Travail de recherche

Extraction de clé RSA par cryptanalyse acoustique basse fréquence

Donat-Bouillud Vivien
Caballol Cédric

Sommaire

I - Introduction

La cryptanalyse acoustique fait parti des attaques sur canaux auxiliaires au même titre que les attaques par faute ou encore les analyses d'émanations électromagnétiques.

Elle consiste à étudier le bruit généré par une machine chiffrant un message afin d'en déduire des informations cachées.

L'attaque par cryptanalyse acoustique dont nous allons parler à été découverte en 2013 et vise l'implémentation GnuPG du standard OpenPGP qui permet de générer des clés RSA pour encrypter et décrypter des messages.

II - Observation de la fuite acoustique

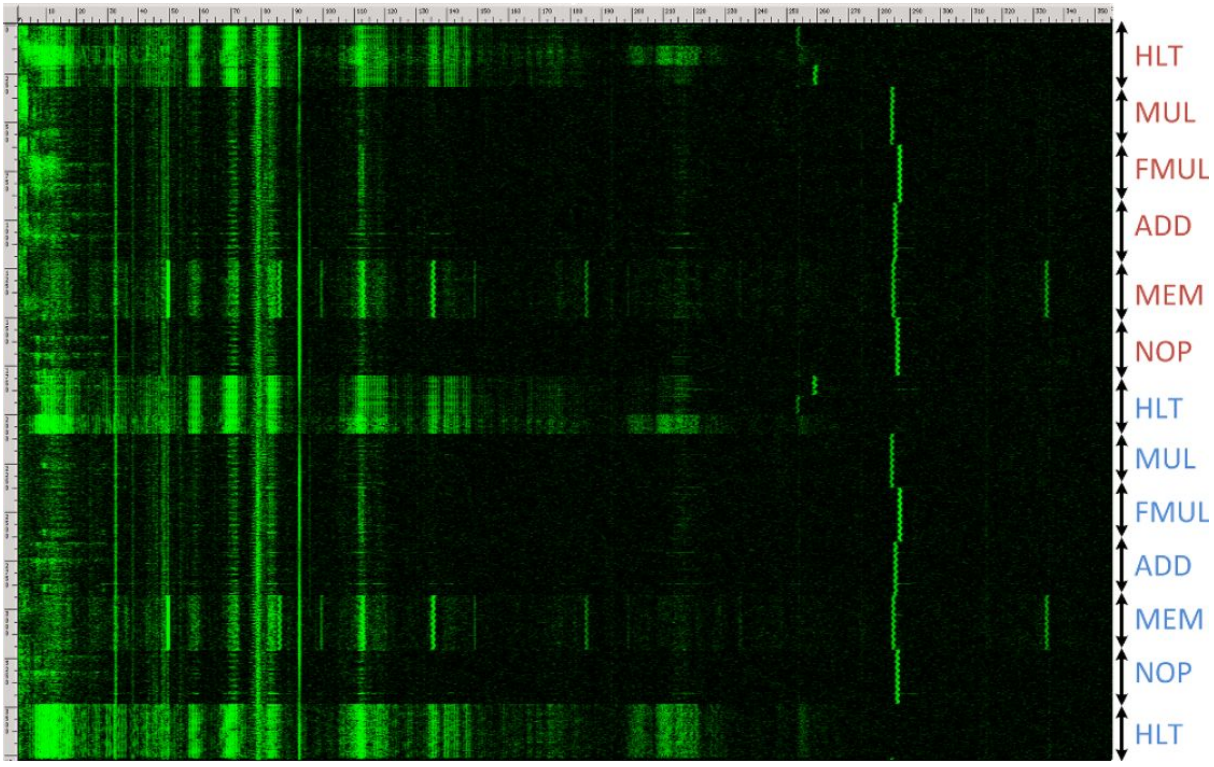
1 - Origine de la fuite

La vitesse d'un processeur est de l'ordre des GHz, par conséquent, il est impossible de faire quelconque analyse acoustique directement sur les instructions du processeur, comme c'est le cas en cryptanalyse électromagnétique, pour la simple et bonne raison que les ondes sonores sont atténuée par l'air, et ce exponentiellement en fonction de la fréquence de l'onde. Il est donc impossible d'entendre quelconque bruit venant du processeur sans être quasiment collé à lui.

