V1.0

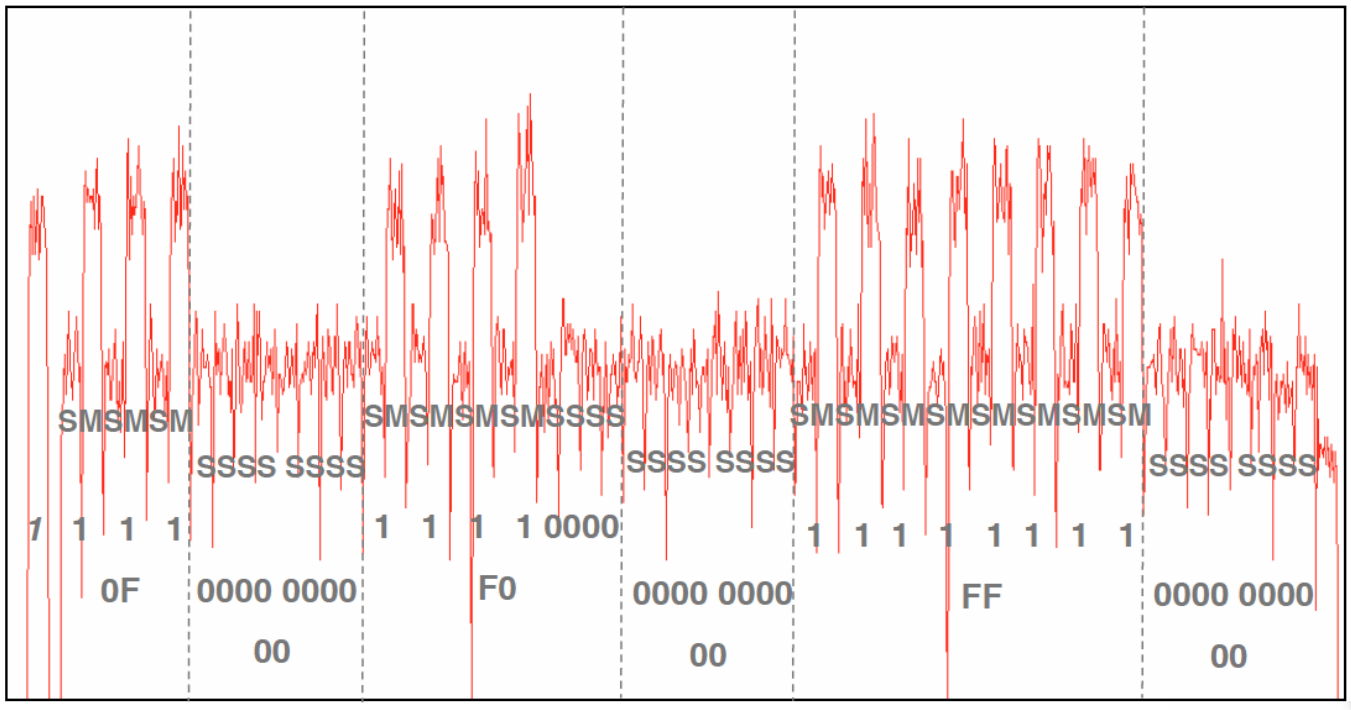
TP EMSE – 2021 : CPA sur RSA

On considère une exponentiation modulaire avec le parcours de l'exposant de gauche à droite suivant l'algorithme :

```
M_d_mod_N(M,d,N)
      T :=M
      For(i=len(d)-2 ;i<=0 ;i--)
            T:=T² mod N
            If(d[i]==1)
                  T:=T*M mod N
            End If
      End For
End
```

On considère la courbe de consommation issue de l'algorithme précédent comme étant un vecteur de nombres C[0..2*(len(d)-2)] où C[i] représente la consommation à l'instant i.

La courbe C représente donc schématiquement la consommation en courant d'un RSA telle qu'illustrée sur la courbe rouge ci-dessous. Chaque C[i] étant une valeur d'échantillonnage de C, c'est à dire un point de la courbe au cours du temps.



