Faculté des sciences et techniques Limoges

Développement de logiciels cryptographiques

Attaques side-channel sur génération de premiers

Master 2 CRYPTIS

Auteurs:

Al Halabi Dalal

Bondi Nada

Brika Lydia

Matet Lise

Professeur:

CLAVIER Christophe

Table des matières

I. Attaque de type SPA sur la génération de premiers par la méthode du crible simple

Les articles étudiés décrivent une analyse simple de courant appelée SPA : Simple Power Analysis s'appliquant à la génération de premiers par la méthode du crible simple.

1. Génération des premiers

L'attaquant est supposé pouvoir observer dans la trace de courant correspondant à la génération d'un premier les différentes étapes successives de tests de divisibilité par chacun des petits premiers.
Pour cela les premiers générés seront enregistrés dans un fichier sous forme de lettre :

A si p_i ne divise pas q.
B si on incrémente q, c'est-à-dire q=q+2 (lorsque p_i divise q ou si test de Miller-Rabin de q retourne composé).
C si q n'est divisible par aucun des petits premiers, donc on vérifie qu'il est bien premier avec le test de Miller-Rabin.

Cette succession de lettres représentera notre trace de courant.
Pour générer les premiers, nous avons eu recours à la bibliothèque GMP et la méthode de crible simple.

Optimized Sieve
1: pick a random l-bit odd integer 2: j ← 2 [pj is the first j th prime] 3: while j ≤ k do 4:   if pj divides q then 5:   q ← q + 2 and go to step 2 6:   j ← j + 1 7:   if T(q) = false then 8:   q ← q + 2 and go to step 2 9: return q

A l'issu de la génération des premiers, la trace de courant est créé sous la forme ci-dessous pour p et q distinctement qui seront représentés par p_m et q_m dans la suite.

```
1 AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAACBAAAAAAABBABAA
```

2. Attaque

a. Description de l'attaque

Dans cette section, on décrit et analyse notre attaque.
En effet, l'attaquant est supposé pouvoir observer dans la trace de courant générée précédemment les différentes étapes successives de tests de divisibilité par chacun des petits premiers vu que ces tests sont différenciés par les lettres.

Pour tout candidat à la primalité qi , il est en conséquence capable de connaitre la valeur du plus petit des petits premiers pj qui divise ce candidat, ou bien de savoir le faite qu'aucun d'entre eux ne le divise.
Lorsqu'une relation déterministe comme l'incrémentation de q : q ← q + 2 relie les candidats successifs, l'ensemble de ces informations modulaires peuvent être ramenées à de l'information sur le dernier candidat testé (le premier q généré).
Dans le cas de la génération d'une clé RSA, les informations modulaires relatives aux générations des deux premiers peuvent être rassemblées et permettre de factoriser le module.

b. Implémentation de l'attaque

A partir de cette étape et à la suite de la génération des traces de courants de p et q, on se met à la place de l'attaquant.
On néglige le fait qu'on connait le secret p_m et q_m et on essaie en tant qu'attaquant de les retrouver en utilisant les traces de courants.
On crée les deux ensemble s_p et s_q, composés des petits nombres premiers tels que :
s_p = {ensemble des p_j tels que p_j divise un des candidats q_j de la génération du premier p}.
s_q = {ensemble des p_j tels que p_j divise un des candidats q_j de la génération du premier q}.

Ces ensembles nous donnent des informations modulaires sur p et q.

Si l'attaquant arrive à récupérer un nombre par exemple 19 appartenant à l'ensemble s_p, alors il sait que p mod 19 est congru à 0 (car 19 divise p) et de même pour les premiers de l'ensemble s_p.
L'attaquant pourra donc retrouver p à partir du système d'équation suivant :

$$\left.\begin{array}{rl} v_j \equiv & 0 \pmod{r_i} \\ v_j = & v_0 + 2j \\ p = v_m = & v_0 + 2m \end{array}\right\} \Rightarrow p = v_j + 2(m-j) \equiv 2(m-j) \pmod{r_i}.$$

Nous combinons toutes les équations du système ci-dessus via le théorème des restes chinois (CRT) modulo le produit des premiers de s_p (noté x).
Cela donne une congruence

$$a_p \equiv p \pmod{s_p} \quad \text{for} \quad s_p := \prod_{r \in \mathcal{S}_p} r$$

Avec a_p connu et comme pq = n on aura donc:

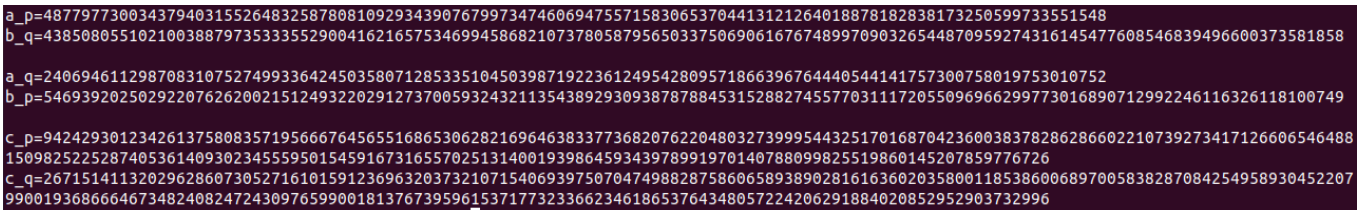
$$a_q :\equiv q \equiv a_p^{-1} n \pmod{s_p}.$$

3. Tests et résultats :

Après avoir étudié l'attaque présentées, on a procéder à l'implémentation en simulant l'obtention des informations issues de la SPA pour l'attaque, tout en s'intéressant à la quantité d'information acquises en relation avec le premier généré.

En simulant l'attaque on a tenté d'afficher les valeurs calculées pour pouvoir ensuite comparer aux résultats analysés dans l'article.

La figure ci-dessous montre les valeurs récupérées de a_p , b_q , A_q et B_q



Pour comparer nos résultats à ceux présents dans l'article nous devons exécuter le processus de génération de RSA 10000 fois pour chacune de plusieurs paires (k, N).

Faute de matériel, nous n'avons pas pu faire les 10000 itérations suite à une insuffisance de capacité.

Cependant, on a pu obtenir les résultats illustrés dans les tableaux ci-dessous pour 1000 itérations:

K=512			
N	R_n	Prob(log2(s)>256	Prob(log2(s)>277
54	251	0.046000	0.018000
60	281	0.041000	0.060000
70	349	0.039000	0.121000

K=1024			
N	R_n	Prob(log2(s)>512	Prob(log2(s)>553
100	541	0.019000	0.008000
110	601	0.029000	0.027000
120	659	0.036000	0.084000

Pour conclure cette section, il faut mentionner qu'on avait finalement obtenu des résultats logiques, mais on a pas pu les comparer aux résultats de l'article suite à l'abscence du matériel performant pour les tests.

II. Attaque de type CPA horizontale sur la génération de premiers par la méthode du crible optimisé.

Nous utiliserons l'article : Side Channel attack against RSA key Generation Algorithms, ainsi que le pseudo code contenu dans le cours.

Dans ce paragraphe, nous effectuons un type d'attaque side channel sur le crible optimisé appelé CPA, où au lieu d'obtenir plusieurs traces de courant basées sur différentes entrées et sorties, nous divisons une trace de courant en plusieurs parties selon le candidat avec lequel nous travaillons.

1. Explications des différentes étapes de cette partie

Lors d'une CPA normale il y a plusieurs traces synchrones, on cherche le moment du calcul des donnée et de l'écriture en mémoire, c'est le point d'intérêt. L'attaquant fait une hypothèse de clé (h) telle que y = s(m XOR h), puis on calcule le poids de Hamming de y. On calcule y pour chaque courbe. On prends chaque valeurs de courant qui sont équivalentes à la valeur du poids de Hamming de y. Si l'hypothèse est fausse on aura une différence entre les traces et les calculs. On étudie tout les points d'intérêts possibles pour obtenir une corellation si on veut des résultats réels.

Lors d'une CPA horizontale, on sait calculer des valeurs et trouver leurs consommations. On doit trouver une bonne hypothèse de clé pour cela. On peut le faire sur plusieurs points d'intérêts, exactement comme pour la CPA verticale. On cherche les résidus des candidats. Les qi sont dépendants les uns des autres. Le h c'est la valeur supposée du résidu par exemple modulo 37. On prends comme inconnue la valeur initiale du candidat. Pour trouver la bonne hypothèse nous devons conduire un test exhaustif. On fait une trace constituée de toutes les valeurs rij. Pour faire un tronçon de la trace et on incrémente. On va jusqu'à 256 par exemple c'est à dire 53 premiers. La trace aura 53 fois le nombre de candidats. Dans la trace on a que les poids de Hamming du résidu (nous travaillerons sur les poids de Hamming). Pour ce projet on n'a mis en place que des traces constituées uniquement de points d'intérêts. Il faut prendre les numéros des premiers (le nième premier) et on prends la valeur de trace numéro le nième premier modulo le premier courant. Et dans la trace on mets les vrais poids de Hamming. Pour plus de réalisme on prends a et b au hazard et on fait a * Hw() + b. Comme c'est linéaire on aura le même résultat en utilisant le coeffiscient de corellation (si g est bon on sera toujours égal à un). Quand on change de premier, par exemple, si le premier vaut 5 on a 5 hypothèses à faire. Si on a beaucoup de mauvaises hypothèses on a des chances de faire un faux positif. Sinon à l'inverse le nombre de bits permet d'avoir des valeurs de poids de Hamming en plus. Du coup quand le poids de Hamming est obligatoirement petit on vois moins de choses car les traces se ressemblent trop. Dans un deuxième temps on bruite donc a et b ça ne change pas le résultat donc c'est les mesures qu'on bruite. Donc par exemple au lieu de 115 on prends 113,6 etc. Pour limiter les erreurs, on prends beaucoup de points. Le niveau de bruit est égal à l'écart type du bruit. On veut un bruit de moyenne 0 et d'écart type au choix. On cherche le taux de succès de l'attaque avec les différents bruits. Il y a un ratio égal à a avec l'écart type sur le poids de Hamming et sur la consommation. On testera différents écarts types.

