Cryptanalyse du système de chiffrement asymétrique RSA par la méthode de Wiener

Ce rapport à été réalisé dans l'intention d'appuyer ma candidature au sein du Master Sécurité des Systèmes d'Information proposé par l'Université de Technologie de Troyes.

Rédigé par
Joachim TECHTACHE

4 Mars 2020

Table des matières

1 Introduction

L'évolution rapide des réseaux informatiques engendre un volume toujours plus important de données sauvegardées et transmises, générant ainsi de nouveaux besoins en matière de sécurité. Dans un monde où l'entreprise dépend de plus en plus de son système informatique, la sécurité est donc devenue une préoccupation primordiale. D'ailleurs, jamais il n'a été autant fait usage de cryptographie qu'aujourd'hui. Diplomates et militaires, autrefois principaux utilisateurs, partagent maintenant ce besoin avec les entreprises et le grand public. L'une des fonctions de la cryptographie qui est d'ailleurs historiquement la première, est d'assurer la confidentialité des données et des transactions. À cette fin, de nombreux systèmes de chiffrement on était conçu, rendant inaccessibles l'information à toute personne non autorisée.

On distingue deux grandes familles de systèmes cryptographiques. Les systèmes symétriques utilisent des clés qui sont des chaines binaires plus ou moins longues. Le plus souvent aucune contrainte n'est imposée pour ces clés, toute chaine binaire peut convenir et en choisir une peut se ramener à tirer à pile ou face autant de fois que le nombre de bits de la clé. Mais tous les algorithmes de chiffrement symétriques ont un problème commun, la transmission de clé. Quel que soit l'algorithme utilisé, si la clé est interceptée par autrui, alors il peut lire les communications, mais aussi se faire passer pour le destinataire et l'expéditeur du message. Ce problème est fondamental car transmettre une clé de chiffrement est très délicat, et même impossible dans certains cas.

Pour résoudre ce problème d'échange de clés de manière sécurisée, Whitfield Diffie et Martin Hellman ont introduit en 1976 le système asymétrique répondant ainsi au principe d'Auguste Kerckhoffs, stipulant que la sécurité d'un système de chiffrement ne doit pas reposer sur le secret de sa procédure, mais uniquement sur le secret d'un paramètre utilisé à chacune de ses mises en œuvre. Les systèmes asymétriques, ne sont jamais des chaines binaires quelconques. Au contraire elles possèdent une structure mathématique telle que la clé privée puisse déchiffrer toute information chiffrée avec la clé publique correspondante. Leur génération ne peut donc pas se limiter à une succession de tirages à pile ou face. L'espace des clés d'un système asymétrique est plus complexe que celui d'un système symétrique, car les clés possèdent une structure mathématique bien précise. Les systèmes asymétriques reposent sur la notion de fonctions à sens unique qui d'ailleurs n'a pas encore été prouvée utilisant des fonctions faciles à calculer mais calculatoirement difficiles à inverser.

Dans ce document, on s'intéressera uniquement au système de chiffrement asymétrique RSA que l'on présentera en instaurant quelques principes mathématiques sur lesquelles repose ce système. On introduira ensuite une méthode d'attaque sur un exposant de déchiffrement faible découvert par le cryptologue Michael J. Wiener. On démontrera également l'une de ces généralisations proposées par le Professeur Benne de Weger. Pour finir on implémentera une attaque de Wiener, en utilisant le langage de programmation Python.

2 Présentation du système

En 1978, les cryptologues Ronald Rivest, Adi Shamir et le chercheur en informatique théorique Leonard Adleman proposent ce premier système asymétrique qu'ils nomment RSA. Ces trois auteurs avaient décidé de travailler ensemble afin de démontrer l'impossibilité logique des systèmes cryptographiques asymétrique mais ils échouèrent en découvrant justement un système de cryptographie asymétrique.

Depuis, le chiffrement RSA est devenu le chiffrement le plus répandu dans le monde. On le trouve dans un nombre toujours croissant de produits commerciaux liés à la sécurisation des échanges de données sur Internet, à la protection de la confidentialité et de l'authenticité du courrier électronique ou au paiement électronique au moyen de cartes à puce. Car bien qu'il soit facile à réaliser, il reste très difficile à casser.

La sécurité de l'algorithme RSA repose sur deux conjectures, casser le système cryptographique RSA nécessite la factorisation de grands nombres qui est l'un des problèmes les plus difficiles en mathématiques. Par difficile, on entend qu'il n'existe pour le moment pas d'algorithmes suffisamment rapide pour résoudre cette question. D'ailleurs, si l'on souhaite être un peu plus précis, on pense qu'il n'existe aucun algorithme ayant une complexité polynomiale en temps qui donne les facteurs premiers d'un nombre quelconque. Mais il est possible que l'une des deux, voir les deux, conjectures soit fausse. Si c'est le cas, alors le système RSA n'est pas sûr.

2.1 Méthodes du système

L'algorithme va dans un premier temps générer deux couples de clés asymétriques, l'une pour l'émetteur qu'on appelle Alice, et l'autre pour le destinataire qu'on appelle Bob. Une fois que chaque personne possède ses deux clés, on peut alors procéder à une communication sécurisée. Bob partage sa clé publique avec Alice avec laquelle elle chiffre son message. Ensuite le message chiffré est transmis à Bob, il procède au déchiffrement grâce à sa clé privée qu'il n'a communiquée à personne. Aucun échange de clés sensibles n'est nécessaire, et seule la clé privée de Bob peut déchiffrer le message, la communication est alors sécurisée. La sécurité de l'algorithme se trouve dans l'utilisation d'une fonction de chiffrement et de déchiffrement à sens unique. Cette fonction est comme pour la génération de clés, très simple à appliquer dans un sens, mais extrêmement complexe dans l'autre.

