Introduction au Chiffrement : Théories et Pratiques du Chiffrement Symétrique et Asymétrique avec une Étude de Cas sur l'Algorithme RSA.

Joshua Meffre

Abstract

Je tiens avant tout à apporter une précision : Je ne suis ni un scientifique ni un professionnel du domaine que j'avance ici. Cet article est le résultat d'énormément de curiosité et de passion de ma part et est né de ma volonté la plus humble de partager mes connaissances et mon enthousiasme pour ce sujet. Cet article est le fruit de mes recherches personnelles et de ma compréhension actuelle en la matière. Il est important de noter que malgré mes efforts pour fournir des informations précises et à jour, cet article peut contenir des erreurs ou des approximations. La complexité du chiffrement en fait une discipline en constante évolution, et ma compréhension reste limitée à ce que j'ai pu apprendre en amateur passionné. En conséquence, cet article ne doit EN AUCUN CAS être utilisé comme une source unique ou définitive pour un travail de recherche sérieux ou pour des utilisations professionnelles. Je vous encourage vivement à consulter des sources spécialisées et fiables dans le domaine, vous pourrez retrouver certaines de ces sources en fin d'article en plus des sources sur lesquelles je me suis moi-même appuyé pour l'écriture de ce dernier. Merci de lire cet article avec un esprit critique et une extrême vigilance. Mon objectif principal est de susciter l'intérêt et la curiosité pour un sujet que je trouve passionnant, et non de fournir une expertise technique exhaustive. CET ARTICLE N'EST PAS UN ARTICLE SCIENTIFIQUE ET IL NE CHERCHE PAS À L'ÊTRE.

Table of Contents

1. Quelques notions avant de commencer.

Cet article contient des formules mathématiques et un jargon qui peuvent paraître inhabituels pour les non-initiés, mais je tiens à vous rassurer, **il n'y a rien de compliqué**. Cette première partie existe justement pour vous aider à comprendre ou à vous remémorer les terminologies mathématiques et le jargon verbal présent dans cet article et nécessaire à sa bonne compréhension. N'hésitez pas à souvent y revenir si vous vous sentez perdu. De plus, chaque fois qu'un terme présent dans cette liste sera utilisé, il sera suivi d'un lien cliquable vous permettant d'accéder facilement à la définition de celui-ci.

1.1 Notions de Mathématiques.

Def 1.1.1 : Un entier naturel($\mathbb{N}$) est un nombre qui permet de compter des unités équivalentes, on les reconnaît parce qu'il ne possède pas de décimales, c'est-à-dire qu'ils ne possèdent pas de nombre après la virgule *(exemple : 30 est un entier naturel alors que 30,6 n'est pas un entier naturel)*.

Def 1.1.2 : La division euclidienne($/$) est une variante de la divison($\div$) et qui a pour particularité de ne travailler qu'avec des entiers naturels($\mathbb{N}$), ce qui donne un résultat en deux parties, le quotient(q) et le reste(r).

Def 1.1.3 : L'opérateur modulo ($\mod$) est un opérateur qui permet le calcul du reste d'une division euclidienne($/$) et qui renvoie le reste(r) comme résultat.

Def 1.1.4 : L'opérateur de congruence($\equiv$) est un opérateur de vérification, comme l'opérateur d'égalité($=$). Il vérifie qu'une valeur a donne bien un résultat c lors d'une opération $a \mod b$. Dans le cas où la vérification est fausse, alors a n'est pas congruent($\not\equiv$) à $c \mod b$

Def 1.1.5 : L'algèbre booléenne, est une algèbre dans lequelle on calcule des valeurs ne pouvant posséder que deux états : $faux, vrai$ | $0, 1$ | $éteint, allumé$.

Def 1.1.6 : Une table de vérité, en algèbre booléenne, est un tableau qui liste toutes les combinaisons possibles des valeurs (0 ou 1) des variables d'une expression booléenne et montre le résultat de l'expression pour chacune de ces combinaisons.

