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

Mathis Lefebvre
lefebvremathis@proton.me

9 août 2024

Résumé

Cet article présente une exploration des principes fondamentaux de la cryptographie à un lecteur curieux, mais non spécialisé. Cet article distingue les mécanismes de chiffrement symétrique et asymétrique, en s'appuyant sur l'exemple de l'algorithme RSA pour illustrer le chiffrement à clé publique. Les notions mathématiques essentielles, une revue rapide de l'histoire du chiffrement et des applications pratiques permettrons au lecteur de saisir les mécanismes clés et l'importance de la place du chiffrement dans le monde actuel. L'objectif est d'offrir une introduction détaillée, mais accessible aux concepts clés de la cryptographie.

Table des matières

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 rassurer le lecteur, **il n'y a rien de compliqué**. Cette première partie existe justement pour l'aider à comprendre ou à se 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 : Un nombre premier($\mathbb{P}$) est un nombre qui est uniquement divisible par 1 et lui-même *(Exemple :* 7 *est divisible par* 1 *et par* 7*, donc lui-même).*

Def 1.1.3 : 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). *(Exemple :* $43/12$ *donne un quotient(q) de* 3 *et un reste(r) de* 7*)*

Def 1.1.4 : 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. *(Exemple :* $43 \bmod 12$ *donne un résultat de* 7*)*

Def 1.1.5 : 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 $a \bmod b$. Dans le cas où la vérification est fausse, alors a n'est pas congruent($\not\equiv$) à $c \bmod b$. *(Exemple :* 43 *est congruent à* 7 *modulo* 12*, on l'écrit ainsi* $43 \equiv 7 (\bmod 12)$*)*

Def 1.1.6 : Le plus grand commun diviseur (gcd) de deux nombres entiers non nuls est égal au plus grand entier qui les divise *(exemple : le* gcd *de* 40 *et* 25 *est de* 5 *puisque leurs diviseurs communs sont* 1 *et* 5*.)*

Def 1.1.7 : L'algèbre booléenne est une algèbre mathématique dans laquelle les valeurs calculées ne peuvent posséder que deux états : $faux, vrai \mid 0, 1 \mid eteint, allume$.

Def 1.1.8 : La factorisation consiste à représenter un entier n comme un produit de nombres premier. Cela est possible en testant la divisibilité de n avec tous les nombres premiers, jusqu'à obtention d'un entier naturel($\mathbb{N}$) comme résultat. *(Exemple : on peut factoriser* 15 *en les nombres premiers* 3 *et* 5*)*

Def 1.1.9 : 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 1.1.10 : La porte logique $XOR(\oplus)$ également appelé *ou exclusif* est un opérateur de l'algèbre booléenne dans lequel le résultat n'est 1 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 Mise en Contexte.

La cryptographie est un domaine indéniablement lié au militaire, surtout en France, qui est longtemps restée l'un des pays les plus restrictifs sur le sujet. Ce n'est qu'après la loi sur la réglementation des télécommunications (1996) [1] que la France commencera à adopter une vision différente des méthodes de chiffrement, qui était jusque ici classé comme arme de guerre de deuxième catégorie. Tout cela n'a pas aidé le grand public à démystifier la cryptographie qui, encore aujourd'hui, peut parfois être considéré comme une technologie réservée aux films d'espionnage ou au militaire. Pourtant, à l'ère d'un monde numérique toujours plus connecté, la cryptographie est présente de manière quasi permanente, bien qu'à des échelles différentes. C'est notamment cette technologie qui sécurise nos transactions bancaires [2], notre navigation sur internet [3] ou encore nos conversations, des plus professionnelles aux plus intimes, via des messageries sécurisées comme WhatsApp [4] ou Signal. Bien que l'existence de la cryptographie se soit démocratisé aux yeux du grand public, la compréhension, même élémentaire de celle-ci et de son impact sur le monde reste méconnue. Or, dans une société qui continue encore et encore de se numériser et qui devient de plus en plus dépendante des technologies de chiffrement sans pour autant les connaître, c'est une société qui se risque à débattre de la nécessité et de la place d'une technologie qu'elle ne comprend pas, et qui est ô combien importante.

