Introduction à la Cryptologie

François Dahmani
Université Grenoble Alpes, Institut Universitaire de France
Master Mathématiques et Applications
M1 Maths Générales

2017

Table des matières

Préambule

Cette UE a pour ambition d'introduire des applications modernes (mais cependant classiques) de cryptologie de concepts mathématiques principalement issus de la théorie des nombres. Elle introduit a l'utilisation d'un outil de calcul formel libre (ici SageMath, mais Xcas serait tout aussi bien).

En principe cette UE se compose de 9 séances de 3h. La modalité de contrôle continu sera sans doute sous la forme d'un projet sur SageMath, à rendre en binôme, avec peut etre une breve interrogation orale (individuelle) pour exposer le protocole proposé.

Le programme ci-dessous détaille 9 séances thématiques, dont le découpage est corrélé au découpage calendaire des séances, mais ne lui correspond peut être pas exactement. Dans chaque séance thématique, quelques aspects plus orientés « cryptographie pratique » et/ou « calcul formel » pourront être abordés si le temps le permet. La référence principale est le livre de G. Zémor [4].

Vous êtes invités, préliminairement, et en autonomie, pendant le mois de janvier, à vous documenter sur l'historique de la cryptographie, et par exemple sur les codes de Vigenere, et les one-time-pads... Il existe de nombreuses sources d'information en ligne.

Vous êtes également invités à **installer** SageMath sur votre ordinateur personnel le plus rapidement possible (voir `http://www.sagemath.org`) [1], et, pourquoi pas, à faire quelques experimentations élémentaires de son interface, et de son aide. J'insiste sur le fait que le serveur en ligne de SageMath est bridé en calcul et ne permetra pas de manipuler des grands nombres comme on peut le souhaiter en cryptologie.

Il va de soi que ces notes sont un support rudimentaire de cours, et que rien n'y est garanti contre les fautes de frappes, les erreurs d'énoncé, et que je me réserve tout droit de changer d'avis quant à la bonne manière d'introduire un concept (ce qui dépendra du public, forcement différent de celui pour lequel ces notes ont été construites).

Gières, 2016,
François Dahmani.

1. Profitez en pour installer également Xcas, `https://www-fourier.ujf-grenoble.fr/~parisse/install_fr` comme cela vous pourrez comparer

1 Séance 1 : arithmétique modulaire, Les anneaux $\mathbb{Z}/m\mathbb{Z}$

1.1 Bref rappels sur $\mathbb{Z}/m\mathbb{Z}$ (en autonomie)

1.1.1 La division Euclidienne des nombres

La division Euclidienne des nombres (apprise en CM1 ?) nous dit que :

$$\forall n \in \mathbb{Z} \quad \forall m \in \mathbb{N} \setminus \{0\}$$
$$\exists! q, r \text{ tels que}$$
$$n = q \times m + r \qquad \text{et} \qquad 0 \leq r < m$$

Exemple : avec la "potence", on sait diviser 12345 par 12, le *quotient* vaut 1028, et le *reste* vaut 9.

On dit que "12349 vaut 9 modulo 12", et on ecrit

$$12345 \equiv 9 \,[\mathrm{mod}\, 12]$$

Observation 1.1.1 *On peut faire deux observations : si n_1 et n_1' ont tous les deux pour reste (par la division Euclidienne par m) le nombre r_1, et si n_2 et n_2' ont tous les deux pour reste r_2, alors*

- *$n_1 + n_2$ et $n_1' + n_2'$ ont tous les deux le même reste ;*
- *$n_1 n_2$ et $n_1' n_2'$ ont tous les deux le même reste.*

L'anneau $\mathbb{Z}/m\mathbb{Z}$ est **l'ensemble des restes possibles par la division Euclidienne par** m, muni des deux opérations dessus : l'addition des restes, et la multiplication des restes. Les regles de bases de l'addition et de la multiplication sont valables (commutativité, associativité, distributivité, 0 est neutre pour l'addition, 1 est neutre pour la multiplication...). On dit que l'on à affaire à une structure d'anneau.

Comme $-1 \equiv m - 1 \,[\mathrm{mod}\, m]$, et même $-r \equiv m - r \,[\mathrm{mod}\, m]$, on pourra ecrire et penser à ces nombres négatifs comme des élements de $\mathbb{Z}/m\mathbb{Z}$ (bien que techniquement ce ne sont pas des restes de division Euclidienne). La manière mathématique de remedier à ce probleme consiste à définir $\mathbb{Z}/m\mathbb{Z}$ comme le *quotient* de l'anneau $\mathbb{Z}$ par l'idéal $m\mathbb{Z}$, ainsi ses éléments sont des classes déquivalences, et on a la possibilité de prendre le representant que l'on préfeère.

On peut "graphiquement" representer $\mathbb{Z}/m\mathbb{Z}$ sur un cercle, avec des points regulierement espacés. On voit alors l'addition comme l'addition des longueurs d'arcs issus du point 0, et la multiplication comme la juxtapositions de n_1 arcs de longueur égale a celle de l'arc $(0, n_2)$...

On peut dessiner ce qui se passe pour $\mathbb{Z}/12\mathbb{Z}$... On peut avoir un sens de ce qui se passe pour $\mathbb{Z}/12543234355643456\mathbb{Z}$! !

1.1.2 $\mathbb{Z}/p\mathbb{Z}$ quand p est premier

Tout se passe comme pour des nombres ?... **Mais !** parfois $n_1 n_2 \equiv 0 \,[\mathrm{mod}\, m]$ sans que n_1 ou n_2 soit nul (par exemple 2×6 modulo 12). et parfois $x^2 \equiv -1 \,[\mathrm{mod}\, m]$. Il n'y a pas de notion de $\leq$ ou $\geq$ interessante dans $\mathbb{Z}/m\mathbb{Z}$

Toutefois, si m est un nombre premier, il n'y a pas de "diviseurs de zero", et tout élément non nul est inversible : pour tout $n \neq 0$ dans $\mathbb{Z}/m\mathbb{Z}$ pour m **premier**, il existe n' qu'on peut noter n^{-1}, aussi dans $\mathbb{Z}/m\mathbb{Z}$, tel que $n \times n^{-1} \equiv 1 \,[\mathrm{mod}\, m]$. On dit dans ce cas que $\mathbb{Z}/p\mathbb{Z}$ (pour p premier) est un **corps**.

Ainsi, dans $\mathbb{Z}/p\mathbb{Z}$ quand p est premier :

- si $n \times k = 0$ alors n ou k vaut 0
- tout element non nul n a un "inverse" n^{-1}
- un element donné a, au maximum, deux racines carrées (peut-être n'en a-t-il pas du tout)
- Plus généralement, tout polynôme $P(X)$ de degré d a au plus d racines. Elles sont simples si P et P' n'ont pas de racine commune.

1.2 Le Théorème Chinois

Grace à l'algorithme d'Euclide, il est (facilement, comme on le verra !) possible de savoir si deux nombres sont premiers entre eux, et même de calculer le pgcd de deux nombres.

Il s'agit de l'algorithme suivant : on a deux nombres a et $b < a$. On reduit a modulo b. Il reste r. La nouvelle valeur de a est b, la nouvelle valeur de b est r, et on recommence. Lorsque r prend la valeur 0, la valeur courante de b est le pgcd des nombres initialements donnés.

Théorème 1.2.1 *Soient n et m des nombres premiers entre eux (c'est à dire qui n'ont pas de facteur premier commun)*

Alors l'application

$$\mathbb{Z}/(n \times m)\mathbb{Z} \to \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}$$

qui a un element r associe le couple (r_1, r_2) de ses restes modulo n et modulo m, est un isomorphisme d'anneau :

$$\mathbb{Z}/(n \times m)\mathbb{Z} \simeq \mathbb{Z}/n\mathbb{Z} \times \mathbb{Z}/m\mathbb{Z}.$$

Plus généralement, si $n_1, n_2 \ldots n_k$ sont premiers entre eux deux à deux, alors

$$\mathbb{Z}/(n_1 \times n_2 \times \ldots n_k)\mathbb{Z} \simeq \mathbb{Z}/n_1\mathbb{Z} \times \mathbb{Z}/n_2 \cdots \times \mathbb{Z}/n_k\mathbb{Z}.$$

Sous cette forme l'interet "pratique" ne saute peut-etre pas directement aux yeux... Nous allons l'illustrer.

Ce théorème, attribué à Sun Tzu, astronome Chinois entre les années 300 et 500.

Voici une application (à peine fictive).

L'Empereur de Chine veut compter ses soldats après une bataille. Sur les conseils de Sun Tzu, il procede ainsi.

- Il leur ordonne de se ranger par colonnes de 5. Tout le monde est bien en rang.
- Il leur ordonne de se ranger par colonnes de 7. Il en reste 3.
- Il leur ordonne de se ranger par colonnes de 11. Il en reste 3.
- Il leur ordonne de se ranger par colonnes de 13. Tout le monde est bien en rang.

Par ailleurs il sait qu'il a moins de 5 000 soldats. Il demande à Sun Tzu s'il peut donner le nombre exact de soldats survivants : Sun Tzu lui dit :

"Oui bien sûr Sire, leur nombre est bien sûr égal à leur nombre "modulo $5 \times 7 \times 11 \times 13 (= 5005)$", ainsi grace à mon brillant théorème, dès que j'aurai trouvé x, y, z, t verifiant

$$\begin{array}{llll} x \equiv 1 \,[\mathrm{mod}\, 5], & x \equiv 0 \,[\mathrm{mod}\, 7], & x \equiv 0 \,[\mathrm{mod}\, 11], & x \equiv 0 \,[\mathrm{mod}\, 13] \\ y \equiv 0 \,[\mathrm{mod}\, 5], & y \equiv 1 \,[\mathrm{mod}\, 7], & y \equiv 0 \,[\mathrm{mod}\, 11], & y \equiv 0 \,[\mathrm{mod}\, 13] \\ z \equiv 0 \,[\mathrm{mod}\, 5], & z \equiv 0 \,[\mathrm{mod}\, 7], & z \equiv 1 \,[\mathrm{mod}\, 11], & z \equiv 0 \,[\mathrm{mod}\, 13] \\ t \equiv 0 \,[\mathrm{mod}\, 5], & t \equiv 0 \,[\mathrm{mod}\, 7], & t \equiv 0 \,[\mathrm{mod}\, 11], & t \equiv 1 \,[\mathrm{mod}\, 13], \end{array}$$

il suffira de prendre $N = 0 \times x + 3 \times y + 3 \times z + 0 \times t$, et de prendre son reste modulo 5005, cela vous donnera le nombre exact de vos soldats.

Or il se trouve que je sais qui peuvent etre x, y, z, t (cela ne dépend pas de vos soldats, et je l'ai calculé avant de venir (ça n'etait pas bien dur)) :

$$x = (7 \times 11 \times 13) \qquad y = (5 \times 11 \times 13) \qquad z = (5 \times 7 \times 13) \times 3 \qquad t = (5 \times 7 \times 11) \times 5.$$

C'est à dire

$$x = 1001 \qquad y = 715 \qquad z = 1365 \qquad t = 1925.$$

Je termine donc le calcul : $N = 3 \times y + 3 \times z = 6240 \equiv 1235 \,[\mathrm{mod}\, 5005]$.

Il vous reste 1235 soldats Sire."

Que pensez vous de cette méthode ? Pourquoi marche-t-elle ? Comment Sun Tzu a-t-il pu si facilement trouver ses elements clé x, y, z, t ?...

D'autres applications classiques concernent la prévision des prochaines concordances de cycles (par exemple, sachant que la lune fait un tour de Terre en 29 jours, et un an dure 365 jours, si le jour de naissance de l'Empereur est une pleine lune, combien d'autre pleine lunes verra-t-il pour ses anniversaires ?)

1.2.1 Application : un cas de secret partagé

Une petite histoire (inventée).

La fille de Sun Tzu et le fils de l'Empereur se fréquentent en cachette. Ils forment des voeux de mariage, et, dans un coffre à cadenas à code (8 chiffres !) ils déposent des diamants sacrés se mettant d'accord pour qu'ils ne soient sortis du coffre qu'à l'occasion de leur mariage, en particulier seulement s'ils sont présents tous les deux, et tous les deux d'accord pour le faire.

La fille de Sun Tzu propose : "Choisis un nombre modulo 3*, et un autre modulo* 7 *que tu ne me diras pas. Pour ma part, je choisis des nombres modulo* 839*, et modulo* 2081 *et je ne te*

les dirai pas. Nous les dirons ensuite séparément à ce serviteur, qui, instruit par la science de mon père, scellera le cadenas par un nombre[2] *qui possède les résidus que nous aurons choisis.*

Ensuite nous truciderons ce serviteur, car il en saura trop...

Enfin, quand nous aurons organisé enfin notre mariage, nous viendrons tous les deux près du coffre, nous nous dévoileront nos secrets, et nous pourrons retrouver le nombre du serviteur, et revêtir les diamants sacrés en parure."

Est-ce que ça marche ? Le serviteur pourra-t-il (facilement !) trouver un nombre à 8 chiffres qui répond aux exigences des jeunes gens ? Le pourrait-t-il avec seulement 7 chiffres ? A-t-il de nombreux choix ? Est-ce que cela répond aux exigences de sécurité fixés par les jeunes amoureux ? (Est-ce que, par exemple, le fils de l'Empereur ne pourrait pas ouvrir le coffre seul afin de revendre les diamants au marché noir ?)

Plus tard le fils de l'Empereur part à la guerre, meurt. La fille de Sun Tzu (après avoir fait son deuil) tombe amoureuse d'un aventurier Tatar qui passait par là. Elle veut partir avec lui et surtout avec les diamants sacrés qu'elle a enfermés dans le coffre. Comme l'aventurier lui fait remarquer que son coffre légendaire est bien connu dans le royaume, qu'on le dit inviolable, et que, le fils de l'Empereur etant mort, elle ne pourra plus jamais l'ouvrir, elle hausse les épaules.

Que veut-elle dire ? ?

1.3 Le symbole de Legendre, et de Jacobi

Théorème 1.3.1 *(Petit théorème de Fermat) Si p est premier, et $x \not\equiv 0 \,[\mathrm{mod}\, p]$, on a*

$$(x)^{p-1} \equiv 1 \,[\mathrm{mod}\, p].$$

Je vous invite à visiter :

`en.wikipedia.org/wiki/Proofs_of_Fermat's_little_theorem`

Une preuve simple consiste à dire : il y a $p-1$ éléments non nuls de $\mathbb{Z}/p\mathbb{Z}$, et ils sont tous inversibles, donc cet ensemble porte une structure de groupe pour la multiplication. L'element x engendre un sous-groupe $\langle x \rangle$, dont les classes $g \times \langle x \rangle$ partitionnent le groupe entier. Ainsi le cardinal r de $\langle x \rangle$ divise $p-1$, et comme $x^r \equiv 1$, on a $(x)^{p-1} \equiv 1 \,[\mathrm{mod}\, p]$.

Si p est un nombre premier, et si $n \in \mathbb{Z}/p\mathbb{Z}, n \neq 0$, on définit le **symbole de Legendre** de n pour p, noté $\left(\frac{n}{p}\right)$, comme étant 1 si n est un carré dans $\mathbb{Z}/p\mathbb{Z}$, et -1 sinon.

Théorème 1.3.2 *(Euler) Si p est premier, impair, et si $x \not\equiv 0 \,[\mathrm{mod}\, p]$, alors on a l'équivalence :*

$$x \text{ est un carré de } \mathbb{Z}/p\mathbb{Z} \qquad \Longleftrightarrow \qquad (x)^{\frac{p-1}{2}} \equiv 1 \,[\mathrm{mod}\, p].$$

2. à moins de 8 chiffres... il faut suivre !

En effet si $x = y^2$ (sous entendu modulo p), on a $x^{(p-1)/2} = y^{p-1}$ qui vaut 1 par "Petit Fermat".

Dans l'autre sens, tout carré est donc racine du polynôme $(X^{(p-1)/2} - 1)$. Il s'en suit qu'il n'y a pas plus de $(p-1)/2$ carrés dans $\mathbb{Z}/p\mathbb{Z} \setminus \{0\}$. Maintenant, je dis qu'il y a exactement ce nombre, car les z^2 pour $z = 1, 2, \ldots, (p-1)/2$ sont tous differents. En effet, sinon $(z_i^2 - z_j^2) = 0$ et donc $(z_i - z_j)(z_i + z_j) = 0$, et donc (comme p est premier), l'un des facteurs est nul. Ca ne peut pas etre le second, car z_i et z_j sont pris parmi $1, 2, \ldots, (p-1)/2$. Donc $z_i = z_j$ et donc $i = j$. Il y a donc bien exactement $(p-1)/2$ carrés, et tous parmi les $(p-1)/2$ racines de $(X^{(p-1)/2} - 1)$, donc toutes les racines du polynôme (dans $\mathbb{Z}/p\mathbb{Z}$), sont des carrés.

Observation 1.3.1 *(Calculer des racines carrées modulo un premier)*

- *Si p est un nombre premier et si $p \equiv 3 \,[\mathrm{mod}\, 4]$, et si a est un carré (mod p), alors $x = a^{(p+1)/4}$ est une racine carrée de a. L'autre racine carrée vaut $-a^{(p+1)/4}$.*
  (Calculer x^2 pour le verifier, en utilisant le théorème précédent).
- *Si p est premier et si $p \equiv 5 \,[\mathrm{mod}\, 8]$, et si a est un carré, et si $a^{(p-1)/4} \equiv 1 \,[\mathrm{mod}\, p]$ alors $x = a^{(p+3)/8}$ est une racine carrée de a.*
  Si à la place de la dernière hypothèse, on a $a^{(p-1)/4} \equiv -1 \,[\mathrm{mod}\, p]$, alors $x = (4a)^{(p+3)/8}/2$ convient.

Si $m = p \times q$ avec p et q premiers, et si $n \in \mathbb{Z}/m\mathbb{Z}, n \neq 0$, le **symbole de Jacobi** de n pour m noté $\left(\frac{n}{m}\right)$, est **le produit des symboles de Legendre** sur p et sur q.

Observation 1.3.2 *Si $m = pq$ avec p et q premiers, et $p \equiv q \equiv 3 \,[\mathrm{mod}\, 4]$, alors*

$$\left(\frac{-1}{p}\right) = \left(\frac{-1}{q}\right) = -1.$$

De plus, pour tout $u \in \mathbb{Z}/m\mathbb{Z}$, $\left(\frac{u}{m}\right) = \left(\frac{-u}{m}\right)$.

En effet, $(p-1)/2$ est impair, et donc par Euler, -1 n'est pas un carré modulo p (et de même pour q). Le symbole de Jacobi est un produit des deux symboles de Legendre. Si les deux changent de signe, les changements se neutralisent.

Théorème 1.3.3 *Soit p et q deux nombres premiers congrus à 3 modulo 4. On pose $n = p \times q$.*

Alors, si y est un carré modulo n, il possède quatre racines carrées dans $\mathbb{Z}/n\mathbb{Z}$ $u, -u, v, -v$ qui verifient

- $\left(\frac{u}{p}\right) = \left(\frac{u}{q}\right) = 1$,
- $\left(\frac{v}{p}\right) = -1$ *et* $\left(\frac{v}{q}\right) = 1$.