Pour générer ces clés, Bob commence par générés deux nombres premiers p et q au hasard, en utilisant un algorithme de test de primalité probabiliste. Il calcule le module de chiffrement N par le produit de ces deux nombres et calcule l'indicatrice d'Euler *(Théorème 2)*. Il choisit ensuite un exposant de chiffrement noté e tel que $pgcd(e, \varphi(N) = 1$ et pour finir il calcule en utilisant l'algorithme d'Euclide, l'inverse modulaire de $e \ mod \ \varphi(N)$: $ed \equiv 1(mod \ \varphi(N))$. Ce nombre d est appelé exposant de déchiffrement tel que $d < \varphi(N)$. Alice souhaite envoyer un message à Bob, elle commence tout d'abord par transformer un message M en un entier positif de façon à ce que $M < N$. Le message chiffré C est ensuite calculé avec par $C \equiv M^e(mod \ N)$. Bob déchiffre ce message C à l'aide de sa clé privée (N, d), il procède simplement au calcul de $M \equiv C^d(mod \ N)$. Un message chiffré C peut être signé numériquement en appliquant l'opération de chiffrement $S \equiv C^d(mod \ N)$ et la signature numérique S peut alors être vérifiée en appliquant l'opération de chiffrement $C \equiv S^e(mod \ N)$.

2.2 Exponentiation modulaire

La fonction à sens unique utilisée par le système RSA s'appelle l'exponentiation modulaire sur des grands nombres, c'est-à-dire, le calcul de $x^e \ (mod\ N)$ pour de grands nombres x, e, N. La méthode évidente pour calculer ceci est de calculer d'abord $t = x^e$ et ensuite $t(mod\ N)$. Mais le nombre x^e est trop grand pour être stocké. Ce nombre, lorsqu'il est écrit en chaine binaire occupe environ $1024 \cdot 2^{1024}$ bits ce qui en fait un nombre bien plus grand que l'estimation du nombre total d'atomes de l'univers. La boucle itérative simple pour le calcul de x^e a besoin de e multiplications ou bien environ 2^{1024} en tout. Ce calcul prendrait plus de temps que l'âge actuel de l'univers estimé à environ 15 milliards d'années.

Nous avons donc besoin d'une astuce pour éliminer le besoin de stocker x^e. Puisque le produit de deux nombres de longueurs k est seulement de longueur $2k$ avant la réduction $mod\ N$ alors 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 modulaire est basé sur l'opération d'élever au carré. Pour calculer $x^e \ (mod\ N)$ avec $e = 2q$, on calcule $x_k = (x_{k-1} \cdot x_{k-1}) \bmod N$. 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 alors 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.

2.3 Test de primalité

On recherche p et q premiers et aussi l'exposant e qui est relativement premier avec $(p-1)$ et $(q-1)$. Si p et q ne sont pas premiers on risque de pouvoir factoriser N facilement et de trouver la clé privée. Il existe deux philosophie pour rechercher des nombres premiers, les tests de primalité probabilistes et les algorithmes de primalité déterministes. Le critère d'évaluation est celui de leur complexité. Le crible d'Eratosthène est l'algorithme le plus simple qui cherche les premiers p tel que $p < \sqrt{(N)}$. Pour un module N à 256 bits $N = pq$ est plus grand que 10^{75}. Donc au crible il faut au moins $0.36 \cdot 10^{36}$ divisions pour décider. Ce qui est beaucoup trop. En pratique, on utilise des tests de primalité, le premier test était basé sur le *(Théorème 5)* qui en essayant pour beaucoup d'entiers test a si $a^p = a\ mod\ p$ alors 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 tels que pour tout a avec $pgcd(a, p) = 1$ et $a^{p-1} = 1\ mod\ p$. Ils s'appellent pseudo-premiers pour le Test de Fermat et la base a.

Mais les implémentations utilisant le test de Fermat ont été la source de nombreuses attaques c'est pourquoi il est préférable d'utiliser des tests de primalité amélioré. Miller et Rabin ont modifié le test de Fermat en s'appuyant sur deux observations, si le nombre premier p divise un produit $u \cdot v$ alors p/u ou p/v. Pour un entier n impaire, on écris $n - 1 = 2^k \cdot d$ avec d impaire. Le petit théorème de Fermat dis que si n est premier $a^{n-1} \equiv 1 \ (mod\ n)$ alors on peut dire que $n/\left(a^{n-1} - 1\right)$. En décomposant $(a^{n-1} - 1)$ 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 $k > 1$.

Dans $x^2 - 1 = (x-1)(x+1)$, si n est premier alors n doit diviser au moins un de ces deux facteurs. Si $k = 1$ on ne décompose plus, sinon on continue et à chaque nouvelle décomposition on vérifie que n divise au moins un des facteurs. S'il divise alors il est peut etre premier, sinon on dit que la base a est témoin du fait que n est composé. Le test de Miller-Rabin n'a pas de nombres pseudo-premiers absolus, donc pour tout nombre composé il y à un témoin. Et ces témoins sont très fréquents. Un théorème nous assure que 75% des nombres de $\mathbb{Z}_n$ sont des témoins pour n.

