

# Rudiments de cryptographie et de théorie des codes correcteurs.

Dans ce document, j'espère vous apporter quelques éléments de cryptographie et de codage, susceptibles de nourrir vos leçons et dossiers d'arithmétique. Il va de soit que, désireux de rester à un niveau élémentaire, je ne propose qu'une entrée superficielle dans le sujet. J'invite ceux qui voudraient aller plus loin à consulter la bibliographie.

Comme je l'ai déjà mentionné, mon objectif est limité : il s'agit de vous présenter les problématiques de la cryptographie et du codage, puis de vous proposer quelques exemples qui constitueraient l'entrée en matière d'un cours à proprement parler.

## 1 Cryptographie.

On use de cryptographie lorsque l'on désire transmettre une information à un tiers de telle sorte qu'une personne non autorisée, quelles que soient ses compétences (celles du meilleur cryptanalyste), soit incapable de déchiffrer le message transmis, dans l'éventualité où ce message aurait été intercepté. Notre correspondant (et lui seul) doit être capable de reconstituer l'information à partir du message transmis. Pour cela, on crypte le message que l'on désire transmettre et on envoie le message crypté (ou cryptogramme). Notre interlocuteur doit alors pouvoir reconstituer le message initial à partir de son cryptogramme.

Les hommes envoient des messages cryptés depuis les temps les plus reculés et ont développé, dans cette science, une grande ingéniosité. Les procédés de cryptage ont été très divers. Leur sécurité reposait le plus souvent sur le caractère secret du protocole de cryptage. Les cryptosystèmes ont ensuite évolué pour devenir plus résistants aux éventuelles fuites. Sont ainsi apparus des cryptosystèmes résistants à la divulgation du protocole de cryptage et dont la sécurité repose sur un paramètre (la clé). La fonction de cryptage est alors construite en suivant un protocole (éventuellement connu de tous) et dépend (fortement) de la clé. Plus récemment, des cryptosystèmes où la fonction de cryptage est connue de tous ont été mis au point. Pour de tel système, il va de soit qu'il doit être difficile de déterminer la fonction de décryptage, même si la fonction de cryptage est connue. C'est pour cette raison que ces cryptosystèmes sont dits asymétriques, les autres étant qualifiés de symétriques.

### 1.1 Cryptosystèmes symétriques (à clé privée).

Les cryptosystèmes symétriques sont ceux où la connaissance de la fonction de cryptage équivaut, à priori, à celle de la fonction de décryptage. La sécurité de tel système repose soit sur le caractère secret du procédé de cryptage, soit sur l'existence d'une clé secrète que l'on peut changer en cas de besoin.

César faisait, par exemple, opérer une permutation circulaire des lettres de l'alphabet sur ses messages. La fonction de cryptage ne dépend que d'un paramètre, l'image de la

lettre A. Ce cryptosystème est évidemment très peu résistant et l'éventuelle clé (l'image de la lettre A) est pour ainsi dire inutile.

Je vous propose ci-dessous un cryptosystème symétrique basé sur une permutation des lettres de l'alphabet. Si il est plus robuste que le chiffrement de César, il résiste pas à une étude des fréquences d'apparition des lettres de l'alphabet dans la langue française. Le second cryptosystème, du à L. Hill, opère sur des paquets de lettres.

## 1.2 Exemples de cryptosystèmes à clé privée.

Dans les deux exemples qui suivent, on identifie les lettres de l'alphabet ou tout autre ensemble de symboles à un groupe cyclique. Pour l'alphabet ce sera  $\mathbb{Z}/26\mathbb{Z}$ .

