Cryptographie à clé publique
Cours 8

Julien Lavauzelle

Université Paris 8

Master 1 mathématiques et applications – parcours ACC

14/04/2023

Programme

On a vu **la semaine dernière** :

- notions sur le modèle de calcul quantique
- besoin d'une cryptographique post-quantique
- notions sur les réseaux euclidiens
- chiffrement NTRU
- problème LWE et chiffrement de Regev

Programme

On a vu **la semaine dernière** :

- notions sur le modèle de calcul quantique
- besoin d'une cryptographique post-quantique
- notions sur les réseaux euclidiens
- chiffrement NTRU
- problème LWE et chiffrement de Regev

Cette semaine : **cryptographie à base de codes** :

- décodage générique
- cryptosystème de McEliece
- protocole d'identification de Stern
- signatures à base de codes

Plan

Rappels de codes correcteurs

Notations.

- Un code correcteur linéaire $\mathcal{C}$ est un sous-espace vectoriel de $\mathbb{F}_q^n$, où $n \geq 1$.
- On note k sa dimension, et
$$d = \min\{|\boldsymbol{c}|,\ \boldsymbol{c} \in \mathcal{C} \setminus \{\boldsymbol{0}\}\}$$
où $|\boldsymbol{x}|$ représente le poids de Hamming de $\boldsymbol{x}$.

Matrices génératrices et de parité.

- Une **matrice génératrice** d'un code $\mathcal{C}$ est une matrice $\boldsymbol{G} \in \mathbb{F}_q^{k \times n}$ telle que
$$\mathcal{C} = \{\boldsymbol{mG}, \boldsymbol{m} \in \mathbb{F}_q^k\}$$
- Une **matrice de parité** d'un code $\mathcal{C}$ est une matrice $\boldsymbol{H} \in \mathbb{F}_q^{(n-k) \times n}$ telle que
$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{F}_q^n \mid \boldsymbol{H}\boldsymbol{x}^\top = \boldsymbol{0}\}$$

À propos du décodage

Bornes et décodage. La borne de Singleton assure que $d \leq n - k + 1$, et il est **théoriquement** possible de décoder jusque $\lfloor \frac{d-1}{2} \rfloor$ erreurs de manière unique.

À propos du décodage

Bornes et décodage. La borne de Singleton assure que $d \leq n - k + 1$, et il est **théoriquement** possible de décoder jusque $\lfloor \frac{d-1}{2} \rfloor$ erreurs de manière unique.

$\rightarrow$ Certains codes sont munis d'**algorithmes de décodage efficaces**, en terme de complexité (polynomiale en n) et en capacité de correction (proche de la limite théorique).

Exemples : codes de Hamming, codes de Goppa, codes BCH, codes de Reed–Solomon.

À propos du décodage

Bornes et décodage. La borne de Singleton assure que $d \leq n - k + 1$, et il est **théoriquement** possible de décoder jusque $\lfloor \frac{d-1}{2} \rfloor$ erreurs de manière unique.

$\rightarrow$ Certains codes sont munis d'**algorithmes de décodage efficaces**, en terme de complexité (polynomiale en n) et en capacité de correction (proche de la limite théorique).

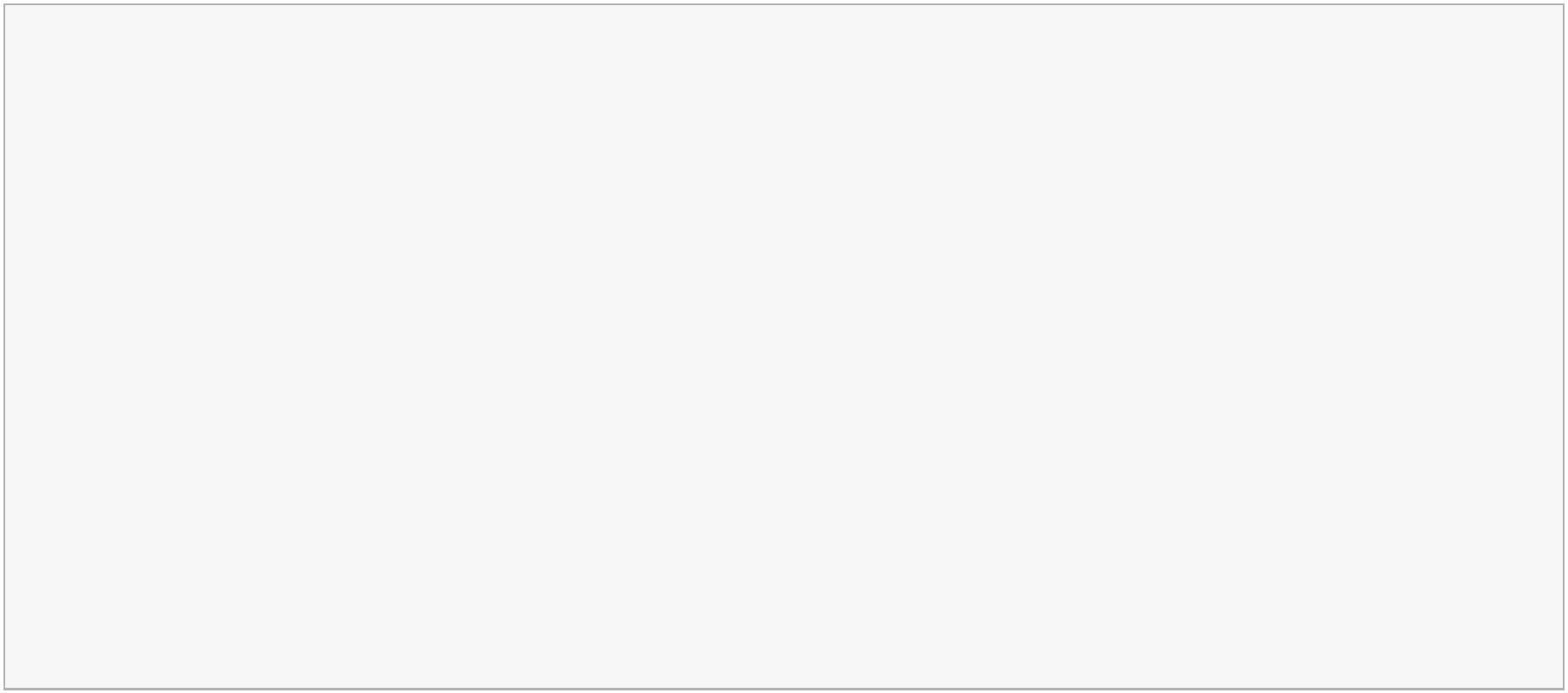
Exemples : codes de Hamming, codes de Goppa, codes BCH, codes de Reed–Solomon.

$\rightarrow$ Pour fonctionner, ces algorithmes ont besoin de **davantage de données** que simplement la structure vectorielle du code.

Exemples : les points d'évaluations pour le code de Reed–Solomon, l'ensemble de définition pour le code BCH.

Le système de McEliece : idée générale

McEliece propose en 1978 (un an après RSA) le système suivant, ici décrit informellement



Le système de McEliece : idée générale

McEliece propose en 1978 (un an après RSA) le système suivant, ici décrit informellement

1. **Génération de clefs**. Alice choisit aléatoirement un code correcteur $\mathcal{C} \subseteq \mathbb{F}_q^n$ de dimension k, qui corrige jusqu'à t erreurs.
   - clé publique : Alice publie t, ainsi qu'un procédé d'encodage aléatoire E du code $\mathcal{C}$
   - clé secrète : Alice garde secrètement un algorithme de décodage pour le code $\mathcal{C}$.

