Sécurisation des communications

1 Motivation

Avec le développement du commerce en ligne dans le Web il est vite apparu que les communications devaient être sécurisées. Divers protocoles ont été créés alors pour chiffrer les communications
Dans un premier temps nous allons nous intéresser au chiffrement à clef privée puis au chiffrement à clef publique enfin au protocole TLS (Transport Security Layer)
L'objectif de la cryptographie est de permettre la circulation sous forme cachée ou chiffrée C d'un message en clair M entre deux personnes, traditionnellement appelées Alice et Bob de telle sorte qu'une troisième personne, appelée Oscar, non autorisée ne puisse pas retrouver M à partir de C dans un temps "raisonnable"

La transformation de M en C appelée **chiffrement** se fait par l'intermédiaire d'une fonction (au sens mathématique et algorithmique aussi) de chiffrement E (encryption) telle que $C = E(M)$

Le **déchiffrement** de C en M se fait par l'intermédiaire d'une fonction D telle que $M = D(C)$

Par conséquent $D(E(M)) = M$

2 Chiffrement à clef privée

On a longtemps cru qu'il n'existait qu'une seule sorte de chiffrement, **le chiffrement à clé privée**.

Les relations fondamentales ci-dessus s'écrivent avec un paramètre K appelé **clé**
$C = E_K(M)$ et $M = D_K(C)$
Cette clé est **partagée secrètement entre Alice et Bob**
Alice et Bob conviennent aussi d'un algorithme de chiffrement et de déchiffrement
La "sécurité de ce système" repose sur la clé qui est acheminée entre Alice et Bob par un "canal sécurisé"

C'est le principe de Kerckhoffs(1883) :

"La sécurité d'un système de chiffrement ne doit pas dépendre de la préservation du secret de l'algorithme. La sécurité ne repose que sur le secret de la clé."

Jusqu'au jour où quelques individus ont remis en cause le principe de Kerckhoffs. (on verra cela dans la révolution du chiffrement à clef publique (1977))

On va étudier quelques algorithmes de chiffrement à clé privée :

2.1 Chiffrement par décalage

Le chiffrement par décalage dit aussi méthode de César est connue depuis l'Antiquité.

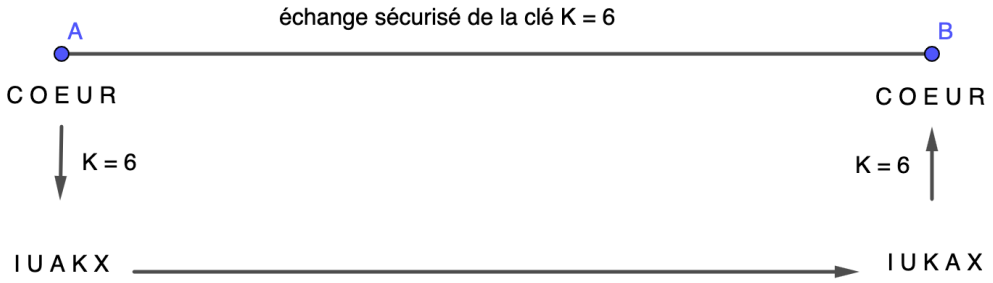
Dans une première approche on suppose que le texte à chiffrer est composé uniquement que de lettres latines non accentuées minuscules

Par exemple : **"les sanglots longs des violons de l'automne blessent mon coeur d'une langueur monotone"**

A chaque lettre est associée un entier, par exemple à la lettre a est associée 0 et à z, 25, et réciproquement à chaque entier de 0 à 25 est associé une lettre de 'a' à 'z'.

La clef est un entier K tel que $1 \leqslant K \leqslant 25$

Supposons que $K = 6$. Voici le chiffrement du mot 'coeur' avec la clé K = 6.



1. A la lettre 'c' est associée le nombre 2, auquel on ajoute 6, ce qui fait 8, qui est l'entier associé à 'i', on chiffre 'c' par 'i'
2. A la lettre 'u' est associée le nombre 20, auquel on ajoute 6, ce qui fait 26, **on sort de la plage des entiers 0..25 consacré à l'alphabet** on doit donc calculer 26 % 26 pour obtenir 0 ce qui correspond à la lettre 'a'

.

Cryptanalyse

La cryptanalyse est un ensemble de méthodes pour déchiffrer un message chiffré **sans posséder la clé de déchiffrement**.

On parle alors de casser le message.

Pour le chiffrement par décalage, casser le message consiste à essayer toutes les clés sur un court message, afin de voir si on obtient un message "en clair".

2.2 La méthode par substitution

Pour expliquer cette méthode on va utiliser le message chiffré que l'on trouve dans la nouvelle d'*Edgar Poe, le scarabé d'or*.

