Une vision pratique et simplifiée de la crypto à clefs publiques

En vue de la préparation à l'usage des certificats et des communications chiffrées sur Internet

---

Patrick Legand
Consultant SSI
contactPL@patrick-legand.com
http://blog.patrick-legand.com

- Novembre 2004 -

Petites connaissances mathématiques préalables

Le bonheur des congruences

Connaissez vous l'expression **a ≡ b (mod n) ?**

Si cela ne vous dit rien, c'est normal ! A moins d'être plongé toute la journée dans des problèmes de cryptologie, il est rare d'avoir à affronter une telle expression.

En revanche, il est possible que, pendant vos cours de mathématiques, vous ayez eu l'occasion d'aborder l'arithmétique modulaire. Hélas, l'expérience montre que le calcul des congruences laisse généralement de mauvais souvenirs, et que les étudiants s'empressent bien vite d'oublier les délices de cette branche mystérieuse de la théorie des nombres.

Malheureusement, la cryptologie fait un usage intensif de l'arithmétique modulaire… et pour bien comprendre le fonctionnement d'algorithmes à clefs publiques tels que RSA, il n'est pas inutile d'acquérir quelques rudiments dans ce domaine.

Que signifie donc a ≡ b (mod n) ? En fait, il s'agit d'une chose très simple : lorsque vous divisez un entier a par un entier n, vous obtenez un reste dont la valeur est b. On dit alors que **a est congru à b modulo n**. L'entier n est appelé module ou modulus (… mais pas moldu !).

Ainsi, quand on divise 21 par 8, l'école nous a appris : " il y va 2 fois et il reste 5 ". Pour reprendre l'expression énoncée au paragraphe précédent, on peut donc affirmer que 21 est congru à 5 modulo 8. Vous conviendrez que jusque là, nous ne dépassons pas le niveau du cours moyen première année.

Nous constatons déjà que tous les nombres inférieurs à 8 sont congrus à eux-mêmes modulo 8 ; par exemple, 3 est égal à 0 fois 8, et il reste 3. Nous pourrions faire le même calcul avec 0, 1, 2... jusqu'à 7, et nous obtiendrions :

```
0 ≡ 0 (mod 8)        4 ≡ 4 (mod 8)
1 ≡ 1 (mod 8)        5 ≡ 5 (mod 8)
2 ≡ 2 (mod 8)        6 ≡ 6 (mod 8)
3 ≡ 3 (mod 8)        7 ≡ 7 (mod 8)
```

Que se passe-t-il pour 8 ? 8 est tout simplement est égal à 1 fois 8, et il reste 0. Donc 8 ≡ 0 (mod 8). Et pour 9 ? 9 ≡ 1 (mod 8). Et ainsi de suite. Si l'on dressait la même séquence que précédemment avec les nombres compris entre 8 et 15, nous obtiendrions :

```
 8 ≡ 0 (mod 8)       12 ≡ 4 (mod 8)
 9 ≡ 1 (mod 8)       13 ≡ 5 (mod 8)
10 ≡ 2 (mod 8)       17 ≡ 6 (mod 8)
11 ≡ 3 (mod 8)       15 ≡ 7 (mod 8)
```

Vous aurez sans doute remarqué le caractère assez répétitif de cette arithmétique. Abordons maintenant quelques opérations plus complexes, ces dernières étant toujours effectuées modulo 8 ; par exemple :

$6+7 \equiv \mathbf{5} \pmod 8$;
$7*9 \equiv \mathbf{7} \pmod 8$;
$43*257 \equiv \mathbf{2} \pmod 8$ car $43*257 = 10\,794 = 1\,349*8$ plus **2** ;
$13^7 \equiv \mathbf{5} \pmod 8$ car $13^7 = 62\,748\,517 = 7\,843\,564 *8$ plus **5** ;

Vous constaterez facilement que, quelles que soient la taille des nombres et la nature de l'opération effectuée modulo 8, le résultat est toujours compris entre 0 et 7. Si la valeur du module était de 587, le résultat de toutes les opérations effectuées modulo 587 serait compris entre 0 et 586.