L'attaque vise plutôt le module de régulation de tension du processeur, qui est composé de multiples condensateurs qui vont émettre un certain son, en fonction de la tension demandée par le processeur. Ce son quant à lui est de l'ordre de l'ultrason environ vers 35kHz (c'est pour ça qu'on parle de cryptanalyse "basse fréquence").

Module de régulation de tension

La fuite d'information vient du fait que lorsque qu'un processeur fait pendant une certaine durée la même suite d'opérations élémentaires, sa tension ne varie pas, et le module de régulation de tension fait un bruit caractéristique pour cette tension. Ci-dessous vous pouvez voir les différents bruits émis par le module lorsqu'on effectue pendant une seconde la même opération assembleur en boucle.



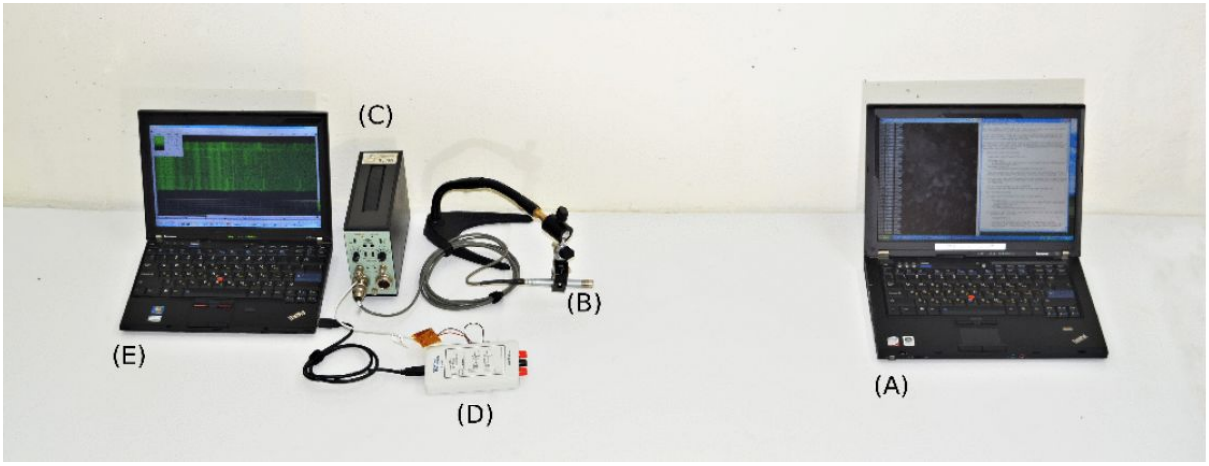
Temps en ordonnée et fréquence en abscisse, mesures effectuées sur 2 CPU différents : en rouge, celui d'un ordinateur portable Compaq Evo N200, en bleu, celui d'un Sony Vaio VGN-T350P

Vous pouvez voir que chaque instruction assembleur demande une tension différente, et émet un bruit particulier, on peut donc les distinguer.

