Sécurité Appliquée : chiffrement asymétrique, TP

Jean-François COUCHOT
couchot[arobase]femto-st[point]fr

21 février 2023

1 RSA : mise en pratique pour du chiffrement.

Exercice 1.1 (Exploitation d'un code pour chiffrer un texte selon RSA).

1. *Installez éventuellement le module* `pycryptodome`.
2. *Récupérez les codes suivants (ceux du CM) et modifiez l'extension des fichiers :*
   - *génération et sauvegarde de clefs RSA ;*
   - *lecture de clef publique, chiffrement selon RSA et sauvegarde du chiffré ;*
   - *déchiffrement selon RSA et affichage du clair.*
3. *Générez votre clef publique et votre clef privée.*
4. *Exécutez le code suivant, expliquez le, expliquez la sortie :*

```
from Crypto.PublicKey import RSA

with open('mPubKey.pem','rb') as f:
    pubKey = RSA.importKey(f.read())

with open('mPrivKey.pem','rb') as fe:
    privKey = RSA.importKey(fe.read())

print(f"Public key:  (n={hex(pubKey.n)}, e={hex(pubKey.e)})")
print(f"Private key: (n={hex(privKey.n)}, d={hex(privKey.d)})")
```

5. *Diffusez à un·e voisin·e votre clef publique et récupérez la sienne.*
6. *Chiffrez un message avec sa clef publique et transmettez lui·elle le message chiffré.*
7. *Déchiffrez son message.*

Exercice 1.2 (Transfert d'une clef AES et chiffrement avec celle-ci).

1. *Générez une clef pour AES à partir d'un générateur pseudo aléatoire crypto-sécurisé et une fonction de hachage sûre (cf. TP précédent sur AES).*
2. *Transférez une version chiffrée de cette clef AES à votre voisin·e selon RSA en exploitant le code de l'exercice ci-dessus.*
3. *Demandez à votre voisin·e de chiffrer une image quelconque avec la clef AES partagée et de vous envoyer ce chiffré.*
4. *Déchiffrez l'image en question.*

2 RSA : mise en pratique pour signer un document.

Exercice 2.1 (Mise en place d'une signature avec `pycryptodome`).

1. *Le code donné à la figure 1 est une implantation d'une signature électronique de documents. Le lire, le comprendre dans les détails.*
2. *Pourquoi la signature est-elle valide dans le premier cas et invalide dans le second ?*
3. *Exploiter ce code pour signer une image de votre choix.*
4. *A partir de cette image ; construisez-en une seconde image en choisissant un pixel au hasard et en diminuant de 1 la valeur de celui-ci.*
5. *Vérifiez que l'image attaquée n'est pas valide pour la signature de l'image originale.*

```python
from Crypto.PublicKey import RSA
from Crypto.Signature.pkcs1_15 import PKCS115_SigScheme
from Crypto.Hash import SHA256
import binascii

# source : https://wizardforcel.gitbooks.io/practical-cryptography-for-developers-book/
#                content/digital-signatures/rsa-sign-verify-examples.html

# Generate 1024-bit RSA key pair (private + public key)
keyPair = RSA.generate(bits=2048)

# Sign the message using the PKCS#1 v1.5 signature scheme (RSASP1)
msg = 'A message for signing!'
hash = SHA256.new(msg.encode())
signer = PKCS115_SigScheme(keyPair)
signature = signer.sign(hash)
print("Signature:", binascii.hexlify(signature))

# Verify valid PKCS#1 v1.5 signature (RSAVP1)
msg = msg
hash = SHA256.new(msg.encode())
signer = PKCS115_SigScheme(keyPair)
try:
    signer.verify(hash, signature)
    print("Signature is valid.")
except:
    print("Signature is invalid.")

# Verify invalid PKCS#1 v1.5 signature (RSAVP1)
msgb = msg[:-1]+":"
hash = SHA256.new(msgb.encode())
signer = PKCS115_SigScheme(keyPair)
try:
    signer.verify(hash, signature)
    print("Signature is valid.")
except:
    print("Signature is invalid.")
```

FIGURE 1 – Code de signature