UN CHIFFREMENT ASYMÉTRIQUE : LE PROTOCOLE RSA (1976)

Notions réinvesties : congruences, exponentiation modulaire rapide, théorèmes de Bézout et de Gauss, petit théorème de Fermat

Il nous entoure sans même que nous le sachions. Il est dans nos cartes bancaires, nos paiements en ligne, nos connexions à des sites sécurisés, nos messageries, nos logiciels…
*Le chiffrement RSA (nommé par les initiales de ses trois inventeurs : Ronald **R**ivest, Adi **S**hamir et Leonard **A**dleman) est un algorithme de cryptographie asymétrique créé en 1977.*

Les circonstances de la découverte sont paradoxales : ces trois auteurs avaient décidé de travailler ensemble afin d'établir que les systèmes cryptographiques « à clef publique » (où la clef de codage du message est connue de tous), inventés quelques mois auparavant par Diffie et Hellman, étaient une impossibilité logique (autrement dit, qu'ils présentaient nécessairement des failles). Ils ne réussirent pas dans leur projet, mais, bien au contraire, découvrirent un nouveau système à clef publique qui supplanta bientôt celui de Diffie et Hellman.

Très utilisé dans le commerce électronique, il l'est plus généralement pour échanger des données confidentielles sur Internet. Cet algorithme a été breveté par le Massachusetts Institute of Technology (MIT) en 1983 aux États-Unis. Ce brevet a expiré le 21 septembre 2000.

A. Shamir R. Rivest L. Adleman

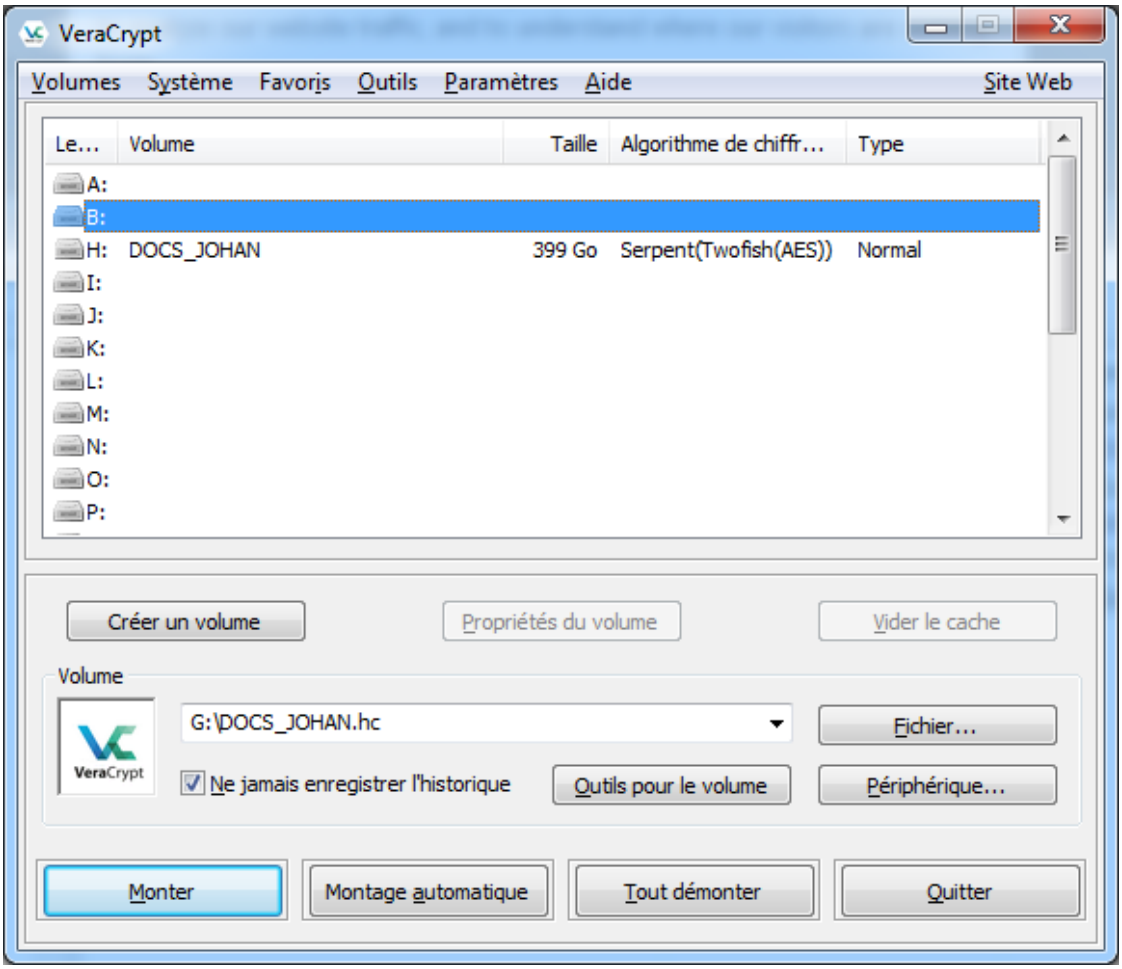
R. Rivest (2015) A. Shamir (2013) L. Adleman (2010)