J'ai pour ma part découvert cette discipline lors de la mise en place de mon serveur informatique personnel, lors de la configuration de SSH, un outil de communication sécurisé entre machines, qui repose sur la cryptographie asymétrique pour garantir la sécurité des communications et qui permet, dans mon cas, de communiquer avec mon serveur partout dans le monde et de manière sécurisée.

La naissance de cet article vient de mon questionnement sur le fonctionnement des méthodes de cryptographie asymétrique, qui ont la particularité de posséder une paire de clés *(dont le fonctionnement précis est détaillé dans la suite de cet article)*, dont une clé publique, qui peut être connue de tous et qui m'ont fait me demander : **jusqu'à quel point ces méthodes sont-elles infaillibles ?**

Ma volonté la plus humble est donc de réussir à fournir tant des informations précises et exactes que de partager mon émerveillement pour la beauté et la complexité mathématique qui composent ces technologies. C'est cette curiosité qui m'a conduit à une meilleure compréhension des enjeux que représentent les technologies pour notre monde, et que j'espère réussir à transmettre.

Pour cela, nous verrons donc une courte révision historique de la cryptographie afin de comprendre d'où vient cette idée étrange, bien qu'importante. Nous verrons ensuite une première méthode de chiffrement, le chiffrement symétrique, en s'appuyant sur deux objets d'études, le Chiffre de César et le One-Time-Pad (OTP). Nous étudierons notamment les faiblesses de ces méthodes qui ont conduit à la création d'une nouvelle : le chiffrement asymétrique, que nous illustrerons avec le protocole Rivet-Shamir-Adleman (RSA) et que nous mettrons à l'épreuve afin de vérifier sa robustesse.

1. *Loi n° 96-659 du 26 juillet 1996 de réglementation des télécommunications - Légifrance.* (1996, 26 Juillet). Legifrance.gouv.fr.
2. Thomas Genet. (2008, 21 Février). *Le protocole cryptographique de paiement par carte bancaire.* Interstices.
3. *Qu'est-ce que le TLS (Transport Layer Security) ?* (n.d.). Cloudflare.
4. *À propos du chiffrement de bout en bout.* (n.d.). WhatsApp.

3 La petite Histoire du Chiffrement.

3.1 Cryptographie Involontaire.

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 est volontairement complexe. Il ne faut pas s'y méprendre : il s'agit en fait 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. Quand bien même, il est ainsi possible de retrouver des phrases telles que "Dans l'année de notre seigneur mille-huit-cent-soixante-trois", plutôt que d'avoir simplement écrit "En 1863".

Bien qu'il ne s'agisse pas d'un type de chiffrement volontaire et explicite, ce texte est partiellement responsable de la naissance de la cryptographie puisqu'il incorpore un élément essentiel de celle-ci : une transformation délibérée de l'écriture. [5]

3.2 Recette de Poterie.

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 compliquée.

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*).

Ajoutons 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 [6].

Ainsi est née la cryptographie.

5. David Kahn. (1968). The Codebreakers. In *Internet Archive* (p. 71). The Macmillan Company. (En Anglais.)
6. David Kahn. (1968). The Codebreakers. In *Internet Archive* (p. 75). The Macmillan Company. (En Anglais.)

3.3 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[7] , 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, ce qui permettra au lecteur de s'habituer à la lecture de formule mathématiques, présentes tout au long de cet article :

Def 3.3.1 : 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 3.3.2 : Définir une valeur n nécessaire au décalage des lettres d'un message (dans le chiffre de César $n = 3$).