### 1.2.1 Le cryptosystème affine.

Ce cryptosystème s'appuie sur les bijections affines de  $\mathbb{Z}/26\mathbb{Z}$ . Considérons deux entiers  $a$  et  $b$  et  $f$  l'application :

$$\begin{aligned} f : \mathbb{Z}/26\mathbb{Z} &\rightarrow \mathbb{Z}/26\mathbb{Z} \\ x &\mapsto ax + b \end{aligned}$$

Cette application est bijective si et seulement si  $a$  et 26 sont premiers entre eux. Pour déterminer sa réciproque, on inverse  $a$  dans  $\mathbb{Z}/26\mathbb{Z}$  en résolvant Bezout :  $au + 26v = 1$ . L'entier  $u$  est alors l'inverse de  $a$  dans  $\mathbb{Z}/26\mathbb{Z}$  et on a :

$$\begin{aligned} f^{-1} : \mathbb{Z}/26\mathbb{Z} &\rightarrow \mathbb{Z}/26\mathbb{Z} \\ x &\mapsto u(x - b) \end{aligned}$$

Le cryptage consiste à appliquer  $f$  à chacune des lettres du message et le décryptage à appliquer  $f^{-1}$ . Une clé est la donnée d'un couple d'entiers  $(a, b)$  avec  $a \wedge 26 = 1$ .

### 1.2.2 Le chiffrement de Lester Hill.

Il s'agit d'une généralisation du cryptosystème précédent, conçue en 1929 pour résister aux analyses de fréquences. Dans ce cryptosystème, on chiffre les lettres par paquets de longueur  $m$ , pour un entier  $m$  donné. On s'arrange donc pour que notre message comporte un nombre de lettres multiple de  $m$ , quitte à ajouter des lettres, de manière aléatoire, à la fin du message.

Pour le chiffrement, on utilise une matrice  $m \times m$  à coefficients entiers  $M$  ; l'application de cryptage est alors :

$$\varphi : (\mathbb{Z}/26\mathbb{Z})^m \longrightarrow (\mathbb{Z}/26\mathbb{Z})^m \\ \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_m \end{pmatrix} \longmapsto M \times \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ x_m \end{pmatrix}$$

Cette application est bijective si et seulement si le déterminant de  $M$  est premier avec 26. Le décryptage s'obtient alors par calcul de la transposée de la comatrice de  $M$ , que l'on

multiplie par un représentant  $u$  de l'inverse de  $\det(M)$  modulo 26 :  $\det(M)u + 26v = 1$ .

Dans ce cryptosystème, une clé est la donnée d'une matrice carrée  $M$  à coefficients entiers dont le déterminant est premier avec 26. Il va de soit que, pour mieux resister aux attaques par fréquences de motifs, il vaut mieux choisir une matrice de taille importante. J'ai néanmoins voulu illustrer ce protocole de chiffrement dans l'exercice 4 du dossier 88 avec un matrice  $2 \times 2$  :

$$M = \begin{pmatrix} 5 & 11 \\ 8 & 3 \end{pmatrix}$$

Son déterminant, 73, est premier à 26 et la transposée de sa comatrice est :

$$\begin{pmatrix} 3 & -11 \\ -8 & 5 \end{pmatrix}$$

L'inverse de  $M$  modulo 26 est donc donné par la matrice :

$$5 \times \begin{pmatrix} 3 & -11 \\ -8 & 5 \end{pmatrix} = \begin{pmatrix} 15 & -3 \\ 12 & -1 \end{pmatrix}$$

### 1.2.3 Les cryptosystèmes symétriques modernes

Les deux cryptosystèmes symétriques que je vous ai présentés ont un intérêt historique et permettent d'illustrer, à l'aide d'outils élémentaires, la problématique du sercret. Ils n'en demeurent pas moins anecdotiques, et vous trouverez de nombreux autres exemples (s'appuyant rarement sur l'arithmétique) dans l'excellent premier chapitre de [2].

Les deux exemples ci-dessus ne résistent pas à la cryptanalyse et ne remplissent pas les critères de choix des cryptosystèmes modernes. Pour ces derniers, on désire avoir des protocoles publics, érigés en standard ou norme, dont la sécurité repose entièrement sur la clé. Les algorithmes de cryptage et de décryptage doivent être extrêmement rapides, pour assurer une communication quasiment en temps réel. On désire de plus disposer d'une clé différente pour chaque transaction. Le standard actuel, AES pour Advanced Encrypton Standard, utilise des clés binaires de 128 bits.