`https://poestories.com/read/goldbug`

```
53‡‡†305))6·;4826)4‡.)4‡);806·;48†8^60))85;161;:‡·8
†83(88)5·†;46(;88·96·?;8)·‡(;485);5·†2:·‡(;4956·2(5
·−4)8^8·;4069285);)6†8)4‡‡;1(‡9;48081;8:8‡1;48†85;4
)485†528806·81(‡9;48;(88;4(‡?34;48)4‡;1‡(;:188;‡?;
```

Ce message a été chiffré en substituant à chaque lettre de la langue anglaise **toujours le même symbole**, en quelque sorte la clé est un dictionnaire Python

`cle = {'e' : '8',....}`

Dans cette nouvelle Poe explique la méthode de chiffrement et la cryptanalyse par l'étude des fréquences des symboles

2.3 La méthode de Vigenère

Pour contrer la cryptanalyse par l'étude de fréquences, Vigenère (XVI^{e} siècle) a eu l'idée d'adapter la méthode de décalage ainsi :

1. On enlève les espaces du texte à chiffrer.
2. La clé est par exemple COEUR. On fait glisser la clé, le long du message pour produire une nouvelle clé aussi longue que le message.
   LESSANGLOTSLONGSDESVIOLONSDE...
   COEURCOEUR....
3. Chaque lettre de la nouvelle clé chiffre la lettre du message par décalage. Par exemple le S de LES est chiffré par la lettre A autrement dit par un décalage de 0 donc ce S est chiffré par S.
   Par contre le S de SANGLOT est chiffré par P dont l'indice est 14 dont on chiffre S avec la lettre d'indice $(18 + 15)\ \%\ 26 = 7$, c'est à dire H.

2.4 Chiffrement affine

Lorsqu'Alice veut chiffrer une lettre dont le code est $n \in \mathbb{Z}_{26}$ elle utilise une fonction affine f définie par $f(x) = ax + b$

Dont la lettre chiffrée est $f(n) = an + b = m \in \mathbb{Z}_{26}$

Pour que Bob puisse déchiffrer $an + b = m$ et retrouver n il est nécessaire que a soit **inversible** dans $\mathbb{Z}_{26}$ c'est à dire il existe $c \in \mathbb{Z}_{26}$

tel que $ac = 1$

Par exemple 15 est inversible dans $\mathbb{Z}_{26}$ car $15 \times 7 = 105 = 4 \times 26 + 1 = 1$ modulo 26

Pour déchiffrer Bob calcule $c(m - b)$ car $c(m - b) = c(an) = (ac)n = 1 \times n = n$

Alice et Bob partagent la clé privée constituée par le couple (a, b) le problème est :

Comment à partir de a Bob peut trouver c ?

Définition 1 *Deux entiers sont dits premiers entre eux s'ils n'ont pas de diviseurs communs*

Exemple : 15 et 26 sont premiers entre eux mais 4 et 26 ne le sont pas car ils ont un diviseur commun 2

Théorème 1 *(Bezout)*

a et b premiers entre eux $\iff$ il existe u et v tel que $au + bv = 1$

Exemple :

$15 \times 7 + 26 \times (-4) = 1$

L' **algorithme d'Euclide étendu** permet d'avoir u et v étant donnés a et b.

Théorème 2 *a est inversible dans $\mathbb{Z}_{26}$ $\iff$ a est premier avec 26*

Preuve :

a est premier avec 26 $\iff$ il existe u et v tel que $au + 26v = 1$ $\iff$ $au = 1 - 26v = 1$ dans $\mathbb{Z}_{26}$ ce qui signifie que a est inversible dans $\mathbb{Z}_{26}$

Le module `sympy` de Python dispose d'une fonction `mod_inverse` permettant de calculer l'inverse d'un nombre a modulo b.

Ainsi

```
>>> mod_inverse(15,26)

7
```

2.5 One-time-Pad

Vernam et **Mauborgne** à la fin de la première guerre mondiale mettent au point le chiffrement à clé jetable (One-time-pad) dont le principe est le suivant :

1. Si on veut chiffrer un message M, il faut générer de manière **aléatoire** une clé K aussi longue que le message M.
2. Le message chiffré C est ensuite obtenu par $C = M \oplus K$
3. Pour déchiffrer C il suffit de faire $C \oplus K = M \oplus K \oplus K = M$
4. **Shannon** en 1949 démontre que le chiffrement à clé jetable est un chiffrement "inconditionnellement" sûr (au sens des probabilités conditionnelles).

   C'est la seule méthode de chiffrement "incassable" à condition que les deux conditions du chiffrement soient respectées.
5. Que se passe -t-il si on réutilise plusieurs fois la même clé ?