2.4 Propriétés du système

Théorème 1. *(Théorème de RSA) Soit p et q deux nombres premiers, on pose alors $N = pq$. Si le nombre e, tel que $1 < e < \varphi(N)$, est premier avec le produit $\varphi(N)$ alors il existe d unique tel que $1 < d < \varphi(N))$ et vérifiant*

$$ed \equiv 1(mod\ \varphi(N))$$

Démonstration. Si les nombres e et $\varphi(N)$ sont premiers entre eux alors il existe d'après le *(Théorème 4)* deux entiers relatifs u_0 et v_0 tels que $u_0 e + v_0\varphi(N) = 1$. Par suite (u, v) est solution de $ue + v\varphi(N) = 1$ si et seulement si il existe un entier k tel que

$$u = u_0 - k\varphi(N) \text{ et } v = v_0 + ke$$

Soit donc k tel que u soit le plus petit des entiers positifs. Dans ces conditions $ue = 1 - v\varphi(N)$ est congru à 1 modulo $\varphi(N)$ et le nombre d recherché est par conséquent égal à u. d est unique car s'il en existait un autre d' alors on aurait $e(d - d') \equiv 0\varphi(N)$

Comme e est premier avec $\varphi(N)$ alors d'après le *(Théorème 3)*, $d - d' \equiv 0\varphi(N)$. Mais comme nous avons $1 < d < \varphi(N)$ et $1 < d' < \varphi(N)$ et bien on ne peut avoir que $d = d'$. Dans les conditions précédentes, si p et q sont différents et si $b \equiv a^e(mod\ N)$ alors $b^d \equiv a(mod\ N)$.Si $b \equiv a^e(mod\ N)$ alors

$$b^d \equiv a^{ed}(mod\ N),\ ed \equiv 1(mod\ \varphi(n))$$

Il existe donc un entier $k \geq 0$ tel que $ed \equiv 1 + k(mod\ \varphi(n))$ On obtient donc

$$a^{ed} = a\left((a^{p-1})^{q-1}\right)^k$$

Si a est divisible par p alors de façon évidente $a^{ed} \equiv a \equiv 0(mod\ p)$, sinon d'après le *(Théorème 5)*, $a^{p-1} \equiv 1(mod\ p)$, d'où $a^{ed} \equiv a(mod\ p)$. De même $a^{ed} \equiv a(mod\ q)$. Il existe donc deux entiers k et k' tels que $a^{ed} = a + kp$ et $a^{ed} = a + k'q$. Ainsi $kp = k'q$, entier qui se trouve donc être multiple de pq puisque p et q sont des nombres premiers différents. On obtient donc

$$a^{ed} = a(mod\ N)$$

Théorème 2. *(Fonction d'Euler) On appelle la fonction d'Euler ou indicatrice d'Euler, la fonction qui pour un entier n associe le nombre d'entiers à n vérifiant* $1 \leq a \leq n$. *Soient p et q deux nombres distincts alors :*

$$\varphi(pq) = (p-1)(q-1)$$

Démonstration. $\varphi(pq)$ est par définition le nombre d'entiers $a : 1 \leq a \leq pq$ premier à pq. Ces entiers a sont les multiples de p et q. Il y p multiples de q dans l'ensemble $\{1, ..., pq\}$ et également q multiple de p. Dans ce dénombrement, on compte deux fois le terme pq

$$\begin{aligned}\varphi(pq) &= pq - (p+q-1) \\ &= (p-1)(q-1)\end{aligned}$$

Théorème 3. *(Théorème de Gauss) Soient a, b et c, trois entiers. Si a est premier avec b et c alors il est premier avec le produit bc.*

$$\forall (a,b,c) \in \mathbb{Z}^3 \ (a|bc \text{ et } a \wedge b = 1) \Rightarrow a|c$$

Théorème 4. *(Théorème de Bézout) Soient a et b deux entiers premiers entre eux si, et seulement s'il existe deux entiers u et v tels que au + bv = 1*

$$a \wedge b = 1 \Leftrightarrow \exists (u,v) \in \mathbb{Z}^2 \ \ au + bv = 1$$

Théorème 5. *(Petit théorème de Fermat) Soit p un nombre premier et a un entier naturel premier avec p alors*

$$a^{p-1} \equiv 1 \ (mod \ p)$$

Démonstration. Supposons dans un premier temps que p divise l'un des termes de la suite $a, 2a, ..., (p-1)a$, disons le terme ka. Comme a et p sont premiers entre eux, par le théorème de Gauss p divise k. Ceci est absurde puisque $1 < k < p$. Donc p ne divise aucun nombre de la suite $a, 2a, ..., (p-1)a$.

On note que les restes de la division de $a, 2a, ..., (p-1)a$ par p sont tous différents. Supposons qu'on ai un reste identique pour ka et $k'a$ avec $k > k'$. Ceci impliquerait que p divise $(k-k')a$ et c'est impossible par le point précédent. Donc à l'ordre des facteurs près, le reste des divisions de $a, 2a, ..., (p-1)a$ par p est $1, 2, ..., p-1$

$$\begin{aligned}&\Rightarrow a \cdot 2a \cdot ... \cdot (p-1)a \equiv 1 \cdot 2 \cdot ... \cdot (p-1) \ (mod \ p) \\ &\Rightarrow a^{p-1} \equiv 1 \ (mod \ p)\end{aligned}$$

3 Approximation diophantiennes

En théorie des nombres, l'approximation diophantienne traite de l'approximation des nombres réels par des nombres rationnels. Il est possible d'approcher tout nombre réel par un rationnel avec une précision arbitrairement grande cette propriété s'appelle la densité de l'ensemble des rationnels dans l'ensemble des réels, muni de la distance usuelle. La valeur absolue de la différence entre le nombre réel à approcher et le nombre rationnel qui l'approche fournit une mesure brute de la précision de l'approximation. La notion de fraction continue est vaste et se retrouve dans de nombreuses branches des mathématiques. Les concepts associés peuvent être relativement simples comme l'algorithme d'Euclide, ou beaucoup plus subtils comme celui de fonction méromorphe. Celles-ci ont été étudiées par les plus grands mathématiciens de toute l'Histoire. On pourra citer notamment Lagrange, Galois, Fermat, Euler, Liouville et bien d'autres encore.