Dans la mise en oeuvre d'une transaction, on génère aléatoirement un binaire de 128 bits que l'on échange en début de communication. Un fois la clé échangée, la communication est crypté par application de DES. L'échange est alors sûr ... à une condition : que la clé n'ait pas été interceptée en debut de communication. Ceci nous amène naturellement au paragraphe suivant.

## 1.3 Cryptosystèmes asymétriques (à clé publique).

Ces cryptosystèmes reposent sur l'existence de bijections  $f$  à partir desquelles il est difficile (couteux) de déterminer la réciproque : même si je connais  $f$ , je ne suis pas capable de déterminer  $f^{-1}$ . De telles fonctions sont dites être des fonctions trappes ou fonctions à sens unique.

La difficulté est là fondamentalement de nature algorithmique : il s'agit de trouver un

algorithme *efficace* permettant de calculer  $f^{-1}$  connaissant  $f$ . La recherche exhaustive doit évidemment être trop *couteuse* pour être envisagée.

Supposons donc que nous avons deux interlocuteurs, Alphonse et Barbara, qui veulent communiquer en toute sécurité. Chacun d'eux se procure un fonction trappe dont ils est le seul à connaître la réciproque. Notons  $c_A$  la fonction trappe d'Alphonse,  $d_A$  sa réciproque,  $c_B$  et  $d_B$  celles de Barbara. Comme la connaissance de  $c_A$  et  $c_B$  ne permet pas de déterminer leurs réciproques, Alphonse et Barbara n'hésitent pas à en faire publicité. Tout le monde connaît donc  $c_A$  et  $c_B$ . Si maintenant Barbara veut envoyer le message  $m$  à Alphonse, elle lui transmet  $c_A(m)$  ; comme Alphonse est le seul à connaître  $d_A$ , il est le seul à pouvoir retrouver  $m$ . Moralité : comme je connais la fonction de cryptage d'Alphonse, je peux l'utiliser pour crypter les messages que je désire lui envoyer. Il sera le seul à même de les décrypter.

Le protocole que je viens de décrire est un simple envoi de message crypté. Mais les cryptosystèmes asymétriques ont également l'avantage de permettre l'authentification. Supposons qu'Alphonse et Barbara entre en communication. Alphonse veut être sûr qu'il ne discute pas avec une tierce personne se faisant passer pour Barbara. Pour cela, il choisit un message  $m$  à lui envoyer, il lui transmet  $c_B(m)$  en lui demandant de le déchiffrer puis de lui retourner. Si la personne qui dit être Barbara retourne  $c_A(m)$ , c'est qu'elle a réussi à déchiffrer  $c_B(m)$ . Selon toute vraisemblance, elle connaît  $d_B$  et comme seule Barbara a connaissance de  $d_B$ , Alphonse est presque sûr de communiquer avec Barbara.

Il est aussi possible d'envoyer des messages cryptés avec authentification de l'expéditeur. Pour envoyer un message  $m$  authentifié à Alphonse, Barbara lui envoie  $c_A d_B(m)$ . Pour décrypter Alphonse applique  $c_B d_A$  à ce qu'il reçoit. Si ce qu'il obtient est intelligible, Barbara est très certainement à l'origine du message reçu.

Je vous donne ci dessous les deux premiers exemples historiques de cryptosystèmes à clé publique. Le premier n'a pas résisté à la cryptanalyse moderne et est maintenant complètement abandonné. Le second est le cryptosystème à clé publique le plus répandu aujourd'hui. Sa robustesse est cependant de plus en plus remise en question et il devrait, dans les années à venir, laisser place à un nouveau protocole basé sur les courbes elliptiques sur les corps finis.

## 1.4 Exemples de cryptosystèmes à clé publique.

Dans cette partie, je me contente de décrire le procédé de constructions des fonctions à sens unique associées à deux protocoles : la méthode sac à dos et RSA.

