X. Les codes correcteurs

X.4. Bornes

Définition

Soit $t = \left\lfloor \frac{d-1}{2} \right\rfloor$, $B(0,t) = \{y \in \mathbb{F}_2^n, d_H(0,y) \leqslant t\}$

$$V_t = \sum_{i=0}^{t} \binom{n}{i} = \mathrm{card} B(0,t)$$

Par définition de t, si $x, y \in C$, on a $B(x,t) \cap B(y,t) = \emptyset$

$$\Rightarrow \coprod_{x \in C} B(x,t) \subset \mathbb{F}_2^n$$

Théorème (Borne de Hamming)

Si C est de type (n, k, d) alors

$$V_t 2^k \leqslant 2^n$$

Cette borne décrit le taux de remplissage de $\mathbb{F}_2^n$ par des boules centrées en les points de C. Quand on a égalité, tous les vecteurs de $\mathbb{F}_2^n$ sont dans des boules !

Définition (Codes parfaits)

Les codes qui vérifient cette borne sont appelés **codes parfaits**.

Si $t = 1$, on trouve $V_t = 1 + n$ donc $n = 2^{n-k} - 1$

Théorème

Borne de Singleton Pour tout code C de type (n, k, d) on a

$$d + k \leqslant n + 1$$

Cette borne signifie que, à dimension du code fixé, on ne peut pas avoir une capacité de correction très élevée tout en conservant une petite longueur.

X. Les codes correcteurs

X.5. Complexité

La table des syndromes est de tailles

$$V_t = \sum_{i=0}^{t} \binom{n}{i}$$

Dans le cas de code de petite longueur, la table des syndrome est petite et la complexité n'a pas vraiment de sens car toutes les données sont de petite taille et il est possible de corriger très vite un code.

À l'opposé, lorsque la longueur du code augmente, on peut montrer que le problème devient NP-complet pour un code "générique"[1]

[1] très peu probable qu'il existe un algorithme permettant de trouver les solutions en temps polynomial en fonction des données du problème.

XI. Retour sur la cryptographie

Il est possible de construire des méthodes d'encryptage en utilisant des codes correcteurs, c'est ce que nous allons voir avec l'algorithme de McEliece, proposé par Robert McEliece en 1978.

XI. Retour sur la cryptographie

XI.1. Création des clefs

Tout d'abord, Bob fixe des entiers k, n et t représentant la dimension, la longueur et la capacité de correction, et choisit une matrice G engendrant un code ayant ces caractéristiques. Nous supposerons de plus qu'il existe un algorithme efficace (en temps polynomial) pour décoder le code engendré par G (de tels algorithmes existent pour les codes de Golay, qui sont des exemples spécifiques de codes linéaires).

Finalement, choisissons deux matrices inversibles, S de taille $k \times k$, et P de taille $n \times n$, supposant de plus que P est une matrice de permutation (exactement un 1 sur chaque ligne).

Considérons finalement la matrice $\hat{G} = PGS$. La clef publique est alors $(\hat{G}, t)$ et la clef privée est (S, G, P).

XI. Retour sur la cryptographie

XI.2. Encryptage et décryptage

Pour envoyer un message à Bob, Alice décide d'abord de l'encoder sous forme d'un vecteur m de longueur k (éventuellement après un salage). Elle choisir ensuite un vecteur aléatoire z de longueur n possédant au plus t entrées à 1. Elle calcule finalement le vecteur $c = \hat{G}m + z$ et l'envoi à Bob.

Recevant le message c, Bob calcule $\hat{c} = P^{-1}c$. Comme P est une matrice de permutation, c'est aussi le cas de P^{-1}, et donc $P^{-1}z$ possède au plus t entrées à 1. en particulier, $\hat{c}$ est obtenu d'un mot du code engendré par G en effectuant au plus t erreurs, et peut donc être corrigé *rapidement* grâce à l'algorithme efficace dont Bob dispose, et il obtient un code corrigé $\hat{M}$, qu'il décode à l'aide d'un inverse à gauche de G en m. Il peut alors calculer $S^{-1}\hat{m}$ qui est le message envoyé par Alice.

XI. Retour sur la cryptographie

XI.3. Sécurité

Dans les faits, il est bien sûr possible de corrigé directement le code $\hat{c}$ à l'aide d'une table de Syndrôme, puis une attaque statistique pour calculer l'inverse de S (facile d'obtenir des données car on connaît $\hat{G}$). Le problème est que résoudre à l'aide de la table des Syndrôme est très gourmand en temps (table de longueur au moins $\binom{t}{n}$), c'est même un problème NP-complet ! L'attaque statistique sur S demandant en plus un grand nombre (dépendant de k) de données, le cassage de l'algorithme de McElies est donc très long !

Dans les faits, afin que ce cryptosystème soit sûr, il semble nécessaire de prendre des clefs très grosses (par exemple, $n = 1024$, $t = 38$, et $k \geqslant 644$).

En raison de la taille des clefs, cet algorithmes a pendant longtemps été laissé de côté. Toutefois, il pourrait revenir sur le devant de la scène car, étant NP-complet, il ne semble pas attaquable par un ordinateur quantique[2].

[2] assertion non prouvée à l'heure actuelle ! mais la plupart des scientifiques pensent que c'est vrai !