   $C_1 = M_1 \oplus K$ et $C_2 = M_2 \oplus K$ donc $C_1 \oplus C_2 = M_1 \oplus M_2$

   Or M_1 et M_2 pourraient contenir des espaces (à moins qu'ils aient été enlevés) dans ce cas ces espaces peuvent dans l'opération $M_1 \oplus M_2$ interférer avec des caractères étant dans [a-zA-Z].

   Ce qui permettrait d'analyser les mots d'une phrase et par exemple le début d'une phrase (mots de deux lettres)

   Si on réussit par exemple à isoler en tête d'une phrase un mot de deux lettres C et si on a un fichier de mots de deux lettres M, en faisant à chaque fois $C \oplus M = M \oplus K \oplus M = K$ ensuite pour chaque clé générée on xor tous les messages chiffrés que l'on a avec chacune de ces clés, si un message est en clair on a trouvé la bonne clé.

3 Système cryptographique à clé publique

"*Comme nous, Ralph était un peu fou. Il faut être fou pour mener jusqu'au bout une recherche originale, et seuls des fous la poursuivront envers et contre tout. Vous avez une première idée, vous vous enthousiasmez, et ça ne marche pas. Alors vous passez à l'idée numéro 2, vous vous enthousiasmez, et ça ne marche pas. Alors vous avez l'idée 99, vous vous enthousiasmez, et ça ne marche pas. Seul un fou sera encore enthousiaste à sa centième idée, et il faudra peut-être avoir essayé cent voies avant que l'une conduise quelque part. A moins d'être assez fou pour retrouver à chaque fois tout votre enthousiasme, vous vous découragerez, et n'aurez pas l'énergie de mener les choses jusqu'au bout. Dieu bénit les fous.*"(Martin Hellman, professeur d"informatique à l'université de Stanford - Californie (1975))

3.1 Le Protocole de Diffie-Hellman (1976)

Alice et Bob vont utiliser une méthode traditionnelle de chiffrement à clé secrète K mais **ils veulent que cette clé ne soit pas acheminée mais déduite par eux seuls par calcul.**

Voici la table de multiplication de {1,2,..,p-1} modulo p premier

```
si p = 5

   x  1 2 3 4
   1  1 2 3 4
   2  2 4 1 3
   3  3 1 4 2
   4  4 3 2 1
```

Si p n'est pas premier on n'a pas de "stabilité" dans les calculs et on observe la présence de 0 sur l'une des lignes de la table, ce qui rend impossible toute multiplication ultérieure..

```
si p = 4

   x  1 2 3
   1  1 2 3
   2  2 0 2
   3  3 2 1
```

Lorsque p est premier il existe au moins un élément g > 1 de {1,2,...,p-1} à partir duquel on peut engendrer tous les autres par élévation à la puissance. On dit que c'est un générateur ou une racine primitive modulo p.

Par exemple si p = 5 g = 2 est un générateur

Car $2^1 = 2$,$2^2 = 4$, $2^3 = 3$ et $2^4 = 1$. On a donc engendré tous les éléments de {1,2,3,4} comme puissances de 2

Définition 2 *p premier*

$(\mathbb{Z}_p^*, \times)$ *est un groupe de* $p-1$ *éléments*

On dit que r est une racine primitive modulo p si :

$\forall x \in \mathbb{Z}_p^* \ \exists \alpha \in \mathbb{N}$ *tel que* $x = r^\alpha$

Exemples

1. Pour $p = 5$

   2 est une racine primitive modulo 5 car les 4 éléments non nuls de $\mathbb{Z}_5$ c'est à dire 1,2,3 et 4 peuvent s'écrire comme des puissances de 2 :

   $2^1 = 2, 2^2 = 4, 2^3 = 3, 2^4 = 1$

2. Pour $p = 7$

   2 n'est pas une racine primitive modulo 7 car

   $2^1 = 2,\ 2^2 = 4,\ 2^3 = 1$ et 5 par exemple ne peut pas s'exprimer comme une puissance de 2 dans $\mathbb{Z}_7^*$

   3 est une racine primitive modulo 7 car :