Def 1.1.7 : La porte logique $XOR(\oplus)$ également appelé *ou exclusif* est un opérateur de l'algèbre booléenne dans lequel le résultat n'est $vrai$ que si les valeurs en entrée sont différentes. *Table de vérité :*

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

1.2 Notions de Vocabulaire.

Def 1.2.1 : En clair *(Ex : "Message en clair")* : En cryptographie, le terme "En clair" fait référence à l'état d'un message ou d'une information qui n'a pas été chiffrée d'une quelconque manière et qui est compréhensible par tout le monde.

Def 1.2.2 : Clé de chiffrement : La clé de chiffrement est la composante qui permet la transformation d'un texte, d'une information ou d'une valeur en clair en utilisant un algorithme de chiffrement.

2. Contexte.

Cet article vise à partager une exploration du domaine de la cryptographie, en partant d'une perspective personnelle. Il me semble important de noter que mon intérêt pour ce sujet n'est pas le fruit d'un attrait naturel, mais plutôt d'une série de circonstances fortuites qui ont éveillé ma curiosité. Cette précision, bien que non essentielle à la compréhension des concepts cryptographiques abordés ici, apporte un éclairage contextuel que je pense pertinent. Le lecteur peut néanmoins choisir de passer directement au chapitre suivant pour se concentrer sur les aspects techniques de l'article.

Étant de nature assez curieuse, et ayant eu très tôt accès à internet, il ne m'est pas rare de m'éparpiller dans les recoins du web et de finir par y trouver un sujet qui me prend de passion rapidement et devient mon obsession pour une durée tout à fait aléatoire. C'est exactement ce qui s'est passé récemment lorsque je suis pris de curiosité pour la pratique du *self-hosting*, dont j'utiliserai le terme français "auto-hébergement". Cette pratique, qu'on pourrait qualifier d'une volonté d'indépendance numérique, consiste à héberger, chez soi, sur son propre serveur informatique, les différents services que l'on retrouve habituellement sur internet. Pourquoi le faire me direz-vous ? Comme évoqué plus haut : l'indépendance, dans un monde de plus en plus régie par les grandes entreprises du numérique, à une époque où celui-ci devient quasiment indispensable, la volonté d'avoir le contrôle sur ce que ces entreprises possèdent ou non devient de plus en plus un problème que l'on cherche à résoudre ; du moins, que **je** cherche à résoudre. De plus, l'implémentation d'un serveur personnel chez soi permet une acquisition de connaissances importantes dans le domaine de l'informatique et dans la compréhension de celui-ci. Cependant, il ne s'agit pas *(encore)* d'un article sur l'auto-hébergement, je vais donc aller au but. Après avoir fait l'acquisition d'une vieille tour d'ordinateur qui me sert aujourd'hui de serveur personnel, il m'a fallu le configurer, et parmi les étapes auquel il est complexe d'échapper, il y a SSH (Secure Shell), un système bien pratique qui, pour simplifier énormément, est un protocole de communication sécurisé, et qui, dans mon exemple, me permet de communiquer avec mon serveur via d'autres appareils partout dans le monde. La sécurité de cet outil repose sur un principe fondamental : le chiffrement asymétrique ou le chiffrement à clé publique. Ce type de chiffrement, qui est le responsable principal de l'existence de cet article tant il a éveillé ma curiosité, possède deux avantages majeurs, dont l'un d'eux me fait poser une question qui viendra plus tard dans l'article, mais qui en est le cœur de mon raisonnement. Cependant, avant de pouvoir l'aborder, j'ai besoin de revenir un peu sur l'histoire du chiffrement et sur des notions importantes qui l'entourent.