2 - Matériel nécessaire pour observer la fuite

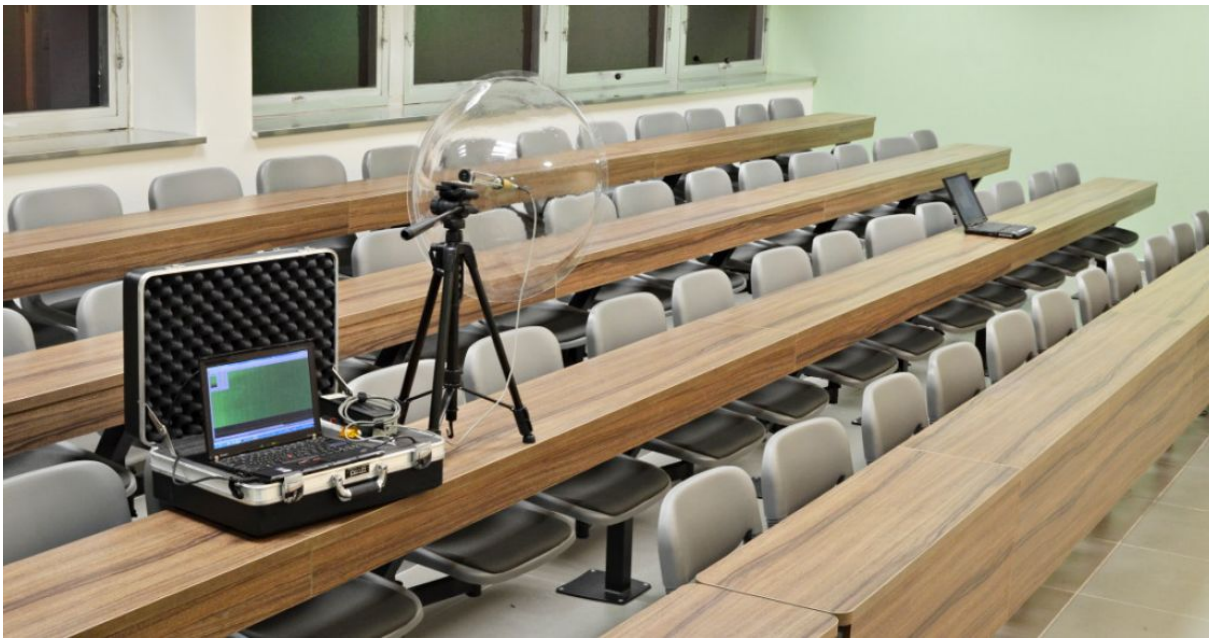
Un des gros avantages de la cryptanalyse acoustique (en tout cas pour exploiter cette fuite), c'est qu'elle ne demande pas de matériel de pointe, cher et peu accessible (comme c'est le cas pour la cryptanalyse électro magnétique).

Seul un micro est indispensable. Pour améliorer les performances, il est possible de rajouter un amplificateur et / ou des filtres fréquentiels, ou encore d'améliorer la qualité du micro mais le principe reste le même, seul la distance d'attaque est impactée.



dispositif “portable” composé d’un micro, d’un amplificateur et d’un filtre, il est possible d’effectuer l’attaque jusqu’à un mètre

dispositif minimal utilisant le micro d’un smartphone, placé en direction de la ventilation de l’ordinateur, l’attaque est possible jusqu’à 30cm de distance



Dispositif utilisant un micro parabolique, l'attaque marche jusqu'à 4 mètres

III - Fonctionnement de RSA

Génération d'une clé

clé privée: (n, d)
avec **n = p x q**
et **p** et **q** deux nombres secrets premiers aléatoires,
et **d** "l'exposant" secret également
clé publique: (n, e)
avec **e** un nombre aléatoire premier avec **(p-1)(q-1)**

Encryptage

$c = m^e \bmod(n)$
Avec **m** le message clair et **c** le message crypté

Décryptage

$m = c^d \bmod(n)$

Cependant, une manière plus rapide de faire est de calculer
$m_p = c^d \bmod(p)$
$m_q = c^d \bmod(q)$

Puis d'utiliser le théorème des restes chinois pour retrouver **m** (ce qui est fait dans GnuPG)

Schéma fonctionnel de RSA

Il est important de noter la décomposition du modulo sur **p** et sur **q** lors du déchiffrement qui est utilisé par GnuPG