Observation 1.3.3 *Si l'on connait n comme dans le théorème, et si l'on connait u, v de symbole de Jacobi opposés et de même carré, alors il est facile de retrouver p et q (c'est à dire de factoriser n).*

En effet, on a $u^2 - v^2 \equiv 0 \,[\mathrm{mod}\, n]$ donc $(u-v)(u+v) \equiv 0 \,[\mathrm{mod}\, n]$.

Comme $u \neq v$ et $u \neq -v$ (ils ont des symboles de Jacobi differents) cela signifie que aucun des facteurs n'est nul. Ainsi cela veut dire que l"entier $(u-v)(u+v)$ est un multiple de n,

donc est divisible par p et q. Comme $u-v$ n'est pas divisible par p (u et v ont des symboles de Legendre differents sur p), c'est forcement $(u+v)$ qui l'est. De même, $(u+v)$ n'est pas divisible par q (u et $-v$ ont des symboles de Legendre differents sur q), $(u-v)$ est donc divisible par q. Il suffit maintenant de calculer le pgcd de n et de $(u-v)$: c'est q, celui de n et $(u+v)$ vaut p. L'algorithme d'Euclide est efficace pour cela.

1.3.1 Application : Protocole de Blum

Alice et Bob, pour des raisons qui leur appartiennent, veulent jouer à "Pile ou Face" l'un contre l'autre, par téléphone, mais chacun craint que l'autre ne triche.

Alice doit tirer pile ou face, et Bob doit deviner. La partie se joue en 100 coups.

Voici un protocole qu'ils décident de suivre, apres en avoir entendu parler dans une UE d'introduction à la cryptologie, à Grenoble.

- Etape 1 : Bob choisit p et q deux premiers differents congrus à 3 modulo 4. Il calcule $n = pq$ et le communique à Alice.
- Etape 2 : Alice choisit ses coups. Elle tire au hasard 100 elements de $\mathbb{Z}/n\mathbb{Z}$. Elle note pour elle même $x_1, \ldots, x_{100}$ ces éléments.
- Etape 3 : Alice calcule dans $\mathbb{Z}/n\mathbb{Z}$ les valeurs $x_1^2, \ldots, x_{100}^2$. Elle communique ces valeurs à Bob.
- Etape 4 : Bob doit répondre (en temps limité) une suite $\epsilon_1, \ldots \epsilon_{100}$ de 1 et de -1. On se met d'accord pour dire que Bob gagne le i-ème coup si ϵ_i, le i-ème terme de sa suite, est le symbole de Jacobi de x_i.
- Etape 5 : Alice révèle les valeurs de $x_1, \ldots, x_{100}$.
- Etape 6 : Bob verifie qu'il n'y a pas d'embrouille trop évidente : les valeurs des x_i^2 doivent bien être celles qu'il a reçues au début.
- Etape 7 : Bob révèle les valeurs de p et q. Alice vérifie que effectivement $n = pq$, et qu'ils sont congrus à 3 modulo 4. Elle verifie aussi que p et q sont bien premiers...
- Etape 8 : Alice et Bob calculent les symboles de Legendre de x_i modulo p, modulo q, en déduisent les symboles de Jacobi modulo n. Ils comparent à la suite des ϵ_i et ils comptent les points.

Que pensez vous de ce protocole ?

Comment est-ce que Alice pourrait tricher ?

Comment est-ce que Bob pourrait tricher ?

Ce protocole sert en pratique à établir entre deux personnes une suite de nombres (il s'agit des valeurs 1, ou 0, de réussite ou non des paris consécutifs de Bob) aléatoires et dont les deux partenaires ont une garantie du caractère aléatoire. C'est une "source d'aléatoire en laquelle ils peuvent avoir tous les deux confiance".

Prenons un exemple : Alice veut biaiser sa suite de nombre de sorte que, par exemple, aux indices multiples de 3, la valeur 1 soit plus probable que -1 (sans que Bob ne le sache). Elle peut le faire. Mais Bob peut independamment forcer l'annulation de ce biais en prenant ses réponses vraiment aléatoire, ou encore le transformer en un biais inconnu d'Alice et de lui, en utilisant des réponses biaisées lui-même.

1.4 Exercices

1. Comment trouver les élḿents x, y, z, t de Sun Tzu en toute circonstances (pour des résidus connus sur des nombres premiers) ?
2. Que dire de l'énoncé du Théorème Chinois si n et m ne sont pas premiers entre eux ?
   En quoi par exemple $\mathbb{Z}/6\mathbb{Z} \times \mathbb{Z}/10\mathbb{Z}$ n'est pas la même chose que $\mathbb{Z}/60\mathbb{Z}$.
3. (Folie des grandeurs) Sur SageMath définir $a = mod(1234, 123)$ et $b = 1234$. Comparer le temps de calcul de a^{10^k+1} et de $mod(b^{10^k+1}, 123)$ pour $k = 2, 3, 4, 5, 6, 7$ et peut être 8.
   Que se passe-t-il selon vous ? Quels sont les types de a et de b ?
4. (Folie des grandeurs 2) Trouver des nombres premiers de 30 chiffres (puis de 35, puis de 40 chiffres). Définir m le produit de deux d'entre eux, consécutifs (nextprime()). Demandez à SageMath de factoriser le nombre m obtenu. (factor(m)).
   Cela donne une idée de la taille des nombres à prendre pour assurer la sécurité du Protocole de Blum.
5. Faire ecrire à SageMath une liste de nombres premiers congrus à 3 modulo 4, et plutot grands. (Disons, de l'ordre de 10^{45} ?)
   Multipliez deux d'entre eux, cela donne N. Trouver des elements qui ont symbole de Jacobi 1 et d'autres qui ont symbole de Jacobi -1 modulo N. On voudrait des listes (pas exhaustives, mais avec du choix !...)
6. Verifier qu'on calcule facilement des racines carrés dans $\mathbb{Z}/p\mathbb{Z}$ pour $p \equiv 3 \,[\mathrm{mod}\, 4]$ et aussi pour des nombres premiers $p \equiv 5 \,[\mathrm{mod}\, 8]$ que l'on aura pris soin de trouver dans une liste assez grande.
7. Faire un scenario à la Sun Tzu, assisté par SageMath. Par exemple jouez le rôle du serviteur (jusqu'à un certain point seulement !) et déterminez un code de cadenas suivant ce que votre partenaire aura décidé comme résidus. Jouez le rôle de la fille et essayez de retrouver le code avec l'information partielle dont vous disposez. (Faire une boucle qui testera vos essais contre le code du cadenas défini par le serviteur). Vous avez un ordinateur, ce que les personnages de mon conte n'avaient pas. Donc vous pouvez prendre des nombres premiers GRANDS, et faire de nombreuses tentatives.
8. Mettre en place un jeu de pile ou face (binome contre binome) sur SageMath, par le protocole de Blum.

2 Séance 2 : Prise en main SageMath

Je vous rappelle que Vous êtes invités à **installer** SageMath sur votre ordinateur personnel le plus rapidement possible (voir `http://www.sagemath.org`). Profitez en pour installer également Xcas, `https://www-fourier.ujf-grenoble.fr/~parisse/install_fr` comme cela vous pourrez comparer.

Depuis un terminal (clic droit quelque part sur le bureau –¿ ouvrir un terminal), lancez le navigateur Chromium (« chromium-browser & ») puis, toujours depuis le terminal, lancer sagemath (« sagemath ») puis entrez « notebook() ». En principe dans la fenetre de votre navigateur chromium, vous obtenez une interface pour ouvrir, editer, sauvegarder des feuilles de travail Sage.

Voici une capture d'une session, ou differentes fonctions sont abordées. Vous êtes encouragés à utiliser l'aide, et les forums de documentations.

J'ai aussi inclu les excellentes fiches de P. Jipsen et W. Stein que l'on trouve à l'adresse `http://wiki.sagemath.org/quickref`.

Prise en main

```
#   On peut travailler dans des corps finis
K=FiniteField(3)
L.<a>=FiniteField(27)
print(K)
print(L)
print(type(K))
print(type(L))
print(L.random_element())
print(L.random_element()+a)
print type(a)
# Le polynome minimal de a est choisi par defaut, mais on peut le
specifier. (cf finite field givaro)
```

```
Finite Field of size 3
Finite Field in a of size 3^3
<class
'sage.rings.finite_rings.finite_field_prime_modn.FiniteField_prime
odn_with_category'>
<class
'sage.rings.finite_rings.finite_field_givaro.FiniteField_givaro_wi
_category'>
a^2 + 2*a + 2
a^2 + 2
<type
'sage.rings.finite_rings.element_givaro.FiniteField_givaroElement'
t;
```

```
# On peut travailler dans l'anneau des entiers, et dans des
quotients Z/nZ
Z=Integers
R=Integers(22)
print(R)
print(type(R))
print(Z)
```

```
Ring of integers modulo 22
<class
'sage.rings.finite_rings.integer_mod_ring.IntegerModRing_generic_w
h_category'>
<class
'sage.rings.finite_rings.integer_mod_ring.IntegerModFactory'>
```

```
# On peut manipuler des matrices
K=FiniteField(29)
M=matrix(K,4,7, K.random_element())
N=matrix(K,7,1,K.random_element())
print(M)
print(N)
```

```
[26  0  0  0  0  0  0]
[ 0 26  0  0  0  0  0]
[ 0  0 26  0  0  0  0]
[ 0  0  0 26  0  0  0]
[6]
[0]
[0]
[0]
[0]
[0]
[0]
```

```
M*N
```

```
[11]
[ 0]
[ 0]
[ 0]
```

```
# Outils de primalité
print next_prime(100)
print is_prime(57)
print factor(12345)
m=R.random_element()
mtilde=lift(m)
print m
print mtilde
print is_prime(m)
is_prime(mtilde)
```

```
101
False
3 * 5 * 823
21
21
False
False
```

```
x=R(454348)
print('x equals')
print(x)
print(type(x))
y=lift(x)
print('y equals ')
print(y)
print(type(y))
print "x égal %s et y égal %s" %(x,y)
```

```
x equals
4
<type 'sage.rings.finite_rings.integer_mod.IntegerMod_int'>
y equals
4
<type 'sage.rings.integer.Integer'>
x égal 4 et y égal 4
```

```
X=2.1
Y=7.124*pi
print "X égal %s et Y égal %s" %(X,Y)
```

```
X égal 2.10000000000000 et Y égal 7.12400000000000*pi
```

```
h=L.random_element()
k=L.random_element()
```

```
print(h+k)
parent(h+k)
```

```
a^2 + a
Finite Field in a of size 3^3
```

```
S = IntegerModRing(5) # SAME THING AS Integers(5)
S.category()
```

```
Join of Category of finite commutative rings and Category of
subquotients of monoids and Category of quotients of semigroups an
Category of finite enumerated sets
```

```
# Rediger des conditions
A=True
B=False
print A==True
if A==True and B==True:
    print 'Yeah'
elif A==True or B==True:
    print 'almost'
else:
    print 'nope.'
```

```
True
almost
```

```
# Definir des fonctions
def add2(x):
    y=x+2
    return y
```

```
# et meme, elle fonctionne
add2(17)
```

```
19
```

```
#Autre fonctions
squaring = lambda x: x*x
```

```
squaring(3)
```

```
9
```

```
f(x)= x^3
```

```
print f
print f(3)
```

```
x |--> x^3
27
```

```
```

Sage Quick Reference (Basic Math)

Peter Jipsen, version 1.1
latest version at `wiki.sagemath.org/quickref`

But : relier les notations standard aux commandes Sage

Interface web (et interface texte)

Pour évaluer une cellule : ⟨shift-enter⟩
com⟨tab⟩ essaye de compléter la *commande*
commande`?`⟨tab⟩ montre la documentation
commande`??`⟨tab⟩ montre la source
`a.`⟨tab⟩ montre toutes les méthodes pour l'objet `a`
`search_doc('`*chaîne ou regexp*`')` cherche dans la doc.
`search_src('`*chaîne ou regexp*`')` cherche dans les sources
`lprint()` bascule le mode sortie LaTeX
`version()` donne la version de Sage
Insérer une cellule : cliquer sur la ligne bleue
Supprimer une cellule : supprimer le contenu puis backspace

Types numériques

Entiers : $\mathbb{Z}$ = `ZZ` par ex. `-2 -1 0 1 10^100`
Rationnels : $\mathbb{Q}$ = `QQ` par ex. `1/2 1/1000 314/100 -42`
Décimaux : $\mathbb{R} \approx$ `RR` par ex. `.5 0.001 3.14 -42.`
Complexes : $\mathbb{C} \approx$ `CC` par ex. `1+i 2.5-3*i`

Constantes et fonctions de base

Constantes : π = `pi` e = `e` i = `i` ∞ = `oo`
Approximation : `pi.n(digits=18)` = 3.14159265358979324
Fonctions : `sin cos tan sec csc cot sinh cosh tanh sech csch coth log ln exp`
ab = `a*b` $\frac{a}{b}$ = `a/b` a^b = `a^b` $\sqrt{x}$ = `sqrt(x)`
$\sqrt[n]{x}$ = `x^(1/n)` $|x|$ = `abs(x)` $\log_b(x)$ = `log(x,b)`
Variables symboliques : `t,u,v,y,z = var('t u v y z')`
Définir une fonction : par ex. $f(x) = x^2$ `f(x)=x^2`
ou `f=lambda x: x^2` ou `def f(x): return x^2`

Opérations sur les expressions

`factor(...)` `expand(...)` `(...).simplify_...`
Équations symboliques : `f(x)==g(x)`
`_` est le résultat précédent
Résoudre $f(x) = g(x)$: `solve(f(x)==g(x),x)`
`solve([f(x,y)==0, g(x,y)==0], x,y)`
`find_root(f(x), a, b)` trouve $x \in [a,b]$ t.q. $f(x) \approx 0$
$\sum_{i=k}^{n} f(i)$ = `sum([f(i) for i in [k..n]])`
$\prod_{i=k}^{n} f(i)$ = `prod([f(i) for i in [k..n]])`

Calcul différentiel et intégral

$\lim_{x\to a} f(x)$ = `limit(f(x), x=a)`
$\lim_{x\to a^-} f(x)$ = `limit(f(x), x=a, dir='minus')`
$\lim_{x\to a^+} f(x)$ = `limit(f(x), x=a, dir='plus')`
$\frac{d}{dx}(f(x))$ = `diff(f(x),x)`
$\frac{\partial}{\partial x}(f(x,y))$ = `diff(f(x,y),x)`
Dériver : `diff` = `differentiate` = `derivative`
$\int f(x)dx$ = `integral(f(x),x)`
Intégrer : `integral` = `integrate`
$\int_a^b f(x)dx$ = `integral(f(x),x,a,b)`
Dev. de Taylor, ordre n en a: `taylor(f(x),x,`a`,`n`)`

Graphiques dans le plan

`line([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*options*`)`
`polygon([(`x_1`,`y_1`),...,(`x_n`,`y_n`)],`*options*`)`
`circle((`x`,`y`),`r`,`*options*`)`
`text("txt",(`x`,`y`),`*options*`)`
options comme dans `plot.options`, par ex. `thickness=`*pixel*,
`rgbcolor=(`r`,`g`,`b`)`, `hue=`h avec $0 \le r,b,g,h \le 1$
utiliser l'option `figsize=[w,h]` pour ajuster le rapport largeur/hauteur
`plot(f(x),`$x_{\min}$`,`$x_{\max}$`,`*options*`)`
`parametric_plot((f(`t`),g(`t`)),`$t_{\min}$`,`$t_{\max}$`,`*options*`)`
`polar_plot(f(`t`),`$t_{\min}$`,`$t_{\max}$`,`*options*`)`
pour combiner : `circle((1,1),1)+line([(0,0),(2,2)])`
`animate(`*liste d'objets graphiques, options*`).show(delay=20)`

Graphiques dans l'espace

`line3d([(`x_1`,`y_1`,`z_1`),...,(`x_n`,`y_n`,`z_n`)],`*options*`)`
`sphere((`x`,`y`,`z`),`r`,`*options*`)`
`tetrahedron((`x`,`y`,`z`),`*size*`,`*options*`)`
`cube((`x`,`y`,`z`),`*size*`,`*options*`)`
options par ex. `aspect_ratio=[1,1,1] color='red' opacity`
`plot3d(f(`x,y`),[`x_b`,`x_e`],[`y_b`,`y_e`],`*options*`)`
ajouter l'option `plot_points=[`m,n`]` ou utiliser `plot3d_adaptive`
`parametric_plot3d((f(`t`),g(`t`),h(`t`)),[`t_b`,`t_e`],`*options*`)`
`parametric_plot3d((f(`u,v`),g(`u,v`),h(`u,v`)),`
`[`u_b`,`u_e`],[`v_b`,`v_e`],`*options*`)`
utiliser + pour combiner des objets graphiques

Math. discrètes

Partie entière $\lfloor x \rfloor$ = `floor(x)` $\lceil x \rceil$ = `ceil(x)`
Reste de la division de n par k = `n%k` $k|n$ ssi `n%k==0`
$n!$ = `factorial(n)` $\binom{x}{m}$ = `binomial(x,m)`
ϕ = `golden_ratio` $\phi(n)$ = `euler_phi(`n`)`
Chaînes : `s = 'Salut' = "Salut" = ""+"Sa"+'lut'`
`s[0]='S'` `s[-1]='t'` `s[1:3]='al'` `s[3:]='ut'`
Listes : par ex. `[1,'Salut',x] = []+[1,'Salut']+[x]`
Tuples : par ex. `(1,'Salut',x)` (non mutable)
Ensembles : $\{1,2,1,a\}$ = `Set([1,2,1,'a'])` $(= \{1,2,a\})$
Création de liste $\approx$ notation ensembliste, par ex.
$\{f(x) : x \in X, x > 0\}$ = `Set([f(x) for x in X if x>0])`

Algèbre linéaire

$\begin{pmatrix}1\\2\end{pmatrix}$ = `vector([1,2])`, $\begin{pmatrix}1&2\\3&4\end{pmatrix}$ = `matrix([[1,2],[3,4]])`
$\begin{vmatrix}1&2\\3&4\end{vmatrix}$ = `det(matrix([[1,2],[3,4]]))`
Av = `A*v` A^{-1} = `A^-1` A^t = `A.transpose()`
méthodes: `nrows() ncols() nullity() rank() trace()`...
nbr de colonnes, nbr de lignes, dim. noyau, rang, trace

Modules et paquetages

`from` *nom_module* `import *` (beaucoup sont préchargés)
`calculus coding combinat crypto functions games geometry graphs groups logic matrix numerical plot probability rings sets stats`
`sage.`*nom_module*`.all.`⟨tab⟩ montre les commandes
Paquetages standards : Maxima GP/PARI GAP Singular R ...
Paquetages opt. : Biopython Fricas(Axiom) Gnuplot ...
`%`*nom_paquetage* pour charger
`time` *commande* pour montrer la durée du calcul