V1.0

Le but de ce TP est de retrouver la clef secrète d qui a servi à faire les exponentiations modulaires de ce « baby RSA ».

Pour ce faire, vous allez développer une attaque CPA sur RSA.

Vous trouverez dans le répertoire étudiant_i (où i représente le numéro de l'étudiant, du binôme qui vous aura été assigné) :

- Des fichiers curve_i.txt avec i variant de 0 à 999 qui représentent les courbes synthétiques discutées plus haut.
- Des fichiers msg_i.txt qui représentent le message M ayant mené à la courbe synthétique curve_i.txt
- Un fichier N.txt représentant le module utilisé pour le calcul
- Pour information, l'exposant public utilisé est $F\_4=2^{16}-1$

Dit autrement curve_i.txt représente la consommation au cours du temps du calcul $msg\_i^d \bmod N$ avec d la valeur de l'exposant secret que vous devez récupérer par CPA.

Vous avez à votre disposition, outre les fichiers précédents :

- Un article de référence sur la DPA RSA
- Un article de référence sur la CPA sur algorithme à clef secrète
- Une thèse dont le chapitre 1 représente une synthèse (en français) des deux articles précédents

Il vous est donc demandé de déduire la CPA RSA des publications précédentes en vous aidant également du cours.

A l'issue des TP vous fournirez :

- Un document expliquant comment vous avez développé votre attaque en expliquant les différentes étapes et en les justifiant ; graphiques/schémas bienvenus. Ce document stipulera aussi si vous avez trouvé la clef ou pas. Faites concis pour le rapport : 5 pages max. Inutile de rappeler la CPA, les formules, etc...
- Votre code source.
    - Ce code source sera sous la forme d'un seul fichier python (pas de librairie, de package ou autre) appelé CPA_RSA.py
    - Si votre code est capable de trouver la clef, l'exécution de ce fichier par l'interpréteur python produira à l'écran l'affichage de la clef sous sa forme ASCII binaire : '0100111….'. Si jamais un bit n'a pas été trouvé /est incertain, vous le remplacerez par le caractère « ? »
    - Si votre code n'en est pas capable, il doit afficher clairement 'clef non trouvée'. Vous expliquerez dans le rapport votre démarche et ce qui a coincé à la fin.
- La clef d que vous aurez trouvée, si vous l'avez trouvée

Informations complémentaires :

V1.0

- Vous pouvez travailler en binôme
- Pour vous faciliter la tâche, on travaille sur de petits nombres pour ne pas avoir à utiliser des librairies de big number, ce qui n'est pas l'objet du TP. Vu qu'il s'agit d'un « baby RSA » donc vous pourriez bien sûr factoriser N pour retrouver d, mais ce n'est pas le but de la manœuvre ; il vous faut plutôt davantage développer l'attaque CPA et valider ensuite sur cet exemple naïf qu'elle fonctionne. De toute façon, à la fin c'est le code python fournissant la clef qui fera foi pas la valeur de la clef elle même.
- Les courbes ont été légèrement bruitées avec du bruit Gaussien afin d'éviter certaines aberrations numériques (e.g. division par un écart type nul); normalement ça ne doit pas empêcher l'attaque.

Modus Operandi

-
-
- A l'issue du TP, vous m'enverrez par email (pas de lien vers un cloud/share ou autre) 3 fichiers :
    - Code source du fichier python nommé CPA_RSA.py renommé en CPA_RSA.txt pour éviter le filtrage par les serveurs de mail
    - Document explicatif (sous format pdf) de l'attaque avec pour nom de fichier votre nom ou le nom du binôme si vous travaillez en binome.
    - Fichier ASCII d.txt qui donne la valeur binaire de d comme par exemple « 1100110001 ». Vous omettrez les 0 non significatifs et mettrez des « ? » Si des bits n'ont pas été trouvés.

Idéalement, votre code source et votre document doivent être en phase et montrer comment vous retrouvez la clef.