Le système de McEliece : idée générale

McEliece propose en 1978 (un an après RSA) le système suivant, ici décrit informellement

1. **Génération de clefs**. Alice choisit aléatoirement un code correcteur $\mathcal{C} \subseteq \mathbb{F}_q^n$ de dimension k, qui corrige jusqu'à t erreurs.
   - clé publique : Alice publie t, ainsi qu'un procédé d'encodage aléatoire E du code $\mathcal{C}$
   - clé secrète : Alice garde secrètement un algorithme de décodage pour le code $\mathcal{C}$.
2. **Chiffrement.** Bob souhaite envoyer un message $\boldsymbol{m} \in \mathbb{F}_q^k$.
   - Il choisit une erreur $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids t, et encode son message $\boldsymbol{m}$ en un mot de code $\boldsymbol{c} = E(\boldsymbol{m}) \in \mathcal{C}$.
   - Le chiffré est $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.

Le système de McEliece : idée générale

McEliece propose en 1978 (un an après RSA) le système suivant, ici décrit informellement

1. **Génération de clefs**. Alice choisit aléatoirement un code correcteur $\mathcal{C} \subseteq \mathbb{F}_q^n$ de dimension k, qui corrige jusqu'à t erreurs.
   - clé publique : Alice publie t, ainsi qu'un procédé d'encodage aléatoire E du code $\mathcal{C}$
   - clé secrète : Alice garde secrètement un algorithme de décodage pour le code $\mathcal{C}$.
2. **Chiffrement.** Bob souhaite envoyer un message $\boldsymbol{m} \in \mathbb{F}_q^k$.
   - Il choisit une erreur $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids t, et encode son message $\boldsymbol{m}$ en un mot de code $\boldsymbol{c} = E(\boldsymbol{m}) \in \mathcal{C}$.
   - Le chiffré est $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.
3. **Déchiffrement.** Alice souhaite retrouver le message $\boldsymbol{m}$ à partir du chiffré $\boldsymbol{y}$.
   - Alice corrige les erreurs dans $\boldsymbol{y}$ grâce à son algorithme de décodage D, et obtient $\boldsymbol{c} \in \mathcal{C}$
   - Alice retrouve le message $\boldsymbol{m}$ associé à $\boldsymbol{c}$.

Le système de McEliece : idée générale

McEliece propose en 1978 (un an après RSA) le système suivant, ici décrit informellement

1. **Génération de clefs**. Alice choisit aléatoirement un code correcteur $\mathcal{C} \subseteq \mathbb{F}_q^n$ de dimension k, qui corrige jusqu'à t erreurs.
   - clé publique : Alice publie t, ainsi qu'un procédé d'encodage aléatoire E du code $\mathcal{C}$
   - clé secrète : Alice garde secrètement un algorithme de décodage pour le code $\mathcal{C}$.
2. **Chiffrement.** Bob souhaite envoyer un message $\boldsymbol{m} \in \mathbb{F}_q^k$.
   - Il choisit une erreur $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids t, et encode son message $\boldsymbol{m}$ en un mot de code $\boldsymbol{c} = E(\boldsymbol{m}) \in \mathcal{C}$.
   - Le chiffré est $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.
3. **Déchiffrement.** Alice souhaite retrouver le message $\boldsymbol{m}$ à partir du chiffré $\boldsymbol{y}$.
   - Alice corrige les erreurs dans $\boldsymbol{y}$ grâce à son algorithme de décodage D, et obtient $\boldsymbol{c} \in \mathcal{C}$
   - Alice retrouve le message $\boldsymbol{m}$ associé à $\boldsymbol{c}$.

La sécurité du système repose sur l'**incapacité** à décoder dans $\mathcal{C}$ à partir (uniquement) du procédé d'encodage E.

Problématiques

Plusieurs questions sur le **système de McEliece** :

1. quels codes choisir ? quel procédé d'encodage ?
2. quelle est la sécurité du système ?
3. quelle performance (taille de clés, complexité) ?

Codes de Goppa binaires

Historiquement, McEliece a proposé de choisir les codes parmi la famille des **codes de Goppa binaires**.

Codes de Goppa binaires

Historiquement, McEliece a proposé de choisir les codes parmi la famille des **codes de Goppa binaires**.

Définition. Soit $q = 2^m$ et $g(x) \in \mathbb{F}_q[X]$ de degré t sans racine multiple. Soit $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ distincts, tels que $g(a_i) \neq 0$ pour tout $i \in [1, n]$. Alors, le **code de Goppa binaire** est

$$\Gamma(g, \boldsymbol{a}) := \left\{ \boldsymbol{c} \in \mathbb{F}_2^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{g(a_i)} \frac{g(x) - g(a_i)}{x - a_i} \equiv 0 \mod g(x) \right\} \subseteq \mathbb{F}_2^n .$$

Codes de Goppa binaires

Historiquement, McEliece a proposé de choisir les codes parmi la famille des **codes de Goppa binaires**.

Définition. Soit $q = 2^m$ et $g(x) \in \mathbb{F}_q[X]$ de degré t sans racine multiple. Soit $\boldsymbol{a} = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$ distincts, tels que $g(a_i) \neq 0$ pour tout $i \in [1, n]$. Alors, le **code de Goppa binaire** est

$$\Gamma(g, \boldsymbol{a}) := \left\{ \boldsymbol{c} \in \mathbb{F}_2^n \;\middle|\; \sum_{i=1}^{n} \frac{c_i}{g(a_i)} \frac{g(x) - g(a_i)}{x - a_i} \equiv 0 \mod g(x) \right\} \subseteq \mathbb{F}_2^n .$$

Propriétés.

1. Si $g(x)$ est irréductible de degré t, alors $\Gamma(g, \boldsymbol{a})$ a distance minimale $2t + 1$ et dimension $k = n - tm$.
2. Les codes de Goppa sont des codes alternants, autrement dit des sous-codes sur un sous-corps de certains codes de Reed-Solomon.
3. Il existe des algorithmes de décodage efficaces pour décoder $\Gamma(g, \boldsymbol{a})$ jusqu'au rayon de décodage unique $t = \lfloor \frac{d-1}{2} \rfloor$
   $\rightarrow$ Exemple : algorithme de Patterson

Génération de clefs

Fixons n, m et t, et $q = 2^m$. On considère $\mathcal{F}$ la famille des codes de Goppa binaires

$$\mathcal{F} = \left\{ \Gamma(g, \boldsymbol{a}) \mid g \in \mathbb{F}_q[X] \text{ irréductible et } \boldsymbol{a} \in \mathbb{F}_q^n \text{ distincts} \right\}$$

Tous ces paramètres sont publics. On note également $k = n - tm$.

Génération de clefs

Fixons n, m et t, et $q = 2^m$. On considère $\mathcal{F}$ la famille des codes de Goppa binaires

$$\mathcal{F} = \left\{ \Gamma(g, \boldsymbol{a}) \mid g \in \mathbb{F}_q[X] \text{ irréductible et } \boldsymbol{a} \in \mathbb{F}_q^n \text{ distincts} \right\}$$

Tous ces paramètres sont publics. On note également $k = n - tm$.

McEliece : génération de clefs

1. Alice choisit aléatoirement $\mathcal{C} = \Gamma(g, \boldsymbol{a}) \in \mathcal{F}$
2. Alice calcule une matrice génératrice (aléatoire) $\boldsymbol{G} \in \mathbb{F}_2^{k \times n}$ de $\mathcal{C}$
3. Les clefs sont :
   - clé privée : g et $\boldsymbol{a}$
   - clé publique : $\boldsymbol{G}$.

Génération de clefs

Fixons n, m et t, et $q = 2^m$. On considère $\mathcal{F}$ la famille des codes de Goppa binaires

$$\mathcal{F} = \left\{ \Gamma(g, \boldsymbol{a}) \mid g \in \mathbb{F}_q[X] \text{ irréductible et } \boldsymbol{a} \in \mathbb{F}_q^n \text{ distincts} \right\}$$

Tous ces paramètres sont publics. On note également $k = n - tm$.