3. La petite histoire du chiffrement.

Les toutes premières traces de chiffrement remontent à 1900 av. J.-C., en Égypte antique lorsqu'un scribe esquisse les hiéroglyphes racontant la vie de son seigneur. En effet, le système d'écriture des hiéroglyphes qu'il utilise est assez inhabituel et volontairement complexe, bien qu'il s'agit plutôt d'une volonté de rendre hommage de manière harmonieuse à son seigneur afin de lui conférer dignité et autorité plutôt qu'une intention explicite de chiffrement. Il est ainsi possible de retrouver des phrases tel que "In the year of Our Lord One thousand eight hundred and sixty three" (Dans l'année de notre seigneur mille-huit-cent-soixante-trois), plutôt que d'avoir simplement écrit "1863". Bien qu'il ne s'agisse pas d'un type de chiffrement volontaire et explicite, ce texte est responsable de la naissance de la cryptographie puisque il incorpore un élément essentiel de celle-ci : une transformation délibérée de l'écriture. [1]

La première apparition s'apparentant à de la cryptographie moderne et explicite date d'environ 1500 av. J.-C. et a été retrouvée en Mésopotamie, ce qui s'apparente actuellement à l'Irak et la Syrie. Elle y apparait sous forme d'une petite tablette contenant une ancienne formule de fabrication de glaçures pour la poterie. Le scribe, qui, visiblement, souhaitait garder sa formule bien secrètement, jouait sur le fait que les signes cunéiformes (signes pouvant représenter différentes syllabes ou sons) pouvaient posséder des valeurs syllabiques différentes pour un même signe et choisissait volontairement les valeurs les moins communes de chaque signe, rendant la lecture compliqué. Le raisonnement de cette méthode ressemble à celle de George Bernard Shaw, avec laquelle il est notamment possible d'écrire le mot anglais *fish* en *ghoti* (le *gh* faisant le son /f/ comme dans le mot *tou**gh***, le *o* faisant le son /i/ de *w**o**men* et *ti* faisant le son /sh/ de *na**ti**on*). Ajoutez à cela que le scribe tronquait également certains sons en ignorant volontairement la consonne finale pour certains signes (par exemple si un signe représente la syllabe *kA*, le signe pouvait simplement être écrit *k*.) et qu'un même mot pouvait être composé de signes différents à chaque itération. Ainsi est née la cryptographie. [2]

[1] Kahn, D. (1968). 2. The First 3,000 Years. In *The Codebreakers: The story of secret writing* (pp. 71-78). essay, The Macmillan Company. (En Anglais.)

[2] Kahn, D. (1968). 2. The First 3,000 Years. In *The Codebreakers: The story of secret writing* (pp. 71-78). essay, The Macmillan Company. (En Anglais.)

4. Le chiffre de César.

Mais l'exemple le plus connu, et qui sera ici le premier objet d'étude, c'est le chiffre de César, [3] qui porte officiellement le nom de chiffrement par décalage. Le concept, s'il n'a pas déjà été saisi, est simple : décaler les lettres d'un message en fonction de leurs positions dans l'alphabet et sur la base d'une fonction mathématique. Ainsi pour un message *"Bonjour"* et pour une fonction $+3$ on obtient *"Erqmrxu"*. Si le concept théorique reste simple à comprendre, il est intéressant de mettre en place une description précise du procédé mathématique de cette méthode de chiffrement. Ainsi cela va permettre au lecteur de s'habituer à la lecture des raisonnement mathématiques qui seront présent tout au long de cet article.

Pour chiffrer un message avec le chiffrement par décalage, on suit le raisonnement suivant : #### Def 4.1.1 : Définir, pour chaque lettre, un nombre en fonction de sa position dans l'alphabet, ainsi $A = 0$ de par sa première position dans l'alphabet, $B = 1$, $C = 2$, etc.

Def 4.1.2 : Définir une valeur n nécessaire au décalage des lettres d'un message (dans le chiffre de César $n = 3$).