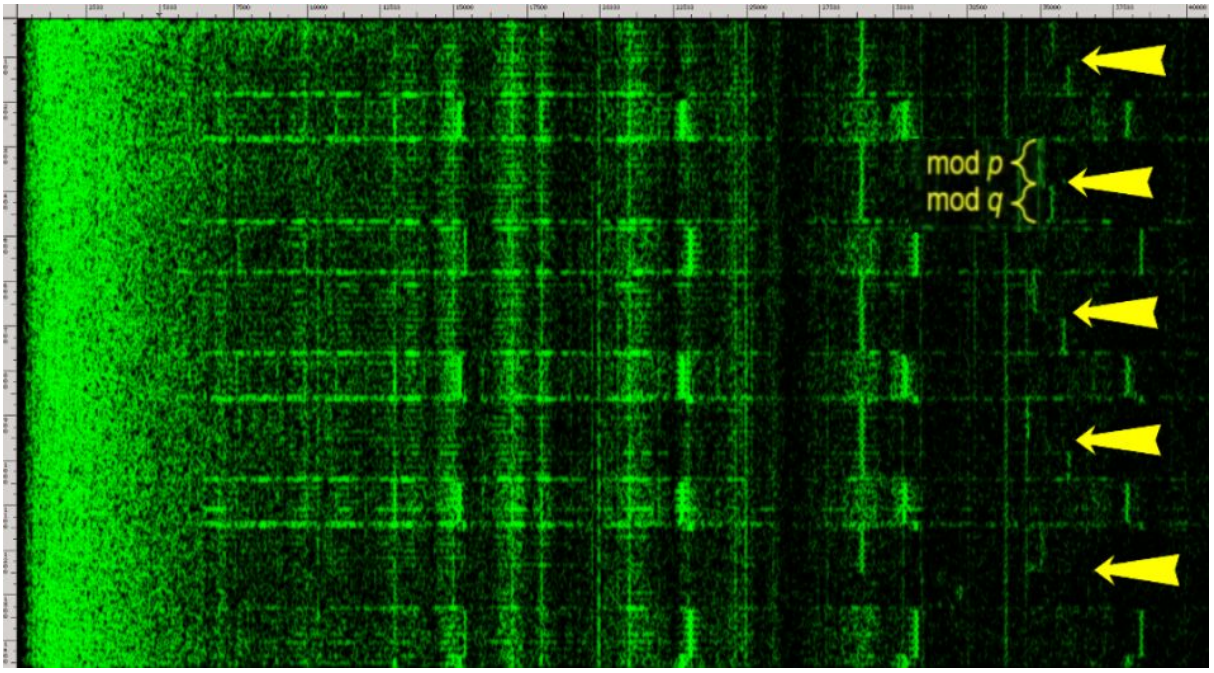
IV - Comment exploiter la fuite ?

1 - Différentiation de clés

Comme rappelé précédemment, l'implémentation de RSA par GnuPG doit, pour décrypter la clé, effectuer les opérations $\mathbf{c^d \bmod p}$ et $\mathbf{c^d \bmod q}$.
Or on sait que p et q sont deux nombres premiers très grands (4096 bits dans la plupart des implémentations) et que la plupart des processeurs actuels fonctionnent sur 64 bits.

On a donc besoin d'une fonction de modulo sur les grands nombres, qui va effectuer plein de modulos intermédiaires sur des nombres plus petits pour arriver à ses fins.

Et comme expliqué dans la section *Origine de la fuite*, effectuer la même opération sur le processeur pendant un certain laps de temps résulte en l'émission d'un son particulier par le module de régulation de tension.



Signature acoustique de 5 déchiffrements de RSA avec des clés différentes, sur chaque déchiffrement on voit le passage du calcul de modulo de p à celui de q (flèches jaunes)

On remarque que chaque clé émet des sons différents pour les modulos de p et de q, et qu'il est donc aisé de distinguer deux clés différentes.
Cela peut paraître bénin, mais cette capacité à distinguer les clés peut présenter un réel intérêt dans certaines applications. On peut prendre pour exemple le fait de

constater qu'une ambassade vient de déchiffrer un message à l'aide d'une clé rarement utilisée, déjà entendue auparavant dans des circonstances diplomatiques spécifiques.

2 - Extraction de la clé par texte chiffrés choisis

Nous avons vu dans l'explication du fonctionnement de RSA qu'une clé privée est composée de *n* et *d*.
Or, on connaît déjà *n* car il fait également parti de la clé publique.
On cherche alors *d*.

Afin de trouver *d*, appuyons-nous sur la façon dont il est choisi :

d est choisi tel que **(ed) mod (p-1)(q-1) = 1**

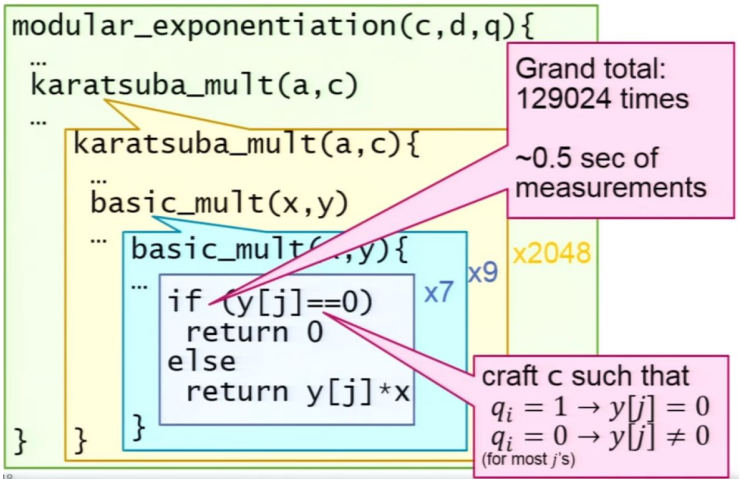
Dans cette formule, il nous manque *d, p* et *q*… Cependant, on sait que *n* (que l'on connaît) vaut *p*q*.

