RSA

A. Dragut, Univ. Aix-Marseille
Cours de cryptographie Chapitre III

Definition. [Le problème de la factorisation entière] *Étant donné un entier N, calculer des facteurs premiers non triviaux de N (différents de 1 et N).*

L'algorithme de cryptage (RSA) de Rivest, Shamir et Adleman est système de chiffrement à clé publique (asymétrique) : $K_e \neq K_d$. Le problème RSA est basé sur le problème de la factorisation entière qui est "difficile" pour $N = pq$ impaire.

Definition. [Le problème RSA] *Étant donné (N, e, y) avec $y \in (\mathbb{Z}/N\mathbb{Z})^*$ et $N = pq$, trouver x tel que $y \equiv x^e \pmod N$.*

Un nombre N de la forme pq où p et q sont deux grands nombres premiers est appelé un module RSA.

La bibliographie de ce chapitre est :

- Understanding Cryptography - (2010) de Christof Paar, Jan Pelzl, Bart Preneel
- An Introduction to Mathematical Cryptography - (2008) de Jeffrey Hoffstein, Jill Pipher, et J.H. Silverman
- Cryptography Theory and Practice - (2002) de Doug Stinson
- Subhamoy Maitra, Santanu Sarkar : Revisiting Wiener's Attack - New Weak Keys in RSA. ISC 2008 : 228-243
- Twenty years of attacks on the RSA cryptosystem. by D. Boneh Notices of the American Mathematical Society (AMS), Vol. 46, No. 2, pp. 203-213, 1999

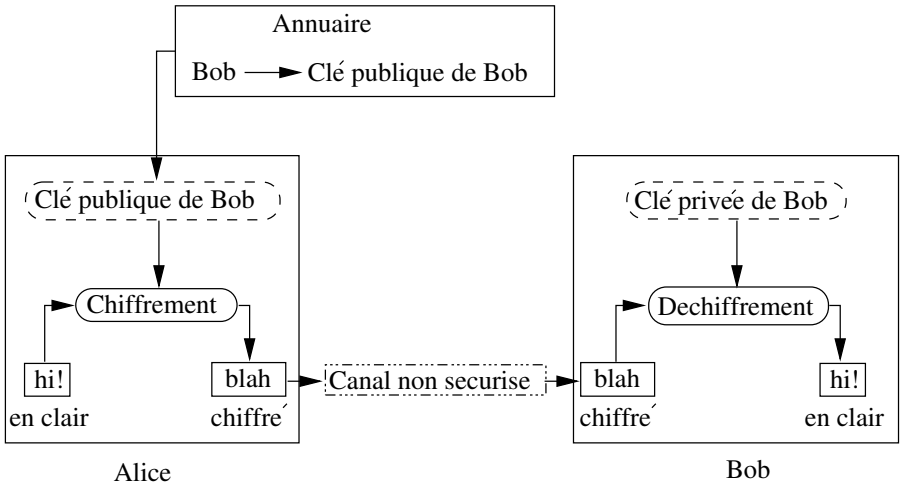

FIGURE 3.1 – Principe du chiffrement à clé publique

3.1 L'algorithme

Première étape : la création des clés de Bob pour RSA

```
Input Bob: rien
Output Bob: Clé publique (N, e); Clé privée d
begin
 Choisir deux grands nombres premiers p et q
 Calculer N = pq et l'indicateur d'Euler φ(N) = (p − 1)(q − 1)
 Choisir un entier 1 < e < φ(N) premier avec φ(N), c.à-d. pgcd(e, φ(N)) = 1
 Calculer d l'inverse de e (mod φ(N))
end
```

Désormais, toute personne ayant accès à la clé publique de Bob peut lui envoyer des messages sécurisés. A aucun moment, Alice n'est intervenue dans la création de la clé publique de Bob.

Remarque. *La fonction $f(x) = x^e \pmod N$ est un exemple de permutation à trappe (trap-door one-way fonction). L'entier d est une trappe qui permet de résoudre le problème RSA.*

Pourquoi ? Bob publie sa clé publique (N, e) dans un annuaire. Sa clé privée est d. La sécurité du protocole RSA vient du fait que seul Bob connaît la factorisation de N. Si on sait factoriser N en trouvant ses facteurs p et q, on peut trouver x car on peut calculer $\phi(N) = (p-1)(q-1)$ et le calcul de la trappe d (et le calcul de l'inverse $y^{-e} \pmod N$) peut être réalisé efficacement en $\log(N)$ étapes grâce à l'algorithme d'Euclide étendu.

Trouver l'inverse modulo se fait en utilisant le Théorème Bachet-Bézout et l'algorithme d'Euclide étendu ou bien en utilisant Le petit Théorème de Fermat et son corollaire d'Euler (voir chapitre : Anneaux modulo N).

Les calcules de RSA se font dans les anneaux de restes modulo N pour le cryptage/décryptage et modulo $\phi(N) = (p-1)(q-1)$ pour la création de la clé publique de RSA. Plus précisément dans leurs sous-groupes multiplicatifs $(\mathbb{Z}/N\mathbb{Z})^*$ et $(\mathbb{Z}/\phi(N)\mathbb{Z})^*$, formés par leurs éléments inversibles. La cardinalité (l'ordre) du sous-groupe multiplicatif $(\mathbb{Z}/N\mathbb{Z})^*$ est égal à la fonction indicatrice d'Euler $\phi(N)$.

Deuxième étape : Protocole RSA

Input general:La clé publique (N, e)
Input Bob: la clé privée d
Input Alice: un message x avec $x < N$
Output Bob: le message x est reçu
begin Cryptage
– Alice calcule $y \equiv x^e \pmod N$
– Alice envoie y à Bob
end Cryptage
begin Décryptage
– Bob calcule $x \equiv y^d \pmod N$
end Décryptage

Le corollaire suivant donne une condition suffisante sur e et sur d pour faire en sorte que le cryptosystème RSA fonctionne pour tout $x \in (\mathbb{Z}/N\mathbb{Z})^*$. :

Lemme 1. ***(RSA fonctionne correctement pour les messages*** $x \in (\mathbb{Z}/N\mathbb{Z})^*$***)*** *Soit deux entiers premiers p et q et $N = pq$. Soit un entier d tel que $ed \equiv 1 \pmod{\phi(N)}$, où $\phi(N)$ est la fonction indicatrice d'Euler. Pour tout $x \in (\mathbb{Z}/N\mathbb{Z})^*$ $x^{ed} \equiv x \pmod N$. Donc $y^d \equiv x \pmod N$ pour $y \equiv x^e \pmod N$.*

Preuve. *On peut réécrire la condition $ed \equiv 1 \pmod{\phi(N)}$ ainsi : il existe un entier k tel que $ed = 1 + k\phi(N)$. Selon le théorème d'Euler pour tout $x \in (\mathbb{Z}/N\mathbb{Z})^*$, $x^{\phi(N)} \equiv 1 \pmod N$. Par conséquent, $y^d \equiv (x^e)^d \equiv x^{ed} \equiv x^{1+k\phi(N)} \equiv x \cdot (x^{\phi(N)})^k \equiv x \cdot 1^k \equiv x \pmod N$.*

La méthode RSA fonctionne pour tout $x \in \mathbb{Z}/N\mathbb{Z}$. Le théorème d'Euler ne s'applique pas aux $x \notin (\mathbb{Z}/N\mathbb{Z})^*$ donc nous devons utiliser le théorème des restes chinois et le fait que

$$y^d \pmod N \equiv x^{ed} \pmod N$$

Théorème. ***(RSA fonctionne correctement)*** *Soit deux entiers premiers p et q et $N = pq$. Soit un entier d tel que $ed \equiv 1 \pmod{\phi(N)}$, où $\phi(N) = (p-1)(q-1)$ est l'indicateur d'Euler. Pour tout $x \in \mathbb{Z}$ $x^{ed} \equiv x \pmod N$. L'entier d est donc une trappe qui permet de résoudre le problème RSA.*

Preuve. *On peut réécrire la condition $ed \equiv 1 \pmod{\phi(N)}$ comme : il existe un entier u tel que $ed = 1 + u\phi(N)$.*

Parce que p et q sont deux entiers premiers, si x^{ed} vérifie $x^{ed} \pmod p \equiv x$ et $x^{ed} \equiv x \pmod q$ on peut appliquer le théorème des restes chinois. Alors pour $N = pq$ le système a une unique solution en $\mathbb{Z}/N\mathbb{Z}$. Donc $x^{ed} \equiv x \pmod N$.