McEliece : génération de clefs

1. Alice choisit aléatoirement $\mathcal{C} = \Gamma(g, \boldsymbol{a}) \in \mathcal{F}$
2. Alice calcule une matrice génératrice (aléatoire) $\boldsymbol{G} \in \mathbb{F}_2^{k \times n}$ de $\mathcal{C}$
3. Les clefs sont :
   - clé privée : g et $\boldsymbol{a}$
   - clé publique : $\boldsymbol{G}$.

Taille des clefs.

- clé privée : $(t + n)m = O(n \log n)$ bits,
- clé publique : naivement kn. On peut réduire à $k(n - k)$ en publiant une forme systématique. Dans tous les cas, $O(n^2)$ bits.

Génération de clefs

Fixons n, m et t, et $q = 2^m$. On considère $\mathcal{F}$ la famille des codes de Goppa binaires

$$\mathcal{F} = \left\{ \Gamma(g, \boldsymbol{a}) \mid g \in \mathbb{F}_q[X] \text{ irréductible et } \boldsymbol{a} \in \mathbb{F}_q^n \text{ distincts} \right\}$$

Tous ces paramètres sont publics. On note également $k = n - tm$.

McEliece : génération de clefs

1. Alice choisit aléatoirement $\mathcal{C} = \Gamma(g, \boldsymbol{a}) \in \mathcal{F}$
2. Alice calcule une matrice génératrice (aléatoire) $\boldsymbol{G} \in \mathbb{F}_2^{k \times n}$ de $\mathcal{C}$
3. Les clefs sont :
   - clé privée : g et $\boldsymbol{a}$
   - clé publique : $\boldsymbol{G}$.

Taille des clefs.

- clé privée : $(t + n)m = O(n \log n)$ bits,
- clé publique : naivement kn. On peut réduire à $k(n - k)$ en publiant une forme systématique. Dans tous les cas, $O(n^2)$ bits.

Complexité. Choix de g en $O^{\sim}(n)$. Calcul de $\boldsymbol{G}$ en $O^{\sim}(n^2)$.

Chiffrement

Bob souhaite chiffrer un message $\boldsymbol{m} \in \mathbb{F}_2^k$.

McEliece : chiffrement

1. Bob choisit aléatoirement un mot $\boldsymbol{e} \in \mathbb{F}_2^n$ de poids t
2. Bob calcule $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$
3. le chiffré est défini par $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$

Chiffrement

Bob souhaite chiffrer un message $\boldsymbol{m} \in \mathbb{F}_2^k$.

McEliece : chiffrement

1. Bob choisit aléatoirement un mot $\boldsymbol{e} \in \mathbb{F}_2^n$ de poids t
2. Bob calcule $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$
3. le chiffré est défini par $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$

Taille des messages.

- clair : k bits
- chiffré : n bits

Chiffrement

Bob souhaite chiffrer un message $\boldsymbol{m} \in \mathbb{F}_2^k$.

McEliece : chiffrement

1. Bob choisit aléatoirement un mot $\boldsymbol{e} \in \mathbb{F}_2^n$ de poids t
2. Bob calcule $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$
3. le chiffré est défini par $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$

Taille des messages.

- clair : k bits
- chiffré : n bits

Complexité. Calcul de $\boldsymbol{e}$ en $O(n)$. Calcul de $\boldsymbol{c}$, puis de $\boldsymbol{y}$, en $O(n^2)$.

Déchiffrement

Alice souhaite déchiffrer $\boldsymbol{y} \in \mathbb{F}_2^k$.

McEliece : déchiffrement

1. Alice utilise son algorithme de décodage (qui dépend de g et $\boldsymbol{a}$) pour décoder $\boldsymbol{y}$.
2. Alice obtient le message $\boldsymbol{m}$ en sortie de l'algorithme de décodage

Complexité. Décodage en $O(n^2)$ (Patterson).

Plan

Deux types d'attaques

Attaques « sur la clé ». Étant donné la clé publique G, on essaie de retrouver la structure de code (de Goppa), ce qui permettrait de décoder.

- attaque indépendante des messages à chiffrer
- pour les codes de Goppa, il existe des résultats confortant l'idée qu'un code de Goppa aléatoire est semblable à un code aléatoire sans structure
- mais, pas de preuve formelle

Deux types d'attaques

Attaques « sur la clé ». Étant donné la clé publique G, on essaie de retrouver la structure de code (de Goppa), ce qui permettrait de décoder.

- attaque indépendante des messages à chiffrer
- pour les codes de Goppa, il existe des résultats confortant l'idée qu'un code de Goppa aléatoire est semblable à un code aléatoire sans structure
- mais, pas de preuve formelle

Attaques « sur le message ». Étant donné un chiffré $y = mG + e$, on essaie de retrouver m.

- dans ce type d'attaque, on suppose généralement que le code n'admet aucune structure connue
- on étudie donc des méthodes de **décodage générique** pour des codes aléatoires.

Plan

Le décodage générique

On suppose maintenant que le code $\mathcal{C}$ n'a aucune structure évidente (autre que vectorielle).

Question. Quelle est la difficulté de décoder un code aléatoire ?

Le décodage générique

On suppose maintenant que le code $\mathcal{C}$ n'a aucune structure évidente (autre que vectorielle).

Question. Quelle est la difficulté de décoder un code aléatoire ?

Remarque. Si $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, alors :

$$\boldsymbol{mG} + \boldsymbol{e} = \boldsymbol{y} \iff \boldsymbol{eH}^\top = \boldsymbol{yH}^\top .$$

Dans la suite, on note $\boldsymbol{s} = \boldsymbol{yH}^\top \in \mathbb{F}_q^{n-k}$ le **syndrome** de $\boldsymbol{y}$.

Le décodage générique

On suppose maintenant que le code $\mathcal{C}$ n'a aucune structure évidente (autre que vectorielle).

Question. Quelle est la difficulté de décoder un code aléatoire ?

Remarque. Si $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, alors :

$$\boldsymbol{mG} + \boldsymbol{e} = \boldsymbol{y} \iff \boldsymbol{eH}^\top = \boldsymbol{yH}^\top .$$

Dans la suite, on note $\boldsymbol{s} = \boldsymbol{yH}^\top \in \mathbb{F}_q^{n-k}$ le **syndrome** de $\boldsymbol{y}$.

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{eH}^\top = \boldsymbol{s}$.

Le décodage générique

On suppose maintenant que le code $\mathcal{C}$ n'a aucune structure évidente (autre que vectorielle).

Question. Quelle est la difficulté de décoder un code aléatoire ?

Remarque. Si $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, alors :

$$\boldsymbol{m}\boldsymbol{G} + \boldsymbol{e} = \boldsymbol{y} \iff \boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{y}\boldsymbol{H}^\top .$$

Dans la suite, on note $\boldsymbol{s} = \boldsymbol{y}\boldsymbol{H}^\top \in \mathbb{F}_q^{n-k}$ le **syndrome** de $\boldsymbol{y}$.

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$.

Théorème (Berlekamp, McEliece et van Tilborg, 1978). SD est un problème NP-difficile.

On the inherent intractability of certain coding problems. Berlekamp, McEliece et van Tilborg. IEEE. **1978**.

Réduction polynomiale vers le problème du mariage 3-dimensionnel (3*D-matching*).

Le décodage générique

On suppose maintenant que le code $\mathcal{C}$ n'a aucune structure évidente (autre que vectorielle).

Question. Quelle est la difficulté de décoder un code aléatoire ?

Remarque. Si $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, alors :

$$\boldsymbol{mG} + \boldsymbol{e} = \boldsymbol{y} \iff \boldsymbol{eH}^\top = \boldsymbol{yH}^\top .$$

Dans la suite, on note $\boldsymbol{s} = \boldsymbol{yH}^\top \in \mathbb{F}_q^{n-k}$ le **syndrome** de $\boldsymbol{y}$.

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{eH}^\top = \boldsymbol{s}$.