On cherche donc soit *p* soit *q*.

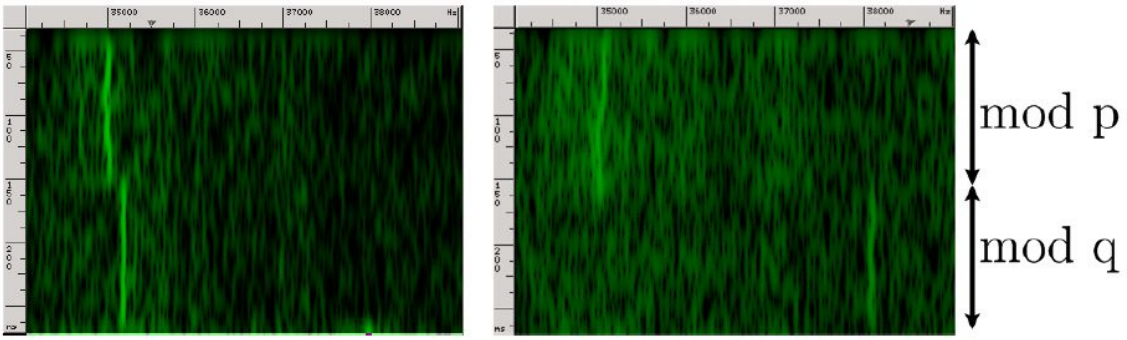
Il a été montré que si on donne un message chiffré *c* choisi tel que *size(c) > size(q)*, alors le calcul de l'exponentiation modulaire de *q* passe en boucle pour chaque bit de *q* dans la branche d'un code faisant une réduction de *c* au modulo *q* et ayant deux sorties possibles :

- la première sortie correspond au cas où q_i vaut 1 ($c \leq q$)
- la deuxième correspond au cas où q_i vaut 0 ($c > q$)

avec q_i étant le i-ème bit de *q*.



Chaque branche émet des sons différents donc on est capable de les distinguer en écoutant :



(a) attacked bit is zero (b) attacked bit is one

Grâce à cette attaque par textes chiffrés choisis, nous pouvons retrouver les 4096 bits de *q* un a un de cette façon :

0) on prend $q_i = 0$ avec $i = 0$
1) on envoie $c = 0111111\ldots$ au déchiffrement
2) en écoutant, on trouve que $c > q$ (par exemple) : q_i vaut donc 0
3) on apporte cette modification à c ($c = 0011111\ldots$), et on incrémente i.
4) on reprend à 1) avec le nouveau message *c*

Maintenant que nous avons *q*, nous pouvons trouver p tel que $p = n/q$.
Enfin, nous pouvons utiliser l'algorithme d'Euclide étendu pour calculer l'inverse du modulo de $(p-1)(q-1)$ afin de trouver *d*.

Le nombre de déchiffrements générés par cette attaque est de $\frac{tailleClé}{2 \cdot 2} \cdot 2$.
On divise par 2 car on a besoin de trouver seulement *q* (et pas *p*).
On re-divise par 2 car il existe une méthode (la méthode de Coppersmith) permettant de trouver *q* en ayant seulement la moitié de ses bits.
On multiplie par 2 car l'écoute de la fuite ne se passe pas toujours bien, donc il y a une correction d'erreurs.
En terme de temps, pour une clé de 4096 bits, l'attaque dure environ 1 heure.

V - Les contre-mesures pour se défendre de la cryptanalyse acoustique

1 - Méthodes inefficaces ou trop difficiles à mettre en oeuvre

1 - 1 - Protections acoustiques physiques

Une idée pourrait être d'absorber le son émis par l'ordinateur cible avec des dispositifs physiques qu'on viendrait placer aux endroits stratégiques… Le problème avec cette méthode est que le bruit dégagé par un ordinateur passe majoritairement par ses ventilations. Or, si on obstrue les ventilations, la durée de vie de l'ordinateur sera fortement réduite...

