Crypto1 - TD 3
2023 – 2024

Énoncé

Exercice 1 :

1.a] Décrire une attaque dans le protocole de mise en accord de clé Diffie-Hellman dans laquelle un attaquant *actif* (*i.e.* qui peut modifier les données pendant le protocole Diffie-Hellman) peut ensuite intercepter, déchiffrer et modifier toutes les communications qu'Alice ou Bob chiffrerait avec sa clé.

Exercice 2 :

Dans cet exercice, on pourra utiliser les résultats numériques suivants :
- $319 \equiv 11 \times 29$; $10^{11} \equiv 263 \mod 319$; $263^2 = 216 \times 319 + 265$;
- $133^3 \equiv 12 \mod 319$; $133^{25} \equiv 133 \mod 319$;
- $11^2 \equiv 121 \mod 280$; $11^4 \equiv 81 \mod 280$; $11^8 \equiv 121 \mod 280$; $11^{16} = 81 \mod 280$;
- $95 = 64 + 31$; $81 \cdot 11 \equiv 51 \mod 280$; $81 \cdot 121 \equiv 1 \mod 280$.

On considère la clef publique RSA $e = 11$ et $N = 319$.

2.a] Quel est le chiffrement avec cette clef du message $M = 100$?

2.b] Calculer la clef privée correspondant à la clef publique.

2.c] Déchiffrer le message $C = 133$.

Exercice 3 :

Étant donné un système de congruences $\{x \equiv a_i \bmod m_i\}_{1 \leq i \leq k}, a_i \in \mathbb{Z}_{m_i}$, $1 \leq i \leq k$, et les m_i sont des entiers premiers entre eux deux à deux. Le théorème des restes chinois (CRT) donne une construction d'une solution x du système de congruences. Pour cela, on pose $m = \prod_{i=1}^{k} m_i$ et :

$$N_i = \frac{m}{m_i}, \text{ et } S_i \equiv N_i^{-1} \bmod m_i, \ 1 \leq i \leq k.$$

Montrer que $x \equiv \sum_{i=1}^{k} a_i N_i S_i \bmod m$ est l'unique solution du système.

Exercice 4 :

Dans tout cet exercice, on supposera que N est une clé publique RSA dont les facteurs premiers sont p et q s. On notera d et e les exposants de (dé)chiffremnt.

4.a] Montrer que pour tout entier a on a $a^d \equiv a^{(d \mod (p-1))} \mod p$.

4.b] A partir de $a_p \equiv a^d \mod p$ et $a_q \equiv a^d \mod q$, comment retrouver $a^d \mod n$?

4.c] Quelle sera alors la complexité de ce calcul ?

Exercice 5 :

Soit $N = p \cdot q$, avec $p > q$ des premiers. On suppose que p et q sont "proches". Finalement, on pose $t = (p + q)/2$ et $s = (p - q)/2$.

5.a] Montrer que $n = t^2 - s^2$, s est petit, et t est plus grand que $\sqrt{n}$.
On considère l'algorithme suivant :

1. $aux := \lceil \sqrt{n} \rceil$
2. $res := aux^2 - n$
3. Tant que res n'est pas un carré parfait
   (a) $aux := aux + 1$
   (b) $res := aux^2 - n$
4. Retourner $aux + \sqrt{res}$

5.b] Utiliser les questions précédentes pour expliquer ce que retourne cet algorithme.

5.c] Dérouler l'algorithme avec $n = 24960007$. On vous donne $\lceil \sqrt{24960007} \rceil = 4996$ et $4996^2 - 24960007 = 9$.

5.d] Donner la complexité de l'algorithme en fonction de t et n.

5.e] En déduire que la complexité s'écrit comme :

$$\frac{(\sqrt{n} - p)^2}{2p}.$$

Exercice 6 :

Soit $N = pq$ avec p et q des premiers.

6.a] Montrer que $\varphi(N) = n - p - q + 1$.

6.b] Montrer alors que p et q sont des racines de $X^2 - (n - \varphi(n) + 1)X + n = 0$.

6.c] En déduire que la connaissance de $\varphi(N)$ permet de factoriser n.

