Chapitre 1

Exercices Crypto symétrique

1.1 Taille des clefs

▷ **Exercice 1.1 :** Le "Cost-Optimized Parallel COde Breaker" est une machine conçue par les universités de Kiel et de Bochum en Allemagne dans la perspective de casser des systèmes de chiffrement par recherche exhaustive. La machine coûte le même prix qu'une petite voiture. Une COPACOBANA contient 16 cartes-support. Chacune de ces cartes contient 8 FPGAs (circuit reprogrammable). Chaque FPGA est capable d'executer 4 "coeurs" de chiffrement DES en parallèle. L'ensemble est cadencé à 140Mhz, et est capable d'effectuer un chiffrement DES par cycle d'horloge.

Combien de chiffrement DES cette machine est-elle capable d'effectuer par seconde ?

▷ **Exercice 1.2 :** En combien de temps une COPACOBANA peut-elle casser le DES (qui a des clefs de 56 bits) ?

▷ **Exercice 1.3 :** En combien de temps une COPACOBANA peut-elle casser SkipJack (qui a des clefs de 80 bits), en admettant qu'un chiffrement skipjack prend le même temps qu'un chiffrement DES ?

▷ **Exercice 1.4 :** Combien y a-t-il de mots de passe de 8 charactères, en autorisant minuscules, majuscules, chiffres et quelques signes de ponctuation ? Quelle est la taille de clef secrète correspondante ? (ceci est parfois nommé l'*entropie* du mot de passe).

▷ **Exercice 1.5 :** Un coeur d'un CPU contemporain, pas particulièrement puissant, est capable de faire 8 millions de chiffrements AES par seconde. Combien de coeurs faut-il pour casser un tel mot de passe en une semaine ?

▷ **Exercice 1.6 :** Le physicien Hans-Joachim Bremermann (1926–1996) a démontré, sur les bases de nos connaissances physiques actuelles, qu'un système matériel auto-suffisant (pas de source d'energie extérieure, pas d'échanges de matière avec l'exterieur) ne peut pas réaliser plus de $1.36 \cdot 10^{50}$ opérations par seconde et par kilo de matière (cette valeur est environ c^2/h, où c est la vitesse de la lumière et h est la constante de Planck).

Sachant que la planète terre pèse $5.972 \cdot 10^{24}$ kg, donner une borne inférieure sur le temps nécessaire pour casser une clef secrète de 512 bits par force brute.

Exercice 1.7 : Considérons l'adversaire suivant, qui prend en argument un message C, chiffré par une clef inconnue de n bits avec un chiffrement authentifié.

```
1: procedure SingleTrial(C, n)
2:     K ← Random bit string of size n
3:     P ← D(K, C)
4:     if P ≠ ⊥ then
5:         Return (P, K)
6:     else
7:         Return ⊥
8:     end if
```

9: **end procedure**

Sachant que K est une clef secrète de n bits générée aléatoirement, quelle est la probabilité que l'algorithme trouve la clef ?

Exercice 1.8 : À l'Euro-Millions, il faut choisir 5 bons nombres parmis 50, et cocher deux bonnes étoiles parmis 12. Il y a donc $\binom{50}{5}\binom{12}{2} = 139\,838\,160$ combinaisons possibles, et une seule rapporte le gros lot.

Qu'est-ce qui est le plus probable : « SingleTrial casse une clef de 128 bits » ou « Je joue 4 fois à l'euro-millions et je gagne les 4 fois ? »

▷ **Exercice 1.9 :** Considérons une version améliorée de l'algorithme précédent :

1: **procedure** ManyTrials(C, n, k)
2: $K \leftarrow$ Random bit string of size n
3: $P \leftarrow \mathcal{D}(K, C)$
4: **repeat**
5: **if** $P \neq \perp$ **then**
6: **Return** (P, K)
7: **end if**
8: **until** k trials have failed
9: **Return** $\perp$
10: **end procedure**

Quelle est sa probabilité de succès, en fonction de n et k ?