Sage Quick Reference:
Elementary Number Theory

William Stein

Sage Version 3.4

http://wiki.sagemath.org/quickref

Everywhere $m, n, a, b, etc.$ are elements of ZZ

ZZ = $\mathbf{Z}$ = all integers

Integers

$\ldots, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, \ldots$

n divided by m has *remainder* `n % m`

`gcd(n,m)`, `gcd(`*list*`)`

extended gcd $g = sa + tb = \gcd(a, b)$: `g,s,t=xgcd(a,b)`

`lcm(n,m)`, `lcm(`*list*`)`

binomial coefficient $\binom{m}{n}$ = `binomial(m,n)`

digits in a given base: `n.digits(`*base*`)`

number of digits: `n.ndigits(`*base*`)`

(*base* is optional and defaults to 10)

divides $n \mid m$: `n.divides(m)` if $nk = m$ some k

divisors – all d with $d \mid n$: `n.divisors()`

factorial – $n!$ = `n.factorial()`

Prime Numbers

$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, \ldots$

factorization: `factor(n)`

primality testing: `is_prime(n)`, `is_pseudoprime(n)`

prime power testing: `is_prime_power(n)`

$\pi(x) = \#\{p : p \leq x \text{ is prime}\}$ = `prime_pi(x)`

set of prime numbers: `Primes()`

$\{p : m \leq p < n \text{ and } p \text{ prime}\}$ =`prime_range(m,n)`

prime powers: `prime_powers(m,n)`

first n primes: `primes_first_n(n)`

next and previous primes: `next_prime(n)`,
`previous_prime(n)`, `next_probable_prime(n)`

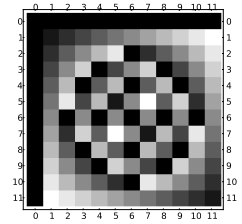
prime powers:
`next_prime_power(n)`, `pevious_prime_power(n)`

Lucas-Lehmer test for primality of $2^p - 1$

```
def is_prime_lucas_lehmer(p):
    s = Mod(4, 2^p - 1)
    for i in range(3, p+1): s = s^2 - 2
    return s == 0
```

Modular Arithmetic and Congruences

```
k=12; m = matrix(ZZ, k, [(i*j)%k for i in [0..k-1] for j in [0..k-1]]); m.plot(cmap='gray')
```



Euler's $\phi(n)$ function: `euler_phi(n)`

Kronecker symbol $\left(\frac{a}{b}\right)$ = `kronecker_symbol(a,b)`

Quadratic residues: `quadratic_residues(n)`

Quadratic non-residues: `quadratic_residues(n)`

ring $\mathbf{Z}/n\mathbf{Z}$ = `Zmod(n) = IntegerModRing(n)`

a modulo n as element of $\mathbf{Z}/n\mathbf{Z}$: `Mod(a, n)`

primitive root modulo n = `primitive_root(n)`

inverse of $n \pmod m$: `n.inverse_mod(m)`

power $a^n \pmod m$: `power_mod(a, n, m)`

Chinese remainder theorem: `x = crt(a,b,m,n)`
finds x with $x \equiv a \pmod m$ and $x \equiv b \pmod n$

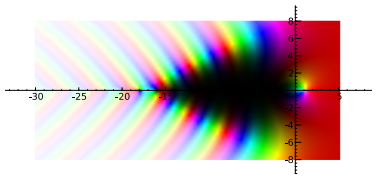
discrete log: `log(Mod(6,7), Mod(3,7))`

order of $a \pmod n$ =`Mod(a,n).multiplicative_order()`

square root of $a \pmod n$ =`Mod(a,n).sqrt()`

Special Functions

```
complex_plot(zeta, (-30,5), (-8,8))
```



$\zeta(s) = \prod_p \frac{1}{1-p^{-s}} = \sum \frac{1}{n^s}$ = `zeta(s)`

$\mathrm{Li}(x) = \int_2^x \frac{1}{\log(t)} dt$ = `Li(x)`

$\Gamma(s) = \int_0^\infty t^{s-1} e^{-t} dt$ = `gamma(s)`

Continued Fractions

```
continued_fraction(pi)
```

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cdots}}}}$$

continued fraction: `c=continued_fraction(x,` *bits*`)`

convergents: `c.convergents()`

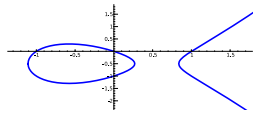
convergent numerator p_n = `c.pn(n)`

convergent denominator q_n = `c.qn(n)`

value: `c.value()`

Elliptic Curves

```
EllipticCurve([0,0,1,-1,0]).plot(plot_points=300,thickness=3)
```



`E = EllipticCurve([`a_1, a_2, a_3, a_4, a_6`])`

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

conductor N of E =`E.conductor()`

discriminant Δ of E =`E.discriminant()`

rank of E = `E.rank()`

free generators for $E(\mathbf{Q})$ = `E.gens()`

j-invariant = `E.j_invariant()`

$N_p = \#\{\text{solutions to } E \text{ modulo } p\}$ = `E.Np(`*prime*`)`

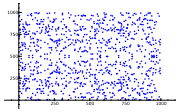
$a_p = p + 1 - N_p$ =`E.ap(`*prime*`)`

$L(E, s) = \sum \frac{a_n}{n^s}$ = `E.lseries()`

$\mathrm{ord}_{s=1} L(E, s)$ = `E.analytic_rank()`

Elliptic Curves Modulo p

```
EllipticCurve(GF(997), [0,0,1,-1,0]).plot()
```



`E = EllipticCurve(GF(p), [`a_1, a_2, a_3, a_4, a_6`])`

$\#E(\mathbf{F}_p)$ = `E.cardinality()`

generators for $E(\mathbf{F}_p)$ = `E.gens()`

$E(\mathbf{F}_p)$ =`E.points()`

Instruction pour exporter un resultat (dans la variable « number ») dans un fichier (le nom 'res' peut etre remplacé par tout nom, y compris par le nom du fichier 'result') :

```
 file('./result', 'w')
res=open(DATA+'result', 'w')
res.seek(0)
res.write(str(number))
res.close()
DATA+'result'
```

(On crée un fichier 'result' avec un droit d'ecriture 'w', on l'ouvre pour ecriture, on se place au 0-eme bit, on ecrit la chaine de caractere qui consitute la variable 'number', et on referme le fichier. La derniere commande indique son emplacement precis, utile si on veut le commniquer à un autre binôme.)

Instruction pour lire un resultat d'un fichier, et le mettre dans une variable :

```
retrieve=open('/Users/Dahmani/.sage/sage_notebook.sagenb/home/admin/7/data/result', 'r')
retrieve.seek(0)
z=int(retrieve.read())
retrieve.close()
z
```

(On ouvre un fichier avec le droit de lecture 'r' (ici, 'retrieve' est juste le nom d'usage de notre fichier ouvert, on peut bien sur mettre ce qu'on veut), on se place au 0-ieme bit, on lit ('retrieve.read()' renvoie une chaine de caracteres), on convertit la chaine lue en 'integer', on met le resultat dans la variable 'z' (qui peut bien sûr s'appeler autrement). Evidemment le chemin vers le fichier est à ecrire selon la réalité du fichier.

3 Séance 3 : Complexité

3.1 « Définition » de la complexité

Il n'est pas facile de définir la notion d'algorithme en général.

Hilbert, en 1900, l'évoquait déjà dans sa célebre liste de 23 problèmes qui, selon lui, étaient primordiaux à la recherche en mathématique à cette époque. Le Xème problème de Hilbert demande « de trouver une méthode, qui, étant donnée une équation Diophantienne *[polynômiale à un certain nombre d'inconnues et à coefficients entiers]*, au moyen d'un nombre fini d'opérations, pourra décider si l'équation est résoluble en nombres entiers. »

La formalisation de ce que signifie « méthode qui au moyen d'un nombre fini d'opérations décide ... » sera faite par Turing, Church et d'autres, dans le concept de Machine de Turing, qui regroupe tout ce que l'on considère aujourd'hui comme algorithme.

Dans ces notes, on reste flou sur la définition d'un algorithme. Mais plus encore, on reste flou sur la définition d'une opération. Par ce terme ou veut parler d'une opération élémentaire, qui peut etre effectuée en temps inferieur à un certain ϵ_0 par un ordinateur.

Par exemple $9 * 4$ est une opération élémentaire, mais

$$\begin{array}{rr} & 5996434543645745253546575675426754363275464357364 64 \\ * & 64534876875435756875698764865746437978 65 \end{array}$$

n'en est certainement pas une (encore que cela dépende de ϵ_0). L'algorithme classique de multiplication utilisera $(2040 = 51 * 40)$ multiplications élémentaires, suivies de $40 * 90$ additions élémentaires. Soit 5640 opérations élémentaires. On peut se convaincre d'une formule explicite qui donne le nombre d'opérations élémentaires pour la multiplication « avec la méthode habituelle », qui est un polynôme de degré 2 en la taille des nombres à multiplier.

La complexité d'un algorithme peut s'expliquer intuitivement : On dit que la complexité d'un algorithme est inférieure à une fonction $f : \mathbb{N} \to \mathbb{N}$ si, pour toute donnée du problème à taille N, le nombre maximal d'opérations de l'algorithme est $O(f(N))$. On dit que la *complexité du pire des cas* est au moins g si la complexité n'est pas inférieure à g.

3.2 Savez vous multiplier des nombres ?

Par exemple, pour calculer un produit de deux nombres à n chiffres, la complexité est inférieure à (n^2). Et pour la méthode naïve, la complexité du pire des cas est en effet (n^2). On peut faire mieux : l'algorithme de Karatsuba donne la multiplication de deux nombres en complexité $O(n^{\log_2(3)})$, c'est à dire $O(n^{1,585})$. La transformée de Fourier discrète permet de multiplier des nombres et des polynômes en $O(n \log(n) \log\log(n))$. On conjecture que la complexité optimale est $O(n \log(n))$ mais on ne connait pas d'algorithme de cette complexité...

D'abord, multipler des nombres ou des polynomes, c'est très similaires. En effet, un nombre est donné par ses chiffres, donc comme une somme $N = \sum_j x_j 10^j$, où chaque x_j est un nombre entier entre 0 et 9. Voyons le comme la valeur d'un certain polynôme P_N en $X = 10$, dont les coefficients sont les x_j : $N = P_N(10)$. Multiplier deux tels polynômes P_{N_1}, P_{N_2}, donne un polynome Q dont les coefficients sont entre 0 et 81. En $X = 10$, cela donne $Q(10) = P_{N_1}(10) \times P_{N_2}(10) = N_1 N_2$. Le seul problème est que les coefficients de Q ne sont pas forcement

inferieurs à 10, ce n'est donc pas le developpement décimal de N_1N_2. Il faut repartir les retenues. Cela se fait tres facilement, je vous laisse y reflechir.

3.2.1 Karatsuba

Multiplions donc les polynômes à coefficients bornés. Voici une description de l'algorithme de Karatsuba.

Observons alors que

$$(aX+b)(cX+d) \;=\; acX^2 \;+\; ((a+b)(c+d) \;-ac-\; bd) \times X \;+\; bd.$$

Le calcul de ac et bd sert deux fois. Appliquons cela à une stratégie « diviser pour régner ».

Soient $A(X)$ et $B(X)$ deux polynômes de degré $2n$.

Ecrivons $A(X) = A_0(X) \times X^n \;+\; A_1(X)$ avec A_0, A_1 de degré moitié (c'est à dire n), et de même $B(X) = B_0(X) \times X^n \;+\; B_1(X)$.

On a d'après l'observation :

$$A \times B \;=\; (A_0 \times B_0) \times X^{2n} \;+\; ((A_0+A_1) \times (B_0+B_1) \;-\; A_0B_0 - A_1B_1) \times X^n \;+\; A_1 \times B_1$$

Le point à observer c'est que presques toutes ces operations se passent sur des objets de taille n à la place de $2n$.

Ainsi, si $T(2n)$ designe le nombre maximal d'operations élémentaires à faire pour multiplier A et B de degré $2n$, on a la formule récursive :

$$T(2n) \leq 3 \times T(n) + 10 \times n$$

En effet, on a trois multiplications de polynômes de degré n à faire, et 4 additions de polynômes n'ayant que $2n$ coefficients non nuls, et deux additions de polynômes n'ayant que n coefficients non nuls (d'où $10n$ opérations élémentaires).

Comme $T(0) = 1$ par convention, on verifie par recurrence que

$$T(2^n) \leq 11 \times 3^n - 10 \times 2^n.$$

Cela montre que la complexité de l'algorithme de Karatsuba est $O(n^{\log_2(3)})$, strictement inferieur à l'algorithme du CE2...

3.2.2 Transformée de Fourier Rapide, et Discrete

Une autre idée pour la multiplication des polynômes consiste à utiliser l'interpolation.

Fixons $m = 2^d$. Soient $P(X)$ et $Q(X)$ deux polynômes de degré $< m$. On souhaite calculer les coefficients de $(P \times Q)(X)$.

On sait a priori que leur produit PQ est de degré $< 2m$. Pour determiner, au moins théoriquement ce produit, il suffit d'en connaitre les valeurs en $2m$ valeurs de X (par le théorème d'interpolation). L'idée est alors d'utiliser les racines $2m$-ième de l'unité $\omega^k = e^{ik\pi/m}$, pour $k = 1, \ldots, 2m$.

Il nous suffit alors de calculer les valeurs de P et de Q en ces points, et de les multiplier. Il s'agira apres cela d'interpoler pour trouver le polynôme de degré $< 2m$ ayant ces valeurs aux racines de l'unité. Encore faut-il que chacune de ces opérations soit facile à mettre en place et à executer.

Nous décrivons ici la Transformation de Fourier Rapide de P, c'est à dire, très concretement, le calcul de P en les racines $2m$-ièmes de l'unité. (1965, Cooley et Tukey)

Posons nos notations. $P(X) = \sum_0^{m-1} x_j \times X^j$ et $Q(X) = \sum_0^{m-1} y_j \times X^j$.

On peut décomposer $P(X) = P_e(X^2) + XP_o(X^2)$ en posant $P_e(T) = \sum_0^{m/2-1} x_{2j+1} \times T^j$ et $P_o(T) = \sum_0^{m/2-1} x_{2j} \times T^{2j}$.

Pour évaluer P au point $\omega^k = e^{ik\pi/m}$, il suffit dévaluer P_e et P_o en ω^{2k} (c'est à dire, comme m est une puissance de 2, en une racine $\frac{m}{2}$-ième de l'unité).

Ainsi, recursivement, si $F(2m)$ est le nombre d'operations pour calculer les $P(\omega^k)$, on a $F(2m) = 2F(m) + 2 \times 2m$. On obtient $F(2m) = 4m \times (d+1) = O(m \log m)$.

Observation 3.2.1 *Cette transformation est une certaine transformation de Fourier (la transformée de Fourier discrete) sur l'espace des fonctions définies sur les points r, $r = 1, \ldots, N$ (parfois interprétés comme $\{\frac{r}{N}, r = 1..N\}$ et les fonctions sont vues comme des discretisations des fonctions définies sur l'interval $[0, 1]$, mais ici ce sont les coefficients des polynomes).*

Il n'appartient pas à ce cours de définir et d'étudier dans un cadre général la transformée de Fourier. Juste un mot : le bon cadre théorique est le cas des groupes topologiques abéliens (et des fonctions définies dessus). Typiquement $\mathbb{S}^1$, $\mathbb{Z}$, ou $\mathbb{R}$, ou encore $\mathbb{Z}/N\mathbb{Z}$. Chaque groupe topologique abélien G possède un groupe dual (topologique abélien aussi) sur lequel les transformées de Fourier des fonctions raisonnables sur G sont définies. Dans le cas de $\mathbb{S}^1$, il s'agit de $\mathbb{Z}$ (et la transformée de Fourier est dans ce cas une serie) dans le case de $\mathbb{R}$ il s'agit de $\mathbb{R}$ lui même. Ici, nous utilisons secretement le groupe $\mathbb{Z}/N\mathbb{Z}$, et son dual est lui même.

Si ψ est une fonction définie sur les r, $r = 1, \ldots, N$, sa transformée $\hat{\psi}$ vaut $\hat{\psi}(k) = \sum_r \psi(r) e^{-2i\pi kr/N}$. La transformée inverse vaut ψ par théorème d'inversion de Fourier, et donne la formule $\psi(r) = \frac{1}{N} \sum_k \hat{\psi}(k) e^{2i\pi rk/N}$.

L'algorithme de calcul de la transformée, décrit ici, est la Transformation de Fourier Rapide (FFT).

Une fois que l'on a obtenu les valeurs de $P(\omega^k)$ et de $Q(\omega^k)$, on utilise le fait que leur module est necessairement inferieur à $m \times \max\{|x_i|\}$ (et $m \times \max\{|y_i|\}$) pour borner la complexité de leur multiplication $P(\omega^k) \times Q(\omega^k)$.

On obtient ainsi $(P \times Q)(\omega^k)$ pour chaque k. Il ne reste qu'à interpoler pour découvrir le polynôme $P \times Q$. Du point de vue de la transformation de Fourier, il suffit d'appliquer la transformation de Fourier inverse, ce qui revient au calcul précédent, en changeant ω en ω^{-1}.

La complexité finale (obtenue par Schönhage-Strassen), est $O(n \log n \log\log n)$.

3.3 L'exponentielle modulaire

Calculer une exponentielle normale m^k (dans $\mathbb{Z}$) coute beaucoup d'opérations élémentaires. En utilisant l'algorithme de Karatsuba, il y en a de l'ordre de

$$\log_2(m)^{\log_2(3)} + 2\log_2(m)^{\log_2(3)} + 3\log_2(m)^{\log_2(3)} + \cdots + (k+1)\log_2(m)^{\log_2(3)} \simeq k^2 \log_2(m)^{\log_2(3)}.$$

C'est à dire $2^{2\log_2(k)} \times \log_2(m)^{\log_2(3)}$. C'est donc exponentiel en terme de la taille de la donnée (qui vaut $\log_2(k) + \log_2(m)$).

Si on se place dans $\mathbb{Z}/N\mathbb{Z}$ la situation est très différente. En effet, la taille des nombres à multiplier ne va jamais exceder $\log_2(N)$. Cependant, on a toujours, en première approche, k multiplications entre nombres de taille $\log_2(N)$ à faire pour calculer m^k, ce qui peut être beaucoup (linéairement beaucoup en k), bien que ces opérations aient toutes taille bornée.

Si l'on connait $\phi(N)$, et si m est premier à N, on peut utiliser le théorème d'Euler $m^{\phi(N)} \equiv 1 \,[\mathrm{mod} N]$ pour remplacer k par son résidu modulo $\phi(N)$. Une fois faite cette réduction (qui consiste à appliquer la division Euclidienne de k par $\phi(N)$) à la fois l'exposant, et l'élément à exponentier sont bornés uniformément (en terme de N).

Enfin, on peut proceder par la méthode des carrés successifs. C'est une méthode d'exponentiation rapide.