   $3^1 = 3, 3^2 = 2, 3^3 = -1, 3^4 = 4, 3^5 = 5, 3^6 = 1$

Théorème 3 *Pour tout p premier il existe une racine primitive modulo p*

Protocole

1. Alice et Bob se mettent d'accord **publiquement** sur un "grand" nombre premier p (ici on prendra $p = 7$) et une racine primitive modulo p soit r (ici $r = 3$)
2. Alice choisit un nombre a **secret**, connu d'elle seule. Choisissez $a =$ .... (de 1 à 6). Et elle transmet à Bob et à qui veut l'entendre le nombre $\alpha = r^a \mod p$.
   Que vaut ici $\alpha = 3^a \mod 7 =$..... ?
3. Bob choisit de même un nombre b **secret** et connu de lui seul,et transmet à Alice $\beta = r^b \mod p$. Choisir une valeur pour b. $b =$ .....
   $\beta =$ .....
4. Ici est le coeur du protocole : **la clé secrète** K **est** $K = \alpha^b \mod p = \beta^a \mod p$
   $\alpha^b \mod 7 =$ ..... et $\beta^a \mod 7 =$ .....
   Autrement dit Alice prend ce que lui a envoyé Bob , c'est à dire β et l'élève à la puissance "le nombre secret" d'Alice et obtient la clé secrète K qui servira à chiffrer.
   De même Bob prend ce que lui a envoyé Alice, c'est à dire α et l'élève à la puissance "le nombre secret" de Bob et obtient ô miracle mathématique le même nombre K
5. Oscar a peut-être intercepté les valeurs de p puis r puis r^a puis r^b mais il arrivera difficilement à obtenir a à partir de α ou b à partir de β, parce que l'exponentiation modulaire est une fonction **à sens unique** (voir plus loin)

Théorème 4 *p premier*
r une racine primitive modulo p
Si $a, b \in [\![1, p-1]\!]$ *et* $\alpha = r^a \ mod\ p$ *et* $\beta = r^b \ mod\ p$
Alors $\alpha^b = \beta^a \ mod\ p$

Preuve
$\alpha^b = (r^a)^b = r^{ab} = (r^b)^a = \beta^a$

Théorème 5 *(Théorème du logarithme discret)*
r une racine primitive modulo p
$\mathbb{Z}_p^* \ni n \to r^n \in \mathbb{Z}_p^*$ *est injective*

Preuve
Si $r^n = r^m$ or on a vu sur des exemples plus haut que la suite des puissances de r^k est périodique de période $p-1$ car l'exponentiation modulaire est surjective et à cause du Théorème de Fermat $r^{p-1} \equiv 1\ [p]$
or n et m appartiennent à $[\![1, p-1]\!]$ donc $n = m$
Etant injective et surjective, l'exponentiation modulaire est **bijective** donc inversible

Définition 3 *La fonction réciproque de l'exponentiation modulaire de base r modulo p premier est :*
le logarithme discret de base r, noté $\log_r$*, où r est une racine primitive modulo p premier et définie par :*
$\log_r(r^n) = n$ *avec* $n \in \mathbb{Z}_p^*$

Pratiquement il se trouve que calculer l'image d'un élément de $\mathbb{Z}_p^*$ par l'exponentiation modulaire est "facile" même si p est un "grand" nombre premier (voir exercice)

Par contre pour calculer l'image d'un élément de $\mathbb{Z}_p^*$ par la fonction réciproque, le logarithme discret de base r, le temps mis sera "très grand".

On dit alors que l'exponentiation modulaire est une **fonction à sens unique** dans le sens où étant donné une image il est difficile de trouver l'antécédent.

Par exemple supposons que la clé publique est (999983, 5) car 5 est un générateur de $\mathbb{Z}_{999983}^*$ pour la multiplication.

Si on intercepte $\alpha = 230618$ saura-t-on trouver dans un temps raisonnable k tel que $5^k \% 999983 = 230618$?

Il n'y a pas de formule d'inversion qui permet d'obtenir k en fonction de 230618, 5 et 999983, donc il nous reste à tester toutes les possibilités entre 2 et 999982. Cela prend à peu près 1 seconde et on trouve $k = 232425$

Maintenant plus la taille de p augmente plus le temps pour casser p^a sera grand.

Par exemple $p = 10^{100} + 43723$ est un nombre premier .

Il existe des algorithmes pour tester la primalité de grands nombres. On peut utiliser des systèmes de calculs comme PARI/GP en ligne `https://pari.math.u-bordeaux.fr/gpwasm.html`

1 s pour un nombre premier de taille 10^7 donc $dfrac 10^{100} 10^7$ pour un nombre premier de taille 10^{100} à peu près 10^{90} secondes. Or 1 année vaut à peu près 3 * 10^7 donc ... cela fait un très grand nombre d'années.

On peut encore augmenter la taille de p si de l'autre côté on utilise plusieurs ordinateurs puissants couplés entre eux.

