Introduction au Chiffrement : Théories et Pratiques du Chiffrement Symétrique et Asymétrique avec une Étude de Cas sur le Protocole RSA

Joshua Meffre

2 Juillet 2024

Résumé et Avertissement

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 NE CHERCHE PAS À L'ÊTRE.

Table des Matières

1. Quelques notions avant de commencer. **1**

1.1 Notions de Mathématiques. 1

1.2 Notions de Vocabulaire. 1

2. Contexte. **2**

3. La petite histoire du chiffrement. **3**

4. Le chiffre de César. **4**

5. Chiffrement asymétrique, ou la solution magique ? **6**

6. Le protocole RSA, incassable ? **9**

7. Non, je ne suis pas un génie. **10**

8. RSA, infaillible à tout jamais ? **11**

9. Conclusion : **12**

1. Quelques notions avant de commencer.

Cet article contient des formules mathématiques et un jargon qui peuvent paraître inhabituelles 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.

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 $a \mod b$

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.

2. Contexte.

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 de ce dernier 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 qu'il 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 faire me direz-vous ? Comme dit un peu 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. De plus, l'implémentation d'un serveur personnel chez soi permet une acquisition de connaissance dans le domaine informatique et dans la compréhension de celui-ci. Mais ce n'est pas un article sur l'auto-hébergement, donc je vais abréger. 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 une des étapes auquel il est complexe d'échapper, c'est SSH (Secure Shell). Un système bien pratique qui, en vulgarisant énormément, est un protocole de communication sécurisé, et qui, dans notre exemple, me permet de communiquer avec mon serveur via d'autres appareils et partout dans le monde. La fondation même de la sécurité de cet outil repose sur un principe : le chiffrement asymétrique ou le chiffrement à clé publique. Ce type de chiffrement possède deux avantages majeurs, que je décrirai plus tard dans cet article. Mais avant d'en comprendre tout les mystères, revenons un peu sur son histoire.

3. La petite histoire du chiffrement.

Les toutes premières traces de chiffrement remontent à 1900 av. J.-C. lorsqu'un scribe esquissa les hiéroglyphes racontant la vie de son seigneur. En effet, le système d'écriture des hiéroglyphes était assez inhabituel, bien qu'il ne s'agisse pas d'un chiffrement explicite, mais plutôt une volonté de rendre hommage de manière harmonieuse à son seigneur afin de lui conférer dignité et autorité, ainsi, on retrouvera écrit "In the year of Our Lord One thousand eight hundred and sixty three" (Dans l'année de notre seigneur mille-huit-cent-soixante-trois), au lieu de simplement écrire "1863". Bien qu'il ne s'agisse pas d'un type de chiffrement volontaire et explicite, ce texte incorpore un élément essentiel de la cryptographie : 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. Cette méthode ressemble à celle de George Bernard Shaw, avec laquelle on peut notamment 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 la consonne finale pour certains signes (par exemple si un signe représente la syllabe *ka*, le signe pouvait simplement être écrit *k*.) et que le même mot pouvait être écrit avec un signe différent. [2]

[1] David Kahn, The Codebreakers : A Comprehensive History of Secret Communication from Ancient Times to the Internet, Revised and Updated | Page 71 (En Anglais.)

[2] David Kahn, The Codebreakers : A Comprehensive History of Secret Communication from Ancient Times to the Internet, Revised and Updated | Page 75 (En Anglais.)

4. Le chiffre de César.

Mais l'exemple le plus connu, c'est le chiffre de César [3] qui, officiellement, porte le nom de chiffrement par décalage. Le concept, si vous ne l'avez pas encore 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"*. Simple à comprendre, me direz-vous, mais intéressons-nous tout de même au raisonnement de cette méthode de chiffrement, ce qui va nous permettre d'échauffer notre esprit logique et de mieux en comprendre la suite.

Pour chiffrer un message avec le chiffrement par décalage, on suit le raisonnement suivant :

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.

2. Définir une valeur n nécessaire pour décaler les lettres d'un message (dans le chiffre de César $n = 3$).

3. On peut ainsi encoder chaque lettre x d'un message via la formule suivante :

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

On peut ici découper l'opération en 2 parties :

1. $(x + n)$ où x représente notre lettre en clair (Def 1.2.1) et où n représente notre valeur de décalage, on additionne ces deux valeurs ensembles pour obtenir un décalage de notre lettre d'origine *(exemple : si $x = 2$ (donc la lettre D) et $n = 3$ le résultat est de 5 (donc la lettre G)*
2. $\bmod 26$ est présent pour s'assurer de ne pas dépasser au-delà de l'alphabet lors du décalage *(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, on utilise alors une fonction modulo (Def 1.1.3) pour s'assurer de l'affirmation suivante : $x + n < 26$. Si tel n'est pas le cas, alors l'opérateur modulo (Def 1.1.3) nous permet de recommencer au début de l'alphabet. Cela est dû au fait que dans l'opération $Z \bmod 26$ (où $Z = x + n$) le résultat donne toujours Z sauf si $Z \geq 26$, ce qui fait que, pour notre exemple plus haut où $x = 25$ et $n = 3$, on obtiendra 2 (donc la lettre D) et non 28 (qui ne correspond à aucune lettre de l'alphabet).)*

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

4. On peut ensuite déchiffrer le message en décodant chaque lettre en utilisant la valeur inversée de n qui est égal à $-n$, via la formule suivante :

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

Ce qu'il faut remarquer ici, c'est que la valeur n, c'est notre clé de chiffrement & de déchiffrement. C'est elle qui nous permet d'appliquer le décalage et par extension de chiffrer notre message. Et le fait que nous utilisions la même clé pour chiffrer et déchiffrer notre message porte un nom : le chiffrement symétrique, de par le fait que la même clé est utilisée dans les deux sens. Seulement cette méthode possède un semi-paradoxe : celui de la transmission de la clé. En effet dans le cas où vous envoyez un message chiffré par une méthode de chiffrement symétrique, il vous faudra envoyer la clé à votre interlocuteur afin qu'il puisse déchiffrer le message et vous répondre, et il faudra le faire sur un canal de communication suffisament sécurisé pour qu'elle ne soit pas interceptée, auquel cas l'action de chiffrer votre message perd son sens. Se présente donc le problème suivant : *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 via ce même canal sécurisé ?* Alors, oui il exuste bien des réponses à ce problème, vous avez surement plusieurs exemples qui vous viennent en tête, notamment celui de définir un canal sécurisé de manière unique pour la transmission de la clé (Ex : une rencontre en personne). Seulement cet exemple reste peu pratique et peut necessiter des moyens plus ou moins grands selon l'importance des informations à transmettre.

5. Chiffrement asymétrique, ou la solution magique ?

Pour palier à ce problème, un type de chiffrement différent est inventé dans les années 70[^4] : le chiffrement asymétrique, le concept est le suivant :

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

Cela est dû au fait que la clé publique, également appelée clé de chiffrement, ne permet pas de déchiffrer les messages qu'elle a chiffrés.

Et c'est à ce moment-là que vous vous demandez peut-être comment c'est possible ?

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 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 !

6. 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.

7. 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.

8. 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.

9. 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.