On écrit $k = \sum_i \epsilon_i 2^i$ avec $\epsilon_i \in \{0, 1\}$ (c'est son expression binaire).

On a alors

$$m^k = \prod_{i, \epsilon_i = 1} m^{2^i}.$$

En utilisant le fait que $m^{2^i} = (m^{2^{i-1}})^2$ on procede alors ainsi : on calcul les carrés successifs de m modulo N, jusqu'au $\log_2(k)$-ieme. Puis on multiplie ensemble ceux dont l'indice i correspond à un $\epsilon_i = 1$. Au final, il n'y a que $3\log_2(k)$ multiplications dans $\mathbb{Z}/N\mathbb{Z}$, ce qui est bien mieux que le « linéairement beaucoup en k » obtenu par la premiere méthode.

Observation. On peut bien sûr optimiser cette esquisse d'algorithme afin d'économiser le stoquage et le nombre d'opérations (par exemple, on n'a pas besoin d'avoir la liste des carrés successifs, mais seulement d'avoir le bon au bon moment). Ces objectifs ne sont pas l'objet de ce cours, et nous les laissons de côté ici, peut être à tort.

3.4 L'algorithme d'Euclide

3.4.1 Euclide simple

Un point central de toute l'approche algorithmique modulaire est que le calcul de pgcd, et plus généralement, l'algorithme d'Euclide (le calcul des coefficients de Bachet-Bézout) est un algorithme *de faible complexité* (au sens algorithmique bien sûr).

Rappellons l'algorithme d'Euclide pour calculer le pgcd entre deux nombres $a > b$.

1. Si $b = 0$, retourner a, et l'algorithme est fini.
2. Poser $r := a \,[\mathrm{mod}\, b]$, et $a := b$ et $b := r$
3. Retourner au premier point.

Géométriquement, l'algorithme s'illustre sur un rectangle de longueur a et de largeur b.

S'il est plat, on retient sa longueur. Sinon, on ôte autant que possible des carrés de côté valant sa largeur.

Le pgcd est le côté du dernier carré utilisé. Il est clair qu'il divise les deux valeurs de a et b, car à chaque étape, le côté du carré utilisé divise le côté du carré utilisé à l'étape précédente,

et donc on peut repaver les « grands carrés » par les nouveaux « petits carrés ». Le fait que le resultat donné soit le plus grand diviseur commun est un exercice.

Le problème est de calculer rapidement le résultat.

Proposition 3.4.1 *Soient $a > b > 0$. Le nombre de division euclidiennes à effectuer dans l'algorithme d'Euclide pour a, b est inferieur à $2\log_2(a)$. Le nombre d'opérations élémentaires est dans $O(\log N)^2$.*

Posons $a_0 = a, b_0 = b$ et en notant $a_i = q_i b_i + r_i$, posons $b_{i+1} = r_i$ et $a_{i+1} = b_i$.
On a $r_2 < r_0/2$. En effet, il y a deux cas.
Soit $r_1 \leq r_0/2$ et comme $r_2 < r_1$ on a l'inégalité directement.
Soit $r_1 > r_0/2$ et alors on a écrit :

$$a_2(= b_1 = r_0) \ = \ b_2 + r_2 \ = r_1 + r_2$$

et $r_2 = r_0 - r_1 < r_0/2$.

De même, pour tout i, $r_{i+2} < r_i/2$. Ainsi $r_{2k} < a/2^k$. Dès que $k \geq \log_2(N)$, $r_{2k} = 0$.

L'estimée sur la complexité de l'algorithme en $O(\log N)^3$ découle de la complexité cubique des divisions euclidiennes. Si l'on prend en compte le fait quantitatif qu'elles deviennent de plus en plus simples (le produit des quotients successifs est plus petit que a) on peut (mais on ne le fait pas ici) améliorier cette complexité en $O((\log N)^2)$ comme annoncé.

3.4.2 L'algorithme d'Euclide étendu

La version étendue de l'algorithme d'Euclide permet de calculer les coefficients de Bachet-Bézout.

Soient $a, b > 0$, et d leur *pgcd*. On cherche u, v tels que $au + bv = d$.

On initialise l'algorithme par $d := a$, et $u := 1$.

1. Si $b = 0$ poser $v := 0$ et retourner (u, v, d) et terminer. Sinon, poser $u_1 := 0$, $d_1 := b$
2. Si $d_1 = 0$ poser $v := (d - au)/b$, retourner (u, v, d) et terminer.
3. Faire la division euclidienne $d = qd_1 + d_2$
4. $u_2 := u - qu_1$ et $(u, d) := (u_1, d_1)$ et $(u_1, d_1) := (u_2, d_2)$, retourner en (2).

Par un raisonnement similaire a l'algorithme classique on peut montrer que la complexité est $O((\log(a))^2)$ si $a > b$.

On calcule tres souvent les inverses modulo N par cet algorithme, même si le théorème d'Euler appliqué à l'exponentiation rapide permet de le faire si l'on connait $\phi(N)$ (en effet $a^{-1} \equiv a^{\phi(N)-1} \,[\mathrm{mod} N]$ si a est inversible).

4 Séance 4 : Codes correcteurs d'erreurs (linéaires)

4.1 Définitions

On se munit du corps $\mathbb{F}_2 = \mathbb{Z}/2\mathbb{Z}$. On considère $V = \mathbb{F}_2^N$. C'est un espace vectoriel, ses vecteurs sont des suites de 0 et de 1 de longueur N.

Notez que on travaille ici modulo 2 et qu'en particulier $1 = -1$, et $x + y = x - y$.

Si $\mathbf{v} \in V$, on peut écrire $\mathbf{v} = (v_1, \ldots, v_N)$ avec $v_i \in \{0, 1\}$. On défini alors le **poids** de $\mathbf{v}$ comme etant le nombre d'indices i tels que $v_i = 1$. Par exemple, le vecteur $(0, 0, \ldots, 0)$ a poids 0 et le vecteur $(1, 1, \ldots, 1)$ a poids N.

La **distance de Hamming** entre deux vecteurs $\mathbf{v}_1$ et $\mathbf{v}_2$ est simplement le poids de $\mathbf{v}_1 - \mathbf{v}_2$.

C'est une veritable distance : Si $d(\mathbf{v}_1, \mathbf{v}_2) = 0$ alors $\mathbf{v}_1 = \mathbf{v}_2$, on a toujours $d(\mathbf{v}_1, \mathbf{v}_2) = d(\mathbf{v}_2, \mathbf{v}_1)$, et enfin elle verifie l'inégalité triangulaire :

$$\forall \mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \qquad d(\mathbf{v}_1, \mathbf{v}_3) \leq d(\mathbf{v}_1, \mathbf{v}_2) + d(\mathbf{v}_2, \mathbf{v}_3).$$

Un **Code** de V est un sous-espace vectoriel.

Par exemple $C_0 = \{(0, v_2, \ldots, v_N) \in V\}$ est un code.

Un autre exemple de code est donné par ${}^t\ker(M)$ où M est une matrice rectangulaire de d lignes et N colonnes. Dans ce cas on dit que M est une **matrice de contrôle** du code $C = \ker(M)$. Pour contrôler qu'un vecteur $\mathbf{v}$ est bien dans C, il suffit de calculer $M({}^t\mathbf{v})$. Si c'est nul, $\mathbf{v} \in C$, si c'est non nul $\mathbf{v} \notin C$. Le **syndrôme** d'un vecteur $\mathbf{v}$ pour le contrôle M est la valeur de $M({}^t\mathbf{v})$.

Les paramètres d'un codes sont :

- sa longueur : c'est la dimension de V.
- sa dimension : c'est la dimension de C comme sous espace vectoriel. Dans le cas d'un code défini par une matrice de contrôle, c'est $N - r$ où r est le rang de la matrice de contrôle.
- sa force : c'est le minimum des distances de Hamming entre les éléments du code :

$$d(C) = \min_{\mathbf{v}_1 \in C \mathbf{v}_2 \in C \mathbf{v}_1 \neq \mathbf{v}_2} d(\mathbf{v}_1, \mathbf{v}_2).$$

  C'est aussi le poids minimal des vecteurs non nuls du code.

Par exemple, la force du code C_0 est 1, sa dimension est $N-1$ et sa longueur est N. Pouvez vous trouver un matrice de contrôle pour C_0 ?

Exemple 4.1.1 *Dans cet example, on prend $N = 10$ et une matrice de contrôle M de 3 lignes :*

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{pmatrix}.$$

la longueur du code est $N = 10$, la dimension du code est $dim(C) = N - 3$ car les trois lignes forment une famille libre. Un vecteur $\mathbf{v} = (v_1, \ldots, v_{10})$ est dans le code C si et seulement si

$$\sum_{i=0}^{4} v_{2i+1} = 0 \quad et \quad \sum_{i=5}^{10} v_i = 0 \quad et \ (v_1 + v_2 + v_5 + v_6 + v_9 + v_{10}) = 0$$

Voit-on facilemement quelle est sa force ?

4.2 Le rôle d'un code

On suppose qu'un message transmis est dégradé (volontairement ou non) : c'est à dire que chaque bit qui le compose est (peut-être aléatoirement) modifié avant que son destinataire ne le receptionne.

On suppose que "peu" de bits sont ainsi modifiés. Peut on avoir établi un protocole a priori pour détecter

1. Si effectivement il y a eu une dégradation du message
2. Quels bits ont été modifiés.

Les codes correcteurs d'erreurs linéaires sont des choix de "codes" dans $\mathbb{F}_2^N$: les messages transmis doivent appartenir au code pour avoir un sens.

Un code sera d'autant meilleur que sa force et sa dimension seront grande (pour une longueur donnée). En effet, un code de force f permettra de corriger à coup sûr le message reçu, à la seule condition qu'il ne contienne pas plus de $(f-1)/2$ erreurs. Par exemple un code de force 5 permet de corriger à coup sûr s'il n'y a que 2 erreurs.

Afin de toujours transmettre des messages du code, on peut se munir d'une base $\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_d$ du sous espace vectoriel C, et envoyer le message $\sum_i \epsilon_i \mathbf{e}_i$, si l'on veut envoyer "en clair" $(\epsilon_1, \ldots, \epsilon_d)$.

4.3 Décodage

Dans les bons cas, le syndrôme d'un vecteur va nous permettre de retrouver l'élément du code le plus proche (et ainsi de "deviner" où s'est placée l'erreur).

Proposition 4.3.1 *Soit C un code de matrice de contrôle M. S'il existe m colonnes de M qui, $c_1, \ldots c_m$ telles que $\sum_{i=1}^{m} c_i = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$, alors la force $f(C)$ de C est $\leq m$.*

Si m est minimal pour cette propriété, alors $f(C) = m$.

Preuve : Si $\mathbf{v} = (v_1, \ldots, v_N)$ est un element de C de poids minimal, disons p, comme $M({}^t\mathbf{v}) = 0$, on obtient que les p colonnes de M indexées par les indices des v_i non nuls ont somme nulle. Reciproquement, si une collection de r colonnes de M a une somme nulle, on trouve un vecteur $\mathbf{w}$ de poids r tel que $M({}^t\mathbf{w}) = 0$ (en choisissant ses coordonnées non nulles aux indices des colonnes données). Cela montre que la force du code est le nombre minimal de colonnes de M ayant une somme nulle.

Corollaire 4.3.1 *Si les colonnes de M sont toutes non-nulles, et toutes différentes, alors la force du code est ≥ 3.*

Dans ce cas, si $\mathbf{w}$ est un element de V à distance précidément 1 de C, il existe un unique $\mathbf{v} \in C$ à distance 1 de $\mathbf{w}$ et il diffère de $\mathbf{w}$ d'une seule coordonnée i_0 qui vérifie :

$$M({}^t\mathbf{w}) = c_{i_0}$$

Preuve : Premier point : d'après la proposition, la force ne peut pas être 1 car aucune colonne n'est nulle, ni 2 car on a supposé que deux colonnes sont toujours différentes. Donc c'est au moins trois.

Second point. L'unicité de $\mathbf{v}$ est simplement l'inégalité triangulaire dans le cas ou la force est ≥ 3. Disons que $\mathbf{w} = \mathbf{v} + \mathbf{e}$ avec $\mathbf{e}$ de poids 1.

$$M({}^t\mathbf{w}) \quad = \quad M({}^t\mathbf{v}) + M({}^t\mathbf{e}) \quad = \quad M({}^t\mathbf{e}) \quad = \quad c_{i_0}$$

pour c_{i_0} la colonne de M de numero l'indice du seul coefficient non nul de $\mathbf{e}$.

4.4 Exercices

Exercice 4.4.1 *Quels sont les paramètres du code de matrice de contrôle suivante ?*

$$M = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

Exercice 4.4.2 *Alice et Bob veulent communiquer à distance.*

Mais un système de brouillage introduit une erreur aléatoirement dans chaque paquet de 15 bits.

Alice et Bob se mettent d'accord pour utiliser un code correcteur d'erreur.

(1) Quelle est la taille des colonnes à prévoir pour avoir 15 colonnes non nulles différentes ?

(2) Quelle dimension peut avoir un code de taille 15 corrigeant automatiquement une erreur arbitraire ?

(3) Proposer une matrice de controle.

(4) Combien de vecteurs possede un base du code (une base du sous-espace vectoriel noyau de cette matrice) ? Comment les trouver ? (on ne demande pas de le faire explicitement... on pourra demander à SageMath, le cas echéant.)

(5) Comment proceder pour transmettre un message de 20 bits
comme par exemple 01001010011100010110 ?

Comment faire pour le décoder ?

Exercice 4.4.3 Wire tap

On considère la situation (générale) suivante.

— *Alice veut envoyer un message à Bob.*

— *Charles veut écouter le message. Mais son outillage provoque des erreurs aléatoires.*
— *Ni Alice ni Bob ne veulent que Charles intercepte le message.*

Comment nos héros peuvent ils s'y prendre ?

Une solution pour Alice et Bob consiste à choisir n, se mettre d'accord pour tirer au sort des éléments aléatoires de $(\mathbb{F}_2)^n$ et à la place d'envoyer 0, envoyer un élément (aléatoire) de somme nulle, et à la place d'envoyer 1, envoyer un élément de somme 1.

Ainsi Bob peut interpreter les messages d'Alice facilement, et Charles ne le peut que si le nombre d'erreur sur le n-uplet qu'il voit passer est pair.

Si on suppose que les lectures de Charles se font toutes avec une probabilité indépendantes p d'erreur ($p = 0,05$ par exemple), on se demande comment choisir n.

(1) Calculer la probabilité qu'il n'y ait pas d'erreur dans un n-uplet donné.

(2) Calculer la probabilité qu'il y ait 1 erreur.

(3) On note N le nombre d'erreur sur le n-uplet. C'est un nombre aléatoire. Quelle loi de probabilité satisfait-il ?

(4) Montrer que l'esperence du nombre d'erreurs $\mathbb{E}(N)$ vaut np.

Montrer que la probabilité qu'il y ait k erreurs vaut $\mathrm{C}_n^k p^k (1-p)^{n-k}$.

(5) Calculer la probabilté qu'il y ait $\lfloor \mathbb{E}(N) \rfloor$ erreurs, et celle qu'il y ait $\lfloor \mathbb{E}(N) + 1 \rfloor$.

(6) Estimer une bonne valeur pour le nombre n.

Le problème de cette méthode c'est qu'il faut beaucoup d'information pour communiquer juste quelques bits. Le débit de transmission est divisé par n.

On procède plutot comme suit.

Soit C_1 un code de $(\mathbb{F}_2)^n$, de longueur n et de dimension $n - r$.

L'espace $(\mathbb{F}_2)^n$ se partitionne en 2^r translatés de C_1 : $C_i = \gamma_i + C_1$,

$$(\mathbb{F}_2)^n = C_1 \cup C_2 \cup \cdots \cup C_{2^r}.$$

Soit M une matrice de contrôle pour C_1.

(7) Verifier que le syndrome d'un élément $\mathbf{v}$ de $(\mathbb{F}_2)^n$ est un élément de $(\mathbb{F}_2)^r$ qui ne dépend que de l'indice i tel que $\mathbf{v} \in C_i$.

(8) Le protocole d'Alice et Bob, pour transmettre un message $\mathbf{m}$ de taille r, est de choisir un élément aléatoire $\mathbf{s}$ de C_1 et de transmettre $\gamma_i + \mathbf{s}$ pour i choisi tel que le syndrome de γ_i soit $\mathbf{m}$.

(i) Le protocole précédent est un cas particulier de celui-ci, mais pour quels paramètres ?

(ii) On fixe $n = 2r$. Par combien le débit de transmission est-il divisé ?

(iii) On suppose qu'on a un code où le codage et décodage est efficace. Essayer de dire heuristiquement pourquoi il fonctionne (en particulier en est on toujours si sûr avec $n = r+1$?).

4.5 Parité généralisée ; introduction à $\mathbb{F}_{256}$

Un **corps** est un ensemble avec une addition et une multiplication définies sur ses éléments verifiant des propriétés familières : elles sont associatives, commutatives, la multiplication est distributive sur l'addition, il y a un élément 0 (neutre pour l'addition) et un élément 1 (neutre

pour la multiplication, et tout élément non nul x possède un opposé $-x$ tel que $x + (-x) = 0$ et surtout un inverse x^{-1} tel que $x \times x^{-1} = 1$.

Si K est un corps, l'anneau des polynômes $K[X]$ a un comportement "similaire" à celui des entiers $\mathbb{Z}$. Per exemple voici une propriété importante.

Par exemple, on peut faire des **divisions euclidiennes** de polynômes.

On dit qu'un polynôme est irréductible s'il n'est pas le produit de deux polynômes de degré strictement inferieur. C'est l'analogue de la propriété d'être premier pour un nombre.

Proposition 4.5.1 *Si K est un corps, et si $P(X)$ est un polynôme* **irreductible** *dans $K[X]$, alors $K[X]/(P)$, l'ensemble des restes possibles par division Euclidienne par P est un corps. Si K est un corps fini, alors le cardinal de $K[X]/(P)$ vaut $|K|^{\deg P}$.*

Comme $\mathbb{Z}/p\mathbb{Z}$ est un corps (pour p premier), on s'attend à trouver des corps de cardinalité p^n. C'est en effet ce qui arrive.

Proposition 4.5.2 *Tous les corps finis ont cardinal p^n pour un certain premier p et un certain nombre n, et par ailleurs, deux corps finis ayant même cardinal sont isomorphes.*

On note $\mathbb{F}_{p^n}$ l'unique corps à p^n éléments.

Proposition 4.5.3 *Si un corps L contient un sous corps K alors L peut aussi être vu comme K-espace vectoriel. En particulier, $|L|$ est une puissance de $|K|$.*

Il se trouve donc qu'il existe des corps $\mathbb{F}_2, \mathbb{F}_4, \mathbb{F}_8, \mathbb{F}_{16}$... et $\mathbb{F}_{256}$ respectivement à 2, 4, 8, 16... et 256 éléments.