Def 4.1.3 : Ainsi, il est possible d'encoder chaque lettre x d'un message via la formule suivante :

$$E_n(x) = (x + n) \bmod 26$$

Dans un objectif de clarification, il est permis de découper en deux la présente opération :

1. $(x + n)$ où x représente la valeur en clair *(Def 1.2.1)* d'une lettre et où n représente la valeur de décalage. En additionnant ces deux valeurs ensembles, on obtient un nombre qui, une fois reconverti en lettre via la méthode utilisé en *Def 4.1.1*, donne une lettre Z qui à pour objectif de servir de substitut de chiffrement à la lettre en clair initiale *(exemple : si $x = 2$ (donc la lettre D) et $n = 3$ le résultat de l'addition des deux valeurs x et n est de 5 (donc la lettre G)*.
2. $\bmod 26$ est présent pour s'assurer que le résultat du décalage d'une lettre ne sorte pas des limites de l'alphabet *(exemple : si $x = 25$ (donc la lettre Z) et $n = 3$, le résultat de $x + n$ est de 28, or 28 n'est égal à aucune lettre de l'alphabet, l'opérateur modulo (Def 1.1.3) est alors utilisée pour s'assurer de l'affirmation suivante : $x + n < 26$. Si tel n'est pas le cas, alors l'opération $\bmod 26$ permet de replacer la valeur de décalage au début de la liste (ici l'alphabet). Cela est dû au fait que dans l'opération $Z \bmod 26$ (où $Z = x + n$) le résultat vaut toujours Z sauf si $Z \geq 26$, ce qui*

[3] Nom donné en rapport au fait que Jules César utilisait ce type de chiffrement pour ses communications secrètes. (En Anglais.)

fait que, pour l'exemple cité plus haut où $x = 25$ et $n = 3$, le résultat de $x + n \mod 26$ sera de 2 (donc la lettre D) et non 28 (qui ne correspond à aucune lettre de l'alphabet).)

Def 4.1.4 : Pour déchiffrer le message il suffit de décoder chaque lettre en utilisant la valeur inverse de n qui est égal à $-n$, via la formule suivante :

$$D_n(x) = (x - n) \bmod 26$$

Il est important de noter que la valeur n constitue notre clé de chiffrement et de déchiffrement. C'est cette clé qui permet d'appliquer le décalage, et par conséquent le chiffrement notre message. Le fait que la même clé est utilisée pour chiffrer et déchiffrer un message est appelé chiffrement symétrique, car la même clé est utilisée dans les deux sens.

5. One Time Pad et chiffrement symétrique.

Il existe plusieurs méthodes de chiffrement symétrique, chacune présentant avantages et inconvénients, mais une analyse exhaustive de chaque méthode serait longue et fastidieuse. Par conséquent, je me concentrerai sur une méthode particulière : le One Time Pad (OTP), qui se distingue des autres par une promesse unique : **être incassable**. Cette affirmation est particulièrement remarquable dans le domaine de la cryptographie, où l'on tend à éviter tout sentiment de sécurité absolue et à considérer tout système comme théoriquement vulnérable. Cependant, la promesse peut bel et bien être tenue si des conditions bien précises sont réunies :

Def 5.1.1 : La clé de chiffrement doit être générée de manière véritablement aléatoire, sans motifs prédictibles.

Def 5.1.2 : La longueur de la clé doit être égale à celle du message à chiffrer.

Def 5.1.3 : Chaque clé générée ne doit être utilisée qu'une seule fois, afin d'éviter les attaques par analyse statistique.