Def 3.3.3 : 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 Z qui a 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. mod 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 *(cf. Def 1.1.3)* est alors utilisé pour garantir que $x + n < 26$. Si tel n'est pas le cas, alors l'opération mod 26 permet de repositionner la valeur de décalage au début de l'alphabet. Cela est dû au fait que dans l'opération Z mod 26 (où $Z = x + n$) le résultat vaut toujours Z tant que $Z < 26$. Si $Z \geq 26$ l'opération modulo ajuste Z pour rester dans les limites de l'alphabet.

   Ainsi, dans l'exemple donné, où $x = 25$ et $n = 3$, le résultat de $x + n \mod 26$ sera 2 (donc la lettre C) et non 28, qui ne correspond à aucune lettre de l'alphabet.

Def 3.3.4 : 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.

7. Nom donné en rapport au fait que Jules César utilisait ce type de chiffrement pour ses communications secrètes : Halsall, P. (2019). *Internet History Sourcebooks*. Fordham.edu. (En Anglais.)

Sans surprise, un telle méthode de chiffrement connaît des failles, la première, c'est l'attaque par force brute, en effet, le nombre de clés ne se limite qu'à 26 ce qui, même pour un être humain, ne prend que peu de temps à tester toutes les clés n possible une par une, jusqu'à obtention d'un résultat ressemblant à un message. Une autre méthode de cryptanalyse, appelée attaque par analyse fréquentielle, consiste à prendre en compte la lettre la plus courante de la langue dans laquelle le message a été écrit (en français, il s'agit de la lettre E) et de voir quelle lettre revient le plus dans le message chiffré. Par exemple, dans le message *"SL JOPMMYLTLUA LZA CYHPTLUA BUL KPZJPWSPUL PUALYLZZHUAL!"* La lettre avec le plus d'itération est la lettre L, or, on sait que le message d'origine est écrit en français, on peut donc supposer que L représente E. La différence de position entre les deux lettres est de sept, ainsi, si l'on décale le message de sept lettres en arrière, on obtient *"LE CHIFFREMENT EST VRAIMENT UNE DISCIPLINE INTÉRESSANTE!"*.

Si tant est qu'à l'époque de l'Empire Romain, cela ne posait pas vraiment de problèmes dû à un taux d'alphabétisation relativement bas, il se trouve, qu'avec le temps, les méthodes de cryptanalyse se sont considérablement améliorées, et ont fini par imposer la nécessité de créer des algorithmes de chiffrement symétrique bien plus robustes.

C'est ainsi que, bien plus tard dans l'histoire, au début de l'année 1883, Auguste Kerckhoffs, un cryptologue militaire néerlandais énonce le principe de Kerckhoffs [8] , qui établit plusieurs règles afin de garantir la sécurité d'un système de chiffrement. Mais celle à retenir est la suivante : *(en parlant d'un système de chiffrement) "Il faut qu'il n'exige pas le secret, et qu'il puisse sans inconvénient tomber entre les mains de l'ennemi."* Principe selon lequel la sécurité d'un système de chiffrement ne doit pas reposer sur la méconnaissance par l'adversaire de l'algorithme utilisé, mais uniquement sur sa méconnaissance de la clé. Ainsi, l'algorithme doit être assez sophistiqué pour pouvoir être aux mains d'un attaquant sans que celui-ci puisse être en capacité de déchiffrer des messages chiffrés par ce dit algorithme. C'est cette règle en particulier qui régira la conception de système cryptographique symétriques plus robustes.

8. Kerckhoffs, A. (1883). LA CRYPTOGRAPHIE MILITAIRE. *Journal Des Sciences Militaires*, IX, 161–191.

4 Chiffrement Symétrique.

4.1 One Time Pad.

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, nous nous concentrerons 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 4.1.1 : La clé de chiffrement doit être générée de manière véritablement aléatoire, sans motifs prédictibles.

Def 4.1.2 : La longueur de la clé doit être égale à celle du message à chiffrer.

Def 4.1.3 : 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 1.1.10)* 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 1.1.10*). 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 chaque bit du message en le calculant avec le bit de la clé situé à la même position. Par exemple, on peut calculer le premier bit du message et le premier bit de la clé avec 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 également 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.