L'arithmétique modulaire définit donc **un espace fini**, à l'intérieur duquel nous sommes condamnés à évoluer… impossible de s'échapper. Quels que soient les stratagèmes compliqués, les calculs savants, les nombres importants, point de salut, le module nous ramènera brutalement, quoi qu'il arrive, à l'intérieur de notre espace fini…

A quoi bon lutter ? Il faut se faire une raison : à partir de maintenant, nous allons " oublier " la plus grande part de chaque nombre mis en jeu dans nos opérations, pour nous intéresser à une toute petite valeur, insignifiante, tout à fait résiduelle, celle se situant à l'intérieur de notre espace fini, celle que nous appellerons : **le résidu**.

Il s'agit peut-être de l'aspect le plus déroutant de l'arithmétique modulaire. Certes, il y a bien cette petite gymnastique intellectuelle de la division et du reste, mais nous allons très vite l'oublier pour nous concentrer sur la notion de résidu. Ainsi, à partir de 62 748 517 nous ne conservons qu'un résidu d'information, le 5 ! Quel est l'intérêt de perdre une telle quantité de données, sachant, de surcroît, qu'il existe infinité d'entiers congrus à 5 modulo 8 ?

Mon premier élément de réponse est une question : avons nous vraiment besoin de toute l'information ? Le 5 ne suffirait-il pas ? Les choses ne seraient-elles pas un perpétuel recommencement ?

Dans la vie courante, chacun pratique quotidiennement cette notion de congruence sans s'en apercevoir. Lorsque, par exemple, vous décollez à 22 heures pour un vol long courrier en direction du grand Sud-Est : si la durée du vol est de 10 h 30, vous vous dites en toute logique que vers 8 h 30 le lendemain matin vous avez de bonnes chances de contempler les magnifiques paysages de la Réunion. Jamais vous n'auriez eu l'idée d'évoquer un atterrissage aux alentours de 32 h 30. Vous vous êtes dit : " il est 22 h, pour aller à 24 h il y a 2 h, que je retire à 10 h 30, ce qui fait 8 h 30 demain matin ". Ou alors vous avez réellement calculé 32h30, auxquels vous avez retranché 24 h. Vous avez intuitivement effectué une réduction de 32 h 30 modulo 24 et retenu le résidu, seule information dont vous ayez réellement besoin.

Comme vous pouvez le constater, nous sommes tous, finalement, très familiers de l'arithmétique modulaire !

Mais à quoi tout cela peut-il servir ?

Poursuivons l'exemple très simple de l'horloge, et supposons que, pour simplifier, une seule aiguille donne à la fois l'heure, les minutes et les secondes. Supposons, en outre, que l'aiguille soit montée folle sur son axe, c'est-à-dire qu'elle soit parfaitement libre de tourner dans un sens, disons dans le sens des aiguilles d'une montre, et que son déplacement ne soit provoqué que par l'impulsion d'une force extérieure. Enfin, supposons qu'un mécanisme extrêmement fiable l'empêche de rebrousser chemin ; tout déplacement de l'aiguille est donc définitif : il s'agit d'une aiguille " **à sens unique** ". Nous évoluons donc à l'intérieur d'un espace fini compris entre 0 et 23 h 59 mn et 59 sec.

Actuellement, notre aiguille marque 16 h 43 mn et 22 sec, … et c'est l'heure du rendez-vous que souhaite me fixer Alice, afin que nous échangions les informations secrètes, qui, réunies, nous permettront de trouver le trésor. Or, il ne faut surtout pas que l'équipe adverse sache à quelle heure nous allons nous rencontrer, car elle chercherait certainement à nous dérober nos documents pour accéder au trésor avant nous : cette information doit absolument rester secrète.

Aussi, comment Alice va-t-elle procéder pour me faire parvenir cette information ? Elle va avoir recours à un artifice très simple : rappelez vous que l'aiguille de l'horloge est libre de tourner sur son axe, autant qu'elle le veut, dans le sens des aiguilles d'une montre. Alice va donc communiquer à cette aiguille une forte impulsion, voire même pourquoi pas une impulsion considérable. Attention toutefois, la force utilisée par Alice n'est pas choisie au hasard. C'est moi, Bernard, grâce aux merveilleux moyens de communication de l'époque moderne, qui lui ai indiqué sa valeur. A l'issue de sa course folle, l'aiguille s'arrête en face d'un point marquant 4 h 32 mn et 17 sec. Elle peut avoir fait plusieurs centaines de tours. Quelle évidence, et, pourtant, comme c'est extraordinaire ! Après avoir parcouru un tel voyage, avoir follement enchaîné les tours, notre aiguille vient s'arrêter en face d'une valeur tout aussi banale que la précédente, une simple valeur située dans notre espace fini. Car c'est le résultat de cette grande aventure qui importe à Alice, le tout petit résidu insignifiant de la valeur immense qu'à dû atteindre l'aiguille, réduite à néant par cet intransigeant " modulo 24 ", le précieux 4 h 32 mn et 17 sec.