Théorème (Berlekamp, McEliece et van Tilborg, 1978). SD est un problème NP-difficile.

On the inherent intractability of certain coding problems. Berlekamp, McEliece et van Tilborg. IEEE. **1978**.

Réduction polynomiale vers le problème du mariage 3-dimensionnel (3*D-matching*).

Conséquence. Il existe des instances difficiles. Que dire de l'instance **moyenne** ?

Le décodage générique

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$.

Le décodage générique

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$.

Remarque. Sans la contrainte de poids, le problème est facile (algèbre linaire).

Le décodage générique

Problème du décodage par syndrome (*syndrome decoding*, SD).

- **Paramètres :** n, k, q, w.
- **Instance :** $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$.
- **Chercher :** $\boldsymbol{e} \in \mathbb{F}_q^n$ de poids w tel que $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$.

Remarque. Sans la contrainte de poids, le problème est facile (algèbre linaire).

Questions.

1. **Existe** t-il toujours au moins une solution ?
2. Si oui, **combien** (en moyenne) ?

Entropie binaire

Définition. La fonction d'**entropie binaire** est

Entropie binaire

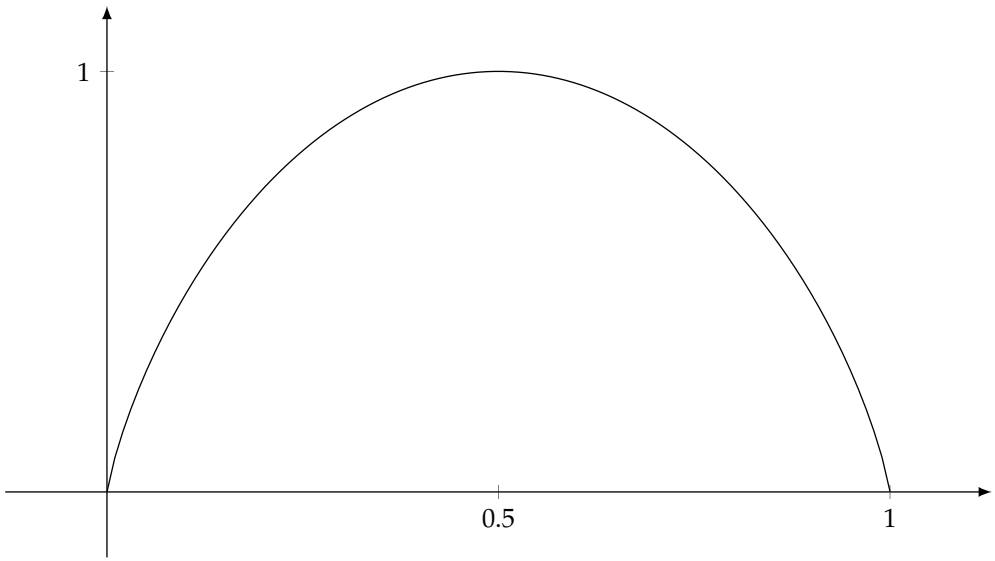
Définition. La fonction d'**entropie binaire** est

$$h(x) = -(1-x)\log_2(1-x) - x\log_2(x)$$

Entropie binaire

Définition. La fonction d'**entropie binaire** est

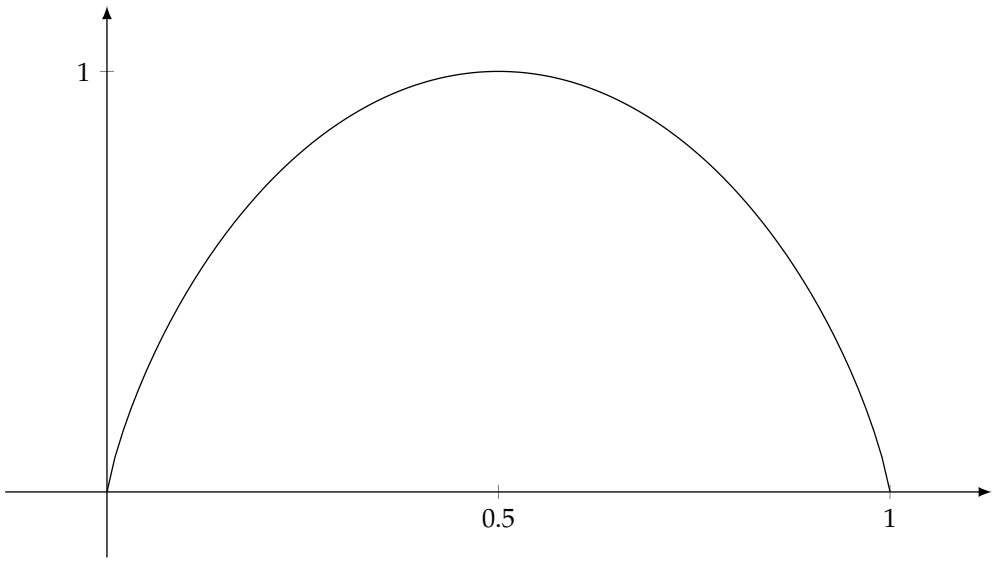
$$h(x) = -(1-x)\log_2(1-x) - x\log_2(x)$$



Entropie binaire

Définition. La fonction d'**entropie binaire** est

$$h(x) = -(1-x)\log_2(1-x) - x\log_2(x)$$



Remarque. Très utile en théorie de l'information.

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Donc,

$$\binom{n}{w} = \frac{n!}{w!(n-w)!}$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Donc,

$$\binom{n}{w} = \frac{n!}{w!(n-w)!} \sim \sqrt{\frac{n}{2\pi\, w(n-w)}}\, 2^{n(\log n - \log e) - w(\log w - \log e) - (n-w)(\log(n-w) - \log e)}$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Donc,

$$\begin{aligned}\binom{n}{w} = \frac{n!}{w!(n-w)!} &\sim \sqrt{\frac{n}{2\pi\, w(n-w)}}\, 2^{n(\log n-\log e)-w(\log w-\log e)-(n-w)(\log(n-w)-\log e)} \\ &\sim \mathrm{poly}(n,w)\cdot 2^{n(\log n-\omega(\log\omega+\log n)-(1-\omega)(\log(1-\omega)+\log n))}\end{aligned}$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Donc,

$$\begin{aligned}\binom{n}{w} = \frac{n!}{w!(n-w)!} &\sim \sqrt{\frac{n}{2\pi\, w(n-w)}}\, 2^{n(\log n-\log e)-w(\log w-\log e)-(n-w)(\log(n-w)-\log e)} \\ &\sim \mathrm{poly}(n,w)\cdot 2^{n(\log n-\omega(\log\omega+\log n)-(1-\omega)(\log(1-\omega)+\log n))} \\ &\sim \mathrm{poly}(n,w)\cdot 2^{nh(\omega)}\end{aligned}$$

Asymptotique du coefficient binomial

Lemme. Si $\omega := \frac{w}{n}$ et $n \to \infty$, alors on a

$$\binom{n}{w} = 2^{n(h(\omega)+o(1))}.$$

Preuve. La formule de Stirling assure que

$$n! \sim \sqrt{2\pi n}\, 2^{n(\log n - \log e)}$$

Donc,

$$\begin{aligned}
\binom{n}{w} = \frac{n!}{w!(n-w)!} &\sim \sqrt{\frac{n}{2\pi\, w(n-w)}}\, 2^{n(\log n-\log e)-w(\log w-\log e)-(n-w)(\log(n-w)-\log e)} \\
&\sim \mathrm{poly}(n,w)\cdot 2^{n(\log n-\omega(\log\omega+\log n)-(1-\omega)(\log(1-\omega)+\log n))} \\
&\sim \mathrm{poly}(n,w)\cdot 2^{nh(\omega)} \\
&= 2^{n(h(\omega)+o(1))}
\end{aligned}$$

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} =$$

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

où $R = \frac{k}{n}$ est le taux d'information du code.