6.d] Utiliser cette technique pour factoriser $N = 21$ et sachant que $\varphi(21) = 12$.

Exercice 7 :

Soit $\mathbb{G}$ un groupe commutatif (noté multiplicativement). Pour simplifier, on peut considérer que $\mathbb{G} = (\mathbb{Z}/n\mathbb{Z})^*$
Proposer un algorithme qui étant donnés t éléments $g_1, \ldots, g_t$ du groupe $\mathbb{G}$ et des entiers positifs $n_1, \ldots, n_t$ calcule le produit $g_1^{n_1} \ldots g_t^{n_t} \in \mathbb{G}$ en $O(\ell + 2^t)$ multiplications dans $\mathbb{G}$ (où ℓ est la taille en bits de $\max(n_1, \ldots, n_t)$).

Exercice 8 : Algorithme de Shanks

Considérons un groupe multiplicatif cyclique $\mathbb{G}$ engendré par $g \in \mathbb{G}$ d'ordre connu q (autrement dit, nous avons $\mathbb{G} = \{1, g, g^2, \ldots, g^{q-1}\}$). Proposer un algorithme de résolution de logarithme discret par compromis temps-mémoire de complexité $O(\sqrt{q})$ opérations de groupe en temps et $O(\sqrt{q})$ éléments de groupe en mémoire.

Indication. On pourra remarquer que pour tout élément $h = g^x \in \langle g \rangle$, l'entier x s'écrit sous la forme $x = x_1 T + x_0$ avec $0 \leq x_0 < T$ et $0 \leq x_1 < T$ pour $T = \lceil \sqrt{q} \rceil + 1$.

Exercice 9 :

Soient N un module RSA et e un nombre entier premier avec $\varphi(N)$. Considérons un algorithme $\mathcal{A}$ qui prend en entrée un élément de $\mathbb{Z}_N^*$ et retourne un élément de $\mathbb{Z}_N^*$, en temps τ (dans le pire des cas) où τ représente au moins le coût d'une exponentiation dans $\mathbb{Z}_N^*$.

Supposons qu'il existe un sous-ensemble E de $\mathbb{Z}_N^*$ avec $\#E \geq \epsilon N$ et $\epsilon \in]0, 1]$ pour lequel lorsque $\mathcal{A}$ est exécuté sur un élément $x \in E$, l'élément y retourné par $\mathcal{A}$ vérifie $y^e = x \bmod N$.

9.a] Montrer qu'il existe un algorithme $\mathcal{B}$ qui résout le problème RSA dans $\mathbb{Z}_N^*$ en un temps espéré $O(\tau/\epsilon)$.

Exercice 10 :

10.a] Montrer que le protocole de chiffrement RSA naïf n'est pas sémantiquement sûr sous une attaque à clairs choisis.

10.b] Montrer que le protocole de chiffrement RSA naïf est inversible sous une attaque à un chiffré choisi.

Exercice 11 :

L'algorithme de chiffrement d'ElGamal est un algorithme de cryptographie asymétrique basé sur le problème du logarithme discret. Il a été créé par T. ELGAMAL en 1985 :

Génération des clés : L'utilisateur choisit un groupe $\mathbb{G}$ d'ordre q dans lequel le problème du logarithme discret est jugé difficile et g un générateur de $\mathbb{G}$. Il tire uniformément aléatoirement $x \in \mathbb{Z}_q^*$ et calcule $y = g^x \in \mathbb{G}$. La clé publique est (q, g, y) et la clé secrète associée est x.

Chiffrement : étant donné un message clair $m \in \mathbb{G}$, l'algorithme de chiffrement tire uniformément aléatoirement $r \in \mathbb{Z}_q^*$ et calcule $c_1 = g^r \in \mathbb{G}$ et $c_2 = m \cdot y^r \in \mathbb{G}$. Le chiffré de m est le couple (c_1, c_2).

Déchiffrement : étant donné un chiffré $(c_1, c_2) \in \mathbb{G}^2$, l'algorithme de déchiffrement retourne (c_2/c_1^x).

11.a] Montrer que le protocole de chiffrement ElGamal n'est pas à sens-unique sous une attaque à un chiffré choisi.