Car cette valeur est vraiment précieuse. Tellement précieuse, d'ailleurs, qu'Alice va s'empresser de me la faire parvenir. Comment ? Peu importe : elle peut l'écrire sur un bout de papier et la confier à un messager, elle peut me faire des signaux de fumée sur la colline ou alors, si elle veut, me la crier très fort pour que j'arrive à l'entendre de l'autre côté du village. Bien sûr Estelle, un redoutable agent de renseignement de l'équipe adverse, a réussi à capter cette information ; il est même probable qu'elle soit parvenue à construite exactement la même horloge. Elle connaît la valeur 4 h 32 mn et 17 sec, elle connaît la force précise avec laquelle Alice a lancé son aiguille (elle entend toutes les informations véhiculées par merveilleux moyens de communication de l'époque moderne), il lui suffit donc d'effectuer l'opération inverse pour découvrir l'heure exacte du rendez-vous. Seulement voilà : elle ne peut pas ; l'aiguille est incapable de tourner dans le sens contraire, c'est une aiguille **à sens unique**, personne, pas même moi, ne peut effectuer cette opération.

Alors que vais-je faire de ce 4 h 32 mn et 17 sec ? En fait, Alice n'a pas employé n'importe quelle force pour lancer son aiguille. Pourquoi ? Parce que j'ai ma botte secrète, je dispose d'une autre force de lancement, une force que l'on pourrait qualifier de " complémentaire " ; autant la force utilisée par Alice pour chiffrer son message peut être connue de tous, autant la valeur de cette deuxième force, croyez moi, je me suis bien gardé de faire état de sa valeur ! Celle d'Alice est publique, la mienne, celle de Bernard, est tout ce qu'il y a de plus privé. Que fais-je donc ? Exactement comme Alice, je lance l'aiguille avec ma force privée, l'aiguille étant positionnée au départ sur 4 h 32 mn et 17 sec. Au bout de quelques centaines de tours, toujours dans le sens des aiguilles d'une montre, l'aiguille vient tranquillement s'arrêter sur 16 h 43 mn et 22 sec, et je connais donc l'heure exacte de mon rendez-vous !

C'est magique ? Non, mais indéniablement astucieux, car il est évident qu'il existe une relation étroite entre la " force publique ", ou la " clef publique " d'Alice et la " force privée ", ou " clef privée " de Bernard. La connaissance de cette clef publique permet-elle de deviner la valeur de la clef privée ? Connaissant la force utilisée par Alice pour lancer son aiguille, quelque chose vous permet-il de déduire la moindre information en ce qui concerne la force employée par Bernard ? Rien, absolument rien. Le fait que tout le monde puisse construire la même horloge, le fait que tout le monde connaisse la valeur de la clef publique ainsi que celle du texte chiffré, permettent-ils de connaître la valeur du message ? Non, absolument pas. Seule la connaissance de la clef privée permet d'accéder au message clair original.

Tous ces éléments illustrent presque exactement les principes et le fonctionnement de l'algorithme à clefs publiques RSA. La seule différence se situe au niveau de l'horloge et les actions entreprises pour l'élaboration des messages chiffrés. Avec RSA, ces mécanismes sont réalisés par des fonctions mathématiques.

RSA

RSA est le plus célèbre et le plus répandu des algorithmes de chiffrement à clefs publiques. Il a été inventé en 1977 par les mathématiciens et cryptologues Ron **Rivest**, Adi **Shamir** et Leonard **Adleman**.

RSA fonctionne comme l'horloge à sens unique dont nous venons de décrire le comportement : il repose sur la mise en œuvre d'une fonction mathématique **à sens unique, " à brèche secrète "**.

