Chapitre 2

Exercices Crypto Asymétrique

Exercice 2.1 : Casser une clef RSA de 1024 bits demande combien de fois plus de calculs qu'une clef de 768 bits ?

▷ **Exercice 2.2 :** Quel niveau de sécurité (=taille de clef secrète équivalente) offrent les clefs de 2048 bits recommandées aujourd'hui ?

▷ **Exercice 2.3 :** Quelle taille de clef RSA faut-il choisir pour s'assurer un niveau de sécurité équivalent à des clefs secrètes de 128 bits ? 256 bits ?

▷ **Exercice 2.4 :** Le meilleur code publiquement disponible de factorisation, CADO-NFS, a besoin de 90 jours pour factoriser un nombre de 512 bits sur un coeur à 2Ghz. On peut estimer qu'un serveur qui contient 36 coeurs comparables coûte 4000 euros, et consomme 400W. Quel budget est nécessaire pour casser RSA-1024 en 3 mois ?

▷ **Exercice 2.5 :** Quelle puissance electrique est nécessaire ?

▷ **Exercice 2.6 :** Ceci est-il a la portée des Etats des pays riches ? (note : en 2016, l'Etat français a commandé 6 sous-marins nucléaires pour $8.5 \cdot 10^9$€. Chaque sous-marin embarquerait un réacteur nucléaire de 150MW.)

▷ **Exercice 2.7 :** Casser une clef de 2048 bits dans les mêmes délais demandera combien de fois plus de calculs (donc d'argent et de puissance électrique) ?

2.1 Protocole de Needham-Schroeder

Le protocole de Needham-Schroeder *à clef publique* (ne pas confondre avec celui à clef secrète) est un protocole qui date de 1978. Alice et Bob l'exécutent pour s'assurer de leurs identités respectives (Alice veut être sûre qu'elle parle à Bob, et Bob veut être sûr qu'il parle à Alice). Alice et Bob utilisent tous les deux une méthode de chiffrement à clef publique, et ils possèdent chacun la clef publique de l'autre. L'un des deux participants (ici Alice) lance le protocole, qui fonctionne en trois temps :

Protocole NS$_{1978}$ (*protocole de Needham-Schroeder à clef publique, version 1978*). Bob démontre à Alice qu'il peut déchiffrer avec sk_b (donc qu'il est bien Bob), puis Alice démontre elle aussi la connaissance de sk_a en effectuant un déchiffrement.

NS1. [A défie B.] $A \to B : \{N_a\}_{pk_b}$ (N_a nonce aléatoire)

NS2. [B répond, défie A.] $B \to A : \{N_a, N_b\}_{pk_a}$ (N_b nonce aléatoire)

NS3. [A répond.] $A \to B : \{N_b\}_{pk_b}$

Alors que ce protocole paraissait sûr, Gavin Lowe a trouvé en 1995 (presque 20 ans après !) une faille en utilisant un programme de recherche automatique d'attaques qu'il avait conçu. L'attaque permet permet à Irène (l'*intrus*) de se faire passer pour Alice auprès de Bob si, au même moment, Alice exécute légitimement le protocole avec elle (ce qui permet à Irène de relayer des messages d'Alice à Bob, alors qu'ils ne lui étaient pas destinés).

Exercice 2.8 : Décrivez une attaque contre le protocole de Needham-Schroeder à clef publique de 1978. Alice initie légitimement le protocole avec Irène, et Irène en profite pour se faire passer pour Alice auprès de Bob.

On peut réparer le protocole de la façon suivante : chaque fois qu'un participant envoie un contenu chiffré, il ajoute son identité à l'intérieur du message chiffré. Le protocole devient donc :

Protocole NSL (*protocole de Needham-Schroeder-Lowe à clef publique*). Bob démontre à Alice qu'il peut déchiffrer avec sk_b (donc qu'il est bien Bob), puis Alice démontre elle aussi la connaissance de sk_a en effectuant un déchiffrement.

NSL1. [A défie B.] $A \to B : \{A, N_a\}_{pk_b}$ (N_a nonce aléatoire)

NSL2. [B répond, défie A.] $B \to A : \{B, N_a, N_b\}_{pk_a}$ (N_b nonce aléatoire)

NSL3. [A répond.] $A \to B : \{A, N_b\}_{pk_b}$

Exercice 2.9 : Pourquoi le protocole modifié évite-t-il l'attaque ?

▷ **Exercice 2.10 :** Détaillez les opérations effectuées par mon navigateur web pour effectuer la vérification de la signature du serveur HTTPS de l'université.