Prouvons-les.

Si $x \equiv 0 \pmod p$, x est divisible par p. Donc x^{ed} est divisible par p et $x^{ed} \equiv 0 \pmod p$. Si $x \neq 0 \pmod p$, x est premier avec p et en appliquant le Petit Théorème du Fermat $x^{p-1} \equiv 1 \pmod p$. Alors $x^{ed} \equiv x^{1+u\phi(N)}$ Comme $\phi(N) = (p-1)(q-1)$ nous avons que $x^{ed} \equiv x(x^{p-1})^{u(q-1)} \equiv x \cdot 1^{k(q-1)} \equiv x \pmod N$. Pour tout entier x l'égalité $x^{ed} \equiv x \pmod p$ est vérifié. Pareil pour q.

Remarque. ***(Failles RSA) Que dire des messages $x \in \mathbb{Z}/N\mathbb{Z} - (\mathbb{Z}/N\mathbb{Z})^*$?***

Le nombre de messages dans $\mathbb{Z}/N\mathbb{Z} - (\mathbb{Z}/N\mathbb{Z})^*$ est de seulement $N - \phi(N) = pq - (p-1)(q-1) = p + q - 1$, alors que le nombre total de messages possibles est $N = pq$. Si p et q ont chacun 512 bits, alors la probabilité de choisir aléatoirement un mauvais message est de seulement de $2 \cdot 2^{512}/2^{1024} = 1/2^{511}$.

Si $x \in \mathbb{Z}/N\mathbb{Z} - (\mathbb{Z}/N\mathbb{Z})^*$, alors $pgcd(x, N) > 1$. Mais $N = pq$ avec p, q nombres premiers, donc soit $p|x$ soit $q|x$ (mais pas les deux, car $x < N = pq$). Supposons que $pgcd(x, N) = p$, alors $pgcd(y, N) = pgcd(x^e, N) = p$ aussi.

Si Alice envoie un tel message x et Eve est suffisamment rusée pour calculer le pgcd(y, N) (ce qu'elle peut faire facilement, i.e. complexité logarithmique, avec l'algorithme d'Euclide étendu), alors Eve réussira à déchiffrer le message et même à trouver la clé publique parce que en sachant p elle peut factoriser N et donc calculer la clé priée.

Même si

3.2 Détails implementation RSA

3.2.1 Décryptage rapide. OpenSSL

Pour accélérer le déchiffrement RSA on utilise le théorème chinois des restes. Pour décrypter y on doit faire y^d et $N = pq$. On définit $d_p \equiv d \pmod{p-1}$, $d_q \equiv d \pmod{q-1}$ et supposons qu'on calcule à l'avance les termes $R_p \equiv q^{p-1} \pmod N$ et $R_q \equiv p^{q-1} \pmod N$; puis on effectue séparément les opérations de décryptage suivantes :

$$x_p \equiv y^{d_p} \pmod p$$

$$x_q \equiv y^{d_q} \pmod q$$

Parce que $pgcd(p, q) = 1$ une solution particulière de ce système de congruences est par conséquent $x \equiv R_p \cdot x_p + R_q \cdot x_q \pmod N$ et les autres solutions sont les entiers congrus modulo le produit $N = pq$. Donc on retrouve $x = y^d \pmod N$.

3.2.2 Difficultés de l'exponentiation modulaire

L'opération de base de RSA est l'exponentiation modulaire de grands nombres, c.à-d. le calcul de $x^e \bmod N$ pour de grands nombres x, e, et N. La méthode évidente pour calculer ceci est de calculer d'abord $t = x^e$ et ensuite $t \pmod N$.

1. Le nombre x^e est trop grand pour être stocké ! Ce nombre, lorsqu'il est écrit en binaire, occupe environ $1024 \cdot 2^{1024}$ bits, ce qui est un nombre bien plus grand que le nombre total d'atomes de l'univers (estimé aux alentours de $1080 \cdot 2^{266}$).
2. La boucle itérative simple pour le calcul de x^e a besoin de e multiplications, ou bien environ 2^{1024} opérations en tout. Ce calcul prendrait plus de temps que l'âge actuel de l'universe (estimé à 15 milliards d'années).

Supposant qu'une itération d'une boucle pourraît être effectuée en une microseconde (estimation très optimiste, car on doit calculer à chaque tel pas un produit et un reste de division entière de grands nombres), seulement $30 \cdot 10^{12}$ pourrait être effectuées par an, et seulement $450 \cdot 10^{21}$ itérations pendant toute la durée de vie de l'universe. Mais $450 \cdot 10^{21} \cong 2^{79}$, ce qui est beaucoup moins que $e - 1$.

Remarque (Astuce pour éliminer le besoin de stocker x^e)**.** *Puisque le produit de deux nombres de longueur k est seulement de longueur $2k$ avant la réduction* mod$N$*, l'astuce consiste à combiner les deux pas en réduisant le résultat modulo N après chaque opération arithmétique. On utilise donc le fait que*

$$a \equiv bc \bmod N \equiv (b \bmod N) \cdot (c \bmod N) \bmod N$$

pour décomposer en ses parties un nombre qui pourrait en principe être très grand, et les combiner plus facilement pour obtenir la valeur finale.

Un algorithme plus efficace pour l'exponentiation est basé sur l'opération d'élever au carré. Pour calculer $x^e \bmod N$ avec $e = 2q$, on calcule

$x_0 = x$
$x_1 = (x_0 \cdot x_0) \bmod N$
$x_2 = (x_1 \cdot x_1) \bmod N$
$\ldots$
$x_k = (x_{k-1} \cdot x_{k-1}) \bmod N$.

Pour des valeurs de e qui ne sont pas des puissances de 2, x^e peut être obtenu en tant que produit modulo N de certains x_i. On écrit en binaire $e = \overline{b_s b_{(s-1)} \cdots b_2 b_1 b_0}_2$. Si $b_i = 1$ on inclut x_i dans le produit final pour obtenir x_e.

Les programmes C sont pour des nombres plus petits que 216, après il faut utiliser des bibliothèques pour des opérations avec des grand nombres.

```
int ExpMod( int x, int e, int N) {
 int r = 1;
 tant que ( e > 0 ) {
   si e est impaire alors r = r*x(mod N);/* en C: if((e&1) == 1)*/
   e = e/2;
   x = x*x(mod N);
 }
 retourne r;
}
```

Exemple. *Soit* $N = 11 = 8 + 2 + 1$. *Donc en binaire* $N = 1011$. *Donc pour calculer* x^N, *nous allons utiliser les puissances suivantes :* 1, x, x^2, 1, x^8. *Dans l'algorithme pour calculer* x^8 *on utilise* $(((x^2)^2)^2)$.

3.2.3 Rechercher des nombres premiers

En RSA on recherche p et q premiers et aussi la clé publique e qui est relativement première avec $(p-1)$ et $(q-1)$, c.à-d. $pgcd(e,(p-1)(q-1)) = 1$. Si p et$q$ ne sont pas premiers on risque de pouvoir factoriser N facilement et de trouver la clé privée. Mais le pire est que l'algorithme RSA ne fonctionne plus. Les calculs pour la production de la clé privée doivent se faire modulo $\phi(N)$. Si p et$q$ ne sont pas premiers, alors la fonction indicatrice d'Euler $\phi(N) \neq (p-1)(q-1)$.

Il y a deux philosophies pour rechercher les nombres premiers : les tests de primalité (probabiliste) et les algorithmes de primalité (déterministe). Le critère d'evaluation est celui de leur complexité.