3.2 Le cryptosystème RSA (Rivest, Shamir, Adleman) (1977). Chiffrement asymétrique et à clé publique

Chiffrement RSA

1. Alice et Bob ont chacun leurs clés publiques P_A et P_B (comme des numéros de téléphone) et leurs clés secrètes S_A et S_B. Ces clés sont des **paires** d'entiers.
2. Les clés d'Alice (et de même pour Bob) définissent des fonctions $P_A()$ et $S_A()$ à sens unique, **commutatives** c'est à dire :

$$S_A(P_A(M)) = P_A(S_A(M)) = M$$

3. Supposons que Bob veuille envoyer un message M chiffré à Alice. Il se procure la clé publique d'Alice P_A comme un numéro de téléphone dans un annuaire.
4. Bob calcule le texte chiffré $C = P_A(M)$ et envoie C à Alice.
5. Lorsque Alice reçoit C elle calcule $S_A(C) = S_A(P_A(M)) = M$ et elle a accès au texte en clair M.

Comment être sûr qu'un message vient bien de la personne censée l'avoir écrit ?

RSA permet de définir la : **Signature numérique d'un message**

1. Alice envoie un message M' à X . Elle associe à M' une signature définie par $s = S_A(M')$

2. Lorsque X reçoit (M',s) pour s'assurer que c'est bien Alice qui a envoyé M' ,il calcule $P_A(s)$ et regarde s'il obtient M'
3. Si $s = S_A(M')$ alors $P_A(s) = P_A(S_A(M')) = M'$, sinon si $s \neq S_A(M')$ il n'obtiendra pas M'

Un message peut être à la fois **chiffré** et **signé** (voir exercice)

Comment ça marche mathématiquement ?

Pour bien comprendre comment fonctionne RSA il nous faut préciser quelques notions mathématiques :

Regardons la table de multiplication dans {1,2,3,...,14 } modulo 15

Où 15 est le produit de deux nombres premiers.

si n = 15 = 3 x 5

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	1	2	3	4	5	6	7	8	9	10	11	12	13	14
2	2	4	6	8	10	12	14	1	3	5	7	9	11	13
3	3	6	9	12	0	3	6	9	12	0	3	6	9	12
4	4	8	12	1	5	9	13	2	6	10	14	3	7	11

etc...

Si on enlève à {1, ..,14} tous les éléments ayant un facteur commun avec 15, c'est à dire 3,6,9,12 et 5,10, le reste {1,2,4,7,8,11,13,14} a une table de multiplication stable

x	1	2	4	7	8	11	13	14
1	1	2	4	7	8	11	13	14
2	2	4	8	14	1	7	11	13
4	4	8	1	13	2	14	7	11

etc...

Dans le cas général où $n = p \times q$ où p et q sont premiers.

On a $pq - 1$ éléments de 1 jusqu'à $pq - 1$

On enlève les multiples de p, p,$2p$ jusqu'à $(q-1)p$ donc $q - 1$ éléments

On enlève les multiples de q donc $p - 1$ éléments donc il reste :

$pq - 1 - (q-1) - (p-1) = pq - p - q + 1 = (p-1)(q-1)$

Définition 4 *Attention n un entier non premier produit de deux nombres premiers p et q*

On note $\mathbb{Z}_n^*$ l'ensemble des entiers strictement positifs et ***premiers avec*** *n (c'est à dire n'ayant pas de facteur premier en commun avec n)*

Le nombre d'éléments de $\mathbb{Z}_n^*$ est noté $\phi(n)$ où la fonction ϕ est appelée indicatrice d'Euler

Théorème 6 *1. $\phi(n) = (p-1)(q-1)$*

2. $\forall a \in \mathbb{Z}_n^*$ on a $a^{\phi(n)} = a^{(p-1)(q-1)} \equiv 1\ [n]$

Preuve Soit a un élément quelconque de $\mathbb{Z}_n^*$

Considérons la fonction $f_a : \mathbb{Z}_n^* \ni x \to ax \in \mathbb{Z}_n^*$

injective car si $ax = ax'$ en multipliant à gauche par l'inverse de a on obtient $x = x'$

surjective car pour tout $y \in \mathbb{Z}_n^*$ il existe $x \in \mathbb{Z}_n^*$ tel que $y = ax$

En effet $x = a^{-1}y$

Donc f_a est bijective et on peut voir f_a comme une **permutation** sur l'ensemble **fini** $\mathbb{Z}_n^*$

Donc $\prod_{x\in\mathbb{Z}_n^*} f_a(x) = \prod_{x\in\mathbb{Z}_n^*} x$

Or $\prod_{x\in\mathbb{Z}_n^*} f_a(x) = a^{\phi(n)} \prod_{x\in\mathbb{Z}_n^*} x$ (commutativité de $\times$)

Et donc $a^{\phi(n)} \prod_{x\in\mathbb{Z}_n^*} x = \prod_{x\in\mathbb{Z}_n^*} x$ donc $a^{\phi(n)} \equiv 1\ [n]$