Observons le fait suivant :

Proposition 4.5.4 *Tout corps de la forme $\mathbb{F}_{p^n}$ contient $\mathbb{Z}/p\mathbb{Z}$, réalisé comme les multiples de 1 dans $\mathbb{F}_{p^n}$.*

En effet, par finitude il existe un premier multiple de 1 qui vaut 0, et cela ne peut être qu'un multiple premier car sinon il existerait un "diviseur de zero" donc un élément non inversible. Enfin ce nombre premier ne peut être que p par la proposition precedente sur les cardinaux.

Attention $\mathbb{F}_4$ **n'est pas** $\mathbb{Z}/4\mathbb{Z}$. Ca n'est pas non plus l'anneau $(\mathbb{Z}/2\mathbb{Z}) \times (\mathbb{Z}/2\mathbb{Z})$ D'ailleurs, on peut ecrire à la main sa table de multiplication.

Disons que ses éléments sont $0, 1, a, b$ on a la table :

$\times$	0	1	a	b
0	0	0	0	0
1	0	1	a	b
a	0	a		
b	0	b		

a^2 ne peut pas être 1 car $X^2 - 1$ a déjà une racine double (c'est 1) et il ne peut pas avoir plus de racines. Il ne peut pas etre 0, sinon a n'est pas inversible. Il ne peut pas être a car $X^2 - X$

a déjà 2 racines (0 et 1). C'est donc forcement b. Par le même argument $b^2 = a$. Finalement $ab = ba = 1$ car il faut bien que a et b aient un inverse.

Par ailleurs $\mathbb{F}_4$ est isomorphe à $\mathbb{F}_2[X]/(X^2 + X + 1)$ car $X^2 + X + 1$ est irreductible sur $\mathbb{F}_2$.

Pour calculer dans $\mathbb{F}_4$, il suffit de voir ses elements comme des $\alpha X + \beta$ avec $\alpha, \beta \in \mathbb{F}_2$. L'addition, et la multiplication se font comme celle des polynômes, en prenant systematiquement le reste par division Euclidienne par $X^2 + X + 1$. On retrouve la table de multiplication ainsi.

Exercice 4.5.1 *En utilisant une proposition précédente, montrer que $\mathbb{F}_{256}$ ne contient pas le sous corps isomorphe à $\mathbb{F}_8$ ni à $\mathbb{F}_{128}$.*

Montrer que si q et q_0 sont tous les deux des puissances de 2, et que $q = q_0^k$ alors $\mathbb{F}_q$ contient un sous corps isomorphe à $\mathbb{F}_{q_0}$. (Indication : montrer que l'ensemble des racines de $X^{q_0} - X$ est un sous-corps de $\mathbb{F}_q$ et qu'il contient q_0 éléments.)

En particulier $\mathbb{F}_{256}$ contient $\mathbb{F}_{16}$ qui contient $\mathbb{F}_4$ qui contient $\mathbb{F}_2$.

Exercice 4.5.2 *(Avec Sage) Trouver des polynômes $P_2, \ldots, P_{20}$ irréductibles sur $\mathbb{F}_2$ tels que pour chaque i, $\mathbb{F}_{2^i} = K[X]/(P_i)$.*

Exercice 4.5.3 *Verifier que*

$$\mathbb{F}_{256} = \mathbb{F}_2[X]/(X^8 + X^4 + X^3 + X + 1).$$

Cette description permet de travailler dans $\mathbb{F}_{256}$ directement. Ses elements sont donc de la forme

$$\gamma = a_7X^7 + a_6X^6 + a_5X^5 + a_4X^4 + a_3X^3 + a_2X^2 + a_1X + a_0$$

Il faut donc un octet $(a_7, \ldots, a_0)$ pour decrire chaque élément, l'addition se fait terme a terme (avec bien sur $1 + 1 = 0$), mais la multiplication est plus subtile, elle se fait comme la pmultiplication des polynômes, en prenant le reste de la division par $X^8 + X^4 + X^3 + X + 1$.

Nous verrons un interet crucial de cette structure de corps dans l'énoncé suivant.

Théorème 4.5.1 *Dans tout corps fini $\mathbb{F}_{p^n}$, il existe un élément non nul γ_0 tel que la suite des γ_0^k pour ($k = 0, 1, \ldots, p^n - 1$) énumère tous les éléments non nuls de $\mathbb{F}_{p^n}$. On dit que γ_0 est un élément primitif.*

Proposition 4.5.5 *Le nombre d'éléments primitifs est donné par la fonction indicatrice "ϕ" d'Euler de $(p^n - 1)$, que l'on peut appeller sur SageMath par*

```
euler_phi(p^n -1)
```

D'une manière générale, si N est un nombre de décomposition en facteurs premiers : $N = p_1^{\alpha_1} p_2^{\alpha_2} \ldots p_k^{\alpha_k}$, alors

$$\phi(N) = \prod_1^k (p_i - 1) p_i^{\alpha_i - 1} = N \times \prod_1^k \left(1 - \frac{1}{p_i}\right)$$

Un **code de parité généralisée** est le noyau **dans** $(\mathbb{F}_2)^m$ d'une matrice de contrôle à coefficients dans $\mathbb{F}_{2^n}$ pour un certain n. Par exemple, typiquement, on prendra une matrice M à coefficients dans $\mathbb{F}_{256}$ et l'on définira un code sur $(\mathbb{F}_2)^N$ comme les éléments de $(\mathbb{F}_2)^N$ qui sont dans le noyau de M.

4.6 Exercices/TP

Exercice 4.6.1 *Sur Sage.*

(a) Combien d'éléments primitifs possède $\mathbb{F}_{256}$ *?*

(b) Trouver des puissances de 2 *assez grandes qui ont la propriété que le corps* $\mathbb{F}_{2^m}$ *correspondants possèdent* **moins de** *0,1% d'éléments* **non-primitifs**. *(Dans ces corps, un éléments "pris au hasard" serait donc probablement primitif...)*

(c) Trouver des puissances de 2 *assez grandes qui ont la propriété que le corps* $\mathbb{F}_{2^m}$ *correspondants possèdent moins de 50% d'éléments non-primitifs.*

(d) Trouver p *premier tel que, dans* $\mathbb{F}_{p^2}$*, il y ait moins de 80% d'éléments non-primitifs. (indication dans le cas d'une recherche "naïve" : cherchez dans une fourchette de 50000 éléments maximum). Pouvez vous trouver des premiers tels qu'il y ait mons de 50% d'éléments non-primitifs par exemple ? Avez vous une explication à cela ?*

(e) Pour de tels nombres premiers p*, verifier par Sage pour lesquels est-ce que* $X^2 + X + 1$ *est irréductible, et pour lesquels il ne l'est pas.*

(f) Choisissez p *pour que* X^2+X+1 *soit irréductible. En identifiant* $\mathbb{F}_{p^2}$ *à* $\mathbb{F}_p[X]/(X^2+X+1)$*, on identifie les éléments de* $\mathbb{F}_{p^2}$ *aux coupes* (a, b) *par l'attribution* $(a, b) \mapsto aX + b$*.*

(f-i) Tirer au sort un élément $t = a_0X + b_0$*.*

(f-ii) Calculer la liste de ses 1000 ou 10000 premières puissances $t^i, i = 0, \ldots, 1000$ *(ou* 10000*).*

(f-iii) Tracer le nuage de points dans le carré $[0, p] \times [0, p]$ *correspondant a leurs coordonnés* (a, b)*. Est-ce que cela a l'air aléatoire ?*

(f-iv) En comparaison (sur la même figure, d'une couleur différente) tracer le nuage de points correspondants non pas aux puissances mais aux multiple $i \times t, i = 1, \ldots, 1000$*.*

Exercice 4.6.2 *(Codes de Hamming)*

Soit α *un élément non nul de* $\mathbb{F}_{2^m}$*. On défini le code sur les messages binaires* C *par sa matrice de controle qui ne contient qu'une ligne :*

$$M = (\alpha, \alpha^2, \ldots, \alpha^{2^m-1})$$

Montrer que la force du code est 3 *si et seulement si* α *est primitif. Quels sont les paramètres de ce code ?*

Exercice 4.6.3 *(Codes BCH)*

Soit α *un élément primitif de* $\mathbb{F}_{2^m}$*. On défini le code sur les messages binaires* C *par sa matrice de controle*

$$M = \begin{pmatrix} \alpha & \alpha^2 & \alpha^3 & \ldots & \alpha^{2^m-1} \\ \alpha^3 & \alpha^6 & \alpha^9 & \ldots & (\alpha^3)^{2^m-1} \end{pmatrix}$$

(a) Quelle est la dimension de ce code ?

(b) Montrer qu'un vecteur (binaire) orthogonal aux deux lignes (c'est à dire dans le noyau de M) est aussi dans le noyau de

$$M' = \begin{pmatrix} \alpha & \alpha^2 & \alpha^3 & \dots & \alpha^{2^m-1} \\ \alpha^2 & \alpha^4 & \alpha^6 & \dots & (\alpha^2)^{2^m-1} \\ \alpha^3 & \alpha^6 & \alpha^9 & \dots & (\alpha^3)^{2^m-1} \\ \alpha^4 & \alpha^8 & \alpha^{12} & \dots & (\alpha^4)^{2^m-1} \end{pmatrix}.$$

(c) Montrer que quatres colonnes quelqonques de M' sont toujours linéairement indépendantes sur $\mathbb{F}_{2^m}$, et sur $\mathbb{F}_2$. En déduire que la force du code est ≥ 5.

Exercice 4.6.4 *(Code de Goppa)*

Si l'on se donne $\alpha_1, \dots \alpha_m$ des éléments de $\mathbb{F}_{256}$, et un polynôme irreductible P **sur** *$\mathbb{F}_{256}$, on définit le* **code de Goppa** *comme étant le code de $(\mathbb{F}_2)^m$ (donc sur des messages binaires !) dont la matrice de contrôle est*

$$M = \begin{pmatrix} \frac{1}{P(\alpha_1)} & \dots & \frac{1}{P(\alpha_m)} \\ \frac{\alpha_1}{P(\alpha_1)} & \dots & \frac{\alpha_m}{P(\alpha_m)} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \frac{\alpha_1^{\deg(P)-1}}{P(\alpha_1)} & \dots & \frac{\alpha_m^{\deg(P)-1}}{P(\alpha_m)} \end{pmatrix}$$

(a) Montrer que la dimension du code est $m - 8\deg(P)$.

(b) Montrer qu'on peut le protocole suivant définit le le même code : on note $\mathbf{x} = (x_1, \dots, x_8)$. Soit $R(\mathbf{x}, X) = \sum_{i=1}^{m} \frac{x_i}{X-\alpha_i}$ Le code Γ est l'ensemble des $\mathbf{x}$ tels que $R(\mathbf{x}, X) \equiv 0 \,[\mathrm{mod} P(X)]$.

(c) (i) Montrer que si $\mathbf{x}$ a poids w,

$$R(\mathbf{x}, X) = \frac{f'_{\mathbf{x}}(X)}{f_{\mathbf{x}}(X)}$$

où $f_{\mathbf{x}}(X)$ est le produit de w facteurs de la forme $(X - \alpha_i)$. En déduire que $P(X)$ divise $f'_{\mathbf{x}}(X)$.

(c)(ii) En remarquant que la dérivée d'un polynôme sur $\mathbb{F}_{2^m}$ est toujours le carré d'un autre polynôme, et que P n'a que des racines simples, conclure que la force du code est $2\deg(P) + 1$.

5 Séance 5 : Registres à décalages, et générateurs pseudo-aléatoires

5.1 Registres à décalages

Un **registre à décalages** est la donnée d'un nombre m, et de coefficients $h_0, \ldots, h_{m-1}$ dans $\mathbb{F}_2$. Cette famille de coefficients s'appelle le **générateur du registre**. On appelle aussi **polynôme de rétroaction du registre** le polynôme $(X^m + h_{m-1}X^{m-1} + \cdots + h_0)$.

Si $U_0 = \begin{pmatrix} u_0 \\ \vdots \\ u_{m-1} \end{pmatrix}$ est un vecteur de $(\mathbb{F}_2)^m$, on note U_k le vecteur obtenu par

$$U_k = M \times U_{k-1} \text{ pour } M = \begin{pmatrix} 0 & 1 & 0 & \ldots & 0 \\ \vdots & \ddots & 1 & \ddots & 0 \\ 0 & \ldots & 0 & \ddots & 0 \\ 0 & \ldots & \ldots & 0 & 1 \\ h_0 & h_1 & \ldots & h_{m-2} & h_{m-1} \end{pmatrix}$$

Il est équivalent de définir $u_n = \sum_0^{m-1} h_i u_{n-m+i}$.

La donnée d'un générateur et d'un vecteur initial U_0 (appelé **graine**) défini un processus, c'est à dire une suite $(U_n)_{n\in\mathbb{N}}$ et une suite $(u_n)_{n\in\mathbb{N}}$.

Exercice 5.1.1 *Le processus ainsi défini est ultimement periodique. Il est periodique si $h_0 = 1$, ou de manière équivalente, si M est inversible.*

Dans la suite on supposera toujours $h_0 \neq 0$. Chaque ligne du vecteur U_k est un **registre**, le processus décale les registres et produit dans le dernier registre une combinaison des m précédents, et oublie le plus m-ième plus ancien. On dit donc que *le processus utilise donc m registres*. Si le processus produit une suite (U_n) de periode maximale (c'est à dire $2^m - 1$) on dit que la suite (u_n) est une m-sequence.

Exercice 5.1.2 *Si la suite (u_n) est une m-sequence, alors pour tout $k \leq m$, toute suite de 0 et de 1 de longueur k apparait exactement 2^{m-k} fois comme « sous mot » dans toute suite de la forme $(u_i, \ldots, u_{i+2^m-1})$.*

Exercice 5.1.3 *Quel est le polynôme caractéristique de la matrice définissant le processus ? Montrer que s'il est irréductible sur $\mathbb{F}_2$, l'ordre de cette matrice dans $GL_n(\mathbb{F}_2)$ est égal à l'ordre de n'importe laquelle de ses racines dans $\mathbb{F}_{2^m}$. Montrer, toujours si le polynôme caracteristique est irréductible, que cet ordre est la periode du processus (si la graine est non nulle). En déduire que pour tout m, il existe des processus produisant des m-sequences.*

5.2 Serie formelle caracteristique du processus

Prenons un processus de registre à décalage donné par un générateur $h = (h_0, \ldots, h_{m-1})$ et une graine U_0.

Grace à la suite (u_n), construisons la série formelle

$$A(X) = \sum_{i=0}^{\infty} u_i X^i \quad = \quad u_0 + u_1 X + u_2 X^2 + \cdots + u_n X^n + \ldots$$

Proposition 5.2.1 *C'est en fait une fraction rationelle, c'est à dire un quotient de deux polynômes. Ecrite sous forme irréductible, le degré du dénominateur vaut le nombre minimal de registres necessaires pour engendrer le processus.*

Preuve (la même que celle qui dit que les nombres à développement décimal periodique sont rationnels) : soit P la periode du processus.

$$\text{Posons } G(X) = u_0 + u_1 X + \cdots + u_{P-1} X^{P-1}.$$

$$\text{On a donc } A(X) = G(X) + X^P G(X) + X^{2P} G(X) + \ldots \quad = G(X) \times (\sum_{k \geq 0} X^{kP}).$$

Ce qui signifie

$$A(X) = \frac{G(X)}{1 - X^P} = \frac{G(X)}{1 + X^P}.$$

C'est donc une fraction rationelle.

On peut être plus précis et simplifier la fraction rationelle par un diviseur commun eventuel de G et de $X^P + 1$.

Définition 5.2.1 *On définit la* **complexité** *d'un registre à décalages (pour une donnée de générateur et de graine) comme étant le degré du dénominateur de la fraction rationelle $A(X)$ exprimée sous forme réduite. Comme le dit la proposition, c'est le plus petit nombre de registres qu'il faut pour engendrer A.*

On peut parler de la même manière de complexité d'une suite periodique.

A priori cette opération de réduction de la fraction peut sembler banale et probablement ineteressante. Il n'en est rien. La proposition suivante indique que, bien que pour l'instant notre majoration de la complxité vaille P la periode, c'est à dire potentiellement $2^m - 1$, en fait la complexité mesiure le nombre de registres necessaires à engendrer la suite, c'est à dire moins que m.

Proposition 5.2.2 *Écrite sous forme de fraction irreductible, le degré du dénominateur de la fraction rationelle $A(X)$ est égal au nombre minimal de registres nécessaires pour engendrer ce processus.*

Commençons par majorer la complexité. Supposons qu'on a m registres. On va arranger l'expression de A comme quotient. On va ecrire la relation de recurrence linéaire sur les coefficients, et rassembler les facteurs de h_j donnés. Puis pour chaque, on rassemble un terme qui a $A(X)$ en facteur. Cela nous donnera un polynôme de degré m en facteur. Voyons le calcul. On complete d'abord la suite (u_j) pour des valeures de j négative, simplement par periodicité. Puis on écrit que pour tout i, $u_i = \sum_{j=1}^{m} h_{m-j} u_{i-j}$.

$$\begin{aligned}
A(X) &= \sum_{i=0}^{\infty} \left(\sum_{j=1}^{m} h_{m-j} u_{i-j} \right) X^i && \text{on inverse les signes somme} \\
A(X) &= \sum_{j=1}^{m} h_{m-j} \left(\sum_{i=0}^{\infty} u_{i-j} X^{i-j} \right) \times X^j && \text{on fait apparaitre } A(X) \\
A(X) &= \sum_{j=1}^{m} h_{m-j} \left(\left(\sum_{i=0}^{j-1} u_{i-j} X^{i-j} \right) + A(X) \right) \times X^j && \text{on rassemble les termes facteurs de } A(X)
\end{aligned}$$

$$A(X) \times (1 - h_{m-1}X - h_{m-2}X^2 - \cdots - h_0 X^m) = \sum_{j=1}^{m} h_{m-j} \left(\sum_{i=0}^{j-1} u_{i-j} X^{i-j} \right) \times X^j$$

Notons $g(X)$ le membre de droite. On a bien $A(X) = g(X)/H(X)$ pour

$$H(X) = 1 + h_{m-1}X + h_{m-2}X^2 + \cdots + h_0 X^m$$

qui est bien de degré m.

Cela montre que le degré du dénominateur de la fraction rationelle A est *inferieur* au nombre minimal de registre qu'il faut pour engendrer A.

Réciproquement, supposons que la série génératrice $A(X)$ est un quotient de polynômes $g(X)/H(X)$ avec $deg(H) = m_0$. On doit montrer que lers coefficients de A verifient une relation de recurrence linéaire de longueur m_0.

On sait par ailleurs que le degré de g est inferieur strictement au degré du dénominateur car c'est le cas pour l'autre expression de $A = G/(X^P + 1)$

Ecrivons $H(X) = 1 + \hbar_{m-1}X + \ldots \hbar_0 X^{m_0}$.