Le crible d'Ératosthène est l'algorithme le plus simple qui cherche les premiers p tel que $p < \sqrt{(N)}$

```
/* Le crible d'Eratosthene */

int Crible( int N, tableau prime[]: prime[1], ... , prime[N]){
    pour tout 1<=k<=N prime[k]= vrai  /* au debut tous les nombres sont premiers*/
    tant que (i<N et prime[N]==vrai){
        si (prime[i]==vrai) alors    /* on vient de trouver un nouveau premier */
            pour tout 1<j<N  { prime[j*i]=faux;}
        i=i+1;
    }
    si (prime[N]== faux) alors i-1 est un facteur de N;
    retourne i-1;
}
```

Pour un module RSA N à 256 bits $N = pq$ est plus grand que 10^{75}. Donc au crible il faut au moins $0,36 \times 10^{36}$ divisions pour décider. Trop !

En pratique nous utilisons des tests de primalité.

Le premier test était basé sur le petit théorème de Fermat (d'Euler).

Algorithme d'Euler

- `Soit` n `un nombre à tester`
- `Tirer au hasard` a `tel que` $1 < a < n$
- `si` $\text{pgcd}(a, n) \neq 1$ `(c.à-d.` a `facteur de` n`) alors` n `pas premier`
- `sinon, si` $a^{n-1} \neq 1 \pmod{n}$`,` n `pas premier`
- `sinon` n `est peut-être premier`

En essayant pour beaucoup d'entiers test a si $a^p = a \bmod p$ on peut déduire que ces entiers a ne sont pas des facteurs de p et que donc peut-être p est un premier.

Il existe des p non premiers (nombres composés) tels que pour tout a avec $\text{pgcd}(a, p) = 1$ et $a^{p-1} = 1 \bmod p$. Ils s'appellent pseudo-premiers pour le Test de Fermat et la base a.

Exemple. *Vérifiez que* $341 = 11 \cdot 31$ *est un pseudo-premier pour le Test de Fermat et la base* 2. *Vérifiez que la base* 3 *est témoin que* 341 *est un nombre composé.*

On vérifie en utilisant l'exponentiation rapide si $2^340 = 1 \pmod{341}$.

$340 = 256 + 64 + 16 + 4 = 2^8 + 2^6 + 2^4 + 2^2$

On sait que $z^{b+c} = z^b \cdot z^c$. Alors

$$\begin{aligned}
2^{340} &= 2^{2^8} \cdot 2^{2^6} \cdot 2^{2^4} \cdot 2^{2^2} \\
2^{340} &= (((((((2^2)^2)^2)^2)^2)^2)^2)^2 \cdot (((((2^2)^2)^2)^2)^2) \cdot 2^{2^4} \cdot 2^{2^2} \\
2^{340} &= (((((256)^2)^2)^2)^2)^2 \cdot (((256)^2)^2)^2 \cdot 256^2 \cdot 16 \\
2^{340} &= (((((256)^2)^2)^2)^2)^2 \cdot (((256)^2)^2)^2 \cdot 65536 \cdot 16
\end{aligned}$$

Mais $65536 > 341$ et on peut réduire modulo 341. $65536 = 192 \cdot 341 + 64 \equiv 64 \pmod{341}$

$$\begin{aligned}
2^{340} &\equiv ((((64)^2)^2)^2)^2 \cdot ((64)^2)^2 \cdot 64 \cdot 16 \pmod{341} \\
2^{340} &\equiv (((4096)^2)^2)^2 \cdot (4096)^2 \cdot 1024 \pmod{341}
\end{aligned}$$

Mais $4096 = 12 \cdot 341 + 4 \equiv 4 \pmod{341}$ et $1024 = 3 \cdot 341 + 1 \equiv 1 \pmod{341}$

$$\begin{aligned}
2^{340} &\equiv ((4^2)^2)^2 \cdot (4)^2 \cdot 1 \pmod{341} \\
2^{340} &\equiv (256)^2 \cdot 16 \pmod{341} \\
2^{340} &\equiv 64 \cdot 16 \equiv 1 \pmod{341}
\end{aligned}$$

Donc 341 est un pseudo-premier pour le test Fermat et la base 2. Similairement on peut calculer $3^{340} \equiv 56 \neq 1 \pmod{341}$. La base 3 est témoin pour le test Fermat du fait que 341 est un nombre composé.

Un nombre n pseudo-premier pour toutes les valeurs de a tels que $pgcd(a, n) = 1$ est un pseudo-premier absolu pour le Test de Fermat. Les nombres de Carmichael sont des tels pseudo-premiers absolus.

Ils ont été la source de nombreux attaques sur les implementations RSA utilisant le teste de Fermat/Euler. Maintenant, il y des testes de primalité amelioré comme Miller-Rabin et Solovay-Strassen.

Remarque. [nombres de Carmichael] *Soit les entiers premiers* $(6k+1), (12k+1), (18k+1)$. *Les nombres* $n = (6k+1)(12k+1)(18k+1) = 1296k^3 + 396k^2 + 36k + 1$, $n-1$ *sont des nombres de Carmichael, parce que ils sont des multiples* $36k$ *et* $lpcm(6k, 12k, 18k) = 36k$. *Nous avons* $a^{(n-1)} \equiv 1 \bmod 6k+1, 12k+1, 18k+1$, *et donc* $a^{(n-1)} \equiv 1 \bmod n$. *Il y en a d'autres.*

Théorème (Korselt)**.** *Un nombre entier posititf composé* n *est un nombre de Carmichael si et seulement si aucun carré de nombre premier ne divise* n *et pour chaque diviseur premier* p *de* n, *le nombre* $p-1$ *divise* $n-1$.

Exemple. *Vérifiez que* $561 = 3 \times 11 \times 17$ *(le premier nombre de Carmichael) est un pseudo-premier absolu pour le test Fermat, i.e.* $a^{560} \equiv 1 \pmod{561}$ *pour tout* $a = 1, 2, \cdots, 560$.

On peut utiliser un logiciel de calcul modulaire comme celui de l'adresse :
http ://ptrow.com/perl/calculator.pl

Miller et Rabin ont modifié le Test Fermat en faisant deux observations :

- si nombre premier p divise un produit $u \cdot v$, alors p/u ou p/v, mais au moins un d'entre eux.
- our un n entier impaire, on peut écrire $n - 1 = 2^k \cdot d$, avec d impaire, ils ont essayé de characteriser les facteurs de $a^{n-1} - 1$.

Le Petit Th. de de Fermat disait que si n est premier $a^{n-1} \equiv 1 \pmod{n}$, alors on peut dire que $n/(a^{n-1} - 1)$. En decomposant $(a^{n-1} - 1)$ en facteurs, le test de Miller-Rabin vérifie si après chaque décomposition n divise au moins un des facteurs. En sachant que tout nombre premier plus grand que 2 est impaire et $n - 1 = 2^k \cdot d$, avec d impaire, alors $k > 1$. On peut donc écrire :

$$a^{n-1} - 1 = a^{2^k \cdot d} - 1 = a^{2 \cdot 2^{k-1} \cdot d} - 1 = (a^{2^{k-1} \cdot d})^2 - 1 = (a^{2^{k-1} \cdot d} - 1)(a^{2^{k-1} \cdot d} + 1) \pmod{n}$$

parce que dans un $x^2 - 1 = (x-1)(x+1)$.

Si n est premier $n/(a^{n-1} - 1)$, alors n doit diviser au moins un des ces deux facteurs. Si $n/(a^{2^{k-1} \cdot d} - 1)$ alors $a^{2^{k-1} \cdot d} \equiv 1 \pmod{n}$. Sinon n doit diviser $(a^{2^{k-1} \cdot d} + 1)$ ce qui équivalent à dire $a^{2^{k-1} \cdot d} \equiv -1 \pmod{n}$

Si $k = 1$ on ne décompose plus, sinon on continue et à chaque nouvelle dècomposition on vérifie que n divise au moins un des facteurs. S'il divise il est peut-être premier, sinon on dit que la base a est témoin du fait que n est composé. Le test de Miller-Rabin n'as pas de nombres pseudo-premiers absolus, donc pour tout nombre composé il y a un témoin. Et ces témoins sont très frequents. Un théorème nous assure que 75% des nombres de $\mathbb{Z}_n$ sont des témoins pour n.

