TP 1: RSA - Sécurité et Pratiques

Avertissement

L'objectif du TP consiste à consolider les notions vues en cours et en TD. **Il ne s'agit nullement d'un TP destiné à l'industrialisation !** Il existe des outils conçus pour réaliser des connexions sécurisées.

Il y a ainsi quelques faiblesses dans le TP. Voici les plus importantes :

- Taille des clés très faible et l'utilisation de RSA,
- Il n'y a ni le bourrage optimal, ni le réseau de Feistel asymétrique (Optimal asymmetric encryption padding : OAEP)
- Le HMAC s'appuie seulement sur une fonction de hachage pour la signature et cette fonction n'est pas sécurisée (md5). Bien que le remplacement de cette fonction soit demandé, l'échange n'est pas sécurisé.
- Le TP utilise la conversion en décimal (mauvaise manipulation de l'encodage) pour un objectif pédagogique.

En outre, l'ensemble des fonctions commencent par home pour rappeler qu'il s'agit bien d'une approche simplifiée faite maison.

Objectifs pédagogiques

Ce TP a pour but de prendre conscience des éléments suivants :

- Les mécanismes fondamentaux du chiffrement asymétrique via RSA,
- Les attaques par canaux auxiliaires,
- L'optimisation par le théorème du reste chinois (CRT),
- Le rôle du bourrage (padding) dans le cadre du chiffrement.

Sujet

Le TP vise à illustrer le déploiement d'un ensemble de mécanismes cryptographiques pour sécuriser l'échange entre deux parties : Alice et Bob. Le TP considère un message envoyé de Bob vers Alice, où chacun dispose d'une paire de clés (publique, privée) pour le chiffrement RSA. Bob chiffre le message avec la clé publique d'Alice. Il procède aussi à la signature de l'empreinte numérique du message avec sa clé privée. Alice reçoit le message le déchiffre et vérifie la signature de Bob.

Pour ce faire, le TP est accompagné d'un fichier RSA_B_A.py (version simplifiée) contenant les éléments donnés dans le tableau suivant :

Variables	Alice	Bob
Les deux nombres premiers	x1a et x2a	x1b et x2b
La fonction d'Euler et n	phia, na	phib , nb
Exposants publique, privé	ea, da	eb, db
Le message en clair	dechif (en string)	secret (en string), num_sec (en nombre décimal)
Le message chiffré	chif (message chiffré en nombre décimal)	chif (message chiffré en nombre décimal)
Le hash (empreinte)	Ahachis3 (en nombre décimal)	Bhachis3 (en nombre décimal)
Vérification de la signature	designe (en nombre décimal, déchiffré avec la clé publique de Bob)	signe (en nombre décimal, chiffré avec la clé privée de Bob)

Partie 1 : Implémentation de base du RSA

Reprenez le fichier RSA_B_A.py (version simplifiée fournie).

1. Complétez les fonctions suivantes :
    a. home_mod_expnoent(x, y, n) : exponentiation modulaire rapide.
    b. home_ext_euclide(y, b) : algorithme d'Euclide étendu.
2. Simulez un échange :
    a. Bob chiffre un message avec la clé publique d'Alice.
    b. Il signe le hash du message avec sa propre clé privée.
    c. Alice déchiffre le message et vérifie l'intégrité via la signature.
3. L'utilisation de la fonction MD5 est dépréciée. Nous vous recommandons de passer à SHA256 :
    a. Remplacer la md5 par sha256 et apporter les modifications nécessaires
    b. Avec le changement de la fonction MD5, il est possible d'écrire des messages plus longs. Définir la nouvelle limite du message (sur la version initiale, la limite de 10 caractères).

Partie 2 : Attaque par canal auxiliaire (Simulation) et Optimisation via le CRT (Chinese Remainder Theorem)

1. Implémentez une fonction home_mod_expnoent_trace qui :
    a. Réalise l'exponentiation modulaire,
    b. Compte le nombre d'opérations de multiplication conditionnelles (simulation de l'observation des pics d'activité.).

   c. Définir ce que l'attaquant peut déduire sur la clé privée en observant les pics d'activité.
2. (Bonus) Comparez les empreintes temporelles de plusieurs messages.
3. Modifier l'algorithme de RSA afin qu'il soit plus léger grâce au théorème du reste chinois (Voir le document CRT_RSA.pdf) :
   a. Calculez dp = d % (p-1) et dq = d % (q-1)
   b. Utilisez le CRT pour reconstituer le message.
   c. (Bonus) Expliquez pourquoi CRT réduit la surface d'attaque.

Partie 3 : Bourrage (Padding PKCS#1v1.5-like)

Enfin, vous remarquez que la taille du message échangé entre Alice et Bob est limitée par la taille de la clé. Afin d'étendre la taille du message et de permettre de chiffrer correctement les messages de valeurs faibles, il est possible de procéder au découpage du message par blocs et au bourrage.

1. En suivant l'hypothèse selon laquelle Bob envoie le message m à Alice, il faut procéder de la manière suivante :
   a. Définir une taille limite de chaque bloc que nous notons k
   b. Bob : chaque bloc ne doit pas contenir plus de 50% du message que nous notons $m_i$et dont la taille en octets est j.
   c. Ainsi, m=$m_1 \| m_2 \| m_3 \| \ldots \| m_i \| \ldots \| m_n$ et $j \leq k/2$.
   d. Bob : Générer $k-j-3$ octets non nuls (Bob utilise alea%255+1 pour chaque octet). Le nombre issu de cette génération est noté x.
   e. Bob : Constituer le bloc de la forme suivante : $00\|02\|x\|00\|mi\|$, avec 00, 02 sont des octets valant 0 et 2 en hexadécimal.
   f. Bob : Chiffrer le bloc avec RSA.
   g. Alice : Déchiffrer le bloc avec RSA
   h. Alice : Eliminer $00\|02\|x\|00$ de chaque bloc pour obtenir mi
   i. Alice : Concaténer l'ensemble de mi pour constituer le message initial.

Cette approche est inspirée de PKCS#1v1.5. Dans PKCS#1v1.5 la taille minimale de x est de 8 octets. Il est important de souligner que cette approche est vulnérable aux attaques avec un serveur qui nous informe lorsqu'il s'agit d'un mauvais bourrage (Un exemple d'attaque similaire sera vu en TD).

2. Veuillez vous rendre sur le site https://link.springer.com/journal/43684 et observer le certificat de *.springer.com et identifier les éléments vus dans ce TP à savoir :
   a. L'algorithme de chiffrement asymétrique, la taille de la clé publique et la valeur de son exposant public
   b. L'algorithme de calcul de l'empreinte numérique
   c. L'algorithme de bourrage

À rendre

Le rapport de TP et le code sont à rendre individuellement. Vous déposez le rapport et le code dans la section devoir dédiée à ce TP. Seuls les codes en Pythons sont acceptés et les rapports en format pdf. Donc vous avez deux/trois fichiers à télécharger :

- Nom_Prenom_main.py : implémentation principale
- Nom_Prenom_util.py (si nécessaire)
- Nom_Prenom.pdf : Explication des éléments réalisés dans chaque partie

Bon courage