Avant d'entrer dans un discours plus formel, prenons le temps d'examiner la fonction RSA. L'équation employée par RSA est de la forme :

$$\text{Message Transformé} = (\text{Message})^{K} \bmod n$$

`Message` étant le message à chiffrer ou à déchiffrer, `n` le **module**, `K` un nombre entier formant, lorsqu'il est associé au module n, la **clef de chiffrement** ou la **clef de déchiffrement**.

Prenons de suite un exemple simple afin d'expérimenter concrètement de cette formule : choisissons un module n égal à 15, et un exposant de chiffrement égal à 3. La fonction RSA est s'exprime alors de la façon suivante :

$$\text{Message chiffré} = (\text{Message})^{3} \bmod 15$$

Imaginons que nous souhaitions chiffrer le message " 7 ". Le calcul de cette valeur chiffrée s'obtient très simplement en appliquant la formule :

$$(7)^{3} \bmod 15$$

Calculons :

$$(7)^{3} \text{ est égal à } 7 \times 7 \times 7 = 343$$
$$343 = 22 \times 15 + \mathbf{13}$$
$$(7)^{3} \equiv \mathbf{13} \pmod{15}$$

La valeur chiffrée de 7 est donc 13, lorsque la clef de chiffrement est égale à (3, 15). Ce n'est pas plus difficile que cela, tout au moins dans le principe.

Cependant la réalité est un peu plus complexe. En effet, si 13 est la valeur chiffrée d'un message satisfaisant à la relation $(\text{message})^{3} \equiv 13 \pmod{15}$, Estelle a-t-elle vraiment du mal à retrouver la valeur de " message " ? Evidemment non : Estelle avait de très bon résultats en arithmétique modulaire à l'école, il est probable que cette équation ne va pas lui résister bien longtemps. Vous-même, d'ailleurs, à la lumière de vos connaissances récentes êtes en mesure de trouver le nombre inférieur à 15 qui, élevé au cube et diminué de 13, donne un multiple de 15 ? Ce n'est pas 0, ce n'est pas 1, ni 2, ni 3 non plus... mais on s'aperçoit rapidement que 7 convient très bien. Connaissant la fonction RSA utilisée ainsi que la valeur chiffrée du message, il est donc parfaitement aisé pour un adversaire de recalculer le message clair original et d'accéder à vos secrets.

En fait, ce n'est pas la fonction RSA qui est en cause, ce sont uniquement les paramètres de la fonction RSA qu'il faut modifier : ils sont beaucoup trop petits. Si l'on essayait l'équation suivante :

$$\text{Message chiffré} = (\text{Message})^{13} \bmod 85$$

Sauriez vous retrouver le message clair si je vous disais que le message chiffré est 28 ? Cette équation est déjà plus compliquée à résoudre, mais avec une bonne calculatrice, un peu de courage et quelques crampes aux doigts, on découvre assez vite que 78 est la bonne réponse. Que diriez vous maintenant de l'équation :

$$\text{Message chiffré} = (\text{Message})^{28\,477} \bmod 77\,837$$

Si vous interceptez le message chiffré " 56 846 ", trouver la valeur claire correspondante devient cette fois plus pénible et vous devez avoir recours à des moyens de calcul sophistiqués. Mais ici encore, un bon tableur et quelques routines Visual Basic en viennent à bout.

Que faire alors ? Pour empêcher de façon définitive la résolution de cette équation, il va falloir utiliser des modules et des exposants de taille suffisamment importante pour défier la puissance des calculateurs du monde entier. Dans l'état actuel des technologies, les nombres utilisés dans les calculs RSA sont composés de plusieurs centaines de chiffres, de l'ordre de trois cents à six cents chiffres, voire plus. C'est à dire qu'ils reposent sur des modules de 1024 à 2048 bits, voire plus.

Mais avant d'entrer dans le sujet proprement dit, où sont donc passés l'horloge, l'aiguille et le sens unique de notre exemple précédent ?

Le sens unique, non seulement nous venons de l'évoquer, mais nous venons de montrer que, sous certaines conditions, le sens interdit s'avérait difficile, voire impossible à prendre. En effet, nous avons vu qu'il était relativement aisé de calculer une valeur C telle que $C = (\text{Message})^K \pmod n$, mais que la résolution en " message " de cette même équation mettait en échec tous les ordinateurs actuels. C'est la raison pour laquelle la fonction RSA est, tout comme l'aiguille de notre horloge, qualifiée de fonction à sens unique.