Algorithme Miller-Rabin

- `Soit` n `un nombre à tester`
- `Tirer au hasard` a `tel que` $1 < a < n$
- `si` $\text{pgcd}(a, n) \neq 1$ `(`a `facteur de` n`) alors` n `pas premier`
- sinon on écrit $n = 2^k \cdot d$, , avec d impaire
  - `sinon, si` $a^d \neq 1 \pmod{n}$ `et pour tout` $r \in 0, 1, \cdots, k-1, a^{2^r d} \neq -1 \pmod{n}$ `alors` n `pas premier`
  - `sinon` n `est peut-être premier`

Exemple. *Vérifiez que pour* $n = 561 = 3 \cdot 11 \cdot 17$ *la base* $a = 2$ *est un témoin pour le test Miller-Rabin du fait que* 561 *est un nombre composé.*

Pour le test Miller -Rabin on écrit $n - 1 = 2^k \cdot d$. On remarque que

$$560 = 16 \cdot 35 = 2^4 \cdot 35, k = 4, d = 35$$

Il faut vérifier que $a^d \neq 1 \pmod{n}$ et pour tout $r \in 1, \cdots, k-1$ $a^{2^s \dot{d}} \neq -1 \pmod{n}$.

On remarque que $35 = 32 + 2 + 1$, donc on peut écrire

$$\begin{aligned} 2^{35} &= 2^{2^5} \cdot 2^2 \cdot 2 \\ 2^{35} &= ((2)^{2^3})^2)^2 \cdot 2^2 \cdot 2 \\ 2^{35} &= ((256)^2)^2 \cdot 2^2 \cdot 2 \\ 2^{35} &= (65536)^2 \cdot 2^2 \cdot 2 \end{aligned}$$

Mais, $65536 > 561$ et on peut réduire modulo 561, donc $65536 = 116 \cdot 561 + 460 \equiv 460 \pmod{561}$. Nous avons aussi $460^2 = 211600 = 377 \cdot 561 + 103$ Maintenant on fait le test

$$\begin{aligned} 2^{35} &\equiv 103 \cdot 8 \equiv 824 \equiv 263 \neq 1 \pmod{561} \\ 2^{2 \cdot 35} &\equiv 263^2 \equiv 166 \neq -1 \pmod{561} \\ 2^{4 \cdot 35} &\equiv 263^4 \equiv (263^2)^2 \equiv 166 \cdot 166 \equiv 67 \neq -1 \pmod{561} \\ 2^{4 \cdot 35} &\equiv 263^8 \equiv (263^4)^2 \equiv 67 \cdot 67 \equiv 1 \neq -1 \pmod{561} \end{aligned}$$

Donc, 561 n'as pas passé le test Miller-Rabin.

Exemple. *Le même problème pour* $41041 = 7 \cdot 11 \cdot 13 \cdot 41$.

3.3 Exemples RSA

Exemple (codage RSA simple)**.** *Soit les nombres premiers* $p = 11$ *et* $q = 5$.

1. $N = pq = 11 \cdot 5 = 55$. $\phi = (p-1)(q-1) = 10 \cdot 4 = 40$
2. *Choisissons* $e = 3$.
3. *Vérifions* $\text{pgcd}(e, p-1) = \text{pgcd}(3, 10) = 1$. *Vérifions* $\text{pgcd}(e, q-1) = \text{pgcd}(3, 4) = 1$. *Par conséquent,* $\text{pgcd}(e, \phi) = \text{pgcd}(e, (p-1)(q-1)) = \text{pgcd}(3, 40) = 1$
4. *Calculons* d *tel que* $ed \equiv 1 \pmod{\phi}$, *c.à-d.* $d \equiv e^{-1} \pmod{\phi} \equiv 3^{-1} \pmod{40}$. *Trouver une valeur pour* d *telle que* ϕ *divide* $(ed-1)$, *c.à-d. trouver* d *tel que* 40 *divide* $3d-1$. *Une suite de tests simples* ($d = 1$, $d = 2$, *...) donne* $d = 27$. *Vérification :* $ed - 1 = 3 \cdot 27 - 1 = 80$, *qui est divisible par* ϕ.
5. *La clé publique est* $(N, e) = (55, 3)$
6. *La clé privée est* $(N, d) = (55, 27)$

Soit le méssage à chiffrer $x = 3$. Alors $y \equiv x^e \pmod{N} \equiv 3^3 \pmod{55} \equiv 27 \pmod{55} \equiv 27$. Donc le méssage chiffré est $y = 27$.

Pour vérifier le déchiffrement on calcule $y' \equiv y^d \pmod{N} \equiv 27^{27} \pmod{55}$.

Nous n'avons pas besoin de calculer 27^{27} en entier ici. Nous avons déjà vu une manière de calculer y' par exponentiation rapide en utilisant

$$a \equiv bc \bmod N \equiv (b \bmod N) \cdot (c \bmod N) \bmod N$$

Donc :

$$\begin{aligned} y' &\equiv 27^{27} \bmod 55 \equiv 27^{9+9+9} \bmod 55 \\ &\equiv (27^9 \bmod 55) \cdot (27^9 \bmod 55) \cdot (27^9 \bmod 55) \end{aligned}$$

$$\begin{aligned} 27^9 \bmod 55 &\equiv 27^{(2+2+2+2+1)} \bmod 55 \equiv 27^2 \cdot 27^2 \cdot 27^2 \cdot 27^2 \cdot 27 \bmod 55 \\ &\equiv (27^2 \bmod 55)^4 \cdot (27 \bmod 55) \bmod 55 \\ &\equiv (729 \bmod 55)^4 \cdot (27 \bmod 55) \bmod 55 \\ &\equiv 14^4 \cdot 27 \bmod 55 \equiv 38416 \cdot 27 \bmod 55 \\ &\equiv 26 \cdot 27 \bmod 55 \\ &\equiv 702 \bmod 55 \\ &\equiv 42 \bmod 55. \end{aligned}$$

$$\begin{aligned} y' &\equiv 27^{27} \bmod 55 \equiv (42 \bmod 55) \cdot (42 \bmod 55) \cdot (42 \bmod 55) \\ &\equiv 42^3 \bmod 55 \equiv 74088 \bmod 55 \equiv 3 \end{aligned}$$

Exemple (codage RSA simple)**.** *Soit les nombres premiers* $p = 11$ *et* $q = 3$.

1. $N = pq = 11 \cdot 3 = 33$. $\phi = (p-1)(q-1) = 10 \cdot 2 = 20$
2. *Choisissons* $e = 3$. *Vérifions* $\text{pgcd}(e, p-1) = \text{pgcd}(3, 10) = 1$ *(c.à-d.* 3 *et* 10 *n'ont pas de diviseurs communs à part* 1*). Vérifions* $\text{pgcd}(e, q-1) = \text{pgcd}(3, 2) = 1$. *Par conséquent,* $\text{pgcd}(e, \phi) = \text{pgcd}(e, (p-1)(q-1)) = \text{pgcd}(3, 20) = 1$
3. *Calculons* d *tel que* $ed \equiv 1 \pmod{\phi}$*, autrement dit calculons* $d \equiv e^{-1} \pmod{\phi} \equiv 3^{-1} \pmod{20}$*, c.à-d. trouver une valeur pour* d *telle que* ϕ *divide* $(ed-1)$*, c.à-d. trouver* d *tel que* 20 *divide* $3d-1$*. Une suite de tests simples (*$d = 1$*,* $d = 2$*, ...) donne* $d = 7$*. Vérification :* $ed - 1 = 3 \cdot 7 - 1 = 20$*, qui est divisible par* ϕ.
4. *La clé publique est* $(N, e) = (33, 3)$
5. *La clé privée est* $(N, d) = (33, 7)$

Soit le méssage à chiffrer $x = 7$. Alors $y \equiv x^e \pmod{N} \equiv 7^3 \pmod{33} \equiv 343 \pmod{33} \equiv 13$. Donc le méssage chiffré est $y = 13$.