Théorème 7 *(Théorème des restes chinois) Soit n_1 et n_2 deux entiers* ***premiers entre eux*** *et $n = n_1 n_2$ (ce qui est le cas lorsque $n = pq$ où p et q sont premiers)*

Soit f la fonction définie par :

$Z_n \ni a \to (a_1, a_2) \in Z_{n_1} \times Z_{n_2}$ où $a \equiv a_i\ [n_i]$ avec $i = 1$ et 2

Cette fonction est bijective et respecte l'addition et la multiplication au sens où :

$f(a + b) = f(a) + f(b)$ et $f(ab) = f(a) \times f(b)$

Par exemple dans $\mathbb{Z}_{15}$ la deuxième ligne est la première modulo 3 et la troisième est la première modulo 5

```
si n = 15 = 3 x 5

  n=15    1   2   3   4   5   6   7   8   9   10   11   12   13   14
  p=3     1   2   0   1   2   0   1   2   0    1    2    0    1    2
  q=5     1   2   3   4   0   1   2   3   4    0    1    2    3    4
```

Réciproquement comment retrouver x dans $\mathbb{Z}_{15}$ tel que $x\%3 = 2$ et $x\%5 = 1$

Dans le tableau on voit que $x = 11$ mais comment trouver ce résultat par calcul ?

Preuve

Montrons l'existence directement de f^{-1}

Puisque n_1 et n_2 sont **premiers entre eux** donc n_1 est inversible dans Z_{n_2} et n_2 est inversible dans Z_{n_1}

Soit $c_1 = n_2 \times (n_2^{-1} \mathrm{mod} n_1)$ on a $c_1 \equiv 1\ [n_1]$ et $c_1 \equiv 0\ [n_2]$

De même $c_2 = n_1 \times (n_1^{-1} \mathrm{mod} n_2)$ on a $c_2 \equiv 1\ [n_2]$ et $c_2 \equiv 0\ [n_1]$

On définit $f^{-1}(a_1, a_2) = (c_1 a_1 + c_2 a_2) \mathrm{mod}\ \ n$

Et $(c_1 a_1 + c_2 a_2) \equiv a_1\ [n_1]$ et $(c_1 a_1 + c_2 a_2) \equiv a_2\ [n_2]$

Exemple 1

$a \equiv 2\ [3]$ et $a \equiv 1\ [5]$

L'inverse de 5 modulo 3, noté est 2, L'inverse de 3 modulo 5, est 2

$c_1 = 5 \times 2 = 10$ et $c_2 = 3 \times 2 = 6$

Donc $a = 10 \times 2 + 6 \times 1 = 26 \equiv 11\ [15]$

Exemple 2

$a \equiv 27\ [79]$ et $a \equiv 27\ [97]$

Sans calculer $a = 27$

Exemple 3

$a \equiv 127$ [79] et $a \equiv 127$ [97]
Sans calculer $a \equiv 127$ [97×79]

Algorithme RSA

Du côté d'Alice

1. Choisir aléatoirement deux grands nombres premiers p et q différents (de taille chacun de plus de 1024 bits)
2. Calculer $n = pq$
3. Choisir un petit entier e impair premier avec $\phi(n) = (p-1)(q-1)$
4. Calculer d l'inverse de e modulo $\phi(n)$ (Algorithme d'Euclide étendu)
5. $P_A = (e, n)$ est la clé publique RSA d'Alice
6. $S_A = (d, n)$ est la clé secrète RSA d'Alice

Du côté de Bob qui veut envoyer $M \in Z_n$ à Alice :
Le chiffrement est $E(M) = P_A(M) = M^e$ mod n
Déchiffrement du message chiffré $C \in Z_n$ venant de n'importe qui :
Le déchiffrement est $D(C) = S_A(C) = C^d$ mod n

La validité de cet algorithme repose sur le fait que

Pour tout $M \in Z_n$ on doit avoir $P_A(S_A(M) = S_A(P_A(M)) = M$
c'est à dire $M^{ed} \equiv M$ [n]
Est ce bien le cas ?

Preuve

Puisque e et d sont inverses modulo $\phi(n) = (p-1)(q-1)$ alors il existe k entier tel que
$ed = 1 + k(p-1)(q-1)$
Donc

1. Si M n'est pas un multiple de p
   $M^{ed} \equiv M(M^{p-1})^{k(q-1)} \equiv M \times (1)^{k(q-1)} \equiv M$ [p] (d'après le Théorème de Fermat)
2. Si M est un multiple de p alors $M \equiv 0[p]$ alors $M^{ed} \equiv 0[p]$ et $M^{ed} \equiv M[p]$

De même $M^{ed} \equiv M$ [q]
Donc $M^{ed} \equiv M$ [p] et $M^{ed} \equiv M$ [q], d'après le corollaire du théorème des restes chinois
$M^{ed} \equiv M$ [n]

ATTENTION !

e et d sont inversibles dans $\mathbb{Z}^*_{(p-1)(q-1)}$ et non dans $\mathbb{Z}^*_{pq}$ ce n'est pas pareil.