4.2 Problème de Transmission.

Si le OTP 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 4.2.1 : *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 4.1.3*, 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 4.2.1*. De plus, une telle rencontre présente rapidement des limitations pratiques. Elle peut nécessiter des moyens considérables selon la sensibilité des informations à échanger et peut même se révéler impraticable dans les contextes de communications longue distance.

5 Chiffrement Asymétrique.

Ainsi est introduit en 1976, par Whitfield Dillie et Martin E. Hellman, le concept de cryptographie à clé publique, également appelé cryptographie asymétrique, qui sera le terme utilisé dans la suite de l'article. 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 4.2.1*.

Le fonctionnement et la sécurité de cette méthode repose sur le fait que bien que D est déterminé par E, il est en théorie 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.[9]

Une mise en pratique concrète de cette méthode ressemblerait 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 envoi 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 :

Def 5.0.1 : *Pour un individu 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.

9. Diffie, W., & Hellman, M. E. (1976). Multiuser cryptographic techniques. *Proceedings of the June 7-10, 1976, National Computer Conference and Exposition on - AFIPS '76.* (En Anglais.)

5.1 Algorithme RSA :

Dans un premier temps, afin de pouvoir établir les deux formules qui constituent les clés de chiffrement, il est nécessaire de définir toutes les valeurs qui les composent :

Def 5.1.1 : Définir p et q, qui doivent être de grands nombres premiers générés de manière véritablement aléatoire, sans motif de prédictibilité.

$$p, q \in \mathbb{P}$$

Def 5.1.2 : Calculer n, produit de p et q :

$$n = p \times q$$

Def 5.1.3 : Calculer $\phi(n)$, égal au totient de n tel que :

$$\phi(n) = (p-1) \times (q-1)$$

Def 5.1.4 : Définir e tel que :

$$1 < e < \phi(n), \quad \gcd(e, \phi(n)) = 1$$

Def 5.1.5 : Calculer d, égal à l'inverse modulaire de e mod $\phi(n)$ tel que :

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

Une fois toutes ces valeurs définies, il est possible de chiffrer des messages avec la paire de clés via les deux formules suivante :

Def 5.1.6 : Pour la clé publique E :

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

Def 5.1.7 : Pour la clé privée D :

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

Dans une volonté de clarification, nous allons mettre l'algorithme RSA à exécution avec un exemple R concret :

Pour une valeur $p = 11$ et $q = 13$, n est donc égal à 143 en appliquant la *Def 5.1.2* puisque :

$$11 \times 13 = 143$$

En appliquant la *Def 5.1.3*, $\phi(n) = 120$ puisque :

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

En appliquant la *Def 5.1.4*, $e = 7$ puisque :

$$1 < 7 < 120, \quad \gcd(7, 120) = 1$$

En appliquant la *Def 5.1.5*, $d = 103$ puisque :

$$103 \times 7 = 721$$
$$721 \equiv 1 \pmod{120}$$

Ainsi, pour une valeur $M = 104$ *(correspondant à la lettre h en ASCII)*, il est possible d'obtenir une valeur chiffrée C avec la clé publique E via la formule suivante :

$$C = 104^7 \bmod 143 = 91$$

Donc $C = 91$, qu'il est possible de déchiffrer avec la clé privée D via la formule suivante et d'obtenir un résultat T :

$$T = 91^{103} \bmod 143 = 104$$

Dans notre exemple, $T = M$, il est donc possible d'affirmer que le chiffrement a été effectif, puisque le résultat du déchiffrement T est bien égal à la valeur en clair initiale M.

Nous allons maintenant donc analyser la situation décrite dans la problématique *Def 5.0.1* du point de vue d'un attaquant et explorer les méthodes possibles et disponibles de ce point de vue afin de comprendre pourquoi il est impossible de déterminer M à partir des valeurs E et C.