Aussi, la théorie des fractions continues est déjà bien avancée et ses applications ne manquent pas. Afin de présenter correctement la méthode de Wiener, il est primordial de définir quelques propriétés de ces fractions continues qu'utilise cette méthode. En général, on fait appel aux fractions continues car celles-ci sont principalement reconnues comme étant les meilleures approximations rationnelles de réels, notamment parce qu'elles fournissent en un certain sens, les meilleures approximations des réels par des rationnels. Cette propriété est à l'origine d'algorithmes pour l'approximation de racines carrées mais aussi de démonstrations d'irrationalité voire de transcendance. On considère le nombre $x = 3 + \sqrt{10} \approx 6,162277\cdots$ qui est une solution de l'équation $x^2 - 6x - 1$ et donc $x^2 = 6x + 1$. En recommençant le processus, on obtient

$$x = 6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{\cdots}}}}$$

Une telle fraction qui peut être finie s'appelle une fraction continue. Ici, c'est la fraction continue $x = 3 + \sqrt{10}$. Pour une notation plus naturelle, on peut écrire $3 + \sqrt{10} = [6, 6, 6, ...]$. On peut aussi prendre un nombre fini d'éléments 6 pour obtenir

$$[6\ , 6] = 6 + \frac{1}{6} = \frac{37}{6} \approx 6,166666... \qquad [6\ , 6, 6] = 6 + \cfrac{1}{6 + \cfrac{1}{6}} = \frac{228}{37} \approx 6,162162...$$

$$[6\ , 6, 6, 6] = 6 + \cfrac{1}{6 + \cfrac{1}{6 + \cfrac{1}{6}}} = \frac{1405}{37} \approx 6,162280...$$

Chacune des fractions $\frac{37}{6}$, $\frac{228}{37}$, $\frac{1405}{228}$ $\cdots$ s'appelle une réduite ou alors convergentes. On observe très simplement que ces fractions vérifient

$$|6 - x| > \left|\frac{37}{6} - x\right| > \left|\frac{228}{37} - x\right| > \left|\frac{1405}{228} - x\right|$$

3.1 Développement en fraction continue

Théorème 6. *Tout nombre réel positif x possède une unique écriture en fraction continue ou les a_i sont des entiers positifs. De plus, la fraction continue est finie si et si seulement le nombre x est rationnel.*

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

Démonstration. Soit x un nombre réel positif. On pose $x_0 = x$ et on considère la partie entière $a_0 = [x_0]$. Si x_0 est un entier alors $a_0 = x_0$ et la fraction continue de x est $x = [a_0]$. Supposons maintenant que x n'est pas un entier. Dans ce cas, on a $0 < x_0 - a_0 < 1$ et $\frac{1}{x_0 - a_0} > 1$. Puisque $x_0 = a_0 + (x_0 - a_0)$, on définie x_1 par

$$x_1 = \frac{1}{x_0 - a_0} > 1 \qquad\qquad x_0 = a_0 + \frac{1}{x_1}$$

On recommence le processus pour x_1, on a $a_1 = [x_1]$ et si x_1 est un nombre entier, alors $a_1 = x_1$. La fraction continue de x représentant un nombre rationnelle est

$$x = a_0 + \frac{1}{a_1} = \frac{a_0 + a_1}{a_1}$$

Si x_1 n'est pas un entier, on recommence le processus en définissant x_2 par

$$x_2 = \frac{1}{x_1 - a_1} > 1 \qquad\qquad x_0 = a_0 + \cfrac{1}{a_1 + \cfrac{1}{x_2}}$$

Ce processus s'arrête au rang n si et seulement si $a_n = [x_x]$ avec $a_n \geq 1$ et dans ce cas alors, la fraction continue de x représentant un nombre rationnelle est

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

Si x est un nombre irrationnelle, le processus continue indéfiniment et on obtient pour x une fraction continue infinie.

$$x = [a_0, a_1, a_3, \cdots]$$

3.2 Propriétés fondamentales des réduites

Définition 6.1. *Soit $[a_1, a_2, a_3, \cdots]$ une fraction continue d'un nombre x. Les nombres entiers a_i s'appellent les quotients partiels et les nombres rationnels $[a_1, a_2, a_3, \cdots, a_k]$ s'appellent les réduites de x. On peut facilement calculer les convergentes.*

$$[a_0] = a_0 = \frac{p_0}{q_0} \qquad [a_0,\ a_1] = a_0 + \frac{1}{a_1} = \frac{a_0a_1 + 1}{a_1} = \frac{p_1}{q_1}$$

$$[a_0,\ a_1,\ a_2] = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2}} = \frac{a_0a_1a_2 + a_0 + a_2}{a_1a_2 + 1} = \frac{p_2}{q_2}$$

On peux aisément observer l'une des principales propriété des convergentes d'un nombre réel qui est $p_2 = a_2(a_0a_1 + 1) + a_0 = a_2p_1 + p_0$ et $q_2 = a_2a_1 + 1 = a_2p_1 + q_0$

Théorème 7. *Soit $[a_1, a_2, a_3, \cdots]$ la fraction continue d'un nombre x. Alors pour tout $n \geq 0$, on définie les nombres p_n et q_n par $p_n = a_np_{n-1} + p_{n-2}$ et également par $q_n = a_nq_{n-1} + q_{n-2}$ avec la convention $p_{-1} = 1$, $q_{-1} = 0$, $p_{-2} = 0$ et $q_{-2} = 1$. On a donc*

$$\frac{p_n}{q_n} = [a_0, a_1, a_2, \cdots, a_n]$$

Démonstration. La preuve s'effectue par récurrence sur n. On Suppose alors que $p_n = a_np_{n-1} + p_{n-2}$ et $q_n = a_nq_{n-1} + q_{n-2}$.