Pour vérifier le déchiffrement on calcule $y' \equiv y^d \pmod{N} \equiv 13^7 \pmod{33} \equiv 7$. Nous avons déjà vu une manière de calculer y' par exponentiation rapide en utilisant

$$a \equiv bc \bmod N \equiv (b \bmod N) \cdot (c \bmod N) \bmod N$$

Donc :

$$\begin{aligned} y' &\equiv 13^7 \bmod 33 \equiv 13^{(3+3+1)} \bmod 33 \equiv 13^3 \cdot 13^3 \cdot 13 \bmod 33 \\ &\equiv (13^3 \bmod 33) \cdot (13^3 \bmod 33) \cdot (13 \bmod 33) \bmod 33 \\ &\equiv (2197 \bmod 33) \cdot (2197 \bmod 33) \cdot (13 \bmod 33) \bmod 33 \\ &\equiv 19 \cdot 19 \cdot 13 \bmod 33 \equiv 4693 \bmod 33 \\ &\equiv 7. \end{aligned}$$

Le texte chiffré y pour toutes les valeurs possibles de x (de 0 à 32) est :

message x	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
chiffré y	0	1	8	27	31	26	18	13	17	3	10	11	12	19	5	9	4

message x	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
chiffré y	29	24	28	14	21	22	23	30	16	20	15	7	2	6	25	32

Dans ce cas simple toutes les 33 valeurs de x (de 0 à 32) sont codées par un code unique y dans le même domaine, d'une manière aléatoire. Nous avons neuf valeurs de x codées par la même valeur de x – ce sont des messages non-cachés. $x = 0$ et $x = N - 1$ vont toujours être ainsi, quelle que soit la valeur de N. En pratique néanmoins, des valeurs plus grandes ne devraient pas poser problème lorsqu'on utilise de grandes valeurs de N, de l'ordre de plusieurs centaines de bits.

3.4 Implementation de RSA

Un bon algorithme de chiffrement mal implementé n'est pas plus sûr que celui obtenu avec un simple chiffre de substitution.

Exemple (**Mauvais exemple** de codage RSA par substitution)**.** *Supposons qu'on utilise RSA pour des messages texte en codant A= 2, B= 3, ..., Z= 27. On évite* 0 *et* 1 *parce que ils sont non-cachés.*

Alors le message en clair "`HELLOWORLD`" serait représenté par la suite d'entiers $(x_1, x_2, \dots)$

$$(9, 6, 13, 13, 16, 24, 16, 19, 13, 5)$$

Utilisant la table ci-avant, on obtient les entiers $(c_1, c_2, \dots)$ du texte chiffré

$$(3, 18, 19, 19, 4, 30, 4, 28, 19, 26)$$

Pour une substitution monoalphabethique un attaque par analyse fréquentielle sur les répétitions des lettres du message crypté est un success.

Exemple (Exemple de codage RSA par blocs)**.** *Soit le message à crypter* `HELLOWORLD`*. Le message à chiffrer est divisé en blocs à* 3 *bits et chaque bloc est chiffré séparément. On peut représenter nos blocs de trois caractères dans la base 26 utilisant les valeurs A= 0, B= 1, C= 2, ..., Z= 25 :*

HEL $= code(H) \cdot 26^2 + code(E) \cdot 26^1 + code(L) = 7 \cdot 676 + 4 \cdot 26 + 11 = 4847$

La décomposition sur n'importe quelle base est unique, donc le déchiffrement peut retrouver les trois lettres du bloc, après avoir déchiffrer la valeur de HEL

Pour cet exemple simplifié, (c.à-d. sans caractères numériques, signes de ponctuation, ...) la valeur maximale d'un groupe (ZZZ) serait

$$25 \cdot 26^2 + 25 \cdot 26 + 25 = 25(26^2 + 26 + 1) = 25\frac{(26^3 - 1)}{26 - 1} = 26^3 - 1 = 17575$$

donc on a besoin d'un module RSA N plus grand que cette valeur.

1. *On génère les nombres premiers* $p = 137$ *et* $q = 131$ *(On cherche des nombres premiers autour de* $\sqrt{N}$*)*
2. $N = pq = 137 \cdot 131 = 17947$
   $\phi = (p-1)(q-1) = 136 \cdot 130 = 17680$
3. *On choisit* $e = 3$ *on vérifie* $\text{pgcd}(e, p-1) = \text{pgcd}(3, 136) = 1$*, OK et on vérifie* $\text{pgcd}(e, q-1) = \text{pgcd}(3, 130) = 1$*, OK.*
4. *On calcule* $d = e^{-1} \bmod \phi = 3^{-1} \bmod 17680 = 11787$*, c.à-d. on trouve une valeur pour* d *telle que* ϕ *divide* $(ed - 1)$*, c.à-d. on trouve* d *tel que* 17680 *divide* $3d - 1$*.*
   Les vérification itératives $(d = 1, 2, ...)$ *donnent* $d = 11787$*.*
   Vérification : $ed - 1 = 3 \cdot 11787 - 1 = 35360$*, qui est bien divisible par* ϕ*.*

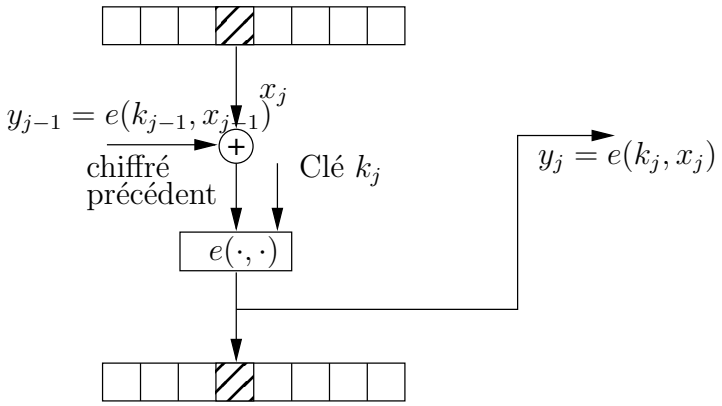

FIGURE 3.2 – Schéma CBC

5. Donc la clé publique est $(N, e) = (17947, 3)$ *et la clé privée* $(N, d) = (17947, 11787)$.

Question : Pourquoi ne pourrions-nous pas utiliser $e = 17$ *ici ? (Parce que* $17 * 1040 = \phi = 17680$*)*

Pour crypter le premier entier qui représente "HEL", on a $y \equiv x^e \bmod N \equiv 4847^3 \equiv 4847^2 \cdot 4847 \equiv 4978 \pmod{17947}$.

On peut vérifier que notre clé privé est valide en décryptant

$x' \equiv y^d mod N \equiv 4978^{11787} \pmod{17947} \equiv 4847 \pmod{17947}$.

Notre texte en clair est representé par l'ensemble d'entiers x : $(4847, 7822, 9917, 2028)$
Notre texte chiffré est representé par l'ensemble d'entiers $y \equiv x^e \bmod N$ *:*
$(4978, 279, 1465, 1278)$

Décryptez les chiffrés en utilisant $x \equiv y^d \pmod{N}$.

Ce genre de chiffrement par blocs à n bits n'as pas non plus une grande résistance à la cryptanalyse. Son grand défaut étant que deux blocs identiquesont le même code. L'attaquant cherche les séquences identiques : attaque par "dictionnaire".

Il y a plusieurs modes de chiffrement par blocs, chacun avec des différentes propriétés de propagation d'erreurs et de résistance à la cryptanalyse.