▷ **Exercice 1.10 :** Qu'est-ce qui est le plus probable : « ManyTrial casse une clef de 128 bits avec un million d'essais » ou « Je joue 4 fois à l'euro-millions et je gagne les 4 fois » ?

1.2 Identification

Dans la littérature sur les protocoles, on écrit souvent :

$$A \to B : \{M\}_K$$

pour dire : « A(lice) envoie à B(ob) le chiffrement du message M par la clef K ». On dit qu'une donnée est *fraiche* si elle générée spécialement pour l'occasion, et qu'elle n'a jamais servi avant. Un *nonce* est un *nombre (arbitraire) à usage unique*[1] La plupart du temps, les nonces sont aléatoires.

Protocole I (*Identification challenge-response*). Alice génère un nombre aléatoire N, le chiffre avec leur clef commune K et transmet le chiffré à Bob. Bob déchiffre ce qu'il a reçu d'Alice avec la clef K et lui envoie le résultat (en principe N). Alice vérifie enfin qu'elle a bien reçu son nombre aléatoire N : si c'est le cas, Bob a donc réussi à effectuer le déchiffrement, et a priori il connaît la clef secrète K.

I1. [Challenge.] $A \to B : \{N\}_K$ (N est un grand nombre aléatoire frais)

I2. [Response.] $B \to A : N$

Exercice 1.11 : Un adversaire actif essaye de se faire passer pour Alice auprès de Bob dans le protocole I. Peut-il obtenir quelque chose si le mécanisme de chiffrement est authentifié ?

Exercice 1.12 : Que risque-t-il de se passer si le nombre aléatoire N est trop petit dans les protocoles précédents ?

Exercice 1.13 : On considère une variante du protocole I donné dans le texte :

Protocole I' (*Identification challenge-response*).

I'1. [Challenge.] $A \to B : N$ (N est un grand nombre aléatoire frais)

I'2. [Response.] $B \to A : \{N\}_K$

Est-il correct ? Quelle condition le mécanisme de chiffrement doit-il satisfaire ?

1. le barbarisme « nonce » vient certainement de Number et de ONCE.

1.3 Disque

Exercice 1.14 : On souhaite mettre en place un système où le contenu entier d'un disque dur est chiffré avec un mot de passe. On souhaite que l'utilisateur puisse changer son mot de passe sans avoir à déchiffer-rechiffrer complètement son disque (ce qui prendrait des heures pour un gros disque moderne). Comment faire ?

1.4 Single Sign-On

Imaginons un réseau où un grand nombre d'utilisateurs veulent communiquer de manière chiffrée. Pour garantir la confidentialité, il faudrait que chaque utilisateur partage une clef secrète différente avec chaque autre... et la gestion des clefs va vite devenir un problème

Les protocoles d'authentification unique ("*Single Sign-On*") permettent de résoudre en partie ce problème en faisant intervenir un *serveur d'authentification*. Dans ce contexte, chaque utilisateur partage un secret avec le serveur d'authentification. Celui-ci permet à une paire utilisateurs d'établir entre eux une *clef de session* fraiche. Une « clef de session » est une clef secrète « jetable », qui n'a vocation qu'à être utilisée pendant la prochaine session de communication, puis jetée après.

Quand Alice veut établir une liaison chiffrée avec Bob, elle contacte d'abord le serveur d'authentification. Ce dernier génère une clef de session fraiche K_{ab} pour Alice et Bob. Il reste à la communiquer de manière sûre aux deux utilisateurs concernés. Une solution potentielle serait :

Protocole P (*le serveur d'identification contacte les deux protagonistes*). Alice contacte le serveur, l'informe qu'elle veut établir une session avec Bob. Le serveur génère une clef de session fraiche, puis la renvoie chiffrée à Alice. Il contacte Bob, l'informe qu'Alice veut communiquer avec lui, et lui transmet la clef de session chiffrée. Alice entre finalement en contact avec Bob, et démontre la possession de la clef partagée.