Ainsi, une fois toutes ces conditions réunies, tout message peut être chiffré à l'aide du One Time Pad, garantissant ainsi l'inviolabilité de l'information ou du message chiffré. Le OTP repose sur le fonctionnement de la porte logique $XOR(\oplus)$ *(Def 1.1.6)* pour transformer une séquence de bits en une autre. L'avantage de $XOR(\oplus)$ réside dans le fait que pour un bit chiffré égal à 1, la probabilité que le bit en clair soit 0 ou 1 est équivalente (voir table de vérité en *Def 1.1.6*). Illustrons cela avec un exemple pour clarifier les choses : soit un message en clair représenté par la séquence de bits 0110100 et une clé 1100101, il est possible de chiffrer le message en commençant par le premier bit, on utilise donc l'opération $XOR(\oplus)$ comme suit :

$$0 \oplus 1 = 1$$

Pour chiffrer le deuxième bit, même opération :

$$1 \oplus 1 = 0$$

Puis récidiver la transformation pour chaque bit jusqu'à obtention du message chiffré 1010001. Ainsi, un adversaire n'ayant accès qu'à ce message chiffré ne pourrait en déduire le message en clair, car chaque bit a une probabilité équivalente d'être 1 ou 0. En d'autres termes, en l'absence de la clé de chiffrement, ce message chiffré pourrait très bien correspondre à un message en clair initial 1111111 ou tout aussi bien 0000000.

Plus concrètement, il existe un nombre de figures exponentiellement grand qui est uniquement limité par la longueur du message. En termes mathématiques, pour un message de longueur n, il y a 2^n configurations possibles.

Si le OPT possède une robustesse qui, en plus d'être impressionnante, n'est pas à écarter, il possède pourtant une faiblesse qu'il partage avec tous ses collègues du chiffrement symétrique : la transmission de la clé. En effet, dans le cas où un message chiffré par une méthode de chiffrement symétrique est envoyé, il faut pouvoir transmettre la clé au récepteur du message afin que ce dernier puisse le déchiffrer. Et il faut le faire sur un canal de communication suffisamment sécurisé pour qu'elle ne soit pas interceptée, auquel cas l'action de chiffrer le message perd son sens. Se présente donc le problème suivant :

Def 5.2.1 : *Si l'on peut transmettre la clé via un canal suffisamment sécurisé pour qu'elle ne soit pas interceptée, pourquoi ne pas directement envoyer le message en clair (Def 1.2.1) via ce même canal sécurisé ?*

En effet, il existe bien des solutions à ce problème, un exemple classique consiste à définir un canal sécurisé de manière unique pour la transmission de la clé, comme une rencontre en personne. Seulement cet exemple reste peu pratique selon les situations et notamment dans le cas du OTP, puisque selon la *Def 5.1.3*, chaque clé doit être générée de manière unique, ce qui nécessiterait donc une rencontre physique à chaque message, on en revient donc à notre problème *Def 5.2.1*. De plus, une telle rencontre présente rapidement des limitations pratiques. Elle peut nécessiter des moyens considérables selon la sensibilité des informations à échanger et qui se révèle même impraticable dans les contextes de communications longue distance.

6. Chiffrement asymétrique et Algorithme RSA.

Ainsi est introduit en 1976, par Whitfield Dillie et Martin E. Hellman, le concept de cryptographie à clé publique. Ce concept repose sur la génération d'une paire de clés de chiffrement, à l'instar du chiffrement symétrique qui utilise une clé unique. La paire de clés se compose respectivement d'une clé privée D de déchiffrement, qui doit rester secrète, et d'une clé publique E de chiffrement, qui, comme son nom l'indique, peut être partagé de manière publique et dont le contenu peut être connu de tous. Cela résout ainsi le problème de transmission de la clé, comme identifié en *Def 5.2.1*.