$$[a_0,\ \cdots,\ a_n,\ a_{n+1}] = \left[a_0,\ \cdots,\ a_n + \tfrac{1}{a_{n+1}}\right]$$

$$= \frac{\left(a_n + \frac{1}{a_{n+1}}\right)p_{n-1} + p_{n-2}}{\left(a_n + \frac{1}{a_{n+1}}\right)q_{n-1} + q_{n-2}}$$

$$= \frac{a_{n+1}(a_np_{n-1} + p_{n-2}) + p_{n-1}}{a_{n+1}(a_nq_{n-1} + q_{n-2}) + q_{n-1}}$$

$$= \frac{a_{n+1}p_n + p_{n-1}}{a_{n+1}q_n + q_{n-1}}$$

$$= \frac{p_{n+1}}{q_{n+1}}$$

Théorème 8. *Soit* $[a_1, a_2, a_3, \cdots]$ *la fraction continue d'un nombre* x*. Alors les convergentes* $\frac{p_n}{q_n}$ *de* x *vérifient pour tout* $n \geq -2$*.*

$$p_n q_{n+1} - q_n p_{n+1} = (-1)^{n+1}$$
$$p_n q_{n+2} - q_n p_{n+2} = (-1)^{n+1} a_{n+2}$$

Démonstration. Considérons la première égalité. Pour $n = -2$ on a bien

$$p_{-2} q_{-1} - q_{-2} p_{-1} = -1 = (-1)^{-2+1}$$

On peut démontrer à l'aide d'une preuve par récurrence que $p_{n-1} q_n - q_{n-1} p_n = (-1)^n$ en utilisant le *(Théorème 7)*.

$$\begin{aligned} p_n q_{n+1} - q_n p_{n+1} &= p_n(a_{n+1} q_n + q_{n-1}) - q_n(a_{n+1} p_n + p_{n-1}) \\ &= p_n q_{n-1} - q_n p_{n-1} \\ &= -(-1)^n \\ &= (-1)^{n+1} \end{aligned}$$

Pour la preuve de la deuxième égalité, celle-ci s'obtient en combinant l'égalité $p_n q_{n+1} - q_n p_{n+1} = (-1)^{n+1}$ et par le *(Théorème 7)*.

$$\begin{aligned} p_n q_{n+2} - q_n p_{n+2} &= p_n(a_{n+2} q_{n+1} + q_n) - q_n(a_{n+2} p_{n+1} + p_n) \\ &= a_{n+2}(p_n q_{n+1} - q_n p_{n+1}) \\ &= a_{n+2}(-1)^{n+1} \end{aligned}$$

3.3 Approximation par les réduites

Théorème 9. *Si* $[a_1, a_2, a_3, \cdots]$ *la fraction continue d'un nombre irrationnel du nombre* x*. Soit* $\frac{p_n}{q_n}$ *une convergente de* x *avec* $n \geq 0$ *et* $\frac{p}{q}$ *un nombre rationnel.*

$$Si\ q < q_{n+1},\ alors\ \left|q_n x - p_n\right| \leq \left|qx - p\right|$$

$$Si\ q \leq q_n,\ alors \left|x - \tfrac{p_n}{q_n}\right| \leq \left|x - \tfrac{p}{q}\right|$$

Démonstration. On suppose que $0 < q < q_{n+1}$. Montrons que $|q_n x - p_n| \leq |qx - p|$. Soit a et b deux entiers tels que

$$\begin{aligned} p &= ap_n + bp_{n+1} \\ q &= aq_n + bq_{n+1} \end{aligned}$$

L'existence de a et b est garantie par le *(Théorème 8)* car $p_n q_{n+1} - p_{n+1} q_n = (-)^{n+1}$. Et donc

$$\begin{aligned} a &= (-1) =^{n+1} (pq_n - qp_{n+1}) \\ a &= (-1) =^{n+1} (qp_n - pq_{n+1}) \end{aligned}$$

On peut discuter les valeurs de p et q suivant les valeurs de a et b. Si $a = 0$, alors $q = bq_{n+1}$ donc $b \geq 1$ ce qui contredit $0 < q < q_{n+1}$. Si $b = 0$, alors $p = ap_{n+1}$ et $q = aq_n$ ce qui donne

$$\left|x - \frac{p_n}{q_n}\right| = \left|x - \frac{p}{q}\right|$$

Supposons donc que $ab \neq 0$. Puisque $q < q_{n+1}$ alors $aq_n + bq_{n+1} < q_{n+1}$ et $aq_n < (1-b)q_{n+1}$. Ce qui montre que $b \geq 1$ et $a < 0$ ou $b \leq -1$. Dans ce cas, on doit avoir $q = aq_n + bq_{n+1} > 0$ donc $a > 0$. Dans les deux cas, $ab < 0$. On a alors

$$|qx - p| = |a(q_n x - p_n)| + |b(q_{n+1} x - p_{n+1})| \geq |q_n x - p_n|$$

Qui termine la preuve si $q < q_{n+1}$ alors $|q_n x - p_n| \leq |qx - p|$ et nous pouvons également conclure que

$$\left|x - \frac{p}{q}\right| = \frac{|qx - p|}{q} \geq \frac{|q_n x - p_n|}{q_n} = \left|x - \frac{p_n}{q_n}\right|$$

Termine la preuve dans le cas ou, si $q \leq q_n$ alors

$$\left|x - \frac{p_n}{q_n}\right| \leq \left|x - \frac{p}{q}\right|$$

Théorème 10. *Si x est un nombre réel. Si $\frac{p}{q}$ est un rationnel qui vérifie l'hypothèse ci dessous, alors $\frac{p}{q}$ est une convergente de x.*

$$x - \frac{p}{q} < \frac{1}{2q^2}$$

Démonstration. Soit $\frac{p}{q}$ est un nombre rationnel. Puisque la suite des dénominateur (q_n) des convergentes $\frac{p_n}{q_n}$ de x est strictement croissantes, alors $q_n \leq q < q_{n+1}$ pour un entier $n \geq 0$. On a alors, en utilisant le *(Théorème 9)* et l'hypothèse ci-dessus.

$$\left|\frac{p}{q} - \frac{p_n}{q_n}\right| \leq \left|\frac{p}{q} - x\right| + \left|x - \frac{p_n}{q_n}\right| \leq 2\left|x - \frac{p}{q}\right| < \frac{1}{q^2}$$