### 1.4.1 La méthode sac à dos.

Ce cryptosystème consiste à modifier, à l'aide de reste modulaire, une fonction  $f$  à *double sens*  $f$ , pour fabriquer une fonction à sens unique. La fonction  $f$  dont on part est associée à une liste d'entiers strictement positifs  $(a_1, \dots, a_n)$  que l'on suppose super-croissante :

$$\forall 1 \leq m \leq n-1, \sum_{k=1}^m a_k < a_{m+1}.$$

Il est évidemment facile de fabriquer une liste super-croissante ; j'en donne un exemple dans l'exercice 5 du dossier 88 : (2, 5, 8, 16, 37, 70).

La fonction  $f$  est alors définie sur les binaires de taille  $n$  et à valeur dans  $\mathbb{N}$  par :

$$\begin{aligned} f : \quad \{0, 1\}^n &\longrightarrow \mathbb{N} \\ \lambda = (\lambda_1, \dots, \lambda_n) &\longmapsto \sum_{i=1}^n \lambda_i \times a_i \end{aligned}$$

Alors  $f$  est injective et réalise une bijection sur son image. On définit sa fonction réciproque par récurrence finie comme suit. Soit  $p = \sum_{i=1}^n \lambda_i a_i$  un entier atteint par  $f$ . Si  $p$  est strictement inférieur à tous les  $a_i$ , alors  $p = 0$ . Sinon, on considère  $i$  le plus grand indice tel que  $p \leq a_i$ . Alors, pour tout  $j > i$ ,  $\lambda_j = 0$ , car  $a_j > p$  et  $a_j$  ne peut donc pas intervenir dans la somme partielle calculant  $p$ . De plus, comme  $\sum_{k=1}^{i-1} a_k < a_i \leq p$ , on ne peut pas atteindre  $p$  en n'utilisant que des sommes partielles de  $a_1, \dots, a_{i-1}$ . Par conséquent  $\lambda_i = 1$ . On applique ensuite ce raisonnement à  $p - a_i$  et en moins de  $n$  itérations, on détermine l'antécédent de  $p$  par  $f$ .

Nous avons donc une bijection  $f$ . Mais la donnée de  $f$  est équivalente à celle de  $f^{-1}$ . Nous allons modifier  $f$  pour corriger ce défaut. On choisit pour cela un entier  $N$  strictement supérieur à  $\sum_{i=1}^n a_i$  et un entier  $u$  premier à  $N$ . On considère alors la liste des entiers  $b_i$ , restes de la division de  $ua_i$  par  $N$ . La liste obtenue n'a alors aucune raison d'être super-croissante, ni même croissante. Je propose par exemple  $N = 139$  et  $u = 111$  dans l'exercice 5, et la liste des  $b_i$  obtenue est (83, 138, 54, 108, 76, 125).

Revenons au cas général et considérons l'application :

$$\begin{aligned} \varphi : \quad \{0, 1\}^n &\longrightarrow \mathbb{N} \\ \lambda = (\lambda_1, \dots, \lambda_n) &\longmapsto \sum_{i=1}^n \lambda_i \times b_i \end{aligned}$$

Pour tout binaire  $\lambda$  de longueur  $n$ , on a alors :

$$\varphi(\lambda) \equiv \sum_{i=1}^n \lambda_i \times ua_i = u \times f(\lambda) \pmod{N}$$

Comme  $u$  est premier avec  $N$ , il est inversible modulo  $N$ . Soit  $v$  un représentant de son inverse. Pour tout binaire  $\lambda$  de longueur  $n$ , on a alors :

$$f(\lambda) \equiv v \times \varphi(\lambda) \pmod{N},$$

et comme  $0 \leq f(\lambda) < N$ , c'est l'unique représentant de  $v\varphi(\lambda)$  modulo  $N$ . Nous savons donc déterminer  $f(\lambda)$  connaissant  $\varphi(\lambda)$ .

Notre fonction à sens unique n'est autre que  $\varphi$ . On rend donc la liste  $(b_1, \dots, b_n)$  publique, en gardant soigneusement le secret sur notre clé : la liste  $(a_1, \dots, a_n)$  et les entiers  $N$  et  $u$ .

Notons que la longueur de la liste doit être grande pour que l'étude exhaustive de tous les cas (il y en a  $2^n$ ) ne soit pas raisonnable. Notre exemple de liste à longueur 6 n'est donc pas pertinent dans la pratique.