Le fonctionnement et la sécurité de cette méthode repose sur le fait que bien que D est déterminé par E, il est impraticable de calculer D à partir de E. Ainsi toute personne en possession de E ne peut en déduire D et n'est donc pas en mesure de déchiffrer les messages chiffrés avec E.[4]

Une mise en pratique concrète de cette méthode ressemblerai à ceci :

- Alice veut pouvoir recevoir des messages de la part de Bob de manière sécurisée.
- Alice génère deux clés, une clé privée D, qu'elle garde pour elle, et une clé publique E, qu'elle envoie à Bob.
- Bob envoit un message M à Alice qu'il chiffre avec la clé publique E.
- Alice reçoit le message chiffré C puis le déchiffre avec la clé privée D et obtient le message M.
- Marie, qui espionnait la conversation depuis tout ce temps, a réussi à intercepter le message chiffré C qu'a envoyé Bob et la clé publique E qu'il a utilisé pour le chiffrer.
- Malgré toutes ses tentatives, Marie n'arrive pas à déchiffrer le message C de Bob.

Vient enfin le moment de poser l'interrogation qui est le cœur du raisonnement de cet article et qui est responsable de son existence :

Pour un sujet A en possession d'une clé de chiffrement publique E et d'un message C chiffré par E, pourquoi A n'est pas en capacité de déchiffrer C ?

Pour aborder cette problématique, nous utiliserons l'algorithme de chiffrement asymétrique Rivest-Shamir-Adleman (RSA) comme objet d'étude. Nous nous concentrerons spécifiquement sur les principes mathématiques régissant la génération de la paire de clés.

Après tout c'est vrai que dans les mathématiques que l'on a apprises à l'école, nous avons toujours plus ou moins inconsciemment compris qu'une fonction mathématique possédait toujours son inverse : on peut inverser une addition avec une soustraction, même chose pour la multiplication qui s'inverse avec une division.

Et, de manière plus générale, dans une équation à 3 valeurs, comme une addition, que l'on peut noter

[4] Diffie, W., & Hellman, M. E. (1976). Multiuser cryptographic techniques. In *Proceedings of the June 7-10, 1976, national computer conference and exposition* (pp. 109–112). essay, Association for Computing Machinery. (En Anglais.)

ainsi :

$$x + y = z$$

Il nous est possible de retrouver l'inconnue à partir du moment où l'on connaît les deux autres valeurs.

Ex (courant) :

$$1 + 2 = ?$$

Dans cet exemple, plus que courant, il suffit d'additionner les deux valeurs pour trouver l'inconnue, qui est notre troisième valeur.

Ex (moins courant) :

$$1 + ? = 3$$

Dans cet exemple, légèrement moins courant, on peut tout de même trouver l'inconnue, il suffit de soustraire 1 à 3 et l'on obtient 2, notre inconnue est donc égale à 2.

Marie se trouve dans cette situation :

$$M + c = C$$

Ici, c représente la clé de chiffrement, C le message chiffré et M le message en clair.

Marie possède donc bien deux valeurs sur trois : c et C, respectivement la clé de chiffrement et le message chiffré. Pourtant, il n'est pas possible de déchiffrer le message, même avec ces deux éléments.

Plongeons-nous alors un petit peu dans le monde des mathématiques pour expliquer le pourquoi du comment.

> Info : À noter que je vais détailler ici la génération de clés pour le protocole RSA, qui est le protocole de chiffrement asymétrique le plus connu et le plus utilisé, cependant retenez bien qu'il ne s'agit pas de l'unique protocole de chiffrement asymétrique.

Pour commencer il nous faut deux valeurs p et q, qui doivent respectivement être des nombres premiers, pour ceux qui auraient oublié, les nombres premiers ont cette particularité de ne se diviser que par 1 et eux-mêmes, par exemple le chiffre 7 n'est divisible que par 7, donc lui-même, et par 1.

Ensuite, il nous faut trouver la valeur n, qui est égal à p multiplié par q. Faisons un exemple : définissons p par 11 et q par 13. Si on multiplie donc ces deux valeurs, on obtient 143.

Ensuite, il nous faut trouver une variante de n, à savoir $\phi(n)$, qui est égal à $(p-1) \times (q-1)$, donc dans notre exemple :

$$(11-1) \times (13-1) = 10 \times 12 = 120$$

Ça va nous permettre de trouver e, qui est égal à un nombre premier quelconque à condition qu'il soit

différent de $\phi(n)$, on peut le noter ainsi :

$$e \in \mathbb{P} \neq \phi(n)$$

Pour notre exemple, 7 fonctionne bien.

