TP 2

Contexte:
Vous travaillez dans le département de criminologie informatique du département de surveillance des cyber-territoires. Les services secrets ont intercepté un message chiffré (texte chiffré) envoyé par une personne (sous surveillance) à son ami. Le message a été chiffré à l'aide de l'algorithme de chiffrement symétrique **AES-256-CBC**. La clé avec laquelle ce message a été chiffré a été dérivée d'un mot de passe envoyé chiffré par l'algorithme de chiffrement asymétrique **RSA**.

Mission:
Vous devez casser la clé et trouver le message initial (texte brut). Selon les premières informations dont vous disposez, l'outil qui a été utilisé pour générer la paire de clés asymétriques a été mal configuré. Par conséquent, la qualité des nombres premiers choisis était médiocre. Le module de chiffrement a été calculé par la multiplication de deux nombres premiers (un grand p et un q relativement petit) choisis au hasard à condition que leur produit soit de la taille souhaitée (**2048 bits** dans cet exemple).

Remarque*:*
Si cette clé RSA avait été correctement générée, il aurait été très difficile de la casser avant des années !

Les éléments rassemblés par les services secrets sont:

1) La clé AES chiffrée par la clé publique:

H98nEFiiFhdijycQeX3STJBdxhNR1ltTowiO7k0VOOYw65Z3PYGb7uwTidQYWE8H3WgKJ8D+oZqL3a2f0wem5W06VAq9LeSx7S2TAN+wgbFneVhsntDEBDVHh7DoHspA+jI5edQGfwNpdCFMDpZ/KVyWyH9hfcdpTh+ZSMNC8niFt94yMHZmfE9cLxhJ+SNIOrIq5Br2caNHoKBY/xKwpY46Jsr1sei6OrTwU732L79yL5XpDquFiy60/pyNhqJCu+93G97L5I6i0wGM9BAG3RDXRNKaBYmA4V/z0hIFbkg0DQqs32bG7ZyLH9A4UXmaX8M3kU1ZJhLXS9+h9j881/rWGg==

2) Message chiffré avec la clé symétrique:

MeJLrKcJ7g/Imr3n20T23QcajdeYOv6z5OEt/F0B3bnhcrY9ttExmp0/C5Fnujl9

3) Clé publique du récepteur:

-----BEGIN PUBLIC KEY-----
MIIBJTANBgkqhkiG9w0BAQEFAAOCARIAMIIBDQKCAQQAjaVpGbUm1FIlrO1L5kUizvBKY5ELn2/+prESVUEBO+RdSLb7JnG3VA5qTgtV46nkxFqNX1SgaZxlMtShKH+ssAixxW411gHcKp4uZlGJ6qNdIte+olLB7PJwMatlfVs16CzecPglnS4U6YbzYuPobnvY5IEqUvLozC9puLDJWXeP2yQKjhfLlXJFcBLYO2xycpDlC459oo/r36v1I9oDuUuUMir0IfnKAq3mBNqrks2cKCRENvEV/b7XbSyFAHrKf85JiTr4DnlVYy7HuJ9W+rAYduD+iCmaNzQNQ5yy4Bs8B+YMiEdCG/4EnVmVQGsm9KCLINFJs8YarqNTHWL4+NHihwIDAQAB
-----END PUBLIC KEY-----

Travail à fournir:

1) Ecrivez un programme dans n'importe quelle langue de votre choix afin de trouver les nombres premiers (p et q) afin de reconstruire la clé privée.

2) Écrivez la clé privée au format ASN.1 DER. ASN.1 est une norme permettant la représentation de données. La structure ASN.1 d'une clé privée RSA est la suivante:

```
RSAPrivateKey ::= SEQUENCE {
version           Version,
modulus           INTEGER,  -- n
publicExponent    INTEGER,  -- e
privateExponent   INTEGER,  -- d
prime1            INTEGER,  -- p
prime2            INTEGER,  -- q
exponent1         INTEGER,  -- d mod (p-1)
exponent2         INTEGER,  -- d mod (q-1)
coefficient       INTEGER,  -- (inverse of q) mod p
otherPrimeInfos   OtherPrimeInfos OPTIONAL
}
```

La version vaut toujours 0 et le *publicExponent* est fixé à 65537 (valeur par défaut pour toutes les clefs générées avec l'outil OpenSSL). Créer un fichier *Private.txt* qui contient :

```
asn1=SEQUENCE:rsa_key

[rsa_key]
version=INTEGER:0
modulus=INTEGER: VALUE OF MODULUS IN INTEGER
pubExp=INTEGER: VALUE OF e IN INTEGER
privExp=INTEGER: VALUE OF d IN INTEGER
p=INTEGER: VALUE OF p IN INTEGER
q=INTEGER: VALUE OF q IN INTEGER
e1=INTEGER: VALUE OF d mod (p-1)  IN INTEGER
e2=INTEGER: VALUE OF d mod (q-1)  IN INTEGER
coeff=INTEGER:14
```

3) Déchiffrer le fichier chiffré (qui contient la clé AES) avec la clé privée et y récupérer le mot de passe.
4) Déchiffrer le message chiffré avec AES-256-CBC (sans *salt*).

Appendix:

La commande **asn1parse** est un utilitaire de diagnostic permettant d'interpréter des structures ASN.1. Il peut aussi être utilisé pour extraire des informations de données au format ASN.1.

OPTIONS

-inform DER|PEM

the input format. **DER** is binary format and **PEM** (the default) is base64 encoded.

-in filename

the input file, default is standard input

-out filename

output file to place the DER encoded data into. If this option is not present then no data will be output. This is most useful when combined with the **-strparse** option.

-noout

don't output the parsed version of the input file.

-offset number

starting offset to begin parsing, default is start of file.

-length number

number of bytes to parse, default is until end of file.

-i

indents the output according to the "depth" of the structures.

-oid filename

a file containing additional OBJECT IDENTIFIERs (OIDs). The format of this file is described in the NOTES section below.

-dump

dump unknown data in hex format.

-dlimit num

like **-dump**, but only the first **num** bytes are output.

-strparse offset

parse the contents octets of the ASN.1 object starting at **offset**. This option can be used multiple times to "drill down" into a nested structure.

-genstr string, -genconf file

generate encoded data based on **string**, **file** or both using ASN1_generate_nconf format. If **file** only is present then the string is obtained from the default section using the name **asn1**. The encoded data is passed through the ASN1 parser and printed out as though it came from a file, the contents can thus be examined and written to a file using the **out** option.

-strictpem

If this option is used then **-inform** will be ignored. Without this option any data in a PEM format input file will be treated as being base64 encoded and processed whether it has the normal PEM BEGIN and END markers or not. This option will ignore any data prior to the start of the BEGIN marker, or after an END marker in a PEM file.