UCBL - Polytech 5A Info
Cryptographie

Correction TD3

Exercice 1. *λ représente le paramètre de sécurité. Il est fixé (préconisé par des instances compétentes) de manière à être certain qu'un attaquant ne puisse pas effectuer 2^λ opérations élémentaires. Actuellement, il est recommandé de choisir $\lambda = 100$, signifiant que personne sur terre n'est capable d'effectuer (en temps raisonnable c'est à dire en moins de quelques dizaines d'années) $2^{100} \approx 10^{30}$ opérations processeurs. Les fonctions KeyGen, Encrypt et Decrypt doivent être de complexité polynomiale par rapport à λ et il doit être garanti qu'il n'existe aucune attaque de complexité polynomiale en λ. De manière générale, toutes les complexités seront mesurées par rapport à ce paramètre.*

Exercice 2.
1 - $ed = 3d = 1$ dans $Z_{\phi(n)}$. Il s'en suit que $0 < d < \phi(n)$ et $3d \equiv 1 \mod \phi(n)$. Ainsi $3d - 1 = \phi(n)$ ou $3d - 1 = 2\phi(n)$. Donc, il existe $i \in \{1, 2\}$ tel que $\phi(n) = (3d - 1)/i$. Ainsi dans la i^{eme} itération de l'algorithme $\phi = \phi(n)$. Ainsi $S = p + q$ et donc p et q sont des solutions de l'equation $x^2 - Sx + n = 0$. Donc $(S + \sqrt{S^2 - 4n})/2$ est un facteur de n.

2 - On peut en déduire que retrouver la clé secrète à partir de la clé publique est aussi difficile que factoriser n. Autrement dit, si on fait l'hypothèse qu'il n'existe pas d'algorithme polynomial de factorisation alors on peut en conclure qu'il n'existe pas d'algorithme polynomial $\mathcal{A}$ qui retourne sk sur l'entrée pk, i.e. $d \leftarrow \mathcal{A}(n, e)$.

3 - Il suffit d'étendre la boucle principale de telle sorte que i varie de 1 à $e - 1$.

4 - Il faut que e soit petit, i.e. $e < p(\lambda)$ où p est un polynôme.

Exercice 3.
1 - Un attaquant (qui écoute le réseau) obtient les encryptions $M_1, \ldots, M_6$ des 6 numéros. Connaissant la clé publique, il peut encrypter les nombres $1, \ldots, 49$ et comparer ces encryptions avec $M_1, \ldots, M_6$ et ainsi obtenir les 6 numeros.

2 - Le nombre de valeurs possibles de m_c est de $49 \times 48 \times \cdots \times 44 \approx 10^{10}$. Ainsi, dans le pire des cas, l'attaquant devra effectuer 10^{10} encryptions, ce qui lui prendra $10^7 s$, soit environ 116 jours. Ainsi, en moyenne, il lui faudra environ 68 jours...

3 - En classant les numéros par ordre croissant, le nombre de valeurs possibles de m_c est divisé par $6!$. Il pourra donc retrouver, à coup sûr, tous les numéros en moins de $4h \approx 116/6!$ jours.

4 - Comme $m_c^{17} < n$ (à justifier) et donc $M = m_c^{17} \mod n = m_c^{17}$ (M étant une encryption de m_c). Aussi, $m_c = M^{1/17}$ (racine dix-septième sur les entiers se calcule rapidement...proposez un algorithme rapide...).

5 - Il faut ajouter de l'aléatoire au message, e.g. concaténer m_c avec un message m_r choisi aléatoirement en s'assurant que le nombre $m_c \| m_r < n$.

Exercice 4. *Encryptons d'abord m_0 et m_1, i.e. $M_i = \textsf{Paillier.Encrypt}(pk, m_i)$. Il suffit ensuite de remarquer que $m_{1-b} = m_0 + m_1 - m_b$. Ainsi, il suffit calculer*

$$M' = M_0 \oplus M_1 \oplus (M \circ -1)$$

Ce qui s'implémente comme suit avec Paillier,

$$M' = M_0 \times M_1 \times M^{-1} \mod n^2$$

Exercice 5. *On supposera que les exercices sont numérotés de 0 à 9.*

1. *Alice génère $(pk_A, sk_A) \leftarrow RSA.KeyGen(\lambda)$, choisit aleatoirement $r_A \in \mathbb{Z}_n^*$, calcule $R_A \leftarrow RSA.Encrypt(pk_A, r_A)$ et envoie (pk_A, R_A) à Bob.*
2. *Bob choisit ensuite aleatoirement $r_B \in \mathbb{Z}_n^*$ et envoie r_B à Alice.*
3. *Alice ...*
4. *Bob ...*
5. *Bob et Alice retournent $r_A + r_B \mod 10$*

Exercice 6.

1 - On propose le protocole suivant :

---

1. *Le prof génère $(pk, sk) \leftarrow Paillier.KeyGen(\lambda)$ et publie pk*
2. *Chaque étudiant $i = 1, \ldots, t$ génère $M_i \leftarrow Paillier.Encrypt(pk, m_i)$, m_i étant sa note et l'envoie au délégué.*
3. *Le délégué génère $M = M_1 \oplus \cdots \oplus M_t$ (correspond au produit des encryptions M_i modulo n^2 avec Paillier) et l'envoie au prof.*
4. *Le prof calcule $res \leftarrow Paillier.Decrypt(sk, M)$ et envoie res/t à tous les étudiants.*

---

2 - Le protocole souffre de plusieurs faiblesses :

- *Un étudiant peut mettre des notes négatives*
- *Aucun controle sur ce que fait le délégué ou le prof qui pourrait, par exemple, envoyer 10 aux étudiants à la fin du protocole.*
- *Si le délégué et le prof se coalisent, ils peuvent connaitre la note de chaque étudiant.*

3 - Il faut que le prof puisse prouver au délégué qu'il a correctement décrypté M. Ceci peut facilement être réalisé (voir propriètés du cryptosystème de Paillier au chapitre 6, haut de la page 37).