Nous avons vu que, comme la liste  $(a_1, \dots, a_n)$  est super-croissante, il est facile de déterminer la réciproque de  $f$  sur son image. La construction de notre fonction trappe a donc consisté à transformer un problème d'inversion facile, celui de  $f$ , en problème d'inversion *difficile*, celui de  $\varphi$ , par multiplication modulaire. Toute la sécurité du système repose donc sur le fait que la suite  $(b_1, \dots, b_n)$  n'est pas une suite super-croissante, et sa connaissance ne permet pas, à priori, de déterminer  $\varphi^{-1}$ .

Ce cryptosystème est l'un des deux premiers exemples historiques de cryptosystème à clé public. Il n'a pas résisté aux attaques des cryptanalystes. Celui qui suit, apparu la même année, est aujourd'hui le cryptosystème asymétrique standard.

### 1.4.2 RSA : Rivest Shamir Adleman.

La sécurité de ce cryptosystème est basée sur la difficulté de factoriser un grand nombre. Le principe est le suivant : on choisit deux grands nombres premiers distincts (pas trop proches, mais du même ordre de grandeur) et on construit leur produit  $N = p \times q$ . Alors, tous les entiers  $a$  premiers à  $(p-1) \times (q-1)$  fournissent une fonction à sens unique :

$$\begin{aligned} \varphi_a : \mathbb{Z}/N\mathbb{Z} &\longrightarrow \mathbb{Z}/N\mathbb{Z} \\ x &\longmapsto x^a \end{aligned}$$

L'inverse de cette application est  $\varphi_b$  où  $b$  est un inverse dans  $\mathbb{N}$  de  $a$  modulo  $(p-1)(q-1)$ . En effet, on a alors :  $ab - u(p-1)(q-1) = 1$ , avec  $u \in \mathbb{N}$ , donc :

$$x^{ab} - x = x(x^{u(p-1)(q-1)} - 1)$$

On déduit alors du petit théorème de Fermat que  $x^{ab} - x \equiv 0 \pmod{N}$ . En effet, pour  $x$  premier avec  $p$  et  $q$ ,  $x^{k(p-1)} \equiv 1 \pmod{p}$  et  $x^{l(q-1)} \equiv 1 \pmod{q}$  pour couple d'entiers  $(k, l)$  ; par conséquent  $x^{u(p-1)(q-1)} - 1$  est divisible par  $N = pq$ . Maintenant, si  $x$  est divisible par  $p$  et premier à  $q$ ,  $x^{u(p-1)(q-1)} \equiv 1 \pmod{q}$  et  $x \times (x^{u(p-1)(q-1)} - 1)$  est donc divisible par  $p$  et  $q$ . De même, si  $x$  est divisible par  $q$  et premier à  $p$ ,  $x^{u(p-1)(q-1)} \equiv 1 \pmod{p}$  et  $x^{ab} \equiv x \pmod{N}$ . Par conséquent  $x^{ab} \equiv x \pmod{N}$ , pour tout entier  $x$ .

La fonction  $\varphi_a$  est publique, donc  $a$  et  $N$  le sont. Mais pour déterminer l'inverse  $\varphi_b$  de  $\varphi_a$ , nous avons recherché une identité de Bezout entre  $a$  et  $(p-1)(q-1)$  ; nous avons utilisé pour cela notre connaissance de  $p$  et  $q$  pour calculer  $(p-1)(q-1)$ . Le caractère à sens unique de notre fonction  $\varphi_a$  dépend (uniquement) de la difficulté à retrouver  $(p-1)(q-1)$  connaissant  $N$  seulement (avis aux cryptanalystes en herbe).

## 1.5 Dans la pratique.

Si les cryptosystèmes asymétriques sont d'invention récente, ce ne sont pas véritablement des concurrents des cryptosystèmes symétriques. Ces derniers sont en effet beaucoup plus

rapides (dans le calcul) et beaucoup plus sûr à taille de clé équivalente (les normes actuelles proposent 128 bits pour AES, 1024 bits pour RSA).

