C. Olivier-Anclin & Y. Zhao

TD 2 – Cryptographie asymétrique, hachage et signatures

Exercise 1 (Chiffrement RSA)

1. Rappeler le système de chiffrement RSA (génération de clés, chiffrement, déchiffrement).
2. Soit $p = 11$ et $q = 23$. Calculer n et $\varphi(n)$.
3. On choisit $e = 7$. Chiffrer le message $M = 9$ avec l'exponentiation modulaire rapide. Détailler les étapes.
4. Calculer la clé privée d associée (algorithme d'Euclide étendu), puis déchiffrer le chiffré $C = 42$.
5. Montrer que le déchiffrement est correct, c'est-à-dire que pour tout message $M \in \mathbb{Z}_n^*$, $M^{ed} \equiv M \pmod{n}$.
6. Rappeler le lien entre le problème de factorisation et la sécurité de RSA : est-ce que casser RSA *équivaut* à factoriser n ?

Exercise 2 (RSA avec module partagé)

Soit $N = pq$ un module RSA avec $p \neq q$ premiers impairs.

1. Pour le chiffrement RSA, parmi les entiers N, p, q, φ, e et d, quels sont ceux qui doivent rester secrets ?
2. À l'aide des éléments publics, montrer que la divulgation d'une quelconque des valeurs privées permet de retrouver toutes les autres valeurs privées.
3. Expliquer pourquoi il est dangereux d'utiliser deux couples de clés (e_1, d_1) et (e_2, d_2) avec $\text{pgcd}(e_1, e_2) = 1$ et le même module $N = pq$ pour deux utilisateurs différents, même si les utilisateurs se font confiance.

Exercise 3 (Maléabilité de RSA)

1. (a) Rappeler la propriété d'homomorphisme multiplicatif de RSA. En déduire une expression de $\mathsf{Dec}(c_1 \times c_2)$ où Dec est la fonction de déchiffrement et c_1, c_2 les chiffrés de m_1, m_2.
   (b) On vous donne un chiffré c à déchiffrer. Vous avez accès à un oracle de déchiffrement (qui refuse de déchiffrer c lui-même mais déchiffre n'importe quel autre message). Comment récupérer le clair de c ?
2. Alice possède une paire de clés RSA de module $N_1 = pq$. Pour générer une seconde paire de clés « pour gagner du temps », elle génère seulement un nouveau premier r et utilise $N_2 = qr$. Décrire comment casser les deux clés en quelques microsecondes.
3. Une implémentation naïve signe avec $\sigma = m^d \bmod n$ sans hacher ni bourrer. Expliquer pourquoi un attaquant peut forger une signature valide pour un message de son choix s'il obtient la signature d'un message arbitraire de son choix. (Indice : multiplicative.)

Remarque : ces attaques expliquent pourquoi on n'utilise jamais « RSA brut » en pratique (mais RSA PKCS#1 OAEP pour le chiffrement, et RSA-PSS ou hachage puis exponentiation pour les signatures), et pourquoi il ne faut jamais coder sa propre cryptographie.

Exercise 4 (Fonctions de hachage et paradoxe des anniversaires)

1. Rappeler les trois propriétés de sécurité d'une fonction de hachage cryptographique (préimage, seconde préimage, résistance aux collisions) et les classer de la plus forte à la plus faible.
2. Une fonction de hachage retourne une sortie uniforme dans un espace de taille 2^n. Après q requêtes, justifier que la probabilité de n'avoir *aucune* collision est $P(\text{pas de collision}) = \prod_{i=0}^{q-1} \left(1 - \frac{i}{2^n}\right)$.
3. En utilisant $1 - x \approx e^{-x}$ pour x petit, montrer que $P(\text{pas de collision}) \approx \exp\left(-\frac{q(q-1)}{2^{n+1}}\right)$.
4. En déduire la probabilité de collision, puis montrer qu'une collision survient avec probabilité $1/2$ dès que $q \approx 1{,}177 \cdot 2^{n/2}$ (borne des anniversaires).

5. Applications : (a) pourquoi une sortie de 64 bits est-elle insuffisante ? (b) quel niveau de sécurité contre les collisions offre SHA-256 ? (c) pourquoi les signatures basées sur des hachés de 128 bits (MD5) ont-elles été abandonnées ?

Exercise 5 (RSA et théorème des restes chinois)
Soit $N = pq$ avec p et q deux nombres premiers vérifiants $p < q < 2p$. Soit e un entier premier avec $\phi(N)$, d l'inverse de e modulp $\phi(N)$. $d_p = d \mod (p-1)$ et $d_q = d \mod (q-1)$

1. Dans un premier temps rappeler le théorème des restes chinois.
2. En utilisant les valeurs de d_p et d_q, expliquez comment accélérer le déchiffrement/la signature RSA. pour le modulo N en commencant par faire le calcul modulo p et modulo q. Montrer que l'algorithme obtenu est (essentiellement) quatre fois plus rapide de l'algorithme naïf.
3. Supposons que par accident (ou par intervention d'un adversaire) l'utilisateur fasse une erreur dans le calcul lors des calculs de la question précédents. Montrer comment obtenir la factorisation de N à partir des valeurs retournées par le déchiffrement/la signature erroné et la signature correct du même message.

   Pour accélérer encore le calcul, une idée est d'utiliser une clé secrète d telle que d_p et d_q sont relativement petits avec $1 < d_p, d_q < K$ (et $d_p \neq d_q$), pour K connu.
4. Proposer un algorithme qui retourne la factorisation de N à partir du couple (N, e) en effectuant $\mathcal{O}(K + \log(e))$ opérations arithmétiques sur des entiers inférieurs à N.

Exercise 6 (Signatures : ElGamal et Schnorr)
On rappelle le schéma de signature ElGamal dans un groupe $G = \langle g \rangle$ d'ordre premier q :

— Clé privée $x \in_R \{1, \ldots, q-1\}$; clé publique $y = g^x$.

— Signer m : choisir $k \in_R \{1, \ldots, q-1\}$ avec $\text{pgcd}(k, q) = 1$; poser $r = g^k$ et $s = k^{-1}(H(m) - xr) \bmod q$. La signature est (r, s).

— Vérifier : $g^{H(m)} \stackrel{?}{=} y^r \cdot r^s$.

1. Montrer que la vérification est correcte.
2. Montrer que si le même aléa k sert à signer deux messages $m_1 \neq m_2$, on peut retrouver la clé privée. (Faire le calcul dans le cas q premier.)
3. Conclure sur l'importance du générateur aléatoire dans les signatures (lien avec le piratage de la PS3 en 2010 et la faille Android `SecureRandom` en 2013).
4. La même faille d'aléa existe-t-elle pour RSA (signature par exponentiation d'un haché) ? Que conclut-on sur les « signatures déterministes » (RFC 6979, EdDSA) ?