Une alternative aux ventilations classique serait l'utilisation d'un module de watercooling, mais sa mise en oeuvre peut être compliquée et coûteuse (surtout pour un ordinateur portable).

1 - 2 - Environnement bruyant

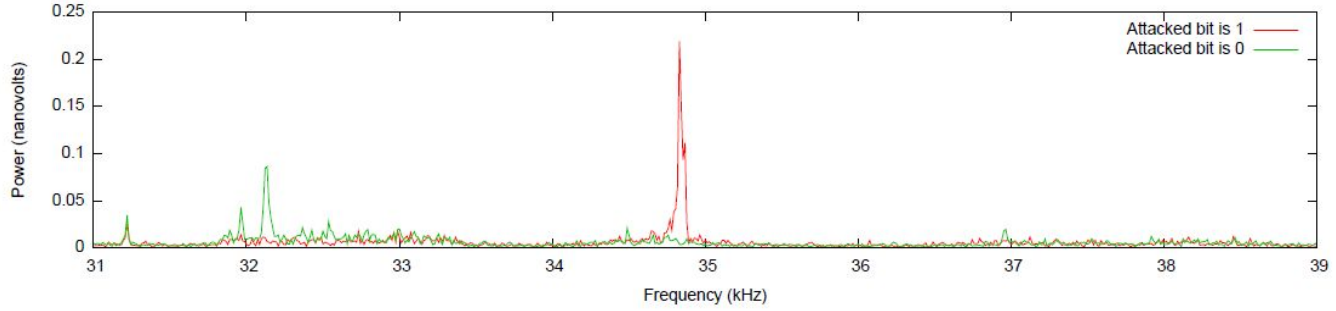
Un environnement bruyant au sens commun du terme (un hall de gare, une salle de concert,...) ne rend pas plus difficile l'extraction de clés privées par cryptanalyse car ces bruits se manifestent principalement dans un spectre en fréquence de 0 à 10 kHz.

Par-contre, un brouilleur assez intelligent pour masquer les signaux aux bonnes fréquences serait efficace. Malheureusement, un tel algorithme de brouillage serait vraiment compliqué à mettre en oeuvre. Il pourrait s'agir d'un algorithme capable de mesurer en temps réel l'impact sonore du déchiffrement sur le module de régulation de tension du processeur. Une fois que la mesure aurait été faite, il devrait être capable de générer un bruit continu dans le même spectre de fréquences afin de masquer l'écoute des déchiffrements.

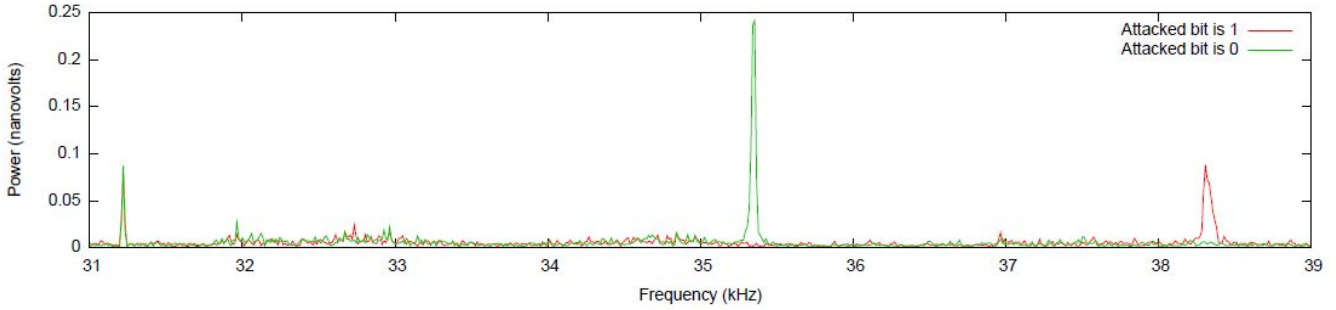
1 - 3 - Exécuter des instructions en parallèle

Exécuter des instructions en parallèle des déchiffrements de messages ne va pas masquer la fuite mais seulement la déplacer dans le spectre en fréquences en la rendant potentiellement plus grave. Comme un son grave, par sa nature, se propage plus facilement qu'un son aigu, il est alors plus facile à capturer, et donc, aide l'attaquant.