Il en résulte alors que

$$\left|pq_n - p_n q\right| < \frac{q_n}{q} \leq 1$$

Ainsi $pq_n - p_n q = 0$ et donc

$$\frac{p}{q} = \frac{p_n}{q_n}$$

4 Cryptanalyse élémentaire

La communauté cryptographique mondiale étudie depuis 25 ans, la solidité de ce système face à toute une panoplie d'attaques. Ces attaques se divisent en trois catégories, les attaques qui cherchent à trouver une faille dans les fondements mathématiques mêmes de l'algorithme, les attaques sur les protocoles construits autour de RSA, enfin, les attaques sur les implémentations du protocole. Toutes ces attaques ont montré que les fondements du système RSA sont solides, on peut donc raisonnablement le considérer comme un algorithme sûr mais que certaines précautions sont nécessaires, notamment sur la taille des clés à utiliser lors de l'utilisation de l'algorithme.

Cependant, dans certaines conditions il est beaucoup plus intéressant d'utiliser un exposant de déchiffrement secret faible afin de pouvoir accélérer le déchiffrement d'un message chiffré dans le cas où il existe une grande différence de puissance de calcul entre deux appareils qui souhaite communiquer. Ce procédé permet ainsi de réduire le traitement requis pour le déchiffrement pour les appareils avec peu de puissance de calcul.

En 1990 le théoricien et chercheur en mathématiques appliquées Michael J. Wiener publie *Cryptanalysis of Short RSA Secret Exponents* et présente alors une attaque cryptanalytique dans le cas d'une utilisation sur un système cryptographique RSA avec un exposant de déchiffrement faible. Cette méthode utilise un algorithme basé sur les fractions continues qui détermine un numérateur et un dénominateur d'une fraction en temps polynomial lorsqu'une estimation suffisamment proche de la fraction est connue. L'exposant public de chiffrement et le module de chiffrement peuvent être utilisés pour créer une estimation d'une fraction qui implique l'exposant secret de déchiffrement. L'algorithme basé sur des fractions continues utilise cette estimation pour découvrir des exposants secrets suffisamment courts dans un cas typique où l'exposant de chiffrement est plus faible que le module de chiffrement.

En 2002, Benne de Weger propose une généralisation de l'attaque de Wiener. Celle-ci permet de démontrer que le choix d'un module avec une petite différence de ses facteurs premiers donnent des améliorations sur les attaques telles que la méthode de Wiener. Bien qu'il existe un certain nombre de généralisations de l'attaque de Wiener, elles ne sont pas dangereuses pour les implémentations qui assurent la sécurité dans le monde numérique. Celles-ci sont néanmoins perspicaces et permettent d'illustrer principalement les pièges à éviter lors de la mise en œuvre de ce système.

4.1 Attaque de Wiener

Théorème 11. *(Théorème de Wiener) Soit $N = pq$ un module RSA. Soit $e < \varphi(N)$ un exposant public pour lequel la clé secrète d est suffisamment faible $d < \frac{1}{3}N^{\frac{1}{4}}$. Connaissant (e, N), on peut alors calculer d et factoriser N.*

Démonstration. Considérons d'abord l'exposant public e et l'exposant privé d d'une instance RSA. Il existe alors un entier k donnant l'équation $ed - k\varphi(N) = 1$. Celle-ci peut être réécrire de cette manière.

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{d\varphi(N)}$$

Par conséquent, $\frac{k}{d}$ est une approximation de $\frac{e}{\varphi(N)}$. Bien que l'attaquant ne connaisse pas la valeur de $\varphi(N)$, il peut intuitivement utiliser N pour l'approximer. En effet, puisque $\varphi(N) = N - p - q + 1$, nous avons

$$\begin{aligned} p + q - 1 &< 3\sqrt{N} \\ N + 1 - \varphi(N) - 1 &< 3\sqrt{N} \end{aligned}$$

En utilisant N à la place de $\varphi(N)$, on obtient alors

$$\begin{aligned} \left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| &= \left|\frac{ed - kN}{Nd}\right| \\ &= \left|\frac{ed - k\varphi(N) - kN + k\varphi(N)}{Nd}\right| \\ &\leq \left|\frac{3k\sqrt{N}}{Nd}\right| \\ &= \frac{3k\sqrt{N}}{\sqrt{N}\sqrt{N}d} \\ &= \frac{3k}{\sqrt{N}} \end{aligned}$$

On sait désormais que $k\varphi(N) = ed-1$ et $e < \varphi(N)$ donc $k\varphi(N) < ed$. Par conséquent $k\varphi(N) < \varphi(N)d$, puisque $k < d$ et $d < \frac{1}{3}N^{\frac{1}{4}}$. On obtient alors

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| = \frac{1}{dN^{\frac{1}{4}}}$$

Si $d < \frac{1}{3}N^{\frac{1}{4}}$ et $2d < 3d$ alors $2d < N^{\frac{1}{4}}$. Donc

$$\frac{1}{2d} > \frac{1}{dN^{\frac{1}{4}}}$$

$$\left|\frac{e}{\varphi(N)} - \frac{k}{d}\right| \leq \frac{3k}{d\sqrt{N}}$$

$$\leq \frac{1}{d \cdot 2d}$$

$$\leq \frac{1}{2d^2}$$

Il s'agit d'une relation d'approximation classique. Le nombre de fractions $\frac{k}{d}$ avec $d < N$ approchant étroitement $\frac{e}{N}$ est délimité par $log_2 N$. Dans les faits, toutes ces fractions sont obtenues en tant que convergentes de la fraction continue et en utilisant le *(Théorème 10)*, on peut en conclure que l'une des convergentes $\frac{k}{d}$ est une réduite de $\frac{e}{N}$. Ainsi il est possible de calculer p et q et donc de factoriser N par la *(Proposition 11.1)* car connaissant désormais k et d, on peut calculer $\varphi(n)$.