▷ **Exercice 2.11 :** La clef publique RSA de DigiCert Inc est de taille 2048 bits, tout comme celle de Terena. Celle de l'université est de taille 4096 bits. Ceci est-il judicieux ?

2.2 Cartes Bleues

▷ **Exercice 2.12 :** Lorsqu'on lui présente une carte EMV « statique », que doit vérifier le terminal de paiement ? Que doit-il contenir à l'avance pour cela ?

▷ **Exercice 2.13 :** Quelle manoeuvre de fraude est rendue impossible par la vérification de l'exercice précédent ?

▷ **Exercice 2.14 :** Quelle autre manoeuvre de fraude n'est *pas* rendue impossible par cette vérification ?

▷ **Exercice 2.15 :** Lorsqu'on lui présente une carte EMV « dynamique », que doit récupérer le terminal de paiement sur la carte ? Que doit-il vérifier ?

▷ **Exercice 2.16 :** Pourquoi le mode dynamique améliore-t-il la sécurité par rapport au mode statique ?

2.3 Théorie de la complexité

Exercice 2.17 : Soit $\mathcal{E}$ un mécanisme de chiffrement à clef publique. Considérons le problème algorithmique suivant :

INPUT — Une clef publique pk.
— Une chaine de bits c.

QUESTION c est-il un chiffré valide ? Autrement dit, existe-t-il un message m et un aléa r tel que $c = \mathcal{E}(pk, m, r)$?

Démontrer que ce problème est dans $NP \cap coNP$ (coNP est la classe des problèmes pour lesquels il existe un certificat verifiable en temps polynomial lorsque la réponse est « NON »). Ceci semble suggérer que ce problème n'est pas NP-complet (car on ne connait aucun problème NP-complet qui soit dans coNP).

Corrections

2.1 En gros, la complexité est $\exp\left(\alpha \sqrt[3]{n \ln^2 n}\right)$ où $\alpha \approx 1.70$. Ce qu'on cherche est donc :

$$\begin{aligned} T(1024)/T(768) &= \exp\left(\alpha \left[\sqrt[3]{1024 \ln^2 1024} - \sqrt[3]{768 \ln^2 768} \right]\right) \\ &\approx e^{7.27} \\ &\approx 1440 \end{aligned}$$

2.8 On note $I(A)$ lorsqu'Irène prétend être Alice.

NS1-a.	A	$\longrightarrow$	I	:	$\{N_a\}_{pk_i}$
NS1-b.	I(A)	$\longrightarrow$	B	:	$\{N_a\}_{pk_b}$
NS2-b.	B	$\longrightarrow$	I(A)	:	$\{N_a, N_b\}_{pk_a}$
NS2-a.	I	$\longrightarrow$	A	:	$\{N_a, N_b\}_{pk_a}$
NS3-a.	A	$\longrightarrow$	I	:	$\{N_b\}_{pk_i}$
NS3-b.	I	$\longrightarrow$	B	:	$\{N_b\}_{pk_b}$

2.9 Dans le protocole modifié, Bob va répondre $\{B, N_a, N_b\}_{pk_a}$ dans l'étape NSL2. La feinte de l'attaque consistait en ceci Irène relayait ce message, sans le modifier (il est opaque pour elle, qui n'a pas la clef secrète d'Alice). Maintenant, si elle le relaye, Alice reçoit un message qui contient « Bob » dedans, alors qu'elle est en train d'exécuter le protocole avec Irène, donc la fraude est détectée.

2.17 Pour montrer que le problème est dans NP, il faut exhiber un certificat vérifiable en temps polynomial lorsque la réponse est « OUI ». En fait, (m, r) est un tel certificat, et la vérification est très simple : il suffit de calculer $\mathcal{E}(pk, m, r)$ et de vérifier si ça donne bien c.

Pour montrer que le problème est dans coNP, il faut exhiber un certificat vérifiable en temps polynomial lorsque la réponse est « NON ». La technique précédente ne marche plus. Mais on peut alors utiliser la clef secrète sk qui correspond à pk comme certificat : il suffit de lancer l'algorithme de déchiffrement et de s'assurer qu'il échoue (au passage, sk peut aussi faire office de certificat pour « OUI »).

Un problème est qu'il faudrait vérifier que sk et pk sont bien liées. Une manière de résoudre ce problème consiste à remarquer que pk et sk sont produit par un algorithme de génération de clef :

$$(sk, pk) \leftarrow \mathcal{KG}(n, r), \qquad r \text{ est aléatoire.}$$

Du coup, au lieu de fournir sk comme certificat, on fournit (n, r). Vérifier le certificat revient à vérifier que pk est correctement obtenu à partir de (n, r), puis effectuer le déchiffrement avec sk.