Enfin, il nous manque d, qui va nous permettre de générer notre clé privée, qui est égal à l'inverse modulaire de $e \bmod \phi(n)$, que l'on peut noter

$$d \times e \equiv 1 (\mathrm{mod} \phi(n))$$

Je vous vois venir, Modulo c'est un dérivé de la division euclidienne, qui a la particularité de ne travailler qu'avec des entiers, ce qui a pour conséquence de nous donner un reste. Par exemple si on divise 45 par 12, on obtient un quotient de 3 et un reste de 9. Eh bien modulo, c'est le calcul de ce reste, en termes plus concret, c'est le calcul du reste d'une division euclidienne.

Pour finir, il nous manque notre valeur M, qui est égal au message qu'on souhaite chiffrer, par exemple, "Hello world". Problème, on ne sait pas encore calculer des lettres, il faut donc convertir ces lettres en nombres, on peut par exemple utiliser la table ASCII pour convertir "H" en 104.

Voilà maintenant, on a toutes nos valeurs pour générer notre clé publique via la formule suivante :

$$C = M^e \bmod n$$

où C représente notre message une fois chiffré.

Et notre clé privée :

$$M = C^d \bmod n$$

En utilisant notre exemple, cela donne, pour notre clé publique, la formule suivante :

$$104^7 \bmod 143$$

Ce qui donne pour résultat 91.

Bravo, vous venez de chiffrer un message !

7. Le protocole RSA, incassable ?

Alors comment se fait-il que l'on ne puisse pas récupérer le message en clair en étant en possession de la clé publique ? Ne peut-on pas simplement inverser la fonction de chiffrement ?

Eh bien… Non, parce que la fonction modulo est une fonction que l'on appelle "à sens unique", il n'est pas possible de l'inverser. L'unique moyen de trouver la valeur originale d'une fonction modulo à partir de son résultat, c'est de tester toutes les valeurs comprises entre 0 et ∞ en entrée jusqu'à ce qu'on obtienne C. Et, entre nous, c'est long, fastidieux et chiant.

Surtout quand on sait qu'il existe une solution plus simple et rapide : générer la clé privée à partir de la clé publique. Mais comment fait-on ? Eh bien il nous faut trouver p et q afin de pouvoir calculer $\phi(n)$ nécessaire à la génération de d (*qui est la composante principale de la clé privée*). Et on a de la chance, on sait que lors de la génération d'une clé RSA, p et q doivent obligatoirement être des nombres premiers, donc pour arriver à nos fins, il nous suffit de factoriser n. Ce qui consiste à tester la divisibilité de n avec tous les nombres premiers, jusqu'à ce que l'on obtienne un entier comme résultat.

Testons avec notre précédent exemple, où $n = 143$, on va donc tenter de le diviser par chaque nombre premier :

- Avec 2 on obtient 71.5 $(143 \div 2)$, ce qui n'est pas bon, vu qu'on obtient un résultat avec décimales (*donc pas un entier*).
- Avec 3 on obtient 47.667 $(143 \div 3)$, ce qui est également incorrect à cause de la décimale.
- Avec 5 on obtient 28.6 $(143 \div 5)$, ce qui est incorrect.
- Avec 7 on obtient 20.428 $(143 \div 7)$, ce qui est incorrect.
- Avec 11 on obtient 13 $(143 \div 11)$, ce qui est correct, on trouve bien un entier (*donc sans décimales*) ce qui signifie que 143 est divisible par 11 et que donc $p = 11$.

Cela nous permet de faire d'une pierre deux coups, puisqu'en divisant 143 par 11, on obtient q, donc ici $q = 13$, on sait donc que $p = 11$ et $q = 13$.