Dans l'univers de la cryptographie, on distingue deux types de chiffrement : ***asymétrique*** et ***symétrique***.

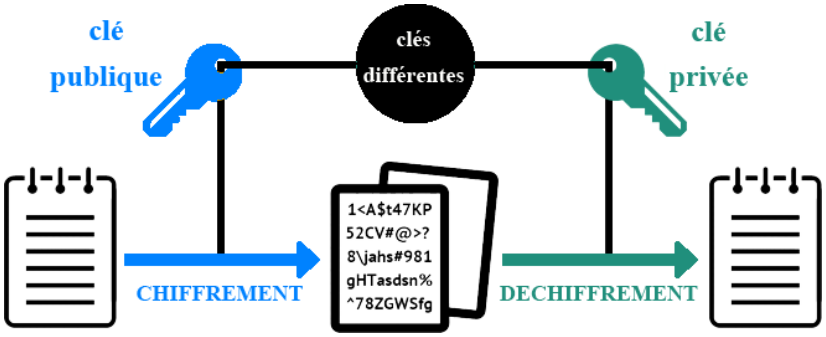
• Les systèmes symétriques utilisent une seule clé pour chiffrer et déchiffrer.

On peut utiliser la métaphore du coffre fort : le coffre fort dispose d'une seule clé qui est la même pour ouvrir et fermer le coffre.

Exemples de chiffrement symétrique : chiffrement affine, chiffrement de Vigénère-Bellaso, chiffrement de Hill… mais aussi les très utilisés AES/Rijndael, Serpent, Twofish, etc. Par exemple, le logiciel VeraCrypt, qui permet de chiffrer des disques durs ou des clés USB, permet d'utiliser plusieurs chiffrements symétriques considérés aujourd'hui comme robustes.



• Les systèmes asymétriques utilisent deux clés. Une pour chiffrer et une autre pour déchiffrer.
La clé servant à chiffrer est ***une clé publique*** et celle servant à déchiffrer est ***une clé privée***.



La clé publique est visible par tout le monde, par exemple dans un annuaire.
La clé privée n'est visible et connue que par son propriétaire.
Si une quelconque personne obtenait une clé privée qui n'est pas la sienne, elle serait alors en mesure de déchiffrer les messages chiffrés qui ne lui sont pas destinés.

On peut ainsi voir la clé publique comme une boite aux lettres (où n'importe qui peut mettre – chiffrer – des messages) dont seul le propriétaire de la clé de la boite (clé privée) peut lire – déchiffrer – les messages.

RSA représente sans conteste le système cryptographique à clé publique le plus utilisé de nos jours puisqu'il est parvenu à survivre aux critiques des spécialistes en cryptographie pendant plus d'un quart de siècle.
La fiabilité de cet algorithme mondialement connu repose sur :

– l'arithmétique modulaire ;
– la difficulté de décomposer en produit de facteurs premiers de gros entiers de l'ordre des nombres décimaux à 100 chiffres ;
– le « petit théorème de Fermat », démontré dans une autre fiche :

– PETIT THÉORÈME DE FERMAT

Soit p un nombre premier et a un entier premier avec p : $a^{p-1} \equiv 1\ [p]$.
ie
Soit a un entier et p un nombre premier ne divisant pas a : $a^{p-1} \equiv 1\ [p]$.