On a $AH = g$, et $deg(g) \leq m_0 - 1$. Le terme de degré m_0 de AH est donc nul :

$$u_{m_0} + \hbar_{m_0-1} u_{m_0-1} + \cdots + \hbar_0 u_0 = 0.$$

Et de même pour tout $i \geq 0$, le terme de degré $m_0 + i$ est nul, et donc

$$u_{m_0+i} + \hbar_{m_0-1} u_{m_0-1+i} + \cdots + \hbar_0 u_0 = 0.$$

Nous avons donc notre recurrence linéaire de longueur m_0.

5.3 Combinaisons, corrélations et transformée de Fourier

Les registres à décalages sont parfaitement déterministes, et, pour un registre de complexité m, en connaissant m valeurs, on peut facilemement retrouver toute l'information, en résolvant un système de m équations à m inconnues.

On peut contourner ce problème en **combinant** plusieurs registres à décalages avec une formule booléenne, ou de maniere equivalente, polynômiale sur $\mathbb{F}_2$.

Il s'agit de s'accorder sur une fonction $f : (\mathbb{F}_2)^k \to \mathbb{F}_2$, comme par exemple

$$f_0 : (x_1, x_2, x_3) \mapsto x_1 + x_1x_2 + x_2x_3$$

et l'appliquer à k registres à décalages « indépendants ». Cela produit ainsi une nouvelle suite que l'on peut esperer *pseudo-aléatoire*.

Exercice 5.3.1 *Verifier que la formule donnée en exemple est équivalente à la formule booléenne « (x_1 ET non-x_2) XOR (x_2 ET x_3) »*

L'utilité de telles manipulations est qu'au prix d'un tres faible artifice de calcul, on peut au moins en principe accroitre sensiblement la complexité.

Proposition 5.3.1 *La complexité d'une somme de suites periodiques est inférieure à la somme des complexités.*

La complexité d'un produit (termes à termes) de deux suites periodiques est inferieure au produit des complexités.

Dans le premier cas, il suffit de mettre au même dénominateur, nous laissons la preuve en exercice. Le second cas est admis pour l'instant.

Malheureusement la complexité n'est pas la seule mesure de la difficulté de prévoir la suite.

Définition 5.3.1 *Une fonction $f : \mathbb{F}_2^k \to \mathbb{F}_2$ est dite* **résitante aux corrélations à l'ordre r** *si : pour tout k-uplet de variables aléatoires indépendantes $(X_1), \ldots, (X_k)$, vallant 0 ou 1 de manière equiprobables, la variable aléatoire $Z = f(X_1, \ldots, X_k)$ est inépendante de n'importe quel vecteur $(X_{i_1}, \ldots, X_{i_r})$ (de longueur r).*

Exemple : dans le cas de f_0 de l'exemple plus haut, on peut verifier que $z_i = x_1$ avec probabilité 3/4 (c'est le cas dès que $x_2 = 0$ et aussi si ($x_2 = 1$ et $x_3 = x_1$)). Ainsi, f_0 n'est pas résistante aux corrélations à l'ordre 1.

Définition 5.3.2 *Soit $f : \mathbb{F}_2^k \to \mathbb{R}$.*

La transformée de Fourier (ou de Walsh, ou de Walsh-Hadamard) de f, que l'on note $\hat{f} : \mathbb{F}_2^k \to \mathbb{R}$ vaut

$$\hat{f}(t_1, \ldots t_k) = \sum_{x \in \mathbb{F}_2^k} f(x)(-1)^{\langle t \cdot x \rangle}$$

Notez des subtilités. Le produit $\langle t \cdot x \rangle$ est la somme, modulo 2, des produits terme à terme des k-uplets (autrement dit, c'est le produit scalaire cannonique dans $\mathbb{F}_2^k$, à valeur dans $\mathbb{F}_2$). Le résultat vaut 0 ou 1 (mais en principe est dans F_2). La valeur $(-1)^{t \cdot x}$ vaut respectivement 1 ou -1 dans $\mathbb{Z}$ (pas dans $\mathbb{F}_2$ bien sur).

Par ailleurs, nous allons utiliser cette transformée de Fourier sur des applications que, pour des raisons pratiques de calcul booléen, nous avions définies à valeurs dans $\mathbb{F}_2$. C'est un abus de langage. Il faut bien comprendre, qu'en fait, nous appliquerons la transformée à la fonction composée avec l'application $0_{\mathbb{F}_2} \mapsto 0_{\mathbb{Z}}$, $1_{\mathbb{F}_2} \mapsto 1_{\mathbb{Z}}$, afin d'être à valeurs dans $\mathbb{Z}$.

Lemme 5.3.1 *Si $t \neq 0$, alors $\sum_{x \in (\mathbb{F}_2)^k} (-1)^{\langle x,t \rangle} = 0$.*
Si $t = 0$ la somme vaut 2^k.

Il suffit de choisir une coordonnée dans le support de t, et de regrouper les x par paires qui ne diffèrent qu'en cette coordonnée.

Ce simple Lemme justifie d'ailleurs le nom de transformée de Fourier : les fonctions $\left(\frac{1}{2^k} \sum (-1)^{\langle t, \cdot \rangle}\right)$ forment une base orthonormée de l'espace des fonctions de $(\mathbb{F}_2)^k$ à valeur dans $\mathbb{R}$. En effet, pour calculer le produit scalaire considérons $\sum_{x \in (\mathbb{F}_2)^k} (-1)^{\langle x,t_1 \rangle} (-1)^{\langle x,t_2 \rangle}$, c'est à dire $\sum_{x \in (\mathbb{F}_2)^k} (-1)^{\langle x,t_1+t_2 \rangle}$, qui vaut 2^k ou 0 selon que $t_1 = t_2$ ou pas. La transformée de Fourier calcule en fait les coordonnées dans cette base, comme le montre la proposition suivante.

Proposition 5.3.2 *(Inverion de Fourier)*
Pour tout $x \in \mathbb{F}_2^k$, on a

$$f(x) = \frac{1}{2^k} \sum_{t \in \mathbb{F}_2^k} \hat{f}(t) (-1)^{\langle t \cdot x \rangle}$$

Autrement dit $\hat{\hat{f}} = 2^k f$ (en utilisant la transformée sur $\hat{f}$ à valeurs dans $\mathbb{Z}$).

Preuve : Il s'agit d'intervertir les sommes. On a, à x fixé :

$$\sum_t \hat{f}(t) (-1)^{\langle t \cdot x \rangle} \;=\; \sum_t (-1)^{\langle t \cdot x \rangle} \sum_y f(y) (-1)^{\langle t \cdot y \rangle} \;=\; \sum_y f(y) \sum_t (-1)^{\langle t \cdot (y+x) \rangle}.$$

Mais $\sum_t (-1)^{\langle t \cdot (y+x) \rangle} = 0$ si $y \neq x$ et $= 2^k$ si $y = x$ par le Lemme.

Exercice 5.3.2 *Calculer $\hat{1}$.*

Exercice 5.3.3 *(Formule sommatoire de Poisson)*
Montrer que $\sum_t \hat{f}(t) = 2^k \times f(0)$, et que $\sum_x f(x) = \hat{f}(0)$.

Proposition 5.3.3 *(Théorème de Xiao et Massey)*
La fonction $f : \mathbb{F}_2^k \to \mathbb{F}_2$ est résistante aux corrélations à l'ordre r si et seulement si $\hat{f}(t) = 0$ pour tout $t \in \mathbb{F}_2^k \setminus \{0\}$ de poids au plus r.

Exercice 5.3.4 *Verifier que ce critère détecte bien la faille de l'exemple f_0 du début.*

La preuve de la Proposition se fait grâce à ce lemme.

Lemme 5.3.2 *Soient $X_1, \ldots, X_k$ sont des des variables aléatoires de valeur 0 ou 1 equiprobables, et soit $f : \mathbb{F}_2^k \to \mathbb{F}_2$. Soit $t \neq 0$ dans $\mathbb{F}_2^k$.*

Alors $Z = f(X_1, \ldots, X_k)$ est indépendant de $\langle t \cdot (X_1, \ldots, X_k)\rangle$ si et seulement si $\hat{f}(t) = 0$.

Preuve. Notons $X = (X_1, \ldots, X_k)$. On a, pour $b = 0$ ou 1

$$P(Z = 1 \,|\, \langle t \cdot X\rangle = b) \;=\; \frac{\sharp\{x, f(x) = 1 \text{ et } \langle t \cdot x\rangle = b\}}{\sharp\{x, \langle t \cdot x\rangle = b\}}$$

ce qui est toujours bien défini car $t \neq 0$. Cela donne :

$$P(Z = 1 \,|\, \langle t \cdot X\rangle = b) \;=\; \frac{1}{2^{k-1}} \sum_{t \cdot x = b} f(x)$$

Prenons la différence :

$$P(Z = 1 \,|\, \langle t \cdot X\rangle = 0) - P(Z = 1 \,|\, \langle t \cdot X\rangle = 1) \;=\; \frac{1}{2^{k-1}} \hat{f}(t).$$

Autrement dit

$$(\hat{f}(t) = 0) \;\Leftrightarrow\; (P(Z = 1 \,|\, \langle t \cdot X\rangle = 0) = P(Z = 1 \,|\, \langle t \cdot X\rangle = 1))$$

On en déduit que l'évenement $\{Z = 1\}$ est indépendant des évenements $\{\langle t \cdot X\rangle = 1\}$ et $\{\langle t \cdot X\rangle = 0\}$, si et seulement si $\hat{f}(t) = 0$.

On peut finir de montrer la proposition.

Si f est résistante aux corrélations d'ordre r : $f(X)$ est une variable aléatoire indépendante de n'importe quel vecteur $X_* = (X_{i_1}, \ldots, X_{i_r})$. Elle est donc aussi indépendante de $t \cdot X_*$ pour tout $t \in \mathbb{F}_2^r$. Elle est donc indépendante de tout $t \cdot X$ pour tout t de poids au plus r. Le Lemme nous assure qu'alors $\hat{f}(t) = 0$ pour tous ces t.

Reciproquement, si $\hat{f}(t) = 0$ pour tout t de poids $\leq r$, le lemme nous assure que $f(X)$ est indépendante de tout $\langle t \cdot X\rangle$ (à valeur dans $\mathbb{F}_2$) pour tout t de poids au plus r. On doit montrer que $f(X)$ est indépendante de tout vecteur $X_* = (X_{i_1}, \ldots, X_{i_r})$. Notons $Z = f(X)$ dans la suite. Remarquons d'emblée qu'il est équivalent de montrer que tout vecteur X_* comme ci-dessus est indépendant de Z, car l'indépendance est une notion symétrique (exercice : montrer cela).

On raisonne à $(i_1, \ldots, i_r)$ fixé.

Fixons z parmi 0 ou 1. Soit $g(x) = P(X_* = x \,|\, Z = z)$, et $h(x) = P(X_* = x)$. On a

$$\hat{g}(t) = \sum_x P(X_* = x | Z = z)(-1)^{\langle t \cdot x\rangle} \qquad \hat{h}(t) = \sum_x P(X_* = x)(-1)^{\langle t \cdot x\rangle}$$

c'est à dire

$$\hat{g}(t) = P(\langle t \cdot X_*\rangle = 0 | Z = z) - P(\langle t \cdot X_*\rangle = 1 | Z = z) \qquad \hat{h}(t) = P(t \cdot X_* = 0) - P(t \cdot X_* = 1).$$

On a supposé que $t \cdot X$ etait indépendant de Z donc on a que $\hat{g}(t) = \hat{h}(t)$.

Comme la transformé de Fourier est inversible, cela veut dire que $g = h$. Ainsi, pour tout z et tout x,

$$P(X_* = x \,|\, Z = z) = P(X_* = x).$$

On a bien l'indépendance cherchée.

Définition 5.3.3 *Soit $f : \mathbb{F}_2^k \to \mathbb{F}_2$.*
On dit que f est équilibrée si le nombre de pré-images de 0 égal celui de 1.
On dit que f est r-résiliente si f est sans corrélation d'ordre r et équilibrée.

Exercice 5.3.5 *Montrer que f est équilibrée si et seulement si $\hat{f}(0) = 2^{k-1}$.*

Exercice 5.3.6 *On définit, pour une fonction booléenne f, la transformée $\tilde{f}$ à valeurs dans $\mathbb{Z}$ définie par*

$$\tilde{f}(t) = \sum_x (-1)^{\langle f(x) + t \cdot x \rangle}.$$

Montrer que $\tilde{f} = \hat{1} - 2\hat{f}$.
Montrer que f est r-résiliente si et seulement si $\tilde{f} \equiv 0$.
Montrer que si $t \in (\mathbb{F}_2)^k$ et si S est un sous-espace vectoriel de $(\mathbb{F}_2)^k$, alors

$$\sum_{x \in S} \tilde{f}(x)(-1)^{\langle t \cdot x \rangle} = |S| \sum_{y \in S^\perp} (-1)^{f(y+t)}$$

(Formule sommatoire de Poisson pour cette transformée).

5.4 Non linéarité, un troisième critère

Un générateur à registres combinés, pour être qualifié de pseudo-aléatoire plausible doit satisfaire les conditions d'équilibre (les proportions de 0 et de 1 obtenus doivent être égales) de résistance aux corrélations, et une autre, que l'on présente maintenant, qui est d'avoir une grande non-linéarité.

La distance $d(f, g)$ entre deux fonctions $(\mathbb{F}_2)^k \to \mathbb{F}_2$ vaut le cardinal du support de $f - g$.

La non-linéarité d'une fonction $f : (\mathbb{F}_2)^k \to \mathbb{F}_2$ est la plus petite distance $d(f, g)$ pour g parcourant l'espace des fonctions affines. Notez que ces fonctions affines sont de la forme $(x \mapsto t \cdot x)$ ou $(x \mapsto t \cdot x + 1)$.

Exercice 5.4.1 *Si l'on définit $F(x) = (-1)^{f(x)}$, montrer que*

$$\sum_{t \in \{0,1\}^k} \hat{F}(t)^2 = 2^{2k}.$$

En déduire que si k est pair, et si $\hat{F}(t) = 2^{k/2}$ pour tout t, alors la non-linéarité de f est maximale.

5.5 TP

— Partons de $m = 10$.
- Creer un polynôme de degré m sur $\mathbb{F}_2$ aléatoire de coefficient constant non nul.
- Question théorique : quelle est la probabilité que ce polynôme soit irréductible ? Que ses racines soient des éléments primitifs de $\mathbb{F}_{2^m}$?
- Creer une graine non-nulle arbitraire (aléatoire ou pas) de taille m. Créer la matrice de rétroaction du processus, ayant le polynôme de rétroaction choisi avant. (La commande integrée matrix(r,s) peut servir)
- Calculer les $(2^m - 1 + m)$ premiers termes de la suite du registre à décalage.
- Calculer la periode du processus ainsi crée.
- Sur plusieurs essais de polynômes aléatoires, estimer une statistique pour la periode maximale $(2^m - 1)$. Est-ce que cela correspond à votre estimation théorique initiale ?
- Faire une liste des periodes observées. Pouvez vous trouver une interpretation algébrique pour chacune ?

— En paires de binomes : l'une des paires engendre des grandes matrices de rétroaction, et des verteurs initiaux, et les propose à l'autre. L'autre paire tente de verifier que le processus verifie les trois critères de Golomb :

1. au sein d'une periode, le nombre de 0 est égal au nombre de 1, à une unité près.
2. au sein d'une periode, le nombre de suites maximales de bits identiques consecutifs de taille k est la moitié du nombre de telles suites de taille $k - 1$.
3. la fonction d'autocorrélation $C(\tau) := \sum_{i=0}^{L-1} (-1)^{u_i + u_{i+\tau}}$ prend deux valeurs suivant que $\tau = 0$ ou non.

— Un binôme engendre une matrice de rétroaction de la taille $L \times L$, et donne à un autre binôme $2L$ termes consécutifs du processus (issu d'une graine non nulle). Le second binôme doit retrouver la matrice. (attaque de Belekamp-Massey)

6 Séance 6 : Techniques symétriques

D'apres Suetone, Jules César, lorsqu'il ecrivait des messages de nature militaire « sensible », procedait à un chiffrement.

> Â« Il y employait, pour les choses tout à fait secrètes, une espèce de chiffre qui en rendait le sens inintelligible (les lettres étant disposées de manière à ne pouvoir jamais former un mot), et qui consistait, je le dis pour ceux qui voudront les déchiffrer, à changer le rang des lettres dans l'alphabet, en écrivant la quatrième pour la première, c'est-à-dire le d pour le a, et ainsi de suite. Â»

Les techniques symétriques de cryptographie sont celles ou les deux utilisateurs Alice et Bob ont les mêmes connaissances. Par exemple le code de César, losque Alice et Bob connaissent la clé secrete (la valeur du décalage) est une technique symétrique.

Ce code de César a pu être utile et mystifiant dans le passé, mais il est facilement déchiffrable, par exemple en faisant l'analyse de la fréquence des lettres, et en l'utilisant pour établir un faible nombre de possibilités pour la clé secrete.

6.1 Substitution : Vigenere, Vernam, one-time pad

Voici des exemples de protocoles symétriques, outre le code de César.

Le code de *Vigenere* utilise une famille finie de chiffres de César pour différentes clés. Essentiellement, la clé secrete est une suite de nombres $s_0, \ldots, s_{k-1}$ et pour la n-ième lettre, on utilise le chiffre de César de clé $s_{\bar{n}}$ où $\bar{n}$ est le residu de n modulo k. En pratique la suite de nombres $s_1, \ldots, s_k$ est donné par un un mot, chaque $-s_i$ etant la position de la i-eme lettre de ce mot dans l'alphabet.

Par exemple le mot secret PASTEQUE indique que pour la premiere lettre du message secret, on doit lui retirer 16, pour la deuxieme lettre on doit lui retirer 1 pour la troisieme, 19 etc... (pour la neuvieme lettre on doit retirer 16 à nouveau et ainsi de suite).

On peut coupler le chiffre de Vigenere avec une clé de chiffrement longue à volonté, engendrée pseudo-aléatoirement par un registre à décalage par exemple. Ainsi le veritable secret (le registre à décalage) est de taille bornée et raisonnable, alors que la clé de chiffrement est de taille aussi longue qu'on veut.

Le code de *Vernam* est un systeme de chiffrement symétrique ou la clé secrete est un texte entier de 0 et de 1. Le message est codé par un XOR (ou exclusif) lettre par lettre avec la clé secrete. Pour interpreter le message, il suffit de refaire un XOR avec la clé secrete. Si la clé est vraiment aléatoire et n'est utilisée qu'une seule fois, c'est un système impénétrable (one-time-pad). En revanche si la même clé est utilisée plusieurs fois, on peut recuperer de l'information sur les messages comme par exemple le resultat par XOR des deux messages clairs entre eux.

En fait, si l'on prend le chiffre de Vigenere avec une clé secrete aléatoire et longue (aussi longue que le message), et utilisée une seule fois, alors on peut montrer qu'il est (équivalent à) un one-time pad.

De nos jours, on doit supposer que l'équippement cryptographique est standardisé et répendu. On doit alors supposer que le cryptanalyste peut soumettre à volonté des textes choisis au système de chiffrement et observer les résultats (on parle d'attaques à clair choisi). Il est assez évident que la clé d'un code de Vigenere à clé réutilisée ne resistera pas longtemps.