Sécurité

La sécurité de RSA repose sur la "difficulté" de factoriser des grands entiers

Exercice

La clé publique d'Alice est le couple (n,e) avec
n = 6324591032675721961071009838204690217216021474909186602746 01
e = 65537
On a intercepté un message chiffré avec la clé publique d'Alice
c = 637754170455445435942814163297673551558350335103827207359 73
On aimerait le déchiffrer.

1. Aller ici `https://pari.math.u-bordeaux.fr/gpwasm.html` pour factoriser n en utilisant la commande `factor()`. On trouve que $n = pq$ où p et q sont premiers
2. Or $c = m^e\ [n]$ et pour déchiffrer c on a besoin de trouver l'inverse d de e modulo $(p-1)(q-1)$
   Car $m = c^d\ [n]$
   (Utiliser `pow(c,-1,(p-1)*(q-1))` pour trouver d puis ...)

Rapidité = Chiffrement hybride

On combine RSA avec des systèmes rapides à clé secrète de la manière suivante si Alice veut envoyer un long message M à Bob

1. Elle choisit une clé courte K et chiffre M avec K pour obtenir C
2. Elle chiffre K avec la clé publique RSA de Bob
3. Elle transmet à Bob l'ensemble (C, $P_B(K)$)

4 Transport Security Layer

Nous avons vu le modèle en couches TCP/IP

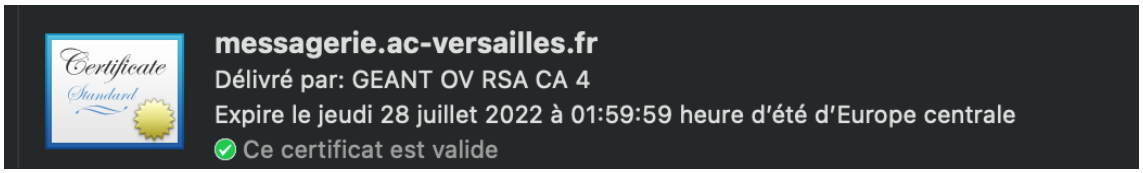
On ajoute une couche supplémentaire entre la couche Application et la couche Transport. Cette couche s'appelle la couche **Session**. C'est à ce niveau qu'on ajoute un en-tête TLS (Transport Security Layer)

Comment TLS se met en place ?

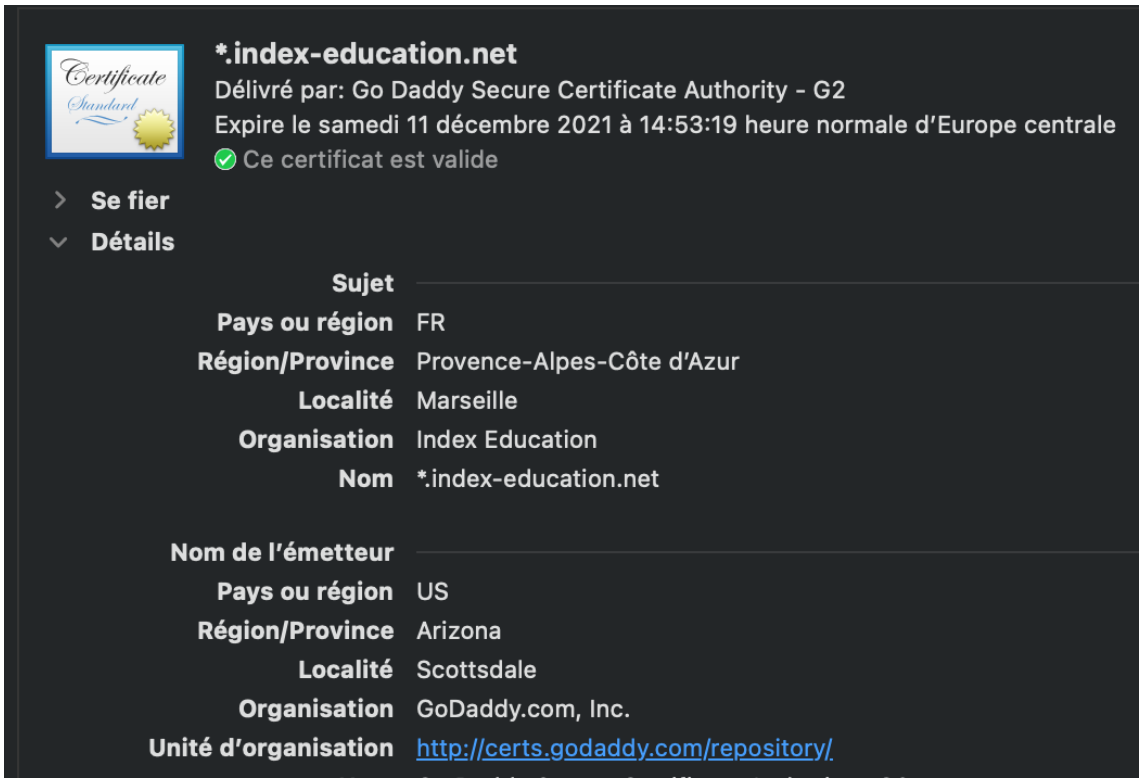
On parle du TLS handshake, et on peut le voir grâce à un logiciel de capture de paquets comme Wireshark