Il ne nous reste donc plus qu'à calculer $\phi(n)$ pour trouver d, et on peut générer la clé privée, puis déchiffrer le message.

8. Non, je ne suis pas un génie.

Et alors autant vous le dire tout de suite, non, je ne suis pas un génie de la cryptographie et non, je ne viens pas de casser l'un des protocoles de sécurité les plus utilisés au monde.

En fait, il y a un petit détail que j'ai oublié de vous préciser : la taille des nombres premiers choisis pour p et q. Pour rendre l'explication plus facile, j'ai volontairement choisi de petits nombres, en réalité, on utilise de bien plus grands nombres, par exemple :

$$5.49 \times 10^{999}$$

oui, c'est pas mal grand.

Et rappelez-vous, pour pouvoir retrouver p et q à partir de n, il nous faut diviser n par tous les nombres premiers jusqu'à obtenir un entier comme résultat, hors le temps nécessaire pour arriver à diviser n par de tels nombres, même pour un superordinateur, se compte en millions d'années.

Ce qui fait donc du protocole RSA un algorithme de chiffrement particulier, parce que théoriquement faillible, mais pas en pratique, de par la durée nécessaire pour factoriser n.

9. RSA, infaillible à tout jamais ?

Est-ce pour autant que nous ne parviendrons jamais à casser une clé RSA ?

Il y a deux situations plausibles : - La première, c'est que nos ordinateurs soient devenus suffisamment puissants pour pouvoir réussir à factoriser n dans des temps concevables pour l'être humain. Mais, soyons honnête, il y a peu de chances que cette situation voit le jour. Déjà parce que cela nécessiterait de faire un bond technologique gigantesque pour atteindre de telles puissances de calculs, ensuite, c'est facilement contournable, il suffit d'encore augmenter la taille des nombres premiers p et q pour croître de façon exponentielle le temps nécessaire pour factoriser n. - La deuxième, c'est l'algorithme de Shor, dont je réserve l'explication pour un article futur. Retenez qu'il s'agit d'un algorithme reposant sur la puissance de l'informatique quantique pour factoriser n dans des temps relativement restreint. Jusqu'à ce jour, l'algorithme de Shor n'a été mis à exécution qu'une fois, en 2001, par IBM, pour factoriser 15 en 3 et 5, bon autant vous dire que, ça, moi aussi, je sais le faire. Mais si une expérience de plus grande envergure voit le jour, elle pourrait alors sérieusement compromettre la sécurité de beaucoup de protocoles utilisant l'algorithme RSA, comme certains réseaux bancaires. Mais ne paniquez pas trop vite, l'attaquant doit tout de même être en possession d'un calculateur quantique pour mettre l'algorithme de Shor à exécution, et ça ne court pas les rues.

10. Conclusion :

Le protocole RSA à encore de beaux jours devant lui, parce que malgré le fait qu'il repose sur des principes mathématiques relativement simples, il offre une sécurité impressionnante grâce à la difficulté pratique de la factorisation de n. Cette complexité rend RSA théoriquement vulnérable, mais pratiquement incassable avec les moyens actuels.

Cependant, il est important de noter que la sécurité en cryptographie n'est jamais absolue. Les avancées technologiques, notamment dans le domaine de l'informatique quantique, pourraient un jour remettre en question la solidité de RSA. L'algorithme de Shor, par exemple, démontre qu'un ordinateur quantique suffisamment puissant pourrait factoriser n dans des temps suffisamment restreint pour pouvoir sérieusement menacer la sécurité des systèmes basés sur RSA.

Néanmoins, la communauté scientifique est consciente de ça et travaille déjà sur des algorithmes pouvant résister à l'informatique quantique. Cela porte d'ailleurs un nom : la cryptographie post-quantique.

Je vous encourage à approfondir vos connaissances en cryptographie et à consulter les sources fiables disponibles ci-dessous pour aller plus loin.