5.2 Attaque par Inversion :

La première méthode qu'il est possible de mettre en place, c'est celle de déduire M en tentant d'inverser la formule de chiffrement *Def 5.1.6*. Le problème rencontré avec cette méthode, c'est que l'opérateur modulo (mod) ne permet pas une telle chose. En effet, Dans une opération O tel que :

$$x \bmod y = w$$

où y et w sont des valeurs connues et x une inconnue, l'une des méthodes, qui sera nommée méthode F, consiste à retrouver x en testant toutes les valeurs comprises entre 0 et ∞ jusqu'à obtention d'une valeur qui satisfait O. Une méthode qui, en plus d'être longue et fastidieuse, ne permet pas d'obtenir un résultat satisfaisant. En effet, si elle est mise à exécution sur l'exemple R décrit plus haut, nous avons ainsi connaissance, en notre position d'attaquant, de la formule de chiffrement E tel que :

$$x^7 \bmod 143 = 91$$

Cette formule de chiffrement peut ainsi être satisfaite avec $x = 104$, 247, 390, 533, 676, 819 ou encore 962 si l'on ne se limite qu'aux valeurs comprises entre 0 et 1000. Une grande quantité de valeurs peuvent ainsi satisfaire la formule *Def 5.1.6*, ce qui rend toute volonté de retrouver la valeur en clair initiale complexe. Une version plus avancée, rapide et logique de cette technique consiste à partir du constat que le message en clair a été encodé avec une norme d'encodage de caractère comme l'ASCII et donc de ne tester que les valeurs comprises dans cette norme d'encodage. Donc pour l'ASCII, seules les valeurs comprises entre 0 et 127, seront testées. Ce qui, dans l'exemple R, fonctionne puisque l'unique résultat obtenu est bel et bien égal à 104. Bien que cette technique puisse sembler efficace pour le moment, il est crucial de mentionner un aspect fondamental souvent négligé : lors du chiffrement d'un message avec RSA, un algorithme de remplissage (padding) est généralement appliqué avant le chiffrement proprement dit. Un algorithme couramment utilisé pour cela est l'OAEP (Optimal Asymmetric Encryption Padding). Bien qu'une description exhaustive de ce processus ne soit pas fournie dans cet article, il est important de noter que l'objectif de ces algorithmes est d'ajouter une couche d'aléatoire à la valeur en clair initiale M et d'augmenter sa longueur avec des données, elles aussi, aléatoires. Cela vise à rendre inefficaces les attaques comme exposées précédemment.

5.3 Attaque par Factorisation :

La deuxième méthode, plus connue, et qui sera nommée Méthode G, consiste à retrouver la valeur d à partir de la clé de chiffrement E. Pour rappel, d est égal à l'inverse modulaire de e mod $\phi(n)$, soit :

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

(comme décrit en *Def 5.1.5*).

Seulement, un attaquant comme décrit dans la problématique *Def 5.0.1*, ne dispose pas de la valeur $\phi(n)$, ni des facteurs premiers p et q qui sont nécessaires au calcul de $\phi(n)$. Par conséquent, cette méthode consiste à déterminer p et q à partir de n. Cela est possible parce qu'un attaquant sait que p et q sont nécessairement des nombres premiers. Ainsi, il peut tenter de factoriser *(Def 1.1.7)* n en tant que produit de p et q. Une fois p et q déterminés, il suffit de calculer $\phi(n)$ comme décrit en *Def 5.1.3*. Une application pratique basée sur l'exemple R est illustrée ainsi :

$$n = 143$$

$$\begin{aligned} 143 \bmod 2 &= 1 \\ 143 \bmod 3 &= 2 \\ 143 \bmod 5 &= 3 \\ 143 \bmod 7 &= 3 \\ 143 \bmod 11 &= 0 \\ 143 \bmod 13 &= 0 \end{aligned}$$

Dans ce cas, $p = 11$ et $q = 13$ puisqu'une opération n mod $\mathbb{P}$ donne 0 pour ces deux valeurs, indiquant que n est divisible par ces nombres premiers.