– COROLLAIRE

Soit p un nombre premier et a un entier naturel : $a^p \equiv a\ [p]$.

Remarque : Fermat cite ce résultat dans une lettre de 1640, mais il ne le prouve pas.
Les premières preuves sont dues à Euler et à Leibniz.

EXERCICE PRÉLIMINAIRE (pas évident)

On note $n = pq$ où p et q sont deux nombres premiers distincts.
On pose $m = (p-1)(q-1)$ et on note c un nombre premier avec m.

On note x un entier naturel.

1. Démontrer qu'il existe un entier naturel d tel que $cd \equiv 1\ [m]$.

2. a) Supposons x divisible par p. Démontrer que : $x^{cd} \equiv x\ [p]$.

b) Supposons x non divisible par p. Démontrer que $x^{p-1} \equiv 1\ [p]$ puis que : $x^{cd} \equiv x\ [p]$.
Aide : théorème de Fermat.

3. De façon analogue, on montrerait que : $x^{cd} \equiv x\ [q]$.
En déduire que : $x^{cd} \equiv x\ [n]$.
Aide : théorème de Gauss.

LE PROTOCOLE RSA

Voici comment fait Bob pour écrire à Alice.

CLÉS PUBLIQUES ET PRIVÉES	
Étape 1	Alice choisit deux grands nombres premiers p et q, et calcule les produits : $m=(p-1)(q-1)$ et $n=pq$.
	Puis Alice détermine un entier naturel c tel que c est premier avec m, et strictement inférieur à m
	Pour décrypter plus tard un message qu'on lui envoie, Alice détermine l'entier d tel que : $cd \equiv 1\ [m]$ avec $1<d<m$.
n et c sont des clés ***publiques*** p, q et donc d sont des clés ***privées***, connues uniquement par Alice	
BOB ENVOIE UN MESSAGE CRYPTÉ À ALICE	
Étape 2	Bob remplace les lettres de son message par leurs positions dans l'alphabet, en les regroupant par blocs* de taille inférieure à n.
	Pour chaque bloc x, Bob détermine b tel que $x^c \equiv b\ [n]$. Il envoie alors cette série de blocs à Alice : c'est le message crypté.
ALICE DÉCRYPTE LE MESSAGE	
Étape 3	Pour chaque bloc b reçu, Alice calcule b^d modulo n.
	La série de blocs obtenue est le message envoyé par Bob.

* on regroupe par blocs car sinon le message pourrait être déchiffré par analyse fréquentielle, qui repose sur le fait que dans un texte, les lettres ont des fréquences différentes. Par exemple, en français, la fréquence de la lettre E est, selon le texte, presque toujours supérieure aux fréquences des autres lettres. Il y a donc de fortes chances pour que, dans un texte codé, la lettre qui apparaît le plus fréquemment représente un E. Les lettres les moins fréquentes représentent probablement un W, un K ou un X… Sur un exemple plus simple, le message chiffré « 763604846 1**890292732** 763604846 1**890292732** » nous indiquerait déjà qu'une lettre est utilisée deux fois, c'est sans doute une voyelle. Des mots de quatre lettres ainsi formés, il n'y en a pas des millions. Ici, le mot chiffré était « caca ».

UN EXEMPLE DE CODAGE

On choisit deux clés privées $p=42139$ et $q=47837$, que l'on admet premiers.

1. a) Calculer n et m. Pourquoi puis-je choisir $c=12\,111\,985$?
Quelles sont les clés ***publiques*** ?

b) Déterminer la clé privée d.

2. Rappelons quelques fonctions Python qui nous serons utiles :

ord() Renvoie le nombre entier représentant le code (dans le standard Unicode) du caractère représenté par la chaîne donnée. Par exemple, *ord('a')* renvoie le nombre entier 97 et *ord('€')* renvoie 8364.

chr() Il s'agit de l'inverse de *ord()*.