P1. [Début.] $A \to S : A, B, N$ (N est un grand nombre aléatoire frais).

P2. [A reçoit K_{ab}.] $S \to A : \{N, A, B, K_{ab}\}_{K_a}$ (K_{ab} est une clef de session fraiche).

P3. [B reçoit K_{ab}.] $S \to B : \{N, A, B, K_{ab}\}_{K_b}$

P4. [A contacte B.] $A \to B : \{N, A, B\}_{K_{ab}}$

Exercice 1.15 : Dans le protocole P, un adversaire essaye de se faire passer pour Alice auprès du serveur. Qu'est-ce qui l'empêche d'aboutir à établir une session avec Bob ?

Exercice 1.16 : Dans le protocole P, un adversaire essaye de se faire passer pour le serveur d'authentification auprès d'Alice. Qu'est-ce qui va mal se passer ?

Exercice 1.17 : On considère le protocole suivant :

Protocole $\mathbf{NSS_{bad}}$ (*Needham-Schroeder sans la confirmation*).

NS1. [Début.] $A \to S : A, B, N_a$

NS2. [Interaction avec S.] $S \to A : \left\{N_a, B, K_{ab}, \{A, K_{ab}\}_{K_b}\right\}_{K_a}$

NS3. [A Contacte B.] $A \to B : \{A, K_{ab}\}_{K_b}$

A et B entament ensuite leur conversation « normale ». Quel problème de sécurité se pose-t-il alors ?

Exercice 1.18 : On considère le protocole suivant :

Protocole $\mathbf{NS_{bad}}$ (*Needham-Schroeder sans les identifiants*).

NS1. [Début.] $A \to S : N_a, A, B$

NS2. [Interaction avec S.] $S \to A : \left\{N_a, K_{ab}, \{K_{ab}\}_{K_b}\right\}_{K_a}$

NS3. [A Contacte B.] $A \to B : \{K_{ab}\}_{K_b}$

NS4. [B défie A.] $B \to A : \{N_b\}_{K_{ab}}$

NS5. [A répond.] $A \to B : \{N_b + 1\}_{K_{ab}}$

Que peut-il se passer ?

Exercice 1.19 : On considère le protocole suivant :

Protocole $\mathbf{NS_{bad}}$ (*Needham-Schroeder sans le nonce N_a*).

NS1. [Début.] $A \to S : A, B$

NS2. [Interaction avec S.] $S \to A : \{B, K_{ab}, \{A, K_{ab}\}_{K_b}\}_{K_a}$

NS3. [A Contacte B.] $A \to B : \{A, K_{ab}\}_{K_b}$

NS4. [B défie A.] $B \to A : \{N_b\}_{K_{ab}}$

NS5. [A répond.] $A \to B : \{N_b + 1\}_{K_{ab}}$

Que peut-il se passer ?

Corrections

1.7 Comme il y a 2^n clefs possibles et qu'on en choisi une au hasard, la probabilité de trouver la bonne est 2^{-n}.

En fait, la probabilité que l'algorithme retourne une paire (P, K) est plus grande que ça et dépend de la qualité du mécanisme « d'authentification » du chiffrement. L'exercice **??** montre en effet que la probabilité d'obtenir autre chose que $\perp$ est supérieure à 2^{-k} lorsque le mécanisme produit des chiffrés k bits plus gros que les clairs. Pour cette raison, on choisit généralement $k = n$. Du coup, la probabilité de succès de SINGLETRIAL est bien 2^{-n}.