Bien que cette méthode puisse sembler efficace, elle néglige un aspect crucial : la longueur de la clé. En effet, dans l'exemple R donné, les valeurs des nombres premiers $p = 11$ et $q = 13$ permettent une factorisation rapide de n. Toutefois, si l'on augmente significativement ces valeurs, le temps nécessaire pour factoriser n augmente de manière exponentielle. Par exemple, pour une clé d'une longueur de 2048 bits, on estime que le temps requis pour effectuer une telle factorisation, même avec les super-ordinateurs les plus puissants actuellement disponibles, s'étendrait sur plusieurs milliards d'années. Cette estimation se base sur les résultats formulés dans une expérience de 2009 faite par une équipe de spécialistes en informatique, en cryptographie et en mathématiques. [10]

Expérience qui a consisté à casser une clé RSA de 768 bits, avec succès et qui aura nécessitée environ deux années. On y apprend notamment que plus de 10^{20} opérations ont été nécessaires à la factorisation d'une valeur n. Ainsi, si l'on utilise un super-ordinateur comme le **Frontier** [11] , capable de $1,1 \times 10^{18}$ opérations par secondes et que l'on considère que 10^{20} opérations ont été nécessaires, il est possible de déterminer le temps que mettrait **Frontier** pour casser une clé RSA de 768 bits :

$$\frac{10^{20}}{1,1 \times 10^{18}} \approx 91$$

Soit environ 91 secondes.

10. Kleinjung, T., Aoki, K., Franke, J., Lenstra, A., Thomé, E., Bos, J., Gaudry, P., Kruppa, A., Montgomery, P., Osvik, D., Te Riele, H., Timofeev, A., & Zimmermann, P. (2010). *Factorization of a 768-bit RSA modulus* (p. 14). (En Anglais.)
11. Super-ordinateur le plus puissant au monde actuellement : https://www.olcf.ornl.gov/frontier/ (En Anglais)

Ainsi, il est possible de considérer une clé RSA de 2048 bits comme étant 10^{16} fois plus complexe à calculer qu'une clé de 768 bits en s'appuyant sur la complexité de l'algorithme General Number Field Sieve (GNFS) comme suit :

Formule :
$L(n) = \exp\left(\left(\frac{64}{9}\right)^{1/3} (\ln n)^{\frac{1}{3}} (\log \ln n)^{\frac{2}{3}}\right)$

Application de la formule sur les valeurs 2048 **et** 768 **:**
$L(2^{2048}) = \exp\left(\left(\frac{64}{9}\right)^{1/3} \left(\ln 2^{2048}\right)^{1/3} \left(\log \ln 2^{2048}\right)^{2/3}\right)$
$L(2^{768}) = \exp\left(\left(\frac{64}{9}\right)^{1/3} \left(\ln 2^{768}\right)^{1/3} \left(\log \ln 2^{768}\right)^{2/3}\right)$

Calcul de $(\ln 2^{2048})$ **et** $(\ln 2^{768})$ **en utilisant la propriété des logarithmes telle que** $\ln(a^b) = b \ln(a)$ **:**
$\ln 2^{2048} = 2048 \times \ln 2 \approx 1419.565$
$\ln 2^{768} = 768 \times \ln 2 \approx 532.337$

Calcul de $(\log \ln 2^{2048})$ **et** $(\log \ln 2^{768})$ **:**
$\log \ln 2^{2048} \approx \log 1419.565 \approx 3.152$
$\log \ln 2^{768} \approx \log 532.337 \approx 2.726$

Ainsi :
$L(2^{2048}) \approx \exp\left(\left(\frac{64}{9}\right)^{1/3} (1419.565)^{1/3} (3.152)^{2/3}\right)$
$L(2^{768}) \approx \exp\left(\left(\frac{64}{9}\right)^{1/3} (532.337)^{1/3} (2.726)^{2/3}\right)$