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

où $R = \frac{k}{n}$ est le taux d'information du code.

Conséquence. Pour une instance aléatoire de SD :

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

où $R = \frac{k}{n}$ est le taux d'information du code.

Conséquence. Pour une instance aléatoire de SD :

1. Si $h(\omega) < 1 - R$, le nombre de solutions tend exponentiellement vers 0,

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

où $R = \frac{k}{n}$ est le taux d'information du code.

Conséquence. Pour une instance aléatoire de SD :

1. Si $h(\omega) < 1 - R$, le nombre de solutions tend exponentiellement vers 0,
2. si $h(\omega) > 1 - R$, le nombre de solutions est exponentiel,

Nombre de solutions attendues

Proposition. Fixons n, k, q et w. Alors, pour $\boldsymbol{H}$ et $\boldsymbol{s}$ tirés uniformément, l'espérance du nombre de solutions $\boldsymbol{e}$ du problème SD est

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} .$$

Preuve. exercice.

Prenons maintenant $q = 2$. Asymptotiquement en n, si $\omega = w/n$, alors

$$\binom{n}{w} \frac{(q-1)^w}{q^{n-k}} = 2^{nh(\omega)-n+k+o(n)} = 2^{n(h(\omega)-(1-R)+o(1))}$$

où $R = \frac{k}{n}$ est le taux d'information du code.

Conséquence. Pour une instance aléatoire de SD :

1. Si $h(\omega) < 1 - R$, le nombre de solutions tend exponentiellement vers 0,
2. si $h(\omega) > 1 - R$, le nombre de solutions est exponentiel,
3. si $h(\omega) = 1 - R$, le nombre de solutions est polynomial en n.

Le décodage générique

Définition. On définit le **poids de Gilbert-Varshamov** w_{GV} comme

$$w_{\mathrm{GV}} := \omega_{\mathrm{GV}}\, n \quad \text{où} \;\; \omega_{\mathrm{GV}} \in [0, \tfrac{1}{2}] \;\; \text{satisfait} \;\; h(\omega_{\mathrm{GV}}) = 1 - R\,.$$

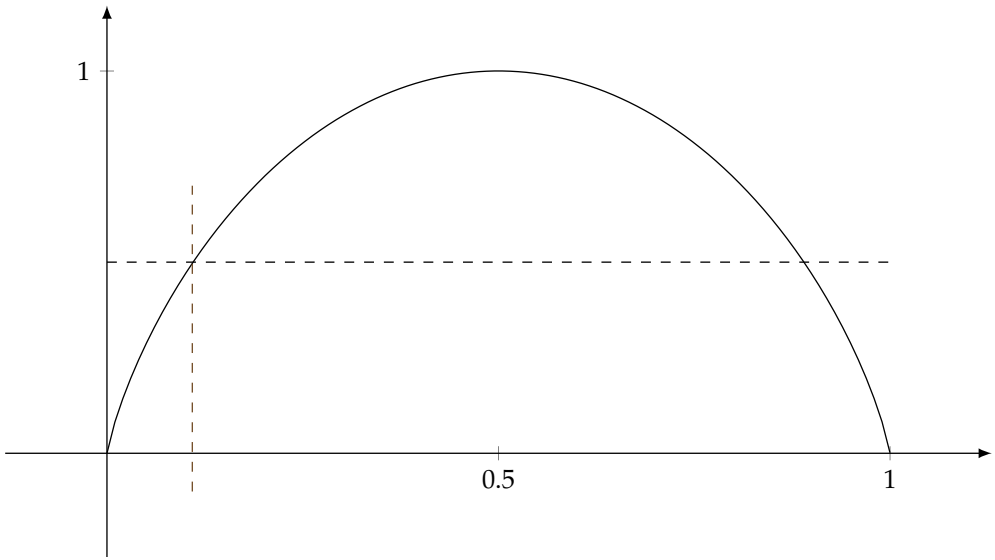
Ce poids dépend du taux d'information $R = \frac{k}{n}$ du code.

Le décodage générique

Définition. On définit le **poids de Gilbert-Varshamov** w_{GV} comme

$$w_{\mathrm{GV}} := \omega_{\mathrm{GV}}\, n \quad \text{où} \;\; \omega_{\mathrm{GV}} \in [0, \tfrac{1}{2}] \;\; \text{satisfait} \;\; h(\omega_{\mathrm{GV}}) = 1 - R\,.$$

Ce poids dépend du taux d'information $R = \frac{k}{n}$ du code.



Sur l'exemple, $R = \frac{1}{2}$ donne $\omega_{\mathrm{GV}} \simeq 0{,}11$.

Plan

Ensembles d'information

Rappels. Pour $I = \{i_1, \ldots, i_k\} \subset \{1, \ldots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I \coloneqq (x_{i_1}, \ldots, x_{i_k}) \in \mathbb{F}_q^k \,.$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Ensembles d'information

Rappels. Pour $I = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I \coloneqq (x_{i_1}, \dots, x_{i_k}) \in \mathbb{F}_q^k \,.$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Le **poinçonnement** de $\mathcal{C} \subseteq \mathbb{F}_q^n$ **en dehors de** I (c'est-à-dire sa restriction à I) est

$$\mathcal{C}_I \coloneqq \{\boldsymbol{c}_I, \boldsymbol{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^k \,.$$

Ensembles d'information

Rappels. Pour $I = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I := (x_{i_1}, \dots, x_{i_k}) \in \mathbb{F}_q^k .$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Le **poinçonnement** de $\mathcal{C} \subseteq \mathbb{F}_q^n$ **en dehors de** I (c'est-à-dire sa restriction à I) est

$$\mathcal{C}_I := \{\boldsymbol{c}_I, \boldsymbol{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^k .$$

Définition. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de dimension k. Un ensemble $I \subseteq \{1, \dots, n\}$ de cardinal k est un **ensemble d'information pour** $\mathcal{C}$ si et seulement si $\mathcal{C}_I = \mathbb{F}_q^k$.

Ensembles d'information

Rappels. Pour $I = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I := (x_{i_1}, \dots, x_{i_k}) \in \mathbb{F}_q^k .$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Le **poinçonnement** de $\mathcal{C} \subseteq \mathbb{F}_q^n$ **en dehors de** I (c'est-à-dire sa restriction à I) est

$$\mathcal{C}_I := \{\boldsymbol{c}_I, \boldsymbol{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^k .$$

Définition. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de dimension k. Un ensemble $I \subseteq \{1, \dots, n\}$ de cardinal k est un **ensemble d'information pour** $\mathcal{C}$ si et seulement si $\mathcal{C}_I = \mathbb{F}_q^k$.

Remarque. Il n'y a pas unicité de l'ensemble d'information d'un code.

Ensembles d'information

Rappels. Pour $I = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I \coloneqq (x_{i_1}, \dots, x_{i_k}) \in \mathbb{F}_q^k .$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Le **poinçonnement** de $\mathcal{C} \subseteq \mathbb{F}_q^n$ **en dehors de** I (c'est-à-dire sa restriction à I) est

$$\mathcal{C}_I \coloneqq \{\boldsymbol{c}_I, \boldsymbol{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^k .$$

Définition. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de dimension k. Un ensemble $I \subseteq \{1, \dots, n\}$ de cardinal k est un **ensemble d'information pour** $\mathcal{C}$ si et seulement si $\mathcal{C}_I = \mathbb{F}_q^k$.

Remarque. Il n'y a pas unicité de l'ensemble d'information d'un code.

Exemple. Le code binaire engendré par

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

possède comme ensembles d'information $\{1,2\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$ et $\{3,4\}$.

Ensembles d'information

Rappels. Pour $I = \{i_1, \dots, i_k\} \subset \{1, \dots, n\}$ et un mot $\boldsymbol{x} \in \mathbb{F}_q^n$, on note

$$\boldsymbol{x}_I := (x_{i_1}, \dots, x_{i_k}) \in \mathbb{F}_q^k .$$

Pour $\boldsymbol{M} \in \mathbb{F}_q^{k \times n}$, on note $\boldsymbol{M}_{|I}$ la matrice constituée des colonnes de $\boldsymbol{M}$ indexées par I.