1.8 La probabilité de gagner 4 fois de suite à l'euro-millions est donc de 1 chance sur $(139\,838\,160)^4 \approx (1.4 \cdot 10^8)^4 \approx 3.85 \cdot 10^{32}$. La probabilité de « deviner » une clef de 128 bits en un seul essai est de 1 chance sur 2^{128}. Qu'est-ce qui est plus grand, 10^{32} ou 2^{128} ? Pour le savoir, le plus simple est de passer au logarithme :

$$\begin{array}{lll} \ln 10^{32} & = 32 \times \ln 10 & \approx 32 \times 2.3 \quad \approx 73.6, \\ \ln 2^{128} & = 128 \times \ln 2 & \approx 128 \times 0.7 \quad \approx 89.6. \end{array}$$

On voit donc que 2^{128} est environ $e^{89.6-73.6} \approx 8\,886\,110$ fois plus grand que 10^{32}. Par conséquent, il est plus probable de gagner 4 fois de suite à l'Euro-millions que de casser une clef de 128 bits par force brute.

1.11 Non : il peut envoyer des chaines de bits arbitraire à Bob, et Bob va essayer de les déchiffrer... Mais comme l'adversaire ne connaît pas K, il ne peut pas produire de nouveaux chiffrés valides que Bob réussirait à déchiffrer correctement.

1.12 Un adversaire pourrait tenter de se faire passer pour Bob avec une certaine chance de succès. Il ne peut pas déchiffrer le message de l'étape I1, mais il pourrait essayer de « deviner » la valeur de N choisie par Alice. Il génère donc N', au hasard, et le renvoie à Alice. Si N est codé sur n bits, il a une chance sur 2^n de réussir. Il faut donc que n soit assez grand ($n = 128$ est un bon choix).

1.13 Bob démontre qu'il est capable de chiffrer avec la clef K (et donc qu'il la connaît). Sans connaître K, il est impossible de répondre. Un adversaire actif peut (en se faisant passer pour Alice) faire chiffrer ce qu'il veut par Bob. Il faut donc que le chiffrement résite aux attaques par clair choisi.

1.14 Il faut stocker $(\{K\}_{pwd}, \{data\}_K)$ sur le disque, où K est une clef aléatoire choisie une bonne fois pour toute, et *pwd* est le mot de passe. Lors d'un changement de mot de passe, il suffit de rechiffrer K.

1.15 Il ne pourra pas lire la réponse chiffrée de l'étape P2. Il ne connaîtra donc pas K_{ab}, et ne pourra pas envoyer correctement le message de l'étape P4.

1.16 Il ne pourra ni fabriquer le message de l'étape P2 (car il ne possède pas K_a).

1.17 B ne vérifie pas que A connait bien K_{ab}. Ceci permet l'attaque (active) suivante : un adversaire enregistre une session du protocole. Plus tard, il ré-emet le message de l'étape NS3 : B ouvre une session croyant parler à A. L'adversaire, qui ne connaît pas K_{ab}, peut néanmoins rejouer les messages précédents (correctement chiffrés avec K_{ab}). C'est ennuyeux si un des messages dit « voici un virement de *xxxxx* € » !

1.18 Dans les étapes d'interaction avec Bob, l'identité de l'initiateur du protocole ne figure dans aucun des messages. Bob ne peut identifier son partenaire que grace à ses propres déclarations ou au protocole réseau (qui n'est pas forcément fiable).

Par exemple, C exécute normalement le protocole pour entrer en contact avec B. Une clef de session K_{cb} est établie et lors de l'étape NS3, C envoie (légitimement) à B le message $\{K_{cb}\}_{K_b}$.

Une fois que tout ceci est terminé, C tente de se faire passer pour A auprès de C. Il suffit qu'il exécute les étapes NS3–NS5 en envoyant exactement les mêmes choses, mais en prétendant être Alice. Comment Bob pourrait-il faire la différence ?

1.19 Le problème c'est que dans l'étape NS2, rien ne lie la réponse de S au message précédent. Un adversaire actif pourrait donc se faire passer pour S auprès de A – ceci dit, s'il ne connaît pas K_a ni K_b, il ne pourra que rejouer des messages de l'étape NS2 précédemment captés. C'est toutefois gênant car cela force Alice et Bob à utiliser toujours la même clef de session K_{ab}.