7 Séance 7 : Techniques asymétriques

7.1 Fonctions à sens unique

7.1.1 Tentative de définition

Une fonction

$$f : \mathcal{E} \to \mathcal{F}$$

est dite à sens unique si

— pour tout $x \in \mathcal{E}$ le calcul de $f(x)$ est « facile »

— pour une proportion significative d'éléments de $\mathcal{F}$, le calcul d'un antécédent est « difficile »

Pour rendre rigoureuse cette définition, on devrait être obligé de parler plutot de fonction à sens unique de parametre ϵ, (cela signifie que la « proportion significative » d'éléments est superieure à $(1-\epsilon)$, ou bien de suites de fonctions à sens unique, pour laquelle cette proportion tend vers 0.

Un autre point à rendre rigoureux est bien sur l'usage des mots « facile » et « difficile ». Ils s'interpretent convenablement en terme de complexité des algorithmes. Informellement, on dit que le calcul de $f(x)$ est facile si le nombre d'operations élémentaires (addition, multiplications...) que l'on éffectue est linéaire ou sous-quadratique en fonction de la longueur de x (la longueur, pas la « valeur »), et que le calcul est difficile si ce nombre est *necessairement* au moins exponentiel. On peut interpreter cela de deux manières : soit il s'agit d'asymptotiques (et avoir pré-enregistrer des antécédents pour certaines valeurs ne change pas l'asymptotique), soit on parle de vrai fonctions qui minorent partout, (et dans ce cas, avoir enregistré quelques valeurs ne change que la proportion des antécédants faciles à calculer ; l'asymptotique est alors sur la suite de fonctions).

Les protocoles asymétriques sont ceux qui utilisent une fonction (que l'on pense être) à sens unique.

7.1.2 Exemple : stockage de mots de passe

Un utilisateur nommé Ursule utilise une machine M avec connection par mot de passe. Cependant, l'espion Eve va sans-doute réussir à capter tout ce que l'utilisateur tapera sur son clavier de connection.

On convient alors du protocole suivant. (Lamport, 1981)

On fixe $f : \mathcal{E} \to \mathcal{F}$ une fonction à sens unique.

Ursule choisit un mot de passe $w \in \mathcal{E}$. Ni M ni Eve n'ont accés à ce mot de passe. Il calcule en secret $f(w), f^2(w), \ldots, f^{1000}(w)$. Il entre, dans la machine M, la valeur de $f^{1000}(w)$. Ce mot est stoqué dans une variable « `motdepassedeUrsule` ».

Eve capte cete information, et Eve connait aussi la fonction f.

A la premiere connexion, Ursule entre $w' = f^{999}(w)$, la machine M calcule $f(w')$ et le compare à `motdepassedeUrsule`.

Si c'est different, on arrete tout. Si ça coincide, M remplace `motdepassedeUrsule` par cette valeur de w', et laisse Ursule utiliser ses services.

A la k-ième connexion, Ursule entre $w' = f^{1000-k}(w)$, puis M calcule $f(w')$ et le compare à `motdepassedeUrsule`. Si c'est different, on arrete tout. Si ça coincide, M remplace `motdepassedeUrsule` par cette valeur de w' et laisse Ursule utiliser ses services.

Ursule peut se connecter 1000 fois, mais si Eve veut se connecter, elle doit inverser f pour l'une des valeurs qu'elle voit passer (dans l'ordre : $f^{1000-k}(w)$ pour $k = 1, \ldots, 1000$. Si ces valeurs sont toutes dans la bonne partie de l'ensemble $\mathcal{F}$, cette tâche est difficile.

7.2 Une fonction à sens unique : l'exponentielle modulaire et le problème du Log discret ?

7.2.1 L'exponentielle modulaire

On choisit p premier, et $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$.

$$\exp_\alpha : \begin{array}{ccc} (\mathbb{Z}/p\mathbb{Z})^* & \to & (\mathbb{Z}/p\mathbb{Z})^* \\ x & \mapsto & \alpha^x \,[\mathrm{mod}\, p] \end{array}$$

Exercice

1. Montrer que α est primitif si et seulement si $\exp_\alpha$ est bijective.
2. Montrer qu'on peut calculer $\exp_\alpha(x)$ en moins de $2\log_2(p) + 1$ multiplications modulo p. (chacune de ces multiplications est donc de « complexité » bornée, car elle intervient entre nombres inferieurs à p.) *(indication : utiliser le developpement binaire de x, $x = \sum 2^{n_k}$ et décomposer (α^x) en $((\alpha^2)^2)^{\cdots 2} \times ((\alpha^2)^2)^{\cdots 2} \times \ldots ((\alpha^2)^2)^{\cdots 2}$ ou il y a au k-ieme facteur, n_k carrés consécutifs...)*
3. Montrer que si on sait factoriser $p - 1$ on sait trouver un élément primitif « facilement » (en faisant des essais successifs facilement testables)
4. Si p n'est plus premier, à quelle condition existe-t-il un α tel que $\exp_\alpha$ est bijective ?

7.3 Protocoles liés à l'exponentielle modulaire : Diffie-Hellman et El Gamal

7.3.1 Diffie Hellman

Afin d'obtenir un **secret commun**, Alice et Bob utilisent le protocole de Diffie Hellman.

1. On choisit p premier, et $\alpha \in (\mathbb{Z}/p\mathbb{Z})^*$. Ces informations sont publiques, Eve les connait.
2. Alice choisit en secret $s_a \in (\mathbb{Z}/p\mathbb{Z})^*$.
3. Bob choisit en secret $s_b \in (\mathbb{Z}/p\mathbb{Z})^*$.
4. Alice calcule $\alpha^{s_a} \,[\mathrm{mod}\, p]$, et envoie à Bob (et Eve peut le voir) la valeur $\alpha^{s_a} \,[\mathrm{mod}\, p]$.
5. Bob envoie de même à Alice (et Eve peut le voir) la valeur $\alpha^{s_b} \,[\mathrm{mod}\, p]$.
6. Alice calcule en secret $(\alpha^{s_b})^{s_a} \,[\mathrm{mod}\, p]$.
7. Bob calcule en secret $(\alpha^{s_a})^{s_b} \,[\mathrm{mod}\, p]$.

Observation : ces deux valeurs coincident et valent $\alpha^{s_a \times s_b} \,[\mathrm{mod}\, p]$. C'est leur secret commun.

Observation : si Eve souhaite calculer le secret, il lui suffit de trouver s_a en sachant $\alpha^{s_a} \,[\mathrm{mod}\, p]$ et α. Si la fonction $\exp_\alpha$ est bien à sens unique, c'est « en général » dûr.

7.3.2 El Gamal

Alice et Bob utilisent le protocole de El Gamal, cette fois pour *chiffrer* un message.

1. On convient d'un nombre premier p. Bob possède une clé secrete $s \in (\mathbb{Z}/p\mathbb{Z})^*$.
2. Il a aussi une clé publique en deux partie α (que Alice et Eve voient) et $\mathbf{P}$ qui vaut $\mathbf{P} = \alpha^s \,[\mathrm{mod}\, p]$.
3. Alice compose un message $M \in (\mathbb{Z}/p\mathbb{Z})^*$.
4. Alice tire au hasard $k \in (\mathbb{Z}/p\mathbb{Z})^*$. Elle seule le connait.
5. Alice calcule $C_1 = \alpha^k \,[\mathrm{mod}\, p]$. Elle calcule aussi $C_2 = \mathbf{P}^k \times M \,[\mathrm{mod}\, p]$.
6. Alice envoie (C_1, C_2) à Bob (et Eve le voit).

Observation : Eve veut décrypter. Il lui suffit de chercher $\mathbf{P}^k$, peut-être en cherchant k. Elle a alors un problème d'inversion de l'exponetielle (ou de logarithme discret).

Observation : Bob veut décrypter. Il ecrit $\mathbf{P}^k = (\alpha^s)^k = \alpha^{sk} = (C_1)^s$. Il peut facilement le calculer. Il suffit alors de l'inverser (trouver $(\mathbf{P}^k)^{-1}$ dans $(\mathbb{Z}/p\mathbb{Z})^*$) et de calculer $M = C_2 \times \mathbf{P}^k)^{-1}$.

On a utilisé le concept de *porte dérobée* à la fonction à sens unique : une fonction $f_t : \mathcal{E} \to \mathcal{F}$ est une fonction à sens unique si l'on ne connait pas t, mais c'est une fonction facilement inversible partout si l'on connait t. Ici le secret de la porte dérobée est le secret s de Bob. En anglais, on trouve le vocable *one-way trap door*.

7.3.3 Schema de signature

Alice veut prouver à Bob que c'est bien elle l'auteur du message « BUPconfirmationquelepayementesteffectueBIPprocederimmediatementalalivraisonBOP »

Bob veut etre certain que c'est bien Alice qui l'a ecrit, et non pas Aline, ou Malice...

1. Alice choisit un secret s.
2. On dispose d'une fonction à sens unique f. Elle rend publique $f(s)$.
3. Alice appose à son message la signature $S = \phi(M, s)$ pour une certain fonction ϕ.

Exemple : Alice a comme avant une clé publique $(\alpha, p, \mathbf{P})$, avec p un grand nombre premier, α un élément primitif de $(\mathbb{Z}/p\mathbb{Z})^*$, et $\mathbf{P} = \alpha^s \,[\mathrm{mod}\, p]$.

Son secret s est un élément de $(\mathbb{Z}/p\mathbb{Z})^*$.

Son message est $M \in (\mathbb{Z}/p\mathbb{Z})^*$ (en clair ; il n'est pas chiffré dans cet exemple).

Elle compose sa signature ainsi.

— Choix de $k \in (\mathbb{Z}/(p-1)\mathbb{Z})^*$.

— Calcul de $u = \alpha^k \,[\mathrm{mod}\, p]$.

— Calcul de l'unique solution v de $M = us + kv \,[\mathrm{mod}\, p-1]$
il s'agit de calculer l'inverse de k (mod $(p-1)$), qui existe bien, et cela peut se faire par l'algorithme d'Euclide. En fait $v = (M - us) \times k^{-1} \,[\mathrm{mod}\, p-1]$
— Signature $= (u, v)$.

Exercice

1. Verifier que Bob, en voyant la signature, et en connaissant le message M, peut facilement calculer α^M et $\mathbf{P}^u \times u^v$ (modulo p).
2. Montrer que si Alice a ecrit le message, ces quantités sont égales.
3. Expliquer comment Bob peut se convaincre que Alice a bien ecrit le message M. (Quel probleme doit résoudre Malice ou Aline pour usurper l'identité d'Alice ?)

7.4 Protocole lié à la factorisation : RSA

Du nom de ses inventeurs Rivest, Shamir, Adelman.

1. Le secret de Bob est une paire de nombres premiers (p, q).
2. La clé publique est $n = p \times q$, et le choix d'un élément e inversible modulo $\phi(n)$.
   Observation. Ici $\phi(n) = (p-1)(q-1)$.
3. Alice compose le message M. Elle calcule $M^e \,[\mathrm{mod}\, n]$ et le publie.
4. Bob déchiffre en calculant $(M^e)^d$, pour d un inverse de e modulo $\phi(n)$.

Observation : Cela fonctionne car $ed = k\phi(n) + 1$ pour un certain k, et donc $ed = k(p-1)(q-1) + 1$. Il suffit de voir que $M^{k(p-1)(q-1)} \equiv 1 \,[\mathrm{mod}\, pq]$. Mais modulo p, $M^{k(p-1)(q-1)}$ vaut bien 1 par Fermat, et bien sur aussi modulo q, $M^{k(p-1)(q-1)}$ vaut bien 1. Ainsi, par le théoreme Chinois, on a bien $M^{k(p-1)(q-1)} \equiv 1 \,[\mathrm{mod}\, pq]$.

Exercice

1. Connaitre $\phi(n)$ c'est équivalent à connaitre p et q.
2. Si l'on connait p et q, et e, on peut facilement connaitre d un inverse de e modulo $\phi(n)$.
3. Connaitre d et e c'est (souvent) connaitre $\phi(n)$.

Eve observe tout. Pour déchiffrer, il lui suffit de calculer d, et pour cela il lui suffit de connaitre $\phi(n)$. Quel problème doit elle résoudre ?

Exercice On utilise RSA avec l'exposant $e = 3$. On suppose qu'un utilisateur veut envoyer le même message M à trois destinataires, de module RSA n_1, n_2, n_3 qui n'utilisent pas deux fois les même facteurs premier. Comment Eve, qui observe les trois valeurs $M^3 \,[\mathrm{mod}\, n_i]$ peut retrouver M ?

7.5 TP « protocoles »

Par paires de binomes, confectionner le protocole codage/décodage El Gamal ou RSA (ou encore un protocole de signature), au choix.

Transformer vos messages en binaire (en ascii) avec par exemple `https://mothereff.in/binary-ascii`, encodez les avec votre protocole, envoyez à l'autre binôme, qui décodera pour obtenir le binaire original, puis pourra retrouver le texte grace a un convertisseur binary-ascii.

Commencer par essayer d'envoyer des nombres tres simples pour des petits parametres de RSA. Puis, quand ca marche, essayer d'envoyer de grands nombres avec de grands parametres RSA. Puis, essayer d'envoyer de vrai messages (comment coder un « vrai » message sous forme de nombre ?)

Il faut que vous choisissiez : comment choisir les nombres premiers impliqués dans votre protocole ? Dans certaines circonstances, est-ce qu'on peut les deviner en voyant seulement N leur produit ? Par exemple : que se passse-t-il si l'on choisit des nombres du genre « nextprime(10^{20}) » ?

Comment choisir e et calculer d ?

Pour RSA : Indiquez àu binôme partenaire la methode que vous utilisez pour choisir vos nombres premiers (juste la methode, pas les nombres). Le binôme d'en face peut il essayer de factoriser n ?...

Communiquer e et d. Le binôme partenaire doit factoriser n...

8 Séance 8 : Sécurité de protocoles et attaques

Désormais, l'espion Eve ne reste plus passif/attentif, mais passe à l'action.

8.1 Quelques approches du problème du Log discret

On veut donc résoudre l'équation dans $(\mathbb{Z}/p\mathbb{Z})^*$, d'inconnue x, dans $\mathbb{Z}/(p-1)\mathbb{Z}$:

$$\alpha^x = \beta$$

où α et β sont donnés, et où l'on suppose α primitif.

Voici une approche, dite du « calcul d'incide ».

— On choisit d'abord un ensemble $B = \{b_0, b_1, \ldots, b_h\}$ d'entiers (typiquement des nombres premiers), dont on connait les logarithmes discrets de base α (elements de $\mathbb{Z}/(p-1)\mathbb{Z}$)).

— On chercher e tel que $\beta\alpha^e$ se décompose modulo p dans B :

$$\beta\alpha^e \equiv \prod_j b_j^{e_j} \ [\mathrm{mod}\, p]$$

— On en déduit que pour les logarithmes discrets (de base α)

$$\log_\alpha(\beta) \equiv \left(\sum_j e_j \log_\alpha(b_j)\right) - e \ [\mathrm{mod}\, p-1].$$

Il reste à choisir B, trouver les log discrets de ses éléments, et trouver e.

La première méthode est le *crible linéaire*. Ici $B = \{-1, p_1, \ldots, p_h\}$ est la liste de -1 et des premiers nombres premiers.

On a $-1 = \alpha^{(p-1)/2}$ parce que α etant primitif, ce n'est pas un carré.

Pour trouver les log discrets des p_j, on choisit c_i au hasard, on évalue $\alpha^{c_j} \bmod p$, dont on prend le representant dans $[-p/2, p/2]$, et on regarde s'il se décompose dans la base B.

Si oui, il admet une factorisation

$$\alpha^{c_i} = \prod_j p_j^{\alpha_{i,j}} \ [\mathrm{mod}\, p].$$

Dès qu'on a trouvé un tel c_i donnant une telle factorisation, on conserve l'equation equivalente :

$$c_i = \sum_j a_{i,j} \log_\alpha(p_j) \ [\mathrm{mod}\, p-1].$$

Quand on a assez déquations pour que le système soit surdeterminé, il suffit de le résoudre comme un système linéaire. On a alors les logarithmes des elements de la base B.

Il existe d'autres cribles mathématiquement tres interessants, comme le crible Gaussien, qui fait intervenir les entiers de Gauss $\mathbb{Z}[i]$, ou encore la méthode de Pohlig-Hellman.

Cette dernière fait un lien entre le problème du log discret, et le problème de la factorisation des entiers. Elle est efficace si $(p-1)$ n'a que des petits facteurs premiers.

Disons que

$$p-1 = p_1^{\lambda_1} p_2^{\lambda_2} \dots p_k^{\lambda_k}.$$

Prenons α primitif dans $(\mathbb{Z}/p\mathbb{Z})^*$, et $\beta = \alpha^x$.

Pour trouver x (connaissant β et α) il suffit de trouver x modulo chaque $p_i^{\lambda_i}$ (par le théorème Chinois). Notons donc x_i le reste de x modulo $p_i^{\lambda_i}$.

Calculons aussi $N_i = (p-1)/(p_i^{\lambda_i})$, c'est à dire que N_i est le produit des $p_j^{\lambda_j}$ pour tous les indices sauf i. On peut calculer cela facilement si l'on connait la decomposition de $p-1$.

On peut observer (exercice de théorie des groupes !) que si $\alpha_i = \alpha^{N_i}$ et si $\beta_i = \beta^{N_i}$, alors

$$\beta_i = \alpha_i^{x_i}.$$

Si p_i est petit, on peut trouver plus facilement x_i, par force brute (recherche exhaustive) ou par une autre méthode, comme un crible, ou une autre méthode comme celle du Baby-step-Giant-step.

Par exemple, montrer que, si l'on ecrit x_i en base p_i comme

$$x_i = z_0 + z_1 p_i + \dots + z_{\lambda_i - 1} p_i^{\lambda_i - 1}$$

et si l'on considère successivement $\alpha_i^{p_i^{\lambda_i-1}}, \alpha_i^{p_i^{\lambda_i-2}}, \dots$, on peut déterminer les z_j en p_i operations chacun.

8.2 Pas de bébé, pas de géant

Une attaque du log discret très commune s'appelle la méthode "baby step giant step". Décrivons là.

Soit $G = (\mathbb{Z}/n\mathbb{Z})^*$, prenons un élément α dedans, et notons $\beta = \alpha^x$. On doit retrouver x modulo $n-1$.

Soit m plus grand que $\lfloor \sqrt{|G|} \rfloor$. On calcule (i, α^i) pour i variant de 1 à m. Ce sont les pas de bébé.

On calcule $(j, \beta \times \alpha^{-jm})$ pour j de 1 à m. Ce sont les pas de géant.

Si on trouve i et j de sorte que $\alpha^i = \beta \times \alpha^{-jm}$, alors simplement, $x = i + jm$.

Exercice Montrer que ce processus découvre le log discret de β. En pratique, comment choisir m ? Quel ordre de grandeur du nombre d'operations élémentaires peut on prévoir, pour ce processus ? Programmer cela en TP, et observer le temps de calcul (éventuellement comparer avec votre ordre de grandeur).