Le **poinçonnement** de $\mathcal{C} \subseteq \mathbb{F}_q^n$ **en dehors de** I (c'est-à-dire sa restriction à I) est

$$\mathcal{C}_I := \{\boldsymbol{c}_I, \boldsymbol{c} \in \mathcal{C}\} \subseteq \mathbb{F}_q^k .$$

Définition. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de dimension k. Un ensemble $I \subseteq \{1, \dots, n\}$ de cardinal k est un **ensemble d'information pour** $\mathcal{C}$ si et seulement si $\mathcal{C}_I = \mathbb{F}_q^k$.

Remarque. Il n'y a pas unicité de l'ensemble d'information d'un code.

Exemple. Le code binaire engendré par

$$\boldsymbol{G} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}$$

possède comme ensembles d'information $\{1,2\}$, $\{1,4\}$, $\{2,3\}$, $\{2,4\}$ et $\{3,4\}$.

Remarque. Un code MDS admet comme ensembles d'information tous les sous-ensembles de taille k de $\{1, \dots, n\}$. C'en est une caractérisation.

Ensembles d'information

Lemme. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de matrice génératrice $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, et de matrice de parité $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$. Soit $I \subseteq \{1,\dots,n\}$ de cardinal k et $\bar{I} = \{1,\dots,n\} \setminus I$ son complémentaire. Alors,

$$I \text{ est un ensemble d'information de } \mathcal{C} \iff \boldsymbol{G}_{|I} \text{ est inversible}$$
$$\iff \boldsymbol{H}_{|\bar{I}} \text{ est inversible.}$$

Ensembles d'information

Lemme. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de matrice génératrice $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, et de matrice de parité $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$. Soit $I \subseteq \{1,\dots,n\}$ de cardinal k et $\bar{I} = \{1,\dots,n\} \setminus I$ son complémentaire. Alors,

$$\begin{aligned} I \text{ est un ensemble d'information de } \mathcal{C} &\iff \boldsymbol{G}_{|I} \text{ est inversible} \\ &\iff \boldsymbol{H}_{|\bar{I}} \text{ est inversible.} \end{aligned}$$

Preuve. La première équivalence est due au fait que $\boldsymbol{G}_{|I}$ engendre $\mathcal{C}_I$. Pour la seconde équivalence :

$$\begin{aligned} \mathcal{C}_I = \mathbb{F}_q^k &\iff \text{aucun mot de } \mathcal{C} \text{ n'est supporté par } \bar{I} \\ &\iff \text{aucune équation de parité de } \mathcal{C}^\perp \text{ n'est supportée par } \bar{I} \\ &\iff \text{les colonnes de } \boldsymbol{H} \text{ indexées par } \bar{I} \text{ sont libres} \\ &\iff (\mathcal{C}^\perp)_{\bar{I}} = \mathbb{F}_q^{n-k} \end{aligned}$$

Ensembles d'information

Lemme. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code de matrice génératrice $\boldsymbol{G} \in \mathbb{F}_q^{k\times n}$, et de matrice de parité $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$. Soit $I \subseteq \{1,\dots,n\}$ de cardinal k et $\bar{I} = \{1,\dots,n\} \setminus I$ son complémentaire. Alors,

$$\begin{aligned} I \text{ est un ensemble d'information de } \mathcal{C} &\iff \boldsymbol{G}_{|I} \text{ est inversible} \\ &\iff \boldsymbol{H}_{|\bar{I}} \text{ est inversible.} \end{aligned}$$

Preuve. La première équivalence est due au fait que $\boldsymbol{G}_{|I}$ engendre $\mathcal{C}_I$. Pour la seconde équivalence :

$$\begin{aligned} \mathcal{C}_I = \mathbb{F}_q^k &\iff \text{aucun mot de } \mathcal{C} \text{ n'est supporté par } \bar{I} \\ &\iff \text{aucune équation de parité de } \mathcal{C}^\perp \text{ n'est supportée par } \bar{I} \\ &\iff \text{les colonnes de } \boldsymbol{H} \text{ indexées par } \bar{I} \text{ sont libres} \\ &\iff (\mathcal{C}^\perp)_{\bar{I}} = \mathbb{F}_q^{n-k} \end{aligned}$$

Conséquences.

- Un mot de code $\boldsymbol{c} \in \mathcal{C}$ est uniquement déterminé par ses coordonnées sur un ensemble d'information du code.
- Tester si un sous-ensemble donné est un ensemble d'information a pour complexité $\mathcal{O}((\min\{k, n-k\})^3)$.

Algorithme de Prange

Idée de l'algorithme de Prange (1962) :

1. on tire aléatoirement un ensemble d'information I pour $\mathcal{C}$,
2. on détermine un mot qui n'a pas d'erreur sur I, et on espère que le nombre d'erreur hors de I est petit.

Algorithme de Prange

Idée de l'algorithme de Prange (1962) :

1. on tire aléatoirement un ensemble d'information I pour $\mathcal{C}$,
2. on détermine un mot qui n'a pas d'erreur sur I, et on espère que le nombre d'erreur hors de I est petit.

ALGORITHME DE PRANGE

Entrée : $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : $\boldsymbol{x} \in \mathbb{F}_q^n$ tel que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement un ensemble d'information I de $\mathcal{C}$.
2. Résoudre le système $\begin{cases} \boldsymbol{x}_{|I} = \boldsymbol{0}, \\ \boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}. \end{cases}$
3. Si la solution $\boldsymbol{x}$ vérifie $|\boldsymbol{x}| = w$, retourner $\boldsymbol{x}$.
4. Sinon, revenir à l'étape 1.

Algorithme de Prange

Idée de l'algorithme de Prange (1962) :

1. on tire aléatoirement un ensemble d'information I pour $\mathcal{C}$,
2. on détermine un mot qui n'a pas d'erreur sur I, et on espère que le nombre d'erreur hors de I est petit.

ALGORITHME DE PRANGE

Entrée : $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : $\boldsymbol{x} \in \mathbb{F}_q^n$ tel que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement un ensemble d'information I de $\mathcal{C}$.
2. Résoudre le système $\begin{cases} \boldsymbol{x}_{|I} = \boldsymbol{0}, \\ \boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}. \end{cases}$
3. Si la solution $\boldsymbol{x}$ vérifie $|\boldsymbol{x}| = w$, retourner $\boldsymbol{x}$.
4. Sinon, revenir à l'étape 1.

Remarque. Pour $w \leq \lfloor \frac{d-1}{2} \rfloor$, si l'algorithme de Prange termine alors il retourne l'erreur $\boldsymbol{e}$ telle que $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$.

Complexité de l'algorithme de Prange ($w < w_{\mathrm{GV}}$)

Proposition. Supposons que $w < w_{\mathrm{GV}}$. Si SD admet une solution en poids w, alors l'algorithme de Prange trouve une solution en temps moyen

$$C_1 = \mathrm{poly}(n) \cdot \frac{\binom{n}{w}}{\binom{n-k}{w}}$$

où $\mathrm{poly}(n)$ représente un facteur polynomial en n.

Complexité de l'algorithme de Prange ($w < w_{\mathrm{GV}}$)

Proposition. Supposons que $w < w_{\mathrm{GV}}$. Si SD admet une solution en poids w, alors l'algorithme de Prange trouve une solution en temps moyen

$$C_1 = \mathrm{poly}(n) \cdot \frac{\binom{n}{w}}{\binom{n-k}{w}}$$

où $\mathrm{poly}(n)$ représente un facteur polynomial en n.

Preuve. Compter le nombre de mots de poids w supportés par le complémentaire de I... (exercice).