- ECB (Electronic CodeBook)
- CBC (Cipher-Block Chaining)
- CFB (Cipher Feedback)
- OFB (Output Feedback)

Le premier schema "utilisable" est le schema du CBC : avant de chiffrer le bloc courant on utilise un OU exclusif (XOR) avec une clé et le chiffrement du bloc précédent.

Pour les chiffrement par blocs implémentés en ECB, CBC le texte en clair doit être un multiple de n=taille du bloc. Si ce n'est pas le cas le padding/bourrage est utilisé. Le design final d'un chiffrement par blocs est divisé en plusieurs rondes/tours/étages pour diffusser les changements dans tout le chiffré. A chaque tour ils utilisent une clé intermédiaire obtenu d'une clé principale (algorithme de génération de clés(key schedule)) Un **blanchiment de clé(key whitening)**avant le premier tour et après le dernier tour combine les données avec des parties de la clé (ex : avec XOR) et empêche le cryptanalyste d'obtenir une paire texte en clair, texte chiffré .

Exemple (Remplissage (padding) aléatoire)**.** *Dans cet exemple si Eve connaît le but de la communication, elle peut déduire le nombre utilisant la longueur du chiffré.*

*ASCII : Offre USD*90000.00
(en hexa) : 4F6666657220243930303030 2E3030
INPUT : 4F6666657220243930303030 2E303001
OUTPUT : 33BEF550BADE4798DDA5C960E2C70EB9
*ASCII : Offre USD*1000000.00
(en hexa) : 4F 66 66 65 72 20243130303030303 02E3030
INPUT : 4F666665722024313030303030302E303007070707070707
OUTPUT : A4B8D1BF3020DB24CDD459BAB6A7BA7B02AC39EE7C1BF090

Si on rajoute un nombre aléatoire d'octets comme "padding" et on l'indique dans le dernier octet rajouté. Cette convention particulière consistant à utiliser seulement le dernieroctet nous limite à 255 octets aléatoires de remplissage L'offre est maintenant :

ASCII PT : Offer USD90000.00
(en hexa) : 4F6666657220243930303030 2E3030
INPUT : 4F6666657220243930303030 2E303012441C0D5E2C60147DF54910B6A6445311
OUTPUT : 33BEF550BADE4798B164164E571A5266B0D488FAD934D6386494FAF528C8ED82
ASCII : Offre USD1000000.00
(en hexa) : 4F666665722024313030303030302E3030
INPUT : 4F666665722024313030303030302E3030CEF8302A84BA07
OUTPUT : A4B8D1BF3020DB24CDD459BAB6A7BA7BC01DF3FCC3B7DC1B

3.5 Attaques sur RSA

Le problème RSA est un problème bien étudié. Pour résoudre le problème RSA général et il ne semble pas y avoir de méthodes autres que la factorisation.

Pour factoriser un nombre, on peut utiliser l'algorithme du crible quadratique de Dixon qui s'exécute en $O(e^{(1+o(1))\sqrt{\ln N \ln\ln N}})$, où N est le module RSA. L'algorithme utilisant les courbes elliptiques a une complexité en moyenne en $O(e^{(1+o(1))\ln N^{1/3}\ln\ln N^{2/3}})$.

Le meilleur algorithme est le crible algébrique où la complexité heuristique est en moyenne de $O(e^{(c+o(1))\ln N^{1/3}\ln\ln N^{2/3}})$, avec $c = (64/9)^{1/3} \simeq 1.923$.

Au-delà de ces valeurs asymptotiques, on peut fixer en pratique quelle est la taille des paramètres RSA. Donc, la taille recommandée est de considérer des modules $N = pq$ de 1024 bits, où p et q sont de taille 512 bits. Dans la formule asymptotique, en remplaçant $\ln N$ par $1024 \cdot \ln 2$, on obtient 287.

En pratique, on peut en considérer d'autres algorithmes. Soit $\psi = |p|$, où p est le plus petit diviseur premier de N. Si le facteur p de $N = pq$ admet une décomposition en "petits" facteurs on peut utiliser l'algorithme de Pollard(méthode $p-1$) pour factoriser N. Mais cette attaque peut être facilement evité en choisissant p, q tels que $2p + 1$,$2q + 1$ soient premiers.

Remarque. *Si $N = pq$ et $\phi(N)$ sont connus, on peut factoriser N en résolvant le système :*

$$N = pq$$

$$\phi(N) = (p-1)(q-1)$$

On substitue $q = \frac{N}{p}$ pour obtenir une équation du second degré

$$p^2 - (N - \phi(N) + 1)p + N = 0$$

Les solutions de cette equation sont p et q avec $N = pq$.

Remarque. *(Rappel sur les équations du second degré) Soit Δ le discriminant de l'équation $ax^2 + bx + c$, et x_1 et x_2 les deux solutions non-nuls de cette équation :*

$$x_1 \times x_2 = \frac{-b - \sqrt{\Delta}}{2a} \times \frac{-b + \sqrt{\Delta}}{2a} = \frac{(-b - \sqrt{\Delta}) \times (-b + \sqrt{\Delta})}{4a^2}$$

En utilisant l'identité : $(a-b) \times (a+b) = a^2 - b^2$ on obtient

$$x_1 \times x_2 = \frac{(-b)^2 - (\sqrt{\Delta})^2}{4a^2} = \frac{b^2 - \Delta}{4a^2} = \frac{b^2 - (b^2 - 4ac)}{4a^2} = \frac{b^2 - b^2 - (-4ac)}{4a^2} = \frac{4ac}{4a^2} = \frac{c}{a}$$

Donc nous avons $x^2 - (x_1 + x_2) \cdot x + (x_1 \cdot x_2)$.

Dans la suite nous allons presenter quelques attaques autres que la factorisation sur des cas RSA particuliers . Ces attaques sont completes quand elles permettent la factorisation de N. Elles sont partielles si on découvre le texte en clair d'un message chiffré.

3.5.1 RSA : Attaques avec exposant public e petit

Prendre $e = 3$ où simplement petit réduit le temps nécessaire au déchiffrement d'un message, mais permet un très grand nombre d'attaques sur RSA. Parmi les plus simples sont les attaques de Wiener, Coppersmith, Hastad.

Le théorème de Coppersmith s'applique en particulier au cas où le message clair M consiste en une partie connue $B = 2^k \cdot b$ et d'une partie inconnue x. Le chiffré est alors $C = M^e = (B+x)^e \mod n$. En utilisant le théorème avec le polynôme $p(x) = (B+x)^e - C$, on retrouve x à partir de C si $|x| < N^{1/e}$. Dans le cas où $e = 3$, on peut retrouver le message clair si on en connaît déjà les deux-tiers.

RSA : L'attaque Wiener, Boneh et Durfee

L'attaque Wiener s'applique quand la taille de d, l'inverse modulaire de l'exposant e est inférieure au quart de la taille de N. Pour un module de 1024 bits, d doit donc être de taille supérieure à 256 bits. La preuve est basée sur l'approximation des fractions continues.

Développement en fraction continue

Definition. [Fraction continue] *Un nombre rationnel x se représente de la manière suivante :*

$$x = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\cdots + \cfrac{1}{a_n}}}}} = [a_0, a_1, a_2, a_3, \cdots, a_n]$$

Les deux notations, avec des barres de fractions ou des crochets signifient la même chose. Si p est un entier inférieur à n, le terme a_p, appelé coefficient d'indice p, désigne un entier strictement positif sauf peut être a_0 qui est un entier quelconque. La fraction qui s'arrête au terme a_p est la réduite d'indice p et si $\frac{1}{x_{p+1}}$ est le complément à ajouter dans l'expression à a_p pour obtenir la valeur exacte de x, alors x_{p+1} est appelé quotient complet d'indice p + 1, ce qui se traduit par l'égalité : $x = [a_0, a_1, a_2, \cdots, a_p, x_{p+1}]$

Ce concept ne se limite pas aux rationnels. Si x est un nombre irrationnel, la suite des coefficients est infinie et celle des réduites est alternée et converge vers x.