Dans la pratique, plus que de s'opposer, les cryptosystèmes symétriques et asymétriques se complètent. Nous avons vu que, pour communiquer sous la protection d'un cryptosystème à clé privé, deux interlocuteurs doivent échanger une clé. Si cet échange se fait en clair, toute personne ayant intercepté la clé sera à même de déchiffrer toute la communication.

Il est donc important de sécuriser l'échange de la clé, et c'est là que les protocoles asymétriques interviennent. L'échange de la clé privée est crypté à l'aide d'un cryptosystème à clé publique : la clé publique permet d'échanger une clé privée. Dans la pratique électronique, ce sont généralement RSA puis AES qui sont mis en oeuvre : on commence par échanger une clé aléatoire de 128 bits sous protocole RSA, puis on sort du protocole RSA, et on crypte la communication en utilisant AES.

## 2 Codes.

### 2.1 Codes : la problématique.

La problématique du codage, complètement différente de celle du cryptage, est celle de la transmission fiable d'informations (ici supposées numériques). Après émission, une telle information subit les aléas des divers canaux de transmission qu'elle emprunte. Elle est ainsi éventuellement (souvent) perturbée (perte ou modification d'un bit par exemple), de telle sorte que le message réceptionné risque d'être erroné (différent du message envoyé). Est-il possible de coder (transformer) le message initial de telle sorte que l'on sache détecter d'éventuelles erreurs de transmission, et dans ce cas, de les corriger (dans une proportion raisonnable) ? Un procédé permettant cela est appelé code détecteur et correcteur d'erreurs.

Il existe des codes correcteurs d'erreur et le langage en est un exemple : imaginez une conversation téléphonique avec de la friture sur la ligne ; en entendant 90% des mots correctement on est en général capable de reconstituer complètement le sens. En effet, dans le langage usuel (pour nous le français), l'information transmise est très largement redondante.

Un *bon* code correcteur d'erreur est un procédé de codage de l'information qui introduit efficacement une telle redondance en restant économique, c'est à dire sans faire exploser la taille du message transmis. Je vous propose ci-dessous l'étude d'un exemple élémentaire de code correcteur d'erreurs.

### 2.2 Un exemple élémentaire.

On code ici un nombre à 10 chiffres  $a_1 \cdots a_{10}$  en ajoutant deux clés : la première est le reste modulo 11 de la somme des dix chiffres ; la seconde est le reste modulo 11 de :  $\sum_{k=1}^{10} ka_k$ . Pour transmettre  $a_1 \cdots a_{10}$ , on envoie donc  $a_1 \cdots a_{10}a_{11}a_{12}$  où :

$$a_{11} \equiv \sum_{k=1}^{10} a_k \pmod{11} \quad \text{et} \quad a_{12} = \sum_{k=1}^{10} ka_k \pmod{11}$$

Nous nous proposons de voir que, si une erreur se produit lors de la transmission, ce code permet de la détecter et de la corriger. Autrement dit que si un chiffre transmis (et un seul) est erroné, on détecte l'erreur et on est capable de la corriger.

Notons pour cela  $b_1 \cdots b_{12}$  le nombre réceptionné et supposons qu'il existe un indice  $i \in \llbracket 1, 12 \rrbracket$  tel que  $b_i \neq a_i$  et  $b_k = a_k$  pour  $k$  différent de  $i$ . Si  $i$  est compris entre 1 et 10, alors :

$$b_{11} - \sum_{k=1}^{10} b_k = a_i - b_i \neq 0 \pmod{11}$$

$$b_{11} - \sum_{k=1}^{10} kb_k = i \times (a_i - b_i) \neq 0 \pmod{11}$$

Donc, si l'erreur est commise dans les dix premiers chiffres, les deux clés ne correspondent pas aux dix premiers chiffres reçus. Si maintenant une erreur a été commise dans la transmission de l'une des clés, un des deux derniers chiffres transmis ne correspond pas aux dix premiers, alors que le second y correspond.