Voici le spectrogramme d'une attaque sur un ordinateur décryptant un message et exécutant des instructions en parallèle :



Pour comparer, voici le spectrogramme d'une attaque sans instructions en parallèle :



2 - Méthodes efficaces

2 - 1 - Randomisation de textes chiffrés

La randomisation de textes chiffrés consiste à modifier le déchiffrement en faisant ces étapes :

- on génère une valeur *r* de 4096 bits aléatoires
- on calcule r^e
- on décrypte $r^e.c$
- on multiplie le résultat par r^{-1}

Cela équivaut à déchiffrer directement *c*. Pour s'assurer que cette affirmation est vraie, voici le détail des calculs faits par une telle fonction de déchiffrement :

$$(r^e \cdot c)^d \cdot r^{-1} \mod n = r^{ed} \cdot r^{-1} \cdot c^d \mod n = r \cdot r^{-1} \cdot c^d \mod n = m$$

Comme les exponentiations modulaires sont calculées sur des bits aléatoires, alors l'attaquant ne peut plus choisir ses textes chiffrés et donc l'attaque ne peut plus avoir lieu.

Une telle modification de l'algorithme de déchiffrement a cependant un coût : celui d'une troisième exponentiation modulaire.

Aussi, elle ne permet pas de contrer la discernabilité des clés qui est, elle, indépendante des textes chiffrés.

2 - 2 - Randomisation de modulos

La randomisation de modulos consiste à rendre le modulo aléatoire à chaque exponentiation tel que :

- $m'_q = c^{dq} \bmod tq$
- $m_q = m'_q \bmod q$

Avec *t* un entier positif de taille moyenne.

Ou encore mieux, pendant la boucle de l'exponentiation modulaire, on pourrait remplacer *q* par des multiples de *q*.

Cette protection ajoute un peu de complexité mais fonctionne aussi bien contre l'attaque par textes chiffrés choisis que pour la discernabilité des clés.

2 - 3 - Normalisation des textes chiffrés

La normalisation des textes chiffrés consiste à enlever tous les 0 inutiles en début des messages chiffrés. Comme les textes chiffrés choisis par l'attaquant en ont souvent (pour pouvoir avoir plus de bits que *p* et ainsi déclencher une réduction modulaire afin d'observer la fuite d'informations), une normalisation les enlèverait.

Cette contre-mesure n'ajoute pas beaucoup de complexité au déchiffrement mais ne contre que l'attaque par textes chiffrés choisis.

VI - Conclusion

Bien que cette attaque ne nécessite pas beaucoup de matériel (juste un smartphone dans certains cas), elle reste tout de même assez compliquée à mettre en oeuvre. Nous n'avons donc pas trouvé de témoignage d'attaques de ce type dans un cadre non-universitaire.

Pour ce qui est des contre-mesures, la cryptanalyse acoustique se démarque d'autres types d'attaques par canaux auxiliaires par le fait qu'il est difficile de les contrer avec des dispositifs physiques (atténuer des plages de fréquences de l'ordre des kHz n'est pas aussi simple à mettre en oeuvre que d'atténuer des ondes électromagnétiques par exemple).

VII - Sitographie

RSA Key Extraction via Low-Bandwidth Acoustic Cryptanalysis
Article de recherche : http://www.cs.tau.ac.il/~tromer/papers/acoustic-20131218.pdf
Explications plus concises et d'autres liens en rapport :
https://m.tau.ac.il/~tromer/acoustic/
Vidéo qui montre l'experimentation :
https://www.youtube.com/watch?v=DU-HruI7Q30

GnuPG : https://gnupg.org/
OpenPGP : https://fr.wikipedia.org/wiki/OpenPGP
Voltage regulator module : https://en.wikipedia.org/wiki/Voltage_regulator_module
Rappels sur RSA : https://sebsauvage.net/comprendre/encryptage/crypto_rsa.html
RSA : https://fr.wikipedia.org/wiki/Chiffrement_RSA
Comment calculer d à partir de n, e, p, q ? :
https://stackoverflow.com/questions/23279208/calculate-d-from-n-e-p-q-in-rsa
Algorithme d'Euclide étendu :
https://fr.wikipedia.org/wiki/Algorithme_d%27Euclide_%C3%A9tendu
Méthode de Coppersmith :
https://en.wikipedia.org/wiki/Coppersmith_method
Attaques par canaux auxiliaires :
https://fr.wikipedia.org/wiki/Attaque_par_canal_auxiliaire