Comparaison de $L(2^{2048})$ **et** $L(2^{768})$ **telle que :**
$$\frac{L(2^{2048})}{L(2^{768})} \approx \frac{\exp\left(\left(\frac{64}{9}\right)^{1/3} (1419.565)^{1/3} (3.152)^{2/3}\right)}{\exp\left(\left(\frac{64}{9}\right)^{1/3} (532.337)^{1/3} (2.726)^{2/3}\right)}$$

Utilisation de la propriété des exponentielles telle que :
$\frac{\exp(a)}{\exp(b)} = \exp(a - b)$

Ainsi :
$\frac{L(2^{2048})}{L(2^{768})} \approx \exp\left(\left(\frac{64}{9}\right)^{1/3} \left((1419.565)^{1/3} (3.152)^{2/3} - (532.337)^{1/3} (2.726)^{2/3}\right)\right)$

Approximation des termes telle que :
$\left(\frac{64}{9}\right)^{1/3} \approx 1.923$
$(1419.565)^{1/3} \approx 11.239$
$(3.152)^{2/3} \approx 2.150$
$(532.337)^{1/3} \approx 532.337$
$(2.726)^{2/3} \approx 1.952$

Ainsi :
$$\frac{L(2^{2048})}{L(2^{768})} \approx \exp\left(1.923\left(11.239 \times 2.150 - 8.105 \times 1.952\right)\right)$$

Approximation des termes telle que :
$11.239 \times 2.150 \approx 24.164$
$8.105 \times 1.952 \approx 15.821$

Ainsi :
$$\frac{L(2^{2048})}{L(2^{768})} \approx \exp(1.923(24.164 - 15.821))$$

Calcul de la différence tel que :
$24.164 - 15.821 = 8.343$

Ainsi :
$$\frac{L(2^{2048})}{L(2^{768})} \approx \exp(1.923 \times 8.343) \approx \exp(16.044) \approx 10^{16.044} \approx 10^{16}$$

Donc, selon cette grossière approximation, le temps nécessaire à **Frontier** pour factoriser n en considérant une clé de 2048 bits est donc approximativement égal à : 91×10^{16} secondes. Soit environ vingt-huit milliards huit cent millions d'années, ce qui rend toute mise en pratique visant à casser une clé RSA via la factorisation de n vaines.

6 L'algorithme RSA est il incassable ?

Comme discuté précédemment, la robustesse de RSA, ainsi que des systèmes de chiffrement asymétrique en général, repose sur le temps nécessaire pour casser les algorithmes de chiffrement et de déchiffrement qui constituent les clés. Cela confère un statut particulier à ces algorithmes, car ils sont théoriquement vulnérables, mais pratiquement incassables dans un délai raisonnable avec les moyens actuels.
Cela signifie-t-il pour autant que nous ne parviendrons jamais à casser une clé RSA dans des temps acceptables pour l'être humain ? Deux scénarios plausibles méritent d'être envisagés :

- Premièrement, il se pourrait que nos ordinateurs deviennent suffisamment puissants pour factoriser n dans des temps considérés comme raisonnables pour l'être humain. Un exemple concret de cela peut être trouvé en *sous-section 5.3 où l'on peut comparer la capacité de calcul entre l'expérience de 2009 et celle du super-ordinateur* ***frontier***. Cependant, cela reste peu probable. Un tel scénario nécessiterait un bond technologique majeur pour atteindre des capacités de calcul astronomiques. De plus, ce défi pourrait être facilement surmonté en augmentant simplement la taille des nombres premiers p et q, rendant ainsi le temps nécessaire à la factorisation exponentiellement plus long.
- Deuxièmement, l'informatique quantique fait des progrès considérables à ne pas prendre à la légère. En effet, il existe l'algorithme de Shor, un algorithme reposant sur les principes de l'informatique quantique, capable de factoriser n en des temps relativement courts. Jusqu'à présent, l'algorithme de Shor a notamment été démontré en 2001, par IBM [12], pour factoriser 15 en 3 et 5. Bien que cette démonstration soit simpliste, une mise en œuvre à plus grande échelle pourrait sérieusement menacer la sécurité de nombreux protocoles utilisant RSA ou d'autres systèmes de chiffrement asymétrique. Il est d'ailleurs estimé qu'un ordinateur quantique équipé de 20 millions de qbits (bits quantiques) pourrait, en théorie, casser une clé RSA de 2048 bits en un temps record de huit heures [13].