Quelle analogie existe-t-il ensuite entre un tour de cadran et la fonction RSA ? Pour comprendre cette analogie, nous allons brièvement parler des aspects pratiques du calcul du nombre $(\text{Message})^K \pmod n$.

Avons nous une idée de la valeur d'un nombre de trois cents chiffres ? Imaginons au hasard un nombre " très grand " : un milliard de milliards de milliards. De combien de chiffres un tel nombre est-il constitué ? Un milliard de milliards de milliards ne comporte que 28 chiffres ! Quelle est la masse d'un électron, cette particule infiniment microscopique ? Elle est de l'ordre de 10^{-31} grammes, c'est à dire un nombre constitué d'un zéro et de 30 zéros après la virgule ! De combien d'atomes l'univers est-il constitué ? environ 10^{77}, un nombre de " seulement " 78 chiffres. Que dire alors de la valeur d'un nombre de trois cents chiffres ? au moins une chose : qu'elle dépasse notre entendement. A fortiori, que dire de la valeur d'un nombre de trois cents chiffres élevé à la puissance d'un autre nombre de trois cents chiffres ? non seulement cette valeur nous dépasse, mais elle dépasse aussi largement les capacités de calcul des plus grands ordinateurs au monde. Comment calculer alors ce nombre immense $(\text{Message})^K \pmod n$?

Nous allons pour cela nous appuyer sur les propriétés de ce que l'on notera désormais ZZn, l'ensemble des entiers $\{0, ..., n - 1\}$. Pour raviver les mauvais souvenirs de la théorie des ensembles, rappelons que l'arithmétique modulo n, munie des deux opérations + et x, est dotée d'une structure algébrique d'anneau. Ce terme exprime de façon générique un ensemble de propriétés qui vont considérablement nous simplifier la vie : l'addition et la multiplication en arithmétique modulaire fonctionnent exactement comme l'addition et la multiplication que nous utilisons tous les jours pour jouer au Monopoly, à la différence toutefois que les résultats sont réduits modulo n.

Supposons que nous souhaitions calculer $a^3 \bmod n$. Lorsque nous avons calculé $7^3 \bmod 15$ dans l'exemple précédent, nous avons d'abord calculé la valeur de 7^3, c'est à dire 343, et effectué ensuite une réduction modulo 15 de 343. Que se passe-t-il avec des nombres de trois cents chiffres ? Un nombre de trois cents chiffres élevé au cube en comporte neuf cents. Travailler

sur neuf cents chiffres est encore envisageable en informatique, mais il ne serait nullement question d'élever ce nombre à la puissance cent. D'autant que dans le contexte d'un calcul RSA, cent est un nombre bien infime...

C'est ici que les propriétés d'anneau de notre espace fini viennent à notre secours :

$$a^3 \bmod n = (((a \times a) \bmod n) \times a) \bmod n$$

En dépit des apparences, je puis vous assurer que nous venons de réaliser un progrès considérable. Imaginons le cheminement du programme informatique effectuant ce calcul :

- l'ordinateur calcule d'abord a x a. Faisons l'hypothèse que l'entier a comporte effectivement plus de trois cents chiffres (1024 bits). Cette opération a toutes les chances de nous faire bondir en dehors des frontières de l'espace ZZn. En effet, le résultat de l'opération a x a compte plus de six cents chiffres, ce qui donne nécessairement une valeur bien supérieure au module n
- l'ordinateur effectue alors une réduction modulo n en calculant (a x a) mod n, ce qui a pour effet de nous faire revenir instantanément à l'intérieur de ZZn. Le résidu peu prendre n'importe quelle valeur à l'intérieur de l'espace {0, ..., n – 1}, il peut très bien être constitué d'un, de cent soixante dix huit, ou trois cents chiffres, peu importe
- ce résidu est maintenant à nouveau multiplié par a ; là encore, nous avons de bonnes chances de sortir de l'espace
- enfin, la valeur obtenue est réduite modulo, n et obtenons ainsi le résidu que nous cherchons.

Vérifions rapidement le fonctionnement de cette technique sur notre exemple précédent $7^3 \bmod 15$:

```
7 x 7 = 49
49 mod 15 = 4
4 x 7 = 28
28 mod 15 = 13
```

13 étant la valeur que nous avions calculée plus haut.