8.3 Le bit de parité

Théorème 8.3.1 *Soit N un module pour RSA impair. Les deux problemes suivants sont d'une difficulté équivalente :*

– Etant donné un message arbitraire chiffré, trouver le message original

– Etant donné un message arbitraire chiffré, trouver la parité (le dernier bit) du message original.

On peut en effet décrire une réduction du premier problème au second, en temps polynomial (en fonction de la longueur de N). Siu cela est établi, cela signifie que n'importe qui sachant faire le deuxieme probleme, saura faire le premier, au prix d'un effort « polynomial », à travers cette réduction.

Le principe de la preuve du théorème repose sur le fait suivant.

Lemme 8.3.1 *Si N est impair, et si $x \in]0, N[$, alors le representant de $2x$ modulo N entre 0 et N est pair si et seulement si $x \leq N/2$.*

(Si $x \leq N/2$, alors $2x \leq N$ et c'est lui son propre representant, et clairement il est pair. Pour la reciproque, si $x > N/2$ alors $N > 2x \leq 2N$ et le representant de $2n$ est exactement $2x - N$ qui est impair.)

On peut alors décrire l'algorithme de reduction du premier problème au second. Pour cet algorithme, on suppose qu'on sait résoudre le second problème. On dit qu'on a un oracle pour le second problème.

On part de $c = m^e \,[\mathrm{mod}\, N]$ un message chiffré.

On sait au départ que $m \in]0, N[$.

On demande à l'oracle quel est le bit de parité de $2^e m^e = (2m)^e$. De sa réponse, grace au lemme, on sait si m se trouve dans $]0, N/2[$ ou dans $]N/2, N[$.

Par récurence, on va supposer qu'on a reussi à la k-ieme étape à découpe $]0, N[$ en 2^k parties égales

$$]0, N/2^k[,\ ]N/2^k, 2N/2^k[,\ ]2N/2^k, 3N/2^k[\ldots](2^k - 1)N/2^k, N[$$

si que l'on sait que (le representant de) m est dans l'un d'entre eux : $]aN/2^k, (a+1)N/2^k[$ de taille $N/2^k$.

On utilise l'oracle pour connaitre le bit de parité de $(2^{k+1})^e m^e = (2^{k+1}m)^e$, et l'on détermine dans quelle moitié de l'intervalle $2^k m$ se trouve.

Par exemple si le bit de parité est 0, alors le representant de $2^k m$ dans $]0, N[$ est dans $]0, N/2[$. Mais $2^k m \leq (a+1)N$ donc en fait $2^k m \leq (a+1)N - N/2$, et donc $m \leq (a+1)N/2^k - N/2^{k+1}$. Cela signifie que m est dans la premiere moitié de $]aN/2^k, (a+1)N/2^k[$. On a donc cerné m dans un intervalle de taille moitié, et on peut continuer la récurrence.

En $\log_2(N) + 1$ opérations, on a réduit l'intervalle a un seul point. Remarquons pour finir que $\log_2(N) + 1$ est bien polynomial (linéaire en fait) en la longueur de N.

8.4 Sensibilité aux impostures : attaques « man-in-the-middle »

Eve peut tenter une imposture en se faisant passer pour Bob dans l'echange des clés.

Si elle intercepte les massages d'Alice pour Bob, et y repond en faisant croire que c'est Bob l'expediteur, elle partage a la fin une clé entre Alice et Eve, alors que Alice pense que c'est une clé entre Alice et Bob. Elle peut déchiffrer les messages que Alice adresse à Bob.

8.5 Faiblesses de « meet in the middle »

Prenons le protocole RSA, et supposons que Eve sait que le message secret est inferieur à 2^ℓ pour un certain ℓ (par exemple, parce qu'elle sait que le message secret est la valeur d'une enchère secrete de Alice sur un objet vendu par Bob, ou parce que c'est un mot de passe court).

Il est possible qu'avec une probabilité non-négligeable, le message (clair) M se factorise $M = m_1 m_2$, avec les deux facteurs m_i inferieurs à $2^{\ell/2}$.

Le message chiffré est $c = M^e = m_1^e m_2^e$ (les égalités sont prises modulo n).

Si Eve a une base de donnée $\{1^e, 2^e, 3^e, \ldots, (2^{\ell/2})^e\}$, elle peut verifier si une paire de ces valeurs donne comme produit c (modulo n).

Si Eve découvre une telle coincidence,, disons $c = i^e \times j^e$ (modulo n), elle peut ecrire $c = (i \times j)^e$. Mais comme e est inversible et que c est aussi égal à m^e, on sait que $ij = m$. Comme Eve connait i et j, elle connait m.

Exercice Estimer l'espace mémoire necessaire à Eve, et le nombre d'operations à faire...

Si on suppose que $m \leq 2^{56}$, combien d'operations utilise cette méthode ? Sur une machine qui permet 10^9 opérations à la secondes, combien de temps prends l'attaque ?

Comparez à une attaque directe, qui essaye de calculer tous les i^e...

Pour une taille de message comprise entre 2^{40} et 2^{64}, la probabilité d'avoir $m = m_1 m_2$ avec chaque m_i de taille environ moitié varie de 18% à 50%.

Exercice Décrire l'attaque « meet in the middle » pour le protocole El Gamal.

8.6 TP « factoriser » ou « meet-in-the-middle »

8.6.1 Log discret

Mettre en place une attaque du log discret : dans un premier temps, produite des nombres premiers p tels que $p-1$ n'ait que des petits facteurs premiers. (Cela permet de potentiellement tester Pohlig-Hellman) Produire α primitif. (On peut anticiper sur le théorème de Lucas)

Mettre en place un crible linéaire, et/ou une méthode de Pohlig-Hellman.

8.6.2 Factoriser avec la connaissance de d

Par paires de binomes faites le défi suivant : Le binôme A selectionne aléatoirement deux grands nombres premiers p_1, p_2, les multiplie pour avoir N, selectionne e acceptable pour RSA, calcule d. Il communique (N, e, d) au binôme B. Le binôme B doit trouver la factorisation de N.

(evidemment, reciproquement le binôme B fait de même pour le A)

8.6.3 « meet in the middle »

Mettre en place un protocole RSA ou El Gamal, au choix. En 1024 bits, prendre comme taille du message à crypter un nombre inferieur à un certain parametre t qu'on prendra initialement $= 2^{40}$, et qu'on pourra faire grandir jusquá 2^{50}.

Mettre en place le dispositif d'attaque brutale (qui essaye toutes les possibilités). A quelle taille de message est-ce que cette methode reste efficace ?

Mettre en place le dispositif de Eve pour utiliser l'attaque « meet in the middle » pour ces chiffrements.

Faire des tests statistiques pour estimer la proportion de message qui sont effectiment attaquable par l'attaque meet-in-the-middle.

A la maison (en faisant tourner l'ordinateur longtemps s'il le faut), essayer de dechiffrer (en jouant le role de Eve) des messages de taille 2^{56}.

9 Séance 9 : Trouver des nombres premiers

On a vu qu'il etait crucial de trouver facilement, rapidement des nombres premiers tres grands, qui ont l'air aléatoires.

Exercice Faire un inventaire des circonstances dans lesquelles on a eu besoin d'un nombre premier, en faisant la difference entre les besoins d'un nombre premier public, ou d'un nombre premier secret.

Théorème des nombres premiers (Hadamard, de la Vallée Poussin) Le nombre $\pi(n)$ de nombres premiers inferieurs à n est asymptotiquement équivalent à $\frac{n}{\log n}$.

9.1 Test de primalité de Fermat

On se souvient du petit théorème de Fermat qui dit que si p est premier et si $a \neq 0$, on a $a^{p-1} \equiv 1 \,[\mathrm{mod}\, p]$.

Ainsi, si n est un nombre, et si $0 \leq a \leq n$, calculer $a^{n-1} \,[\mathrm{mod}\, n]$ est un test de primalité pour n : si par chance on trouve a tel que $a^{n-1} \neq 1 \,[\mathrm{mod}\, n]$, on sait que n n'est pas premier.

La reciproque est fausse : il existe des nombres n qui, bien que satisfaisant au test precedent pour tout a premier avec n, ne sont pas premiers. On les appelle les nombres de Carmichael.

Efficacité du test, en prenant $a = 2$.

Un résultat (Pomerance 1981) montre que le nombre de nombres satisfaisant le test de Fermat pour $a = 2$, et inférieurs à n, est $\pi_2(n)$ verifiant

$$\pi(n) + e^{(\log(n))^{5/14}} \leq \pi_2(n) \leq \pi(n) + ne^{-\frac{1}{2}\log(n)\log\log\log(n)/\log\log(n)}$$

On peut alors verifier que le rapport de $\pi_2(n) - \pi(n)$ sur $\pi(n)$ tend vers 0 rapidement.

Exercice (TP-Sage) Quelle est la proportion de nombres inferieurs à 2^{15} satisfaisant le test de Fermat pour $a = 2$? Quelle est le nombre de nombres premiers inferieurs à 2^{15} ?

Exercice Soit $n = 2^{\alpha_0} \times p_1^{\alpha_1} \times \cdots \times p_r^{\alpha_r}$ avec tous les p_i premiers. Posons $\lambda(2^t) = 2^{t-1}$ si $t < 3$ et $= 2^{t-2}$ si $t \geq 3$. Soit $\lambda(n) = \mathrm{ppcm}(\lambda(2^{\alpha_0}), \phi(p_1^{\alpha_1}), \ldots, \phi(p_r^{\alpha_r})$. Montrer que pour tout a inversible modulo n, on a $a^{\lambda(n)} \equiv 1 \,[\mathrm{mod}\, n]$.

(utiliser les restes Chinois, et le théorème de Gauss suivant : $(\mathbb{Z}/n\mathbb{Z})^*$ est cyclique si et seulement si n est de la forme $2, 4, p^k, 2p^k$ pour p premier impair ; on montrera aussi que si a impair, $a^{\lambda(2^t)} \equiv 1 \,\mathrm{mod}\, 2^t$).

Exercice Verifier que 1729 est un nombre de Carmichael, et plus généralement $(6t+1)(12t+1)(18t+1)$ en est un si les trois facteurs $(6t+1)$, $(12t+1)$, $(18t+1)$ sont premiers.

Commentaire En pratique pour un nombre construit soi même, le test de Fermat est satisfaisant. Mais si c'est un nombre fourni par un tiers, on est en droit de s'inquieter de l'honeteté du fournisseur : s'il fournit un nombre de Carmichael, on ne le detectera pas avec Fermat, et il aura créé une faiblesse dans notre équipement...

9.2 Test de Soloway-Strassen, un test probabiliste

Rappellons un théorème d'Euler.

Théorème d'Euler Si p est un nombre premier impair, alors pour tout x premier avec p, on a

$$x^{(p-1)/2} \equiv \left(\frac{x}{p}\right) \ [\mathrm{mod} p]$$

où le membre de droite est le symbole de Legendre (=1 ou -1 selon que x est un carré ou pas).

Le symbole de Legendre se généralise en symbole de Jacobi si le modulo residuel n'est pas premier. Il vaut alors le produit des symboles de legendre dans les quotients residuels modulo les facteurs premiers.

Lemme On a les deux formules suivantes :

$$\left(\frac{2}{n}\right) = (-1)^{(n^2-1)/8}$$

$$\left(\frac{-1}{n}\right) = (-1)^{(n-1)/2}$$

Loi de reciprocité quadratique : si m et n sont premiers entre eux, impairs (et $\neq 1$) alors

$$\left(\frac{m}{n}\right)\left(\frac{n}{m}\right) = (-1)^{(n-1)(m-1)/4}$$

Exemple d'application Calculons $\left(\frac{31}{91}\right)$.

$$\left(\frac{31}{91}\right) = -\left(\frac{91}{31}\right) = -\left(\frac{29}{31}\right) = -\left(\frac{31}{29}\right) = -\left(\frac{2}{29}\right) = 1.$$

On a un nombre n dont on veut tester la primalité.

Test de Soloway-Strassen Choisir au hasard a premier à n, calculer $a^{(n-1)/2}$ modulo n. Calculer le symbole de Jacobi : $\left(\frac{a}{n}\right)$.

L'entier n a satisfait le test si

$$a^{(n-1)/2} \equiv \left(\frac{a}{n}\right) \ \mathrm{mod}\, n.$$

Proposition Si n n'est pas premier alors l'ensemble des a verifiant le test est un sous-groupe strict de $(\mathbb{Z}/n\mathbb{Z})^*$.

Application : il y a donc au moins autant d'elements ne verifiant pas le test que d'element verifiant le test. En supposant que n n'est pas premier, la probabilité de tomber uniquement sur des éléments a verifiant le test, en k tentatives aléatoires indépendantes est $\leq 2^{-k}$.

Preuve de la proposition Soit G_n les valeurs de a satisfaisant le test. C'est un sous groupe de $(\mathbb{Z}/n\mathbb{Z})^*$ car le symbole de Jacobi est un morphisme de groupe dans $\mathbb{Z}/2\mathbb{Z}$. Supposons que c'est $(\mathbb{Z}/n\mathbb{Z})^*$ tout entier.

Montrons que n n'a pas de facteur carré.

Soit p un diviseur de n et p^t la plus grande puissance qui divise encore n.

D'apres Gauss, il existe un générateur g de $(\mathbb{Z}/p^t\mathbb{Z})^*$. D'apres le théorème Chinois, il existe $a \in (\mathbb{Z}/n\mathbb{Z})^*$ tel que

$$a \equiv g \,[\mathrm{mod}\, p^t] \qquad a \equiv 1 \,[\mathrm{mod}\, n/p^t].$$

Ainsi a est d'ordre $\phi(p^t)$ dans $(\mathbb{Z}/n\mathbb{Z})^*$.

Mais par hypothèse, $a^{n-1} = 1$ dans $\mathbb{Z}/n\mathbb{Z}$. Donc $\phi(p^t)$ divise $n-1$.

Comme $\phi(p^t) = p^{t-1}(p-1)$ et puisque p ne divise pas $n-1$ (il divise n), on a forcement $t = 1$.

On a donc que n n'a pas de facteur carré.

Supposons, en raisonnant par l'absurde, que n n'est pas premier. Sa decomposition en facteurs premiers est donc de la forme $n = p_1 p_2 \dots p_k$ avec $k \geq 2$.

Encore d'apres le théorème des restes Chinois, il existe a qui n'est pas un carré modulo p_1 et tel que $a \equiv 1 \bmod n/p_1$.

Par définition du symbole de Jacobi, on a alors

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)\left(\frac{a}{n/p_1}\right) = -1.$$

Maintenant par hypothèse, a satisfait au test. Donc $a^{(n-1)/2} \equiv -1 \,[\mathrm{mod}\, n]$.

Mais, en premnant la réduction modulo n/p_1, cela signifie que $a^{(n-1)/2} \equiv -1 \,[\mathrm{mod}\, n/p_1]$, ce qui contredit que (par propriété de a) $a \equiv 1 \bmod n/p_1$. Finalement de notre raisonnement par l'absurde, on en retire bien que n est premier. La proposition est démontrée.

9.3 Test de Miller Rabin

Choisir a au hasard, premier à n, ecrire $n - 1 = 2^s t$ avec t impair et calculer

$$a^t,\, a^{2t},\, \dots,\, a^{2^{s-1}t} \pmod n$$

L'entier n satisfait le test si toutes ces valeurs sont 1 ou bien si l'une d'elles vaut -1.

Notez qu'après -1 toutes les valeurs sont 1.

Si n est premier, on a $a^{n-1} = 1$ par Fermat, et ainsi, $a^{(n-1)/2}$ qui est l'une des deux racines de 1, soit 1 soit -1. On peut continuer tant que le résultat est 1 et tant que $(n-1)/2^k$ est pair (pour en prendre la racine). Dès qu'on observe -1, on ne prédit plus le résultat précédent. Ces remarques montrent que si n est premier, il satisfait au test de Miller-Rabin pour tout a premier à n.

Proposition Si un entier est premier, il satisfait au test de Miller Rabin pour tout a.

Proposition Si un entier satisfait au test de Miller-Rabin pour a, il satisfait au test de Soloway Strassen pour a.

9.4 Théorème de Lucas, et TP.

Théorème de Lucas Le nombre n est premier si et seulement s'il existe $\alpha \in (\mathbb{Z}/n\mathbb{Z})^*$ tel que $\alpha^{n-1} \equiv 1 \,[\mathrm{mod}\, n]$, et $\alpha^{(n-1)/p} \neq 1 \,[\mathrm{mod}\, n]$ pour tout diviseur premier p de $(n-1)$.

Utilisation : supposons qu'on ait des nombres premiers à disposition, $p_1, \ldots, p_k$. On peut construire $n = 1 + p_1^{r_1} p_2^{r_2} \ldots p_k^{r_k}$. On connait la factorisation de $n-1$. On peut alors tester pour α aléatoire, si les conditions sont verifiées. Au bout de quelques essais (pour n et pour α) on espere avoir une réponse positive. C'est très pratique pour implementer une exponentielle discrete, mais en revanche cela prête le flanc à l'attaque de Pohlig-Hellman.

Observation : il faut que $p_1 = 2$...

Cette approche fournit quand elle marche un nombre premier p et un élément primitif modulo p.

Preuve du théorème : si n est premier, α primitif convient. Réciproquement, si α convient, il est d'ordre precisement $n-1$, mais donc $n-1$ divise $\phi(n)$, l'indicatrice d'Euler, qui est forcement $\leq n-1$. Donc $\phi(n) = n-1$ et ainsi n est premier.

TP : implementation de Lucas Fabriquer, sur Sage, des nombres premiers grace au théorème de Lucas. (on gardera aussi l'élément primitif du groupe multiplicatif correspondant...)

9.5 Aspects de factorisation

Un paragraphe (adorable) de H.W. Jevons, dans un livre de 1873 (*The principles of Science*, remarquablement précurseur :

« *There are many cases in which we can easily and infallibly do a certain thing but may have trouble in undoing it. [...] Given any two numbers we may by a simple and infallible process obtain their product, but when a large number is given it is quite another matter to determine its factors. Can the reader say what two numbers multiplied together will produce the number*

8 616 460 799 ?

I think it is unlikely that anyone but myself will ever know ; for they are two large prime numbers. »

Comparez au résultat suivant de Lehmer de 1907 qui dit que si n est produit de deux facteurs premiers dont la différence est inférieure à $2(kn)^{1/4}$ alors on factorise facilement n en utilisant le developpement en fractions continues de $\sqrt{kn}$...

Références

[1] J. Hoffstein, J. Pipher, J. Silverman, An Introduction to Mathematical Cryptography, Ed. Springer, Undergraduate Text in Mathematics.

[2] J. Katz, Y. Lindell, Introduction to Modern Cryptography, CRC Press.

[3] Wembo Mao, Modern Cryptography, Theory and Practice. Prentice Hall, HP Publishing partner.

[4] Gilles Zemor, Cours de Cryptography, Ed. Cassini.