12. Vandersypen, L. M. K., Steffen, M., Breyta, G., Yannoni, C. S., Sherwood, M. H., & Chuang, I. L. (2001). Experimental realization of Shor's quantum factoring algorithm using nuclear magnetic resonance. *Nature*, *414*(6866), 883–887. (En Anglais)
13. Gidney, C., & Ekerå, M. (2021). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum*, *5*, 433. (En Anglais)

7 Conclusion.

Nous avons pu voir que, depuis son invention, la cryptographie et le chiffrement n'ont cessé d'évoluer, de s'améliorer afin de proposer des solutions toujours plus complètes, robustes et pratiques. Cette évolution est marquée par une recherche constante de nouvelles techniques pour protéger l'information contre des adversaires possédant des méthodes de plus en plus sophistiquées, partant de la simple volonté de protéger une recette de poterie et allant jusqu'à la sécurisation des systèmes bancaires, de la navigation sur internet, de la transmission de donnée, etc. Les systèmes de chiffrement asymétrique sont aujourd'hui devenus indispensables dans le domaine de la cryptographie et sont par conséquent régulièrement mis à l'épreuve afin de tester leur robustesse et de les rendre toujours plus solide avant que le pire n'arrive.

Nous avons également pu répondre à la problématique *Def 5.0.1* en expliquant que l'individu A ne peut déduire C à partir de E qu'en utilisant des attaques dont deux ont été démontrés et démontés dans cet article. Ces méthodes ont montré leurs limites face à des algorithmes prévus pour les contrer, comme le OAEP, qui montrent l'importance de régulièrement tester les algorithmes de chiffrement, ou ont révélé leur impraticabilité face à des clés suffisamment longues. Par exemple, pour des clés de 2048 bits, le temps de calcul nécessaire pour les casser dépasse largement l'échelle de temps humainement raisonnable, même avec les super-ordinateurs les plus avancés comme le **Frontier**. Les résultats des expériences de factorisation menées en 2009 sur une clé de 768 bits renforcent cette affirmation, indiquant que le temps requis pour une clé de 2048 bits serait de plusieurs milliards d'années.

Enfin, une conclusion définitive sur un tel sujet reste complexe, l'évolution des méthodes de cryptanalyses et de l'informatique quantique laisse à penser que tôt ou tard, les systèmes cryptographiques actuels finiront par céder et montrer leurs faiblesses. Est-ce pour autant que l'avenir de la cryptographie est alors condamné ? Non, l'être humain se fait la course à lui même, car en cherchant à améliorer nos algorithmes de chiffrement, nous ne faisons que repousser les limites de notre compréhension des mathématiques, et par conséquent, de notre propre capacité de raisonnement et de réflexion. Des exemples comme la cryptographie post-quantique, qui est branche de la cryptographie réservée à la mise en place d'algorithmes résistant à l'informatique quantique, nous montrent bien que nous sommes en capacité de repousser nos propres limites.

8 Lectures Recommandées.

- Kahn, D. (1996). *The Codebreakers*. Simon and Schuster. (En Anglais.)
- ANSSI. (2020). *Guide des mécanismes cryptographiques règles et recommandations concernant le choix et le dimensionnement des mécanismes cryptographiques.* ANSSI.
- Diffie, W., & Hellman, M. E. (1976). *Multiuser cryptographic techniques. Proceedings of the June 7-10, 1976, National Computer Conference and Exposition on - AFIPS '76.* (En Anglais.)
- Rauzy, P. (n.d.). *Introduction à la sécurité.* Université Paris 8.
- Rivest, R., Shamir, A., & Adleman, L. (1978). *Programming Techniques A Method for Obtaining Digital Signatures and Public- Key Cryptosystems.*