Nous avons réalisé différentes étapes pour implémenter cette seconde attaque :

1. Ecrire l'algorithme du crible optimisé ainsi que trouver n.
2. Simuler la consommation d'énergie :

Pour comprendre le plan, nous devons nous familiariser avec les notations :
-S = {s0,...sλ-1} est l'ensemble des nombres premiers que nous utilisons, il est appelé crible et son cardinal est égal à λ.
-Nous avons supposé que l'algorithme passe par n itérations, c'est-à-dire que nous avons commencé avec v0 et terminé avec vn-1 le nombre premier généré.
-rij : le reste de ri (i+1ème candidat) par sj (j+1 l'élément du tamis).
-lij est la trace à rij.

Maintenant, nous devons simuler les traces, pour cela nous avons un vecteur long de λ x n, contenant la valeur qui est supposée être des traces de courant. Puisqu'il y a λ nombres premiers, nous devons avoir λ valeurs différentes, ces valeurs doivent être ajoutées aux étapes 7 et 12 de l'algorithme 1 ci-dessous.

3. Réaliser l'attaque :

-écrire le code de l'algorithme 2
-créer un vecteur flottant de taille λ qui contiendra plus tard la valeur de différents score[h] pour différentes valeurs de h.
-Créer un autre vecteur de taille λ qui contiendra les candidats.
-En utilisant l'algorithme 3, calculez le score[h].
-le h qui a le score le plus élevé de score[] est le candidat à l'indice j

4. appliquer l'algorithme des restes chinois sur l'ensemble {candidat[k]mod sk,k ∈ {0,...,λ -1}} et la valeur que nous avons trouvée est le premier ˆp.

2. Le crible optimisé

Pour chaque candidat, nous effectuons une division successive modulo tous les nombres premiers dans l'ensemble des nombres premiers, chacune de ces divisions sera notre point d'intérêt.

Algorithm 1 Optimized Sieve

```
1: pick a random l-bit integer q=v_0
2: for j=2 to k do
3:     r_j ← q mod s_j
4: end for
5: while r_j = 0 for some j ∈ {2,...,λ} do
6:     for j = 2 to k do
7:         r_j ← (r_j + 2)mod s_j
8:     end for
9:     q ← q + τ
10:    if T(q) false then
11:        for j = 2 to k do
12:            r_j ← (r_j + 2) mods_j
13:        end for
14:        q ← q + τ and go to step 5
15:    end if
16: end while
17: return q
```

Commençons par expliquer plus en détail le crible optimisé, nous générons un nombre impair de taille k (en bits), nous testons sa divisibilité sur tous les petits nombres premiers, s'il est divisible par l'un d'entre eux, nous ajoutons deux à ce candidat modulo chaque nombre premier et 2 au candidat lui-même, puis nous répétons le processus, jusqu'à ce qu'aucun de ces petits nombres premiers ne divise le candidat. Nous effectuons ensuite un test de primalité, si le candidat passe le test, nous nous arrêtons

là, sinon nous ajoutons 2 comme nous l'avons fait précédemment et nous répétons le processus jusqu'à ce que notre candidat passe les deux tests.

3. L'attaque

Maintenant que nous avons décrit le crible optimisée, passons à l'attaque. Comme nous l'avons mentionné au début, nos points d'intérêt sont les divisions des candidats par les petits nombres premiers, nous "mesurons" les traces de courant à ces points (puisque nous ne disposons pas d'outils de mesure, nous avons simulé la mesure de courant en calculant le poids de Hamming du candidat i modulo le premier j et nous avons ajouté un bruit gaussien pour simuler le bruit réel.