Proposition 11.1. *(Factorisation de N) Soit N un module RSA. Si on connait $\varphi(n)$ alors on peut factoriser N.*

Démonstration. Supposons que $\varphi(n)$ est connu. Ainsi, on dispose d'un système a deux équations en p et q.

$$\begin{cases} pq & = N \\ p+q & = N+1-\varphi(N) \end{cases}$$

Qui donne l'équation

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

On obtiens ainsi

$$p = \frac{N+1-\varphi(N) + \sqrt{(N+1-\varphi(N))^2 - 4N}}{2}$$

$$q = \frac{N+1-\varphi(N) + \sqrt{(N+1-\varphi(N))^2 - 4N}}{2}$$

Dans le cas d'un module avec $q < p < 2q$, nous présentons ci-dessous une attaque de Wiener fonctionnant sur un module de déchiffrement tel que $d < \frac{\sqrt{6\sqrt{2}}}{6} N^{\frac{1}{4}} + 0.15$.

Lemme 12. *Soit N =pq, un module RSA avec $q < p < 2q$. Alors :*

$$\frac{\sqrt{2}}{2}\sqrt{N} < q < \sqrt{N} < p < \sqrt{2}\sqrt{N} \quad \text{et} \quad 2\sqrt{N} < p + q < \frac{3\sqrt{2}}{2}\sqrt{N}$$

Démonstration. Afin de prouver la première assertion, supposons $q < p < 2q$. En multipliant avec q, on obtient $q^2 < N < 2q^2$. Par conséquent $\frac{\sqrt{2}}{2}\sqrt{N} < q < \sqrt{N}$. En utilisant $p = \frac{N}{q}$ on obtient

$$\sqrt{N} < p < \sqrt{2}\sqrt{N}$$

Pour prouver la deuxième assertion, on observe que $(p+q)^2 = (p-q)^2 + 4N > 4N$, donnant $p + q > 2\sqrt{N}$. D'autre part, nous avons

$$(p+q)^2 = (p-q)^2 + 4N < \left(\sqrt{2}\sqrt{N} - \frac{\sqrt{2}}{2}\sqrt{N}\right)^2 + 4N = \frac{9}{2}N$$

Par conséquent, On peut terminer la preuve avec $p + q < \frac{3\sqrt{2}}{2}\sqrt{N}$

Théorème 13. *(Théorème de Wiener) Soit $n = pq$ un module RSA avec $q < p < 2q$. Soit $e < \varphi(n)$ un exposant public pour lequel la clé secrète d est suffisamment faible $d < \frac{\sqrt{6\sqrt{2}}}{6} N^{\frac{1}{4}} + 0.15$. Connaissant N et e, on peut alors calculer d et factoriser N.*

Démonstration. Nous pouvons réécrire l'équation $ed - k(N + 1 - p - q) = 1$ par $ed - kN = 1 - k(p + q - 1)$. En divisant par Nd. On obtient alors

$$\frac{e}{N} - \frac{k}{d} = \frac{|1 - k(p+q-1|}{Nd} < \frac{k(p+q-1)}{Nd} \tag{1}$$

Puisque $e < \varphi(N)$, alors $k = \frac{ed-1}{\varphi(N)} < \frac{ed}{\varphi(N)} < d$. Par conséquent, l'équation (1) nous donne

$$\frac{e}{N} - \frac{k}{d} < \frac{k(p+q-1)}{N} < \frac{p+q}{N}$$

En utilisant le *(Lemme 12)*, on obtient

$$\frac{e}{N} - \frac{k}{d} < \frac{\frac{3\sqrt{2}}{2}N^{\frac{1}{2}}}{N} < \frac{3\sqrt{2}}{2}N^{-\frac{1}{2}}$$

En supposant que $d < \frac{6\sqrt{2}}{6}N^{-\frac{1}{2}}$, alors

$$\frac{3\sqrt{2}}{2}N^{-\frac{1}{2}} < \frac{1}{2d^2}$$

Et par conséquent, on obtient

$$\frac{e}{N} - \frac{k}{d} < \frac{1}{2d^2}$$

Ainsi, en utilisant le théorème *(Théorème 10)*, on peux en conclure que l'une des convergentes $\frac{k}{d}$ est une réduite de $\frac{e}{N}$. A l'aide de la *(Proposition 11.1)*, il est possible de factoriser N en temps polynomial dans $logN$.

4.2 Généralisation par de Weger

Théorème 14. *(Théorème de de Weger) Soit $n = pq$ un module RSA avec $q < p < 2q$ et $p - q = N^\beta$. Soit $e < \varphi(n)$ un exposant public pour lequel la clé secrète d est suffisamment faible $d < N^\delta$. Connaissant N et e, on peut alors calculer d et factoriser N.*

Démonstration. Considérons d'abord l'exposant public e et l'exposant privé d d'une instance RSA donnant l'équation $ed - k\varphi(N) = 1$. Celle-ci peut être réécrire de cette manière.

$$ed - k\left(N + 1 - 2\sqrt{N}\right) = 1 - k\left(p + q - 2\sqrt{N}\right)$$

En divisant par $\left(N + 1 - 2\sqrt{N}\right)d$ et en utilisant $p + q > 2\sqrt{N}$ que nous avons prouvé dans le Lemme XX, on obtient

$$\frac{e}{N + 1 - 2\sqrt{N}} - \frac{k}{d} = \frac{1 - k\left(p + q - 2\sqrt{N}\right)}{\left(N + 1 - 2\sqrt{N}\right)d} < \frac{k\left(p + q - 2\sqrt{N}\right)}{\left(N + 1 - 2\sqrt{N}\right)d} \tag{2}$$