Ainsi, si  $b_{11} \equiv_{(11)} \sum b_k$  ou  $b_{12} \equiv_{(11)} \sum kb_k$ , alors l'erreur a été commise sur la transmission d'une des deux clés, et il est alors facile de la corriger. Si maintenant  $b_{11}$  et  $b_{12}$  ne correspondent pas aux dix premiers chiffres transmis, alors l'erreur a été commise dans ces dix premiers chiffres. Comme  $b_{11}$  et  $b_{12}$  ne sont pas erronés, on connaît  $a_i - b_i$  et  $i(a_i - b_i)$ . Comme 11 est premier,  $a_i - b_i$  est inversible modulo 11 et on détermine ainsi  $i$  et on restitue alors  $a_i = b_i + b_{11} - \sum b_k$ .

Ainsi, si il y moins d'une erreur, notre code est capable de la détecter et de la corriger.

## 2.3 Les codes linéaires

Les codes linéaires sont des codes définis à partir d'une matrice de la façon suivante. On fixe un corps fini  $\mathbb{F} = \mathbb{F}_q$ . Pour simplifier, vous pouvez considérer  $\mathbb{F} = \mathbb{Z}/p\mathbb{Z}$ , où  $p$  est un nombre premier. On se donne  $G$  une matrice  $n \times k$ . On code les message découpés en blocs de  $k$  nombres strictement inférieurs à  $p$ . Un tel bloc est vu comme un vecteur de  $\mathbb{F}^k$  et on le code en lui appliquant la matrice  $G$  :

$$\begin{array}{ccc} \mathbb{F}^k & \longrightarrow & \mathbb{F}^n \\ \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix} & \longmapsto & G \times \begin{pmatrix} a_1 \\ \vdots \\ a_k \end{pmatrix} \\ \text{message} & & \text{message codé} \end{array}$$

Pour que l'on puisse décoder, il vaut évidemment mieux supposer que  $G$  est injective. Un décodage est fourni par une section à gauche de  $G$ , une matrice  $H$  telle que  $HG = Id_{\mathbb{F}^k}$ . Les erreurs sont détectés grace au propriétés de la matrice  $G$ . Puis, pour chaque type d'erreur, il faut déterminer une matrice de décodage qui cout-circuite cette erreur. Ce travail d'écriture de l'algorithme de décodage avec détection et correction d'erreurs est le plus difficile de la théorie.

Le code que nous avons étudié en (2.2.) est un exemple de code numériques où  $\mathbb{F} = \mathbb{Z}/11\mathbb{Z}$  et de matrice associée :

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \end{pmatrix}$$

## 2.4 Les codes qui vous entourent.

Avec l'expansion de l'électronique et du numérique, notre vie quotidienne est envahie de codes, qui sont de plus en plus sophistiqués. La première utilisation marquante a été celle du code de Reed-Müller pour les transmissions de données en provenance des sondes Mariner à la fin des années 1960. On a ainsi pu considérablement améliorer les débits de transmissions.

On utilise aujourd'hui des codes plusieurs fois par jour sans le savoir. Lorsque qu'on écoute un CD, on utilise par exemple les fameux codes de Reed-Solomon. Ce sont des codes non linéaires introduit en 1960. Leur première utilisation a été la transmission d'images prise par la sonde Voyager. On les trouve maintenant dans le stockage de données sur un disque dur, les CD, les transmissions numériques ...

A chaque problème de transmission correspond une problématique de code différente. On n'utilisera pas le même code si le canal de transmission produit des erreurs uniformément, ou si il affecte un grande zone (perte d'un bloc entier). Il y a donc eu ces dernières années une interaction féconde entre les problématiques de transmission et la théorie des codes. Elle a donné jour, par exemple, aux codes convolutifs et aux codes de Goppa (géométrie algébrique) qui utilisent un noyau mathématique très élaboré.

## References

- [1] M., Demazure, *Cours d'algèbre - primalité divisibilité, codes*, Cassini, 1997.
- [2] A., Salomaa, *Public-Key Cryptography*, Texts in Theoretical Computer Science, Springer Verlag, 1996.
- [3] Van Lint, *Introduction to coding theory*, Graduate texts in Mathematics (86), Springer Verlag, 1999.