Percevez vous l'intérêt de cette méthode ? Le voici : lorsque vous calculez $(\text{Message})^K \pmod n$, même lorsque vous employez des puissances très élevées (l'entier K est généralement immense), jamais l'ordinateur ne devra manipuler des nombres de taille deux fois supérieure à celle du module. Dès qu'il y a multiplication, une réduction modulaire intervient immédiatement après.

Cela ne vous rappelle-t-il pas notre horloge ? Lorsque vous effectuez une multiplication, il se peut que sortiez de l'espace fini ; si c'est le cas, la réduction modulo n vous y ramène et vient pointer sur un nouveau résidu de cet espace ; si ce n'est pas le cas, la réduction modulo n ne changera pas votre valeur, elle même résidu de cet espace. C'est un peu comme si vous lanciez l'aiguille autour du cadran : si elle a suffisamment d'énergie pour dépasser la limite de l'espace fini 23 h 59 mn et 59 sec, elle entame un nouveau tour et vient pointer sur une autre valeur, un autre " résidu " de ce même espace, grâce à la réduction modulo 24 h. Le calcul de $(\text{Message})^K \pmod n$ peut donc s'interpréter comme la succession de très nombreux tours de cadrans avec, au passage, un pointage successif sur différents résidus de l'espace. Cette opération est en quelque sorte un beau voyage de résidu en résidu au sein de ZZn, et la valeur chiffrée du message n'est autre que le résidu final obtenu au terme de ce voyage.

Vous devinez maintenant sans peine que la force de lancement de l'aiguille utilisée par Alice est symbolisée par l'exposant K de cette équation. Plus l'exposant K est élevé, plus l'aiguille a des chances de faire des tours de cadran. Les valeurs de K et du module n peuvent être connus de tout le monde, car, à l'image de notre horloge à sens unique, aucun ordinateur au monde n'a la puissance suffisante pour retrouver Message à partir de sa valeur chiffrée. Et moi, Bernard, pour retrouver le texte clair du message à partir du message chiffré, je vais procéder comme avec l'horloge : continuer à faire des tours de cadran, c'est à dire continuer à élever à la puissance le message chiffré, en d'autres termes, continuer à voyager de résidu en résidu à l'intérieur de ZZn, jusqu'à retomber sur un résidu particulier qui, justement, s'avère être le

message clair lui même. C'est une des propriétés intéressantes de la fonction RSA, la fameuse brèche secrète évoquée plus haut.

Elaboration du module

La première opération consiste à déterminer une valeur pour le nombre n. Un module **n** est très facile à obtenir : il suffit d'engendrer de façon aléatoire deux nombres premiers, **p** et **q** ; le module n est tout simplement le produit de ces deux nombres :

$$\mathbf{n = p \times q}.$$

Nous verrons plus loin pourquoi p et q doivent être premiers et aléatoires.

Lorsqu'on parle d'une clef RSA de 1024 bits, cela veut dire que le module n est un nombre de 1024 bits. Cela veut dire qu'il a été élaboré à partir de deux nombres p et q de 512 bits.

Mais est-ce que le fait de se cantonner à des nombres premiers ne limite pas considérablement les possibilités ? Avez-vous une idée de la quantité de nombres premiers de 512 bits ? Bruce Schneier, dans son remarquable ouvrage intitulé " Cryptographie Appliquée ", apporte une réponse très éloquente : " *le Père Noël ne tombera jamais à court de nombres premiers pour tous les petits garçons et toutes les petites filles sages. En fait, il y a plus de* 10^{151} *nombres premiers de 512 bits de long ou moins. Il y a* 10^{84} *atomes dans l'univers. Si chaque atome de l'univers avait besoin d'un milliard de nombres premiers chaque micro seconde depuis l'origine des temps jusqu'à maintenant, il faudrait* 10^{116} *nombre premiers ; il en resterait encore approximativement* 10^{151}*.* "

Elaboration des clefs de chiffrement et de déchiffrement

Exposant de chiffrement

Les calculs de clefs de chiffrement et de déchiffrement RSA font intervenir un nombre noté φ**(n)**. Il s'agit du cardinal de l'ensemble restreint des résidus modulos n, c'est à dire de l'ensemble des résidus premiers par rapport à n.