En considérant les termes de la partie droite de l'équation *(2)*, on remarque que nous avons $N + 1 - 2\sqrt{N} > \frac{1}{2}N$ et donc, en utilisant le *(Lemme 12)*, on obtient

$$p + q - 2\sqrt{N} = \frac{(p+q)^2 - 4N}{p + q + 2\sqrt{N}} < \frac{(p-q)^2}{4\sqrt{N}}$$

On sait que $e < \varphi(n)$ et par hypothèse initiale on a $k = \dfrac{ed - 1}{\varphi(N)} < \dfrac{ed}{\varphi(N)} < d$. Par conséquent, les inégalités de l'équation (2) nous donnent

$$\frac{e}{N + 1 - 2\sqrt{N}} - \frac{k}{d} < \frac{k}{d} \cdot \frac{\dfrac{(p-q)^2}{4\sqrt{N}}}{\dfrac{1}{2}N} < \frac{(p-q)^2}{2N\sqrt{N}}$$

Pour appliquer le *(Théorème 10)*, nous avons besoin du condition suffisante.

$$\frac{(p-q)^2}{2N\sqrt{N}} < \frac{1}{2d^2}$$

En utilisant $d < N^\delta$ et $p - q = N^\beta$, la condition est satisfaite si $\delta < \frac{3}{4} - \beta$. Nous pouvons donc utiliser le *(Théorème 10)* pour en conclure que l'une des convergentes $\frac{k}{d}$ est une réduite de $\frac{e}{N}$

5 Implémentation avec Python

5.1 Fractions continues

Cette fonction convertit un nombre rationnel $\frac{x}{y}$ en une liste L de quotients partiels. Elle prend en paramètre d'entrée deux nombres entiers x pour la valeur de l'exposant de chiffrement e et y pour la valeur du module N. On utilise la fonction $math.log2$ pour initialiser la variable r qui permet de définir la limite maximum des quotients partiels de y.

```
def cont_frac(x : int, y : int) -> int:

    L = []
    a = x
    b = y
    r = log2(y)

    while(a%b != 0 and r > 0 ):
        q = a // b
        a_t = a
        a = b
        b = a_t % b
        L.append(q)
        r = r-1

    L.append(a // b)

    return L
```

Cette fonction calcule et retourne toutes les convergentes $\frac{k}{d}$ de $\frac{e}{N}$. On définit la variable a pour k et la variable b pour d. Pour parcourir la liste des quotients partiels, on initialise une variable i par la taille d'une liste l qui sera implémentée dans la prochaine fonction.

```
def compute_frac(l : list) -> tuple:

    i = len(l)-2
    c = 1
    a = l[i]
    b = l[i+1]

    while ( i > 0 ) :
        a = l[i]
        a = a * b + c
        c = b
        b = a
        i = i - 1

    a = l[0]
    a = a * b + c

    return (a,b)
```

Cette fonction donne la séquence fractionnaire d'une liste de quotients partiels à l'aide des précédentes fonctions et en prenant en paramètre d'entrée la liste des quotients partiels L. On affecte cette liste à la variable l afin de vérifier la taille de la liste. Avec l'aide d'instructions conditionnelles, on peut alors produire une liste de convergentes que la fonction retournera.

```
def frac_seq(L : list) -> list:

    seq = []

    for i in range(len(L)) :
        l = L[0:i]

        if len(l) == 0 :
            continue

        elif len(l) == 1 :
            seq.insert(i,(L[0],1))

        else :
            seq.insert(i,compute_frac(l))

    return seq
```

5.2 Arithmétique

Cette fonction utilise la méthode babylonienne pour extraire la racine carrée entière pour un nombre n positif uniquement, pris en paramètre d'entrée de la fonction.

```
def isqrt(n):

    if n < 0:
        raise ValueError('square root not defined for negative (n)')

    if n == 0:
        return 0

    a = int(log2(n)) + 1 // 2
    b = int(log2(n)) + 1 % 2
    x = 2**(a+b)

    while True:
        y = (x + n//x)//2

        if y >= x:
            return x

        x = y
```

5.3 Wiener

Cette fonction prend en paramètre deux entiers, soit un exposant de chiffrement e et un module N. Dans celle-ci, on commence tout d'abord par initialiser la liste des quotients partiels et la liste des convergentes, en appelant les fonctions précédentes. Ensuite, nous pouvons vérifier si le couple $\frac{k}{d}$ produit une factorisation de N. À l'aide du calcul de $\varphi(N)$ et de N nous pouvons déduire une éventuelle factorisation en résolvant l'équation du second degré. Pour cela, on commence par calculer le discriminant et on vérifie par la fonction d'arithmétique qu'il est bien un entier. Si le discriminant est inférieur à zero alors l'équation n'admet aucune solution. Dans notre cas, l'équation doit admettre deux solutions distinctes p et q. On vérifie ensuite que le résultat de N correspondant au produit de p et q et si c'est le cas, on retourne l'exposant de déchiffrement d associé à la convergente $\frac{k}{d}$ qui a permis de calculer $\varphi(N)$ et résoudre l'équation.

```
def Wiener(N : int, e : int) -> int :

    quot = ContinueFractions.cont_frac(e,N)
    seq = ContinueFractions.frac_seq(quot)

    for (k,d) in seq :

        if k == 0:
            continue

        phi = (e * d - 1) // k
        a = 1
        b = (N - phi) + 1
        c = N
        discr = b*b - 4 * a * c

        if discr > 0 :
            rdiscr = Arithmetic.isqrt(discr)

            if (rdiscr * rdiscr) == discr :
                p = -((-b - rdiscr) // (2 * a))
                q = -((-b + rdiscr) // (2 * a))

                if N == (p * q) :
                    return d

    print("Not vulnerable to Wiener")

    return 0
```

```
def launch_attack():

     e = 16917260959939977087817625773361191612612876574913 2...
     N = 39092753815604946508629330358212246668992977841527 9...

     d = Wiener(N, e)
```