Str() Convertit en chaîne de caractères. Par exemple *str(2)* renvoit la chaîne *'2'*.

pow() *pow(a, e, n)* permet de calculer a^e modulo n (exponentiation modulaire rapide).

Voici ci-dessous un programme Python qui permet de chiffrer simplement[1] un texte.
Chaque caractère du texte est remplacé par son code Unicode (nombre compris entre 1 et 1114111) et ce nombre est chiffré. L'algorithme affiche le message codé, chaque lettre codée étant séparée d'un espace.

```
def chiffre_rsa(message, n, c) : #n=pq et c premier avec (p-1)(q-1)
    mescode = ""
    for l in message :
        y = pow(ord(l), c, n)
        y = str(y)
        mescode = mescode + y +" "
    return mescode

n, c, d = 2015803343, 12111985, 492080977
message = input('Message à chiffrer : ')
print("Message chiffré : ", chiffre_rsa(message, n, c))
```

a) Bien comprendre chaque ligne de cet algorithme.

b) En vous aidant du travail déjà effectué ci-dessus, écrire un programme Python qui permet de mieux chiffrer un message, en regroupant par blocs de 15 (afin qu'il puisse pas être déchiffré par analyse fréquentielle). Vous pourriez avoir besoin de la fonction suivante, qui peut être programmée mais fait gagner du temps :

zfill() Renvoie une chaîne de caractère avec des 0 à gauche.
Par exemple, *zfill('test', 9)* renvoie *00000test*.

Voici les grandes étapes du programme que vous devez écrire :
– transformer d'abord chaque lettre en son nombre Unicode en rajoutant autant de 0 qu'il faut à gauche pour avoir un nombre codé sur 7 valeurs (car avec la fonction *ord()* de Python, il y a 1114111 valeurs possibles). Par exemple la lettre *'e'* aura un Unicode de 101, que l'on transformera en **0000**101.
– créer une chaîne de caractères qui contient tous les nombres ci-dessus les uns après les autres (du type *'0000101000154700154670000015'*) puis chiffrer chaque bloc de 15 et afficher le résultat final sous la forme de blocs de 15 (à chaque fois, rajouter autant de 0 qu'il faut, à gauche, pour avoir des blocs de 15).
Par exemple, `j'aime les maths` sera d'abord transformé en :

```
000010600000390000097000010500001090000101000003200001080000101000011500000320
000109000009700001160000104000011 5
```

puis chiffré en :

```
000000518630026 000000320748074 000001441642578 000001060515593 00000023859339
9 000000449300379 000000941212706 000000120130546.
```

Lorsque votre programme est fonctionnel, envoyez-le-moi en m'écrivant un mail (avec le sujet « tsspé – RSA » et votre nom dans le mail). Je vous enverrai alors une correction de cette question.

3. Il serait intéressant de faire un programme qui déchiffre un message chiffré, bien sûr en connaissant la taille des blocs choisis pendant le chiffrement. Si ça vous intéresse, vous pouvez le faire et me l'envoyer.

1 sans faire de blocs particuliers, donc attaquable par analyse fréquentielle

SÉCURITÉ : ATTAQUE PAR FORCE BRUTE POSSIBLE ?

1. Une clé de 256 bits comprend combien (valeur approchée) de chiffres décimaux ?

2. a) Si un nombre (une clé) n codée en 256 bits comporte deux facteurs premiers d'environ 39 chiffres et qu'on souhaite factoriser ce nombre, on peut souhaiter diviser n par l'ensemble des nombres premiers à 39 chiffres ou moins.
En supposant que 0,1 % des nombres soient des nombres premiers, à combien de divisions devrons-nous procéder ?

b) En supposant qu'on dispose d'un ordinateur capable de réaliser 10^{20} divisions par seconde[2], combien de temps passerons-nous à attaquer la clé n ?