Prenons quelques exemples pour mieux comprendre. Dans une arithmétique modulo 8, nous avons vu au début de ce chapitre que l'ensemble des résidus modulo 8 était {0, 1, 2, 3, 4, 5, 6, 7}. Dans cet ensemble, 2, 4 et 6 ne sont manifestement pas premiers avec 8 ; l'ensemble restreint des résidus modulo 8 est donc : {1, 3, 5, 7}, le nombre 0 ne faisant jamais partie de cet ensemble. $\varphi(8)$ est donc égal à 4.

Plus généralement, dans une arithmétique modulo p, si p est un nombre premier, $\varphi(p)$ est égal à p – 1 : tous les élément de ZZp, à l'exception de 0, sont premier avec p. Lorsque n est le produit de deux nombres premiers p et q, on démontre aisément que :

$$\varphi(n) \text{ est égal à } (p - 1)(q - 1).$$

Pourquoi introduire cette notion $\varphi(n)$? $\varphi(n)$ est dotée de propriétés intéressantes qui vont nous être très précieuses dans les calculs RSA. Si par exemple n est premier et a n'est pas un multiple de n, alors le petit théorème de Fermat indique que :

$$a^{n-1} \equiv 1 \pmod{n}.$$

Ce théorème est notamment utilisé pour effectuer des tests de primalité sur les grands nombres : on choisit quelques valeurs au nombre " x " et si les opérations $x^{(p-1)} \bmod p$ sont égales à 1, cela veut dire que le nombre p est probablement premier.

Enfin, le petit théorème de Fermat généralisé par Euler montre que $a^{\varphi(n)} \equiv 1 \pmod{n}$. dans le cas où n = p x q, cette formule revient à :

$$\mathbf{a^{(p-1)(q-1)} \equiv 1 \pmod{n}}.$$

Ceci étant brièvement rappelé, nous pouvons maintenant aborder le choix de " e ".

L'exposant de chiffrement, noté " **e** " par convention, est un élément de ZZn choisi au hasard. Il doit toutefois satisfaire à une seule condition :

> Les nombres **e** et φ**(n)** dont la valeur est égale à (p – 1)(q – 1) doivent être **premiers entre eux**.

Cette condition est impérative afin de pouvoir calculer la clef de déchiffrement.

Il existe donc un très grand nombre de possibilités pour e. Une manière de déterminer une valeur de e consiste à engendrer un nombre aléatoire inférieur à n et à tester si ce nombre et (p – 1)(q – 1) sont premiers entre eux. Cette vérification s'effectue simplement en très peu d'opérations grâce à l'algorithme d'Euclide (leur pgcd est égal à 1). Si ce n'est pas le cas, il suffit d'incrémenter ce nombre aléatoire jusqu'à ce que le pgcd de ce nombre et de $\varphi(n)$ soit égal à 1. La valeur ainsi obtenue est un candidat probable pour e. Vous avez généralement une grande latitude pour choisir e.

Exposant de déchiffrement

En revanche, lorsque la valeur de e est fixée, l'exposant de déchiffrement " **d** " est calculé de manière à ce que :

$$\mathbf{e \times d \equiv 1 \pmod{(p-1)(q-1)}}$$

d représente donc l'inverse de e modulo $\varphi(n)$. On démontre que cette équation admet une solution unique si et seulement si e est premier avec $\varphi(n)$. La valeur de d s'obtient très facilement en appliquant le théorème de Bezout.

Clef publique, clef privée

Nous venons donc de calculer deux exposants. Chaque exposant associé au module va devenir une clef : l'une va servir à chiffrer, e par convention, l'autre à déchiffrer l'information. Le fait de ne pas utiliser la même clef pour chiffrer et déchiffrer est une propriété très intéressante. Lorsque vous souhaitez utiliser la fonction RSA pour garantir la confidentialité d'un secret, vous avez tout le loisir de publier la clef de chiffrement : tout le monde pourra effectuer cette opération de chiffrement. En revanche la clef de déchiffrement ne doit être connue que du destinataire. Par convention, les exposants e et d sont utilisés respectivement pour le chiffrement et le déchiffrement.

> Le couple de nombres **e** et **n** constituent ce que l'on appelle la **clef publique**.
>
> Le nombre **d** constitue ce que l'on appelle la **clef privée**.