Time	Source	Destination	Protocol	Length	Info
153.342734	192.168.0.40	87.248.100.168	TLSv1…	583	Client Hello
153.364134	87.248.100.168	192.168.0.40	TCP	66	443 → 52006 [ACK] Seq=1 Ack=518 Win=30336 Len=0 TSva
153.364918	87.248.100.168	192.168.0.40	TLSv1…	1514	Server Hello, Change Cipher Spec, Application Data
153.364983	192.168.0.40	87.248.100.168	TCP	66	52006 → 443 [ACK] Seq=518 Ack=1449 Win=130304 Len=0
153.365913	87.248.100.168	192.168.0.40	TCP	1514	443 → 52006 [ACK] Seq=1449 Ack=518 Win=30336 Len=144
153.365919	87.248.100.168	192.168.0.40	TLSv1…	814	Application Data, Application Data, Application Data
153.365996	192.168.0.40	87.248.100.168	TCP	66	52006 → 443 [ACK] Seq=518 Ack=3645 Win=128832 Len=0
153.371315	192.168.0.40	87.248.100.168	TLSv1…	130	Change Cipher Spec, Application Data
153.372024	192.168.0.40	87.248.100.168	TLSv1…	910	Application Data
153.392200	87.248.100.168	192.168.0.40	TCP	66	443 → 52006 [ACK] Seq=3645 Ack=1426 Win=32000 Len=0
153.393105	87.248.100.168	192.168.0.40	TLSv1…	369	Application Data
153.393108	87.248.100.168	192.168.0.40	TLSv1…	369	Application Data
153.393268	192.168.0.40	87.248.100.168	TCP	66	52006 → 443 [ACK] Seq=1426 Ack=4251 Win=130432 Len=0
153.393453	87.248.100.168	192.168.0.40	TLSv1…	718	Application Data
153.393457	87.248.100.168	192.168.0.40	TLSv1…	804	Application Data

1. Etape 1 : Client Hello : Le client envoie un certain nombre d'informations au serveur en vue de réaliser un échange d'informations cryptées
2. Etape 2 : Serveur Hello : Le serveur répond (en bleu foncé) à ce stade le serveur envoie au client son certificat d'authentification. Voici l'exemple du certificat du webmail de l'académie de Versailles (on peut voir le certificat en cliquant sur le cadenas de la barre URL) :



Voici celui de Pronote :



Il envoie aussi un certain nombre d'informations sur les techniques de chiffrement Voici ci-dessous les informations envoyées par le serveur d'un hébergeur de site Web à un client avant une communication cryptée

Protocole :	TLS1.2	Chiffrement :	AES-256-GCM
Échange de clé :	ECDHE-SECP256R1-RSA-SHA384	Mac :	AEAD

3. Etape 3 : Il y a un autre aller-retour entre le client et le serveur pour finaliser le contrat de communication cryptée entre eux (**échange de clé (Diffie-Hellmann)** et autres informations)
Voici une copie d'écran du RFC 2246 de l'I.E.T.F concernant la version 1 de TLS (1999)

```
   Client                                               Server

   ClientHello                  -------->
                                                   ServerHello
                                                  Certificate*
                                            ServerKeyExchange*
                                           CertificateRequest*
                                <--------      ServerHelloDone
   Certificate*
   ClientKeyExchange
   CertificateVerify*
   [ChangeCipherSpec]
   Finished                     -------->
                                                [ChangeCipherSpec]
                                <--------             Finished
   Application Data             <------->     Application Data

          Fig. 1 - Message flow for a full handshake
```