EN PRATIQUE

L'algorithme, bien qu'assez sûr, est lent ! Dans la pratique, il sert le plus souvent à transmettre une clé servant à déchiffrer un message codé selon une méthode plus rapide, par exemple par chiffrement symétrique, comme l'AES, qui est actuellement le plus utilisé et le plus sûr.
L'AES (**A**dvanced **E**ncryption **S**tandard, soit « standard de chiffrement avancé »), aussi connu sous le nom de Rijndael[3], est un algorithme de chiffrement symétrique. Il remporta en octobre 2000 le concours AES, lancé en 1997 par le NIST[4] et devint le nouveau standard de chiffrement pour les organisations du gouvernement des États-Unis. Il a été approuvé par la NSA.

UN PROTOCOLE SÉCURISÉ… POUR COMBIEN DE TEMPS ?

Le 12 décembre 2009, le plus grand nombre factorisé par force brute était long de 768 bits.
Les clés RSA sont habituellement de longueur comprise entre 1 024 et 2 048 bits.
Quelques experts croient possible que des clés de 1 024 bits seront cassées dans un proche avenir, mais peu voient un moyen de casser de cette manière des clés de 4 096 bits dans un avenir prévisible proche.
On peut néanmoins présumer que RSA reste sûr si la taille de la clé est suffisamment grande.
On peut trouver la factorisation d'une clé de taille inférieure à 256 bits en quelques minutes sur un ordinateur individuel, en utilisant des logiciels librement disponibles.
Pour une taille allant jusqu'à 512 bits, et depuis 1999, il faut faire travailler conjointement plusieurs centaines d'ordinateurs.
Par sûreté, il est couramment recommandé que la taille des clés RSA soit au moins de 2 048 bits.

Si une personne possède un moyen « rapide » de factoriser le nombre n, tous les algorithmes de chiffrement fondés sur ce principe seraient remis en cause ainsi que toutes les données chiffrées dans le passé à l'aide de ces algorithmes.

En 1994, un algorithme permettant de factoriser les nombres en un temps non exponentiel a été écrit pour les ordinateurs quantiques. Il s'agit de l'*algorithme de Shor*. Les applications des ordinateurs quantiques permettent théoriquement de casser le RSA par la force brute, ce qui a activé la recherche sur ce sujet ; mais actuellement ces ordinateurs génèrent des erreurs aléatoires qui les rendent inefficaces.

Les autres types d'attaques, beaucoup plus efficaces, visent la manière dont les nombres premiers p et q ont été générés, et comment c a été choisi, si l'on dispose de messages codés ou de toute autre information qui peut être utilisée. Une partie de la recherche sur ce sujet est publiée mais les techniques

2 Un ordinateur est capable de tester plusieurs centaines de milliers voire quelques millions de mots de passe par seconde.

3 Conçu par Joan Daemen et Vincent Rijmen, deux chercheurs belges.

4 Le National Institute of Standards and Technology, ou NIST (qu'on pourrait traduire par « Institut national des normes et de la technologie »), est une agence du département du Commerce des États-Unis. Son but est de promouvoir l'économie en développant des technologies, la métrologie et des standards de concert avec l'industrie.

les plus récentes développées par les entreprises de cryptanalyse et les organismes de renseignement comme la NSA restent secrètes.

Remarque : il n'a jamais été démontré que le code ne pouvait être cassé sans la connaissance de p et q !

IMPOSSIBLE À DÉCHIFFRER ?

Il semble que les transmissions cryptées qui circulent sur Internet ne soient pas autant sécurisées qu'elles le prétendent. C'est en tout cas ce que Edward Snowden nous apprend en affirmant que la *National Security Agency* américaine (NSA) est tout à fait capable de décrypter les échanges numériques qui l'intéressent.
Les mathématiciens de la NSA auraient-ils découvert un algorithme efficace permettant la factorisation des grands entiers ? Ce ne semble en fait pas être le cas. La NSA a plutôt exploité la puissance commerciale des États-Unis pour imposer au monde entier des algorithmes qui posséderaient des « portes dérobées » (*backdoor*) aidant au déchiffrement.
Les experts soupçonnaient déjà que le *Data Encryption Standard* (le fameux standard DES) comportait une information cachée. Cette fois-ci, ce sont les protocoles de génération de nombres pseudo-aléatoires qui sont suspects. Enfin, les clés de cryptographies des principaux acteurs de l'Internet seraient connues de la NSA…

Source : Tangente Hors-série n° 49 « Les maths de l'impossible »

Conclusion : il n'est pas du tout impensable que les services américains aient prévu (forcés ou négociés) l'installation d'une *backdoor* dans un ou plusieurs algorithmes comme AES ou RSA. À ce sujet, lire cet article.