Si $\frac{h_n}{k_n}$ désigne la réduite d'ordre n, on dispose des relations de récurrence suivantes :

$$h_{n+1} = a_{n+2}h_{n+1} + h_n, \quad k_{n+1} = a_{n+2}k_{n+1} + k_n$$

et

$$[a_0, a_1, \ldots, a_{n-1}, x_n] = \frac{x_n h_{n-1} + h_{n-2}}{x_n k_{n-1} + k_{n-2}}$$

Ce qui montre que les numérateurs et les dénominateurs des réduites forment deux suites qui tendent vers l'infini.

L'algorithme d'Euclide étendu fournit un développement en fraction continue d'un nombre rationnel. Le développement obtenu est fini.

Exemple. *Développez* 355/113 *en fraction continue.*

$$\begin{array}{rcrcr} 355 & = & 3\times & 113 & + & 16, \\ 113 & = & 7\times & 16 & + & 1, \\ 16 & = & 16\times & 1 & + & 0, \end{array}$$

La partie entière de 355/113 est 3, ce qui permet d'écrire

$$\frac{355}{113} = 3 + \frac{16}{113} = 3 + \cfrac{1}{\cfrac{113}{16}}.$$

La fraction 16/113 est plus petite que 1, mais son inverse, possède une partie entière, 7, si on utilise les résultats de la deuxième division euclidienne :

$$\frac{113}{16} = 7 + \frac{1}{16}.$$

Ainsi :

$$\frac{355}{113} = 3 + \frac{1}{7 + \frac{1}{16}}$$

qui est bien une fraction continue finie.

Les fractions continues "coupées" 3, $3 + \frac{1}{7}$, $3 + \frac{1}{7 + \frac{1}{16}}$ sont des approximations fractionnaires de plus en plus précises du nombre $\frac{355}{113}$. On appelle ces fractions continues "coupées" : des réduites.

Théorème. *Soit a et b entiers non-nuls si*

$$\left|\frac{a}{b} - x\right| < \frac{1}{2 \cdot b^2}$$

Alors $\frac{a}{b}$ est une réduite de x.

Le Théorème de Wiener

Théorème (Wiener)**.** *Soit $N = p \cdot q$ avec $q < p < 2 \cdot q$. Soit $3 \cdot d \leq N^{\frac{1}{4}}$.Étant donné (N, e) avec $ed \equiv 1 \pmod{\phi(N)}$ il existe un algorithme polynomial permettant de retrouver d.*

Preuve. *Comme d est calculé dans l'équation $e \cdot d \equiv 1 \pmod{\phi(N)}$, on en déduit qu'il existe un entier $t < d$ tel que $e \cdot d = t \cdot \phi(N) + 1$. Si $t = d$ $d \cdot (e - \phi(N)) \equiv 1 \pmod{\phi(N)}$*

Parce que $\phi(N)$ n'est pas publique, pour la cryptanalyse il nous faut écrire $\phi(N)$ en fonction de N qui est publique.

$$\phi(N) = (p-1)(q-1) = N - (p+q) + 1$$

Donc $e \cdot d = t \cdot (N + 1 - (p+q)) + 1$. En divisant par dN nous avons que :

$$\frac{e}{N} - \frac{t}{d} = \frac{t - t(p+q) + 1}{dN}$$

Par conséquent :

$$\left|\frac{e}{N} - \frac{t}{d}\right| = \frac{t(p+q) - t - 1}{dN} < \frac{tq(\frac{p}{q} + 1)}{dN}$$

Supposons que $q < p < 2q$, alors $N = p \cdot q > q^2$ et $q < \sqrt{N} < p$.

$$\left|\frac{e}{N} - \frac{t}{d}\right| < \frac{3 \cdot t \cdot q}{d \cdot N} < \frac{3 \cdot t}{d \cdot \sqrt{N}}$$

Comme $t < d$, on a que

$$\left|\frac{e}{N} - \frac{t}{d}\right| < \frac{3}{\sqrt{N}}$$

Enfin, puisque $\sqrt{6} \cdot d < 3d < N^{1/4}$*, on a que* $6 \cdot d^2 < \sqrt{N}$

$$\left| \frac{e}{N} - \frac{t}{d} \right| < \frac{1}{2 \cdot d^2}$$

Alors $\frac{t}{d}$ *est une réduite de* $\frac{e}{N}$*.*

Si $\left|\frac{e}{N} - \frac{t}{d}\right| < \frac{1}{3d^2}$ **on peut obtenir une approximation de** $\frac{t}{d}$ **et donc du** $\phi(N) = \frac{(ed-1)}{t}$.

Exemple. *Soit* $N = 160523347$ *et* $e = 60728973$*. On développe* $\frac{e}{N}$ *en fraction continue.*

$$\begin{aligned}
\frac{e}{N} &= 0 + \frac{1}{\frac{N}{e}} \\
N &= 121457946 + 39065401 = 2 \cdot 60728973 + 39065401 \\
\frac{e}{N} &= 0 + \frac{1}{2 + \frac{1}{\frac{60728973}{39065401}}} \\
60728973 &= 1 \cdot 39065401 + 21663572 \\
\frac{e}{N} &= 0 + \frac{1}{2 + \frac{1}{1 + \frac{21663572}{39065401}}} \\
39065401 &= 1 \cdot 21663572 + 17401829 \\
\frac{e}{N} &= 0 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{17401829}{21663572}}}} \\
21663572 &= 1 \cdot 17401829 + 4261743 \\
\frac{e}{N} &= 0 + \frac{1}{2 + \frac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{4261743}{17401829}}}}}
\end{aligned}$$

$$
\begin{aligned}
17401829 &= 17046972 + 354857 = 4 \cdot 4261743 + 354857 \\
\frac{e}{N} &= 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{354857}{4261743}}}}}} \\
4261743 &= 12 \cdot 354857 + 3459 \\
&\cdots\cdots
\end{aligned}
$$

Le développement en fraction continue de $\frac{e}{N}$ *est :*$[0, 2, 1, 1, 1, 4, 12, 102, 1, 1, 2, 3, 2, 2, 36]$.

$$
x = 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{12 + \cdots}}}}}}
$$

Donc $\frac{e}{N}$ *est approximé par 0, 1/2 , 1/3 , 2/5 , 3/8, 14/37,171/572 ...*

$$
\begin{aligned}
f_1 &= 0 + \cfrac{1}{2 + \cfrac{1}{1}} = \frac{1}{3} \\
f_2 &= 0 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1}}} = \cfrac{1}{2 + \cfrac{1}{2}} = \cfrac{1}{\cfrac{5}{2}} = \frac{2}{5} \\
f_3 &= 0 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{1 + \frac{1}{1}}}} = \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{2}}} = \cfrac{1}{2 + \cfrac{1}{\cfrac{3}{2}}} = \cfrac{1}{2 + \frac{2}{3}} = \cfrac{1}{\frac{8}{3}} = \frac{3}{8} \\
f_4 &= 0 + \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{1 + \frac{1}{1 + \frac{1}{4}}}}} = \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{1 + \frac{1}{\frac{5}{4}}}}} = \cfrac{1}{2 + \cfrac{1}{1 + \frac{1}{1 + \frac{4}{5}}}} = \cfrac{1}{2 + \cfrac{1}{1 + \frac{5}{9}}} = \cfrac{1}{2 + \frac{9}{14}} = \cfrac{1}{\frac{37}{14}} = \frac{14}{37}
\end{aligned}
$$

Nous allons essayer une par une ces approximations. Soit $\frac{t}{d} = \frac{1}{2}$. *On prend* $t = 1$ *et*

$d = 2$. *Alors* $\phi(N) = \frac{ed-1}{t} = 60728973 \times 2 - 1 = 121457945$. *Donc un premier essai est* $p = \frac{39065403}{2} - \frac{\sqrt{1526105069459021}}{2}$ *qui n'est pas correct.*

Le premier essai qui arrive à factoriser N *est* $\frac{t}{d} = \frac{14}{37}$. *On prend* $t = 14$, $d = 37$. *Alors* $\phi(N) = \frac{ed-1}{t} = \frac{60728973 \times 37 - 1}{14} = 160498000$ *et nous obtenons l'equation de deuxième degré*

$$0 = x^2 - S \cdot x + P = x^2 - (N - \phi(N) + 1) - N$$

$$x^2 - 25348 \cdot x + 160523347 = 0$$

qui admet comme solutions : $p = 12347$ *et* $q = 13001$.