Complexité de l'algorithme de Prange ($w < w_{\mathrm{GV}}$)

Proposition. Supposons que $w < w_{\mathrm{GV}}$. Si SD admet une solution en poids w, alors l'algorithme de Prange trouve une solution en temps moyen

$$C_1 = \mathrm{poly}(n) \cdot \frac{\binom{n}{w}}{\binom{n-k}{w}}$$

où $\mathrm{poly}(n)$ représente un facteur polynomial en n.

Preuve. Compter le nombre de mots de poids w supportés par le complémentaire de I... (exercice).

Dans le cas $w < w_{\mathrm{GV}}$, on obtient asymptotiquement

$$C_1 = 2^{n(\gamma_1 + o(1))}$$

où

$$\gamma_1 = h(\omega) - (1-R)h(w/(n-k)) = h(\omega) - (1-R)h(\omega/(1-R))$$

Complexité de l'algorithme de Prange ($w > w_{\mathrm{GV}}$)

Proposition. Supposons que $w > w_{\mathrm{GV}}$. L'algorithme de Prange trouve une solution en temps moyen

$$C_2 = \mathrm{poly}(n) \cdot \frac{2^{n-k}}{\binom{n-k}{w}}$$

où $\mathrm{poly}(n)$ représente un facteur polynomial en n.

Preuve. Comme précédemment. Penser à moyenner par le nombre de solutions au problème.

Complexité de l'algorithme de Prange ($w > w_{\mathrm{GV}}$)

Proposition. Supposons que $w > w_{\mathrm{GV}}$. L'algorithme de Prange trouve une solution en temps moyen

$$C_2 = \mathrm{poly}(n) \cdot \frac{2^{n-k}}{\binom{n-k}{w}}$$

où $\mathrm{poly}(n)$ représente un facteur polynomial en n.

Preuve. Comme précédemment. Penser à moyenner par le nombre de solutions au problème.

Dans le cas $w > w_{\mathrm{GV}}$, on obtient asymptotiquement

$$C_2 = 2^{n(\gamma_2 + o(1))}$$

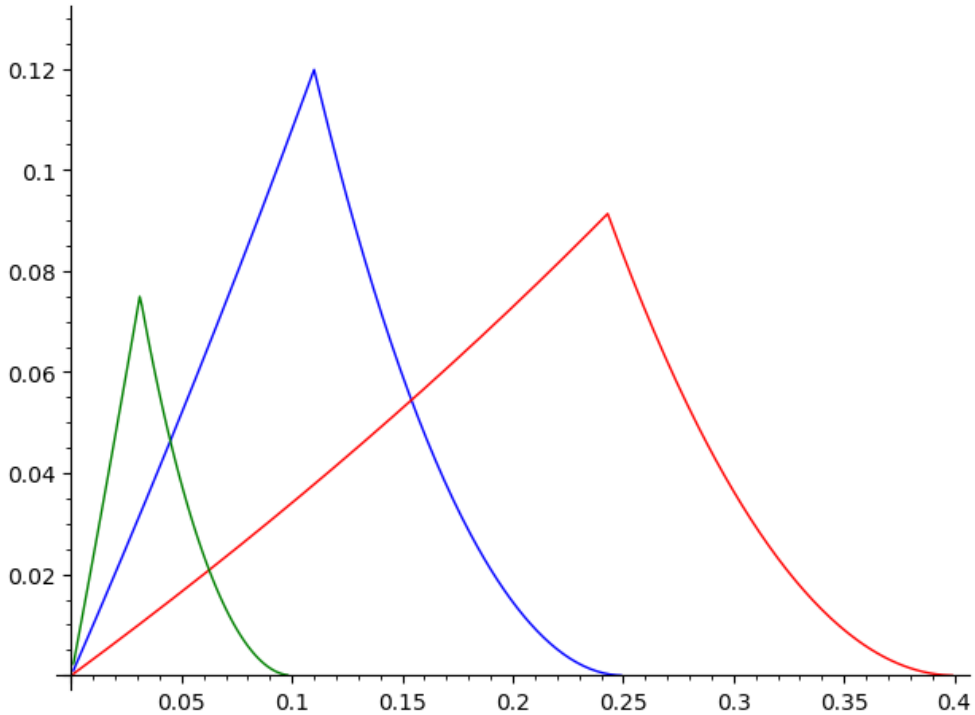
où

$$\gamma_2 = (1-R) - (1-R)h(w/(n-k)) = (1-R)(1-h(\omega/(1-R))$$

Remarque : dès que $w \geq n-k$, il devient très aisé de trouver une solution.

Exposant asymptotique de Prange

Observons l'évolution de l'exposant asymptotique en fonction de ω et R :



Exposant asymptotique $C(R,\omega)$ de la compelxité de l'algorithme de Prange en fonction de ω. Autrement dit, la complexité de l'algorithme est en $2^{C(R,\omega)}$, où ω varie selon l'axe des abscisses, et R selon la couleur.

En abscisse : ω.
En ordonnée : l'exposant asymptotique.
Couleurs : $R = 0.8$, $R = 0.5$, $R = 0.2$.

Plan

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Idée : exploiter le **paradoxe des anniversaires** grâce à un compromis temps-mémoire.

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Idée : exploiter le **paradoxe des anniversaires** grâce à un compromis temps-mémoire.

Pour simplifier, on suppose que w et n sont pairs.

Soit $J_1 \sqcup J_2$ une partition de $[1, n]$, telle que $|J_1| = |J_2| = n/2$. On écrit $\boldsymbol{e}_i := \boldsymbol{e}_{|J_i}$ et $\boldsymbol{H}_i := \boldsymbol{H}_{|J_i}$. Alors,

$$\boldsymbol{e}\boldsymbol{H}^\top - \boldsymbol{s} = \boldsymbol{e}_1\boldsymbol{H}_1^\top + (\boldsymbol{e}_2\boldsymbol{H}_2^\top - \boldsymbol{s})$$

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Idée : exploiter le **paradoxe des anniversaires** grâce à un compromis temps-mémoire.

Pour simplifier, on suppose que w et n sont pairs.

Soit $J_1 \sqcup J_2$ une partition de $[1, n]$, telle que $|J_1| = |J_2| = n/2$. On écrit $\boldsymbol{e}_i := \boldsymbol{e}_{|J_i}$ et $\boldsymbol{H}_i := \boldsymbol{H}_{|J_i}$. Alors,

$$\boldsymbol{e}\boldsymbol{H}^\top - \boldsymbol{s} = \boldsymbol{e}_1\boldsymbol{H}_1^\top + (\boldsymbol{e}_2\boldsymbol{H}_2^\top - \boldsymbol{s})$$

On espère alors que J_1 et J_2 coupent $\boldsymbol{e}$ en deux erreurs de poids $w/2$ et de longueur $n/2$.

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Idée : exploiter le **paradoxe des anniversaires** grâce à un compromis temps-mémoire.

Pour simplifier, on suppose que w et n sont pairs.

Soit $J_1 \sqcup J_2$ une partition de $[1, n]$, telle que $|J_1| = |J_2| = n/2$. On écrit $\boldsymbol{e}_i := \boldsymbol{e}_{|J_i}$ et $\boldsymbol{H}_i := \boldsymbol{H}_{|J_i}$. Alors,

$$\boldsymbol{e}\boldsymbol{H}^\top - \boldsymbol{s} = \boldsymbol{e}_1\boldsymbol{H}_1^\top + (\boldsymbol{e}_2\boldsymbol{H}_2^\top - \boldsymbol{s})$$

On espère alors que J_1 et J_2 coupent $\boldsymbol{e}$ en deux erreurs de poids $w/2$ et de longueur $n/2$.