UN INTÉRÊT ÉNORME : L'AUTHENTIFICATION

sur le site de la banque LCL →

Les avantages de la signature électronique

La signature électronique présente de nombreux intérêts tant pour les particuliers que pour les entreprises. Elle permet notamment de :

- **sécuriser les échanges et les transactions en ligne** : l'identité du signataire et le contenu du document sont authentifiés ;
- **signer un document sans l'imprimer** : en plus des économies d'encre, de papier et de frais d'envoi, les signataires n'ont pas besoin de se rencontrer physiquement. La transmission de documents numériques est simplifiée et instantanée
- **partager un document de plusieurs façons** : le document dématérialisé peut être transmis par mail, mais il peut également être transporté sur un support numérique comme une clé USB ou un disque dur.
- **archiver des documents numériquement**

Dans le protocole RSA décrit ci-dessus, les nombres c et d jouent un rôle interchangeable…
c est une clé publique alors que d est une clé privée, mais si Alice décide de coder elle-même un message en utilisant sa clé privée d, alors Bob pourra décrypter ce message avec la clé publique c.

Si Bob réussit effectivement à décrypter le message avec c, c'est que le message avait bien été crypté avec la clé privée d (que Alice est la seule à connaître).

Il s'agit donc d'une ***signature électronique***, qui garantit l'authenticité du message reçu, et qui peut servir à garantir qu'un message envoyé par Bob a bien été lu par Alice.

Même si, par « l'attaque de l'homme du milieu », un espion avait intercepté les messages envoyés par Alice ou Bob afin de les modifier, vérifier la provenance et l'authenticité du message garantit que les personnes qui s'échangent des informations sont les bonnes.

DE TRÈS GRANDS NOMBRES PREMIERS ?

Le chiffrement demande de pouvoir vérifier que de « très grands » nombres sont des nombres premiers, pour pouvoir trouver p et q, mais aussi que le produit de ces deux très grands nombres ne soit pas factorisable facilement. En effet les algorithmes efficaces connus qui permettent de vérifier qu'un nombre n'est pas premier ne fournissent pas de factorisation.

Le couple de clefs demande de choisir deux nombres premiers de grande taille, de façon qu'il soit calculatoirement impossible de factoriser leur produit.

Pour déterminer un nombre premier de grande taille, on utilise un procédé qui fournit à la demande un entier impair aléatoire d'une taille suffisante, puis un test de primalité permet de déterminer s'il est ou non premier, et on s'arrête dès qu'un nombre premier est obtenu. Le théorème des nombres premiers assure que l'on trouve un nombre premier au bout d'un nombre « raisonnable » d'essais.

La méthode demande cependant un test de primalité très rapide. En pratique on utilise un test probabiliste, le test de primalité de Rabin-Miller ou une variante. Un tel test ne garantit pas exactement que le nombre soit premier, mais seulement une (très) forte probabilité qu'il le soit.
Quand on utilise le test de Rabin-Miller, le risque est réel de tomber sur un nombre composé qui, parce qu'il a une propriété particulière non identifiée, piège l'algorithme… Toutefois, ce risque est équivalent à celui d'une erreur logicielle, bien particulière elle aussi, comme une erreur de microprocesseur ou de compilateur qui pourrait produire le même effet numérique…
De nombreux mathématiciens et passionnés des nombres distinguent soigneusement une affirmation de primalité issue d'un algorithme du type Rabin-Miller, qu'ils jugent douteuse (même si on s'est arrangé pour que la probabilité d'erreur due à la méthode soit inférieure à $10^{-1\,000000}$), d'une affirmation de primalité issue d'un algorithme sûr, qu'ils considèrent comme définitive (même si les risques pris dans la mise en œuvre de l'algorithme donnent une probabilité d'erreur bien supérieure à $10^{-1\,000000}$).