Comment contourner l'attaque tout en réduisant le temps de déchiffrement ?

Souvent, le déchiffrement RSA est réalisé en utilisant le théorème des restes chinois . Pour calculer $x \equiv y^d \pmod N$, on calcule séparément le message x modulo p et modulo q, en calculant $x_p \equiv y^{d_p} \pmod p$ et $x_q \equiv C^{d_q} \pmod q$, avec $d_p \equiv d \pmod{p-1}$ et $d_q \equiv d \pmod{q-1}$. En utilisant le théorème des restes chinois, on obtient l'unique message $x \in \mathbb{Z}_N$ tel que $x \equiv x_p \mod p$ et $x \equiv x_q \mod q$. Une exponentiation modulaire ayant une complexité cubique en la taille du module, chacune des deux exponentiations est 8 fois plus rapide que l'exponentiation $x \equiv y^d \mod N$. Il en résulte un gain en temps de calcul d'un facteur 4.

On peut essayer de choisir d tel que $d_p \equiv d \pmod{p-1}$ et $d_q \equiv d \pmod{q-1}$ soient plus petits (de taille 128 bits), avec un exposant d de la même taille que N. Dans ce cas, les attaques de Wiener et de Boneh et Durfee ne s'appliquent plus.

RSA : L'attaque Hastad "Broadcast attack"

L'attaque Hastad s'applique au cas où un même message est chiffré avec la même clé publique et envoyés à différentes personnes.

Supposons que Bob utilise le même exposant e de chiffrement pour envoyer à k personnes différentes le mème message x. Chaque personne P_i possède sa propre clef publique (N_i, e), tels que $\text{pgcd}(N_i, N_j) = 1$ pour tout $i \neq j$. On suppose que le message x est inférieur à tous les N_i. Pour envoyer x à la i-ème personne P_i, Bob chiffre naïvement le message x avec $C_i = x^e \mod N_i$. L'attaquant Mallory peut espionner la communication et obtenir chacun des k messages chiffrés.

Soit $e = 3$ et le nombre de chiffrés $k \geq 3$. Supposons que Mallory obtienne y_1, y_2, y_3, où :
$y_1 = x^3 \pmod{N_1}$
$y_2 = x^3 \pmod{N_2}$
$y_3 = x^3 \pmod{N_3}$

En appliquant le Théorème des restes chinois à y_1, y_2, y_3, on obtient un entier y' compris entre 0 et $N_1 \cdot N_2 \cdot N_3$ tel que $y' = x^3 \pmod{N_1 \cdot N_2 \cdot N_3}$. Comme le message $x \leq N_i$, on a $x^3 < N_1 \cdot N_2 \cdot N_3$. L'égalité $y' = x^3$ est vérifie et Mallory retrouve x en calculant (dans $\mathbb{Z}$) la racine cubique de y'). L'attaque se généralise à tout exposant de chiffrement e, à condition que le nombre de chiffrés k soit supérieur ou égal à e.

Comment la prévenir ? Avant de chiffrer x, Bob ajoute une information de temps au message x : $x_i = i \cdot 2^L + x$ à la personne P_i, où L est la taille de bits du message. Dans ce cas les messages sont différents. Il y a des versions améliorées de l'attaque pour toute information déterministe rajoutée.

Exemple. *(Attaque Hastad) Soit $e = 3$ l'exposant commun de chiffrement pour envoyer à 3 personnes différentes le mème message x. Chaque personne P_i possède sa propre clef publique (N_i, e) : $N_1 = 2 \times 13 = 26$, $N_2 = 3 \times 11 = 33$, $N_3 = 5 \times 7 = 35$. On connaît $y_1 \equiv x^3 \pmod{26}$, $y_2 \equiv x^3 \pmod{33}$, $y_3 \equiv x^3 \pmod{35}$.*

Selon la preuve du Théorème des restes chinois à y_1, y_2, y_3 il existe les entiers u_1, u_2, u_3 tels que $v_1 N_1 + u_1(N_2 N_3) = 1$ et donc $u_1 N_2 N_3 \equiv 1 \pmod{N_1}$. Similairement $u_2 N_1 N_3 \equiv 1 \pmod{N_2}$, $u_3 N_1 N_3 \equiv 1 \pmod{N_3}$. On calcule les inverses modulaires :
$u_1 \equiv 1155^{-1} \pmod{26} \equiv 19 \pmod{26}$
$u_2 \equiv 910^{-1} \pmod{33} \equiv 7 \pmod{33}$
$u_3 \equiv 858^{-1} \pmod{35} \equiv 2 \pmod{35}$.
La solution du système des congruences est $y' \equiv y_1 u_1 N_2 N_3 + y_2 u_2 N_1 N_3 + y_3 u_3 N_1 N_3 \pmod{26 \cdot 33 \cdot 35}$.

RSA : L'attaque Franklin-Reiter sur les messages liés

L'attaque Franklin-Reiter s'applique au cas où deux messages liés entre eux par une relation connue, sont chiffrés avec la même clé publique et envoyés à différentes personnes. La complexité de l'attaque est quadratique en e. Elle ne s'applique donc que lorsqu'un petit exposant e est utilisé.

Definition. *Les messages x_1, x_2 sont liés si :*
- *ils vérifient une relation polynomiale connue P de la forme $x_2 = P(x_1)$, $\deg(P) = \delta$*
- *les deux chiffrés correspondant y_1 et y_2 sont connus*

Dans ce cas $z = x_1 \pmod{N}$ est une racine commune des deux équations polynomiales :

$$z^e - y_1 = 0 \pmod{N} \text{ et } (P(z))^e - y_2 = 0 \pmod{N}$$

de sorte qu'avec forte probabilité on retrouve x_1 avec :

$$pgcd(z^e - y_1, (P(z))^e - y_2) = z - m_1 \pmod{N}$$

Le cas limite est le même N pour envoyer le même message x à des utilisateurs différents. Si les clés e_1 et e_2 sont des entiers relativement premiers en utilisant le Théorème Bachet-Bézout il existe u_1, u_2 tels que $1 = u_1 e_1 + u_2 e_2$. Les chiffrés du x sont $y_1 = x^{e_1}$, $y_2 = x^{e_2}$ La cryptanalyse de x : $y_1^{u_1} + y_2^{u_2} = x^{u_1 e_1 + u_2 e_2} \equiv x \pmod{N}$

Boneh a généralisé les deux attaques Hastad et Franklin-Reiter en supposant que pour chacune des personnes $P_1, \ldots, P_k$, Bob a déterminé a lié les messages avec un polynôme constant et public $f_i \in \mathbb{Z}_{N_i}[X]$. Il a donc diffusé le chiffré de $f_i(x)$ à la personne P_i.

Eve peut obtenir les chiffrés $y_i = f_i(x)^{e_i} \pmod{N_i}$ pour $i = 1, \ldots, k$. La nouvelle attaque permet de retrouver x à partir des chiffrés y_i s'il y en a suffisamment.

Théorème. [version améliorée Boneh] *Soient $N_1, \ldots, N_k$ des entiers relativement premiers. Soit $N_{\min} = \min(N_i)$. Soient les polynômes $g_i \in \mathbb{Z}_{N_i}[x]$ de degré maximum $\delta \leq k$.*

S'il existe un unique $x_0 < N_{\min}$ tel que $g_i(x_0) \equiv 0 \pmod{N_i}$ pour chaque $i = 1, \cdots, k$, alors on peut retrouver efficacement x_0 à partir des N_i et des polynômes g_i.