Chiffrement et déchiffrement RSA

Pour chiffrer un message, il suffit d'élever ce message à la puissance de la **clef de chiffrement**, cette opération étant effectuée modulo n ; pour déchiffrer le message, il suffit d'élever le

message chiffré à la puissance de la **clef de déchiffrement**, cette opération étant effectuée modulo n :

$$\text{Message chiffré : } \mathbf{c = m^e \bmod n}$$
$$\text{Message clair : } \mathbf{m = c^d \bmod n}$$

Pour les courageux, je propose la démonstration suivante :

Toutes les opérations sont effectuées mod n. Vous déchiffrez, donc vous élevez le message chiffré à la puissance d :

$$c^d = (m^e)^d = m^{ed}$$

Par définition, d est calculé de manière à ce que e x d soit congru à 1 modulo (p – 1)(q – 1). Exprimé en français, cela revient à dire que e x d est un multiple de (p – 1)(q – 1) plus 1 ; exprimé sous une forme mathématique :

$$e \times d = k\,(p-1)(q-1) + 1.$$

Nous pouvons donc écrire :

$$m^{ed} = m^{k(p-1)(q-1)+1} = m * m^{k(p-1)(q-1)}$$

C'est ici que le petit théorème de Fermat généralisé par Euler vient à la rescousse : **lorsque *p* et *q* sont des nombres premiers**, cette expression $m^{k(p-1)(q-1)}$ si compliquée est tout simplement congrue à 1 modulo n. Ce qui nous permet de déduire :

$$m * m^{k(p-1)(q-1)} = m * 1 = m$$

Grâce à notre brèche secrète, nous venons de retrouver notre message clair.

Pourquoi utiliser des nombres si grands ? Vous connaissez les valeurs de n et de e, la clef publique. Supposez que vous réussissiez à factoriser n, c'est à dire à retrouver ses facteurs premiers, les valeurs de p et q à partir de n. Vous savez donc calculer φ(n), vous en déduisez immédiatement la valeur de d, la clef secrète et vous avez cassé le chiffre ! **La sécurité de RSA est donc basée sur la difficulté de factoriser n**. A l'heure actuelle, les techniques de factorisation les plus performantes parviennent à procéder à la factorisation de nombres d'une taille supérieure à 512 bits, et les développements en théorie des nombres, hélas pour les concepteurs d'algorithmes, rendent la factorisation plus facile. Dans l'état actuel des technologies, l'utilisation d'une clef de 1024 bits peut vous préserver la confidentialité de vos secret pour encore une certaine durée... disons d'ores et déjà qu'il est préférable d'utiliser une clef de 2048 bits pour sécuriser des informations vraiment sensibles pour les 10 années à venir.

Exemple

Il s'agit par exemple de chiffrer le message suivant : " **Code : J&B007** ". En code ASCII avec l'en-tête on obtient le message suivant :

```
m = 210456444444326731113100310123225823227423826624824825 5
```

Choisissons aléatoirement deux nombres premiers : p = 83, q = 191

Calculons le module **n** : n = p x q = 15 853

Calculons $\varphi(n) = (p - 1)(q - 1)$: $\varphi(n) = 42 \times 190 = 15\,580$

Choisissons **e** tel que e soit premier avec $\varphi(n)$. Par exemple : e = 771

Calculons la clé privée **d** : d = 1 071

Pour chiffrer le message, il convient d'abord de le décomposer en blocs dont la taille est inférieure à n = 15 853. On chiffre ensuite chaque bloc avec la formule $m^{771} \bmod 15\,853$:

m	$m^{771} \bmod 15\,853$	$m^{1\,071} \bmod 15\,853$
2104	8876	2104
5644	7470	5644
4444	15726	4444
3	10977	3
2673	2482	2673
11131	3761	11131
003101	14977	003101
2322	9391	2322
5823	5271	5823
2274	6999	2274
2382	6700	2382
6624	12049	6624
8248	5240	8248
255	1823	255

On obtient ainsi le message chiffré suivant :

```
c =  8876 7470 15726 10977 2482 3761 14977 9391
     5271 6999 6700 12049 5240 1823
```

Pour déchiffrer ce message on procède au même type de calcul avec la clé privée d. Par exemple 8 876 devient :

$$8\,876^{1\,071} \bmod 15\,853 = 2\,104$$