Après avoir effectué nos mesures, nous commençons l'attaque : pour chaque nombre premier, nos modules candidats actuels peuvent avoir n'importe quelle valeur allant de 1 à p-1 (p premier), donc pour chacune de ces valeurs, nous obtenons une trace de courant calculée comme le poids de Hamming sur la photo, en combinant toutes les valeurs des traces de courant pour chaque h, nous obtenons une trace de courant, nous mesurons ensuite sa corrélation avec la trace réelle et le h avec la plus grande valeur de p sera choisie.
On fait ça pour chaque nombre premier et on obtient une série d'équations modulo tous les petits nombres premiers. En utilisant le théorème du reste chinois, on retrouve les modulos premiers du produit des petits nombres premiers.

Algorithm 2 Horizontal correlation attack

Measurements Phase
for $i = 0$to $n-1$ do
  for $j = 0$to $\lambda - 1$ do
    measure l_{ij}
  end for
end for
Attack phase
for each sieve, perform a partial SCA
for $j = 0$to $\lambda - 1$ do
  test each possible candidate
  for $h = 1$to $s_j - 1$ do
    By processing predictions
    for $i = 0$to $n-1$ do
      $m_{ij} = HW(h - (n-i-1)2 \mod s_j$
    end for
    and applying a statistical distinguisher
    $\rho[h] = \Delta((m_{ij})_i, (l_{ij})_i))$
  end for
  Then select the most likely candidate
  candidate[j]=argmaxh(score[h])
end for
Apply the Chinese Remainder Theorem CRT
$\hat{p} = CRT(candidate[0] \mod s_0, ..., candidate[\lambda - 1] \mod s_{\lambda-1})$

Algorithm 3 Calculating the coefficient of correlation

```
Input:2 vectors A and B of size N
Output:  the correlation coefficient ρ
function SUM(X)
    sum ← 0
    for i = 1toN do
        sum ← sum + X[i]
    end for
    return sum
end function
function AVG(X)
    avg ← Sum(A)/N
    return avg
end function
function OP(X, x)
    for i = 1toN do
        Y[i] = X[i] − x
    end for
end function
function MUL(X, Y)
    for i = 1toN do
        Z[i] = X[i].Y[i]
    end for
end function
Ā = Avg(A) , B̄ = Avg(B)
A′ = Op(A, Ā) ,B′ = Op(B, B̄)
C = Mul(A′, B′)
cov = (1/N) sum(C)
var1 = (1/N) sum(mul(A′A′))
var2 = (1/N) sum(mul(B′, B′))
rho = cov / √(var1.var2)
return ρ
```

4. Tests et résultats :

Dans cette dernière partie nous avons testé sur des nombres de candidats sur 1024 bits et 512 bits. Ce qui nous donne respectivement :

```
vect[1980]=1522
vect[1981]=1522
vect[1982]=1522
vect[1983]=1567
vect[1984]=1567
vect[1985]=1567
vect[1986]=1567
vect[1987]=1624
vect[1988]=1624
vect[1989]=1624
vect[1990]=1851
vect[1991]=1851
vect[1992]=1851
vect[1993]=1851
vect[1994]=1862
vect[1995]=1862
vect[1996]=1862
vect[1997]=2127
vect[1998]=2127
vect[1999]=2127
quar1=89
quart2=288
quart3=487
```

1 résultats sur 1024

```
vect[1966]=660
vect[1967]=660
vect[1968]=660
vect[1969]=660
vect[1970]=660
vect[1971]=660
vect[1972]=715
vect[1973]=715
vect[1974]=715
vect[1975]=715
vect[1976]=715
vect[1977]=715
vect[1978]=715
vect[1979]=715
vect[1980]=715
vect[1981]=738
vect[1982]=738
vect[1983]=738
vect[1984]=738
vect[1985]=738
vect[1986]=738
vect[1987]=738
vect[1988]=738
vect[1989]=738
vect[1990]=738
vect[1991]=979
vect[1992]=979
vect[1993]=979
vect[1994]=979
vect[1995]=979
vect[1996]=979
vect[1997]=979
vect[1998]=979
vect[1999]=979
quar1=54
quart2=159
```

2 résultats sur 512

Or dans le documents les résultats ont été testés sur des premiers de 512 bits , les résultats sur ces premiers attendus étaient 53, 126 et 246. Ce qui paraît cohérent avec les résultats que nous avons obtenus.

and 1024 respectively. In the sequel, we focus on the 512-bit case (even if the outlines of our approach could also be followed to study the two other cases) since generating primes of that size is for instance required when constructing a 1024-bit RSA modulus (*e.g.* for some banking applications) or when generating strong primes according to the ANSI X9.31 standard [1]. For a 512-bit prime, the *median* of the distribution of n as well as the first and third *complementary quartiles*[11], are respectively equal to 53, 126 and 246. The quartiles Q_1, Q_2 and Q_3 related to 75%, 50% and 25% are represented by horizontal lines in Figure 1.

3 Résultats de l'expérience obtenue dans les documents