du DES. En effet elles sont les seuls éléments non linéaire dans cet algorithme et elles confèrent à l'algorithme son niveau de sécurité. La chose qui pourrait paraître étonnante est que les 8 tables de substitutions du DES sont totalement publiques maintenant. Ces tables de substitution se représentent initialement sous la forme d'un tableau à deux entrées. La méthode permettant d'acceder à la case du tableau que l'on recherche est un peu particulière[2]. Nous avons donc initialement un bloc de 48 bits que

[1] le protocole exacte d'une ronde du DES sera expliquée par Sébastien

[2] Encore une fois, c'est Sébastien qui expliquera dans sa partie exactement quand est-ce

nos tables de subsitutions vont transformé en un bloc de 32 bits de façon absolument pas linéaire. Nos tables ont 4 rangs et 16 colonnes. Découpons notre bloc de 48 bits en 8 sous-blocs de 6 bits. Prenons notre premier sous-bloc et nommons les bits qui le compose : b1 b2 b3 b4 b5 b6. Le nombre composé par les deux bits b1 b6 va permettre la selection du rang puis les 4 autres bits b2 b3 b4 b5 vont permettre de selectionner la colonne, le tout sur la table de substitution n° 1. Ainsi nous avons donc selectionner une case de notre tableau qui contient un nombre sur 4 bits. En faisant de même pour les 8 sous-blocs on a donc utilisé les 8 tables de substitutions et récupéré les 32 bits escomptés.

On peut remarquer que la sélection d'un case dans une table n'ai pas forcément une chose aisée dûe à la façon dont on utilise les bits de nos sous-blocs. En effet en réalisation matérielle cette sélection ne pose pas de problème par contre en réalisation logicielle ce n'est pas optimisé. J'ai donc réarrangé les s-box pour obtenir 8 tableaux à une dimension en remettant bien dans l'ordre les bits b1 b2 b3 b4 b5 b6. Ainsi lorsque j'utilise une table je n'ai besoin que du bon sous-bloc de 6 bits pour récuperer sa représentation décimale et avoir le numéro de la case du tableau que je veux ! Par ailleurs, cela me permet aussi de limiter les risques de mauvaise utilisation des tables de substitutions.

Voici un exemple de l'utilisation d'une table de substitution :

S_5		4 bits au centre de l'entrée															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1100**	**1011**	**1100**	**1101**	**1110**	**1111**
Bits externes	**00**	0010	1100	0100	0001	0111	1100	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1100	0011	1001	1000	0110
	10	0100	0010	0001	1011	1100	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1100	0100	0101	0011

Voici une S-box où l'on voit bien le nombre "b1 b6" pour la sélection de la ligne et le nombre "b2 b3 b4 b5" pour la selection de la colonne

Je me suis aussi occupé de la fonction principale. Cette dernière gère les fonctions de chiffrage de données, de déchiffrage ainsi que la fonction de création de la clef. Cette fonction gère aussi tout la partie accès/écriture/lecture dans les différents fichiers utiles. Ainsi, lorsque l'on appelle le programme avec le paramètre –clef, le programme crée un fichier nommé clefDES et écrit une clef dedans avec les bits de parité à jour. Lorsque l'on appelle le programme

que l'on s'en sert.

avec le paramètre –chiffre, il faut lui donner en deuxième paramètre le fichier à chiffrer et en troisième paramètre le fichier contenant une clé DES. Le programme ouvre le fichier contenant la clé pour la récuperer puis crée les 16 sous-clé. Ensuite il ouvre le fichier à chiffrer et récupère les 8 premiers caractères. Comme un caractère est représenté sur un octet donc sur 8 bits, les 8 caractères lu donnent donc 64 bits. Ensuite on chiffre ces 64 bits avec les 16 sous-clefs et on écrit les 8 nouveaux caractères correspondant au bloc de 64 bits chiffré. Puis on fait de même pour chaque bloc de 8 caractère du fichier à chiffrer. Comme un fichier ne contient pas forcément un multiple de 8 caractères et, par contre, le DES à besoin de bloc de 64 bits, nous avons trouvé une astuce pour que le dernier bloc soit aussi chiffré : On complète le bloc avec des caractère espace pour avoir au final un bloc de 8 caractères. C'est pour cela que nos fichiers chiffrés contiennent tout le temps un multiple de 8 caractères.

2.2 Partie de Sébastien Gislais

3 Stéganographie

3.1 Partie de Justin Ganivet

3.2 Partie de Stéphane Ladevie

4 Conclusion

5 Annexe

5.1 Screenshot Cryptographie