On construit donc deux listes

$$L_1 = \{\boldsymbol{x}_1\boldsymbol{H}_1^\top \mid \boldsymbol{x}_1 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_1| = w/2\}$$
$$L_2 = \{\boldsymbol{s} - \boldsymbol{x}_2\boldsymbol{H}_2^\top \mid \boldsymbol{x}_2 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_2| = w/2\}$$

Méthode de Dumer : idée

Rappel. On souhaite résoudre $\boldsymbol{e}\boldsymbol{H}^\top = \boldsymbol{s}$ où $\boldsymbol{e}$ a poids w.

Idée : exploiter le **paradoxe des anniversaires** grâce à un compromis temps-mémoire.

Pour simplifier, on suppose que w et n sont pairs.

Soit $J_1 \sqcup J_2$ une partition de $[1, n]$, telle que $|J_1| = |J_2| = n/2$. On écrit $\boldsymbol{e}_i := \boldsymbol{e}_{|J_i}$ et $\boldsymbol{H}_i := \boldsymbol{H}_{|J_i}$. Alors,

$$\boldsymbol{e}\boldsymbol{H}^\top - \boldsymbol{s} = \boldsymbol{e}_1\boldsymbol{H}_1^\top + (\boldsymbol{e}_2\boldsymbol{H}_2^\top - \boldsymbol{s})$$

On espère alors que J_1 et J_2 coupent $\boldsymbol{e}$ en deux erreurs de poids $w/2$ et de longueur $n/2$.

On construit donc deux listes

$$L_1 = \{\boldsymbol{x}_1\boldsymbol{H}_1^\top \mid \boldsymbol{x}_1 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_1| = w/2\}$$
$$L_2 = \{\boldsymbol{s} - \boldsymbol{x}_2\boldsymbol{H}_2^\top \mid \boldsymbol{x}_2 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_2| = w/2\}$$

Et on retrouve $\boldsymbol{e} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ comme une collision entre les listes.

Algorithme de Dumer

Définition des listes :

$$L_1 = \{\boldsymbol{x}_1 \boldsymbol{H}_1^\top \mid \boldsymbol{x}_1 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_1| = w/2\}$$
$$L_2 = \{\boldsymbol{s} - \boldsymbol{x}_2 \boldsymbol{H}_2^\top \mid \boldsymbol{x}_2 \in \mathbb{F}_q^{n/2}, |\boldsymbol{x}_2| = w/2\}$$

ALGORITHME DE DUMER

Entrée : $\boldsymbol{H}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : un ensemble de vecteurs $\boldsymbol{x}$ tels que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement une partition $J_1 \sqcup J_2$ de $\{1, \ldots, n\}$.
2. Construire les listes L_1 et L_2 comme définies plus haut.
3. Si $L_1 \cap L_2 = \varnothing$, revenir à l'étape 1.
4. Sinon, retourner l'ensemble des $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ provenant de $L_1 \cap L_2$.

Comptage pour Dumer

Lemme. Soient deux listes L_1, L_2 d'éléments de $\mathbb{F}_q^{n-k}$ tirés uniformément, de taille N chacune. Alors, le nombre moyen de collisions entre L_1 et L_2 est :

$$\frac{N^2}{q^{n-k}}$$

Comptage pour Dumer

Lemme. Soient deux listes L_1, L_2 d'éléments de $\mathbb{F}_q^{n-k}$ tirés uniformément, de taille N chacune. Alors, le nombre moyen de collisions entre L_1 et L_2 est :

$$\frac{N^2}{q^{n-k}}$$

Preuve. Ce nombre moyen de collisions est

$$\mathbb{E}\Big(\sum_{x \in \mathbb{F}_q^n} \mathbb{1}_{x \in L_1} \mathbb{1}_{x \in L_2} \Big),$$

où $\mathbb{1}_{\mathcal{E}}$ est la fonction indicatrice de l'évènement $\mathcal{E}$.

Comptage pour Dumer

Lemme. Soient deux listes L_1, L_2 d'éléments de $\mathbb{F}_q^{n-k}$ tirés uniformément, de taille N chacune. Alors, le nombre moyen de collisions entre L_1 et L_2 est :

$$\frac{N^2}{q^{n-k}}$$

Preuve. Ce nombre moyen de collisions est

$$\mathbb{E}\Big(\sum_{x \in \mathbb{F}_q^n} \mathbb{1}_{x \in L_1} \mathbb{1}_{x \in L_2} \Big),$$

où $\mathbb{1}_{\mathcal{E}}$ est la fonction indicatrice de l'évènement $\mathcal{E}$. On fait ensuite jouer la linéarité de l'espérance et l'indépendance des variables :

$$\mathbb{E}\Big(\sum_{x \in \mathbb{F}_q^{n-k}} \mathbb{1}_{x \in L_1} \mathbb{1}_{x \in L_2} \Big) = q^{n-k} \left(\frac{N}{q^{n-k}} \right)^2$$

Comptage pour Dumer

Lemme. Soient deux listes L_1, L_2 d'éléments de $\mathbb{F}_q^{n-k}$ tirés uniformément, de taille N chacune. Alors, le nombre moyen de collisions entre L_1 et L_2 est :

$$\frac{N^2}{q^{n-k}}$$

Preuve. Ce nombre moyen de collisions est

$$\mathbb{E}\Big(\sum_{x \in \mathbb{F}_q^n} \mathbb{1}_{x \in L_1} \mathbb{1}_{x \in L_2} \Big) ,$$

où $\mathbb{1}_{\mathcal{E}}$ est la fonction indicatrice de l'évènement $\mathcal{E}$. On fait ensuite jouer la linéarité de l'espérance et l'indépendance des variables :

$$\mathbb{E}\Big(\sum_{x \in \mathbb{F}_q^{n-k}} \mathbb{1}_{x \in L_1} \mathbb{1}_{x \in L_2} \Big) = q^{n-k} \left(\frac{N}{q^{n-k}} \right)^2$$

Lemme. Le nombre d'éléments dans les listes L_1 et L_2 est

$$N = \binom{n/2}{w/2} (q-1)^{w/2} .$$

Algorithme de Dumer : complexité

ALGORITHME DE DUMER

Entrée : $\boldsymbol{H}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : un ensemble de vecteurs $\boldsymbol{x}$ tels que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement une partition $J_1 \sqcup J_2$ de $\{1, \dots, n\}$.
2. Construire les listes L_1 et L_2 comme définies plus haut.
3. Si $L_1 \cap L_2 = \varnothing$, revenir à l'étape 1.
4. Sinon, retourner l'ensemble des $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ provenant de $L_1 \cap L_2$.

Théorème ($q = 2$). Soit $N = \binom{n/2}{w/2}$ et $M = N^2/2^{n-k}$. La méthode de Dumer fournit en moyenne M solutions au problème de décodage par syndrome, en temps $M + N$ (à un facteur polynomial près).

Algorithme de Dumer : complexité

ALGORITHME DE DUMER

Entrée : $\boldsymbol{H}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : un ensemble de vecteurs $\boldsymbol{x}$ tels que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement une partition $J_1 \sqcup J_2$ de $\{1, \dots, n\}$.
2. Construire les listes L_1 et L_2 comme définies plus haut.
3. Si $L_1 \cap L_2 = \varnothing$, revenir à l'étape 1.
4. Sinon, retourner l'ensemble des $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ provenant de $L_1 \cap L_2$.

Théorème ($q = 2$). Soit $N = \binom{n/2}{w/2}$ et $M = N^2/2^{n-k}$. La méthode de Dumer fournit en moyenne M solutions au problème de décodage par syndrome, en temps $M + N$ (à un facteur polynomial près).

Preuve. Une bonne partition est trouvée avec probabilité $p = \binom{n/2}{w/2}^2 / \binom{n}{w}$. On doit donc effectuer $1/p$ itérations, et on note que $1/p = \mathrm{poly}(n, w)$.

Algorithme de Dumer : complexité

ALGORITHME DE DUMER

Entrée : $\boldsymbol{H}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : un ensemble de vecteurs $\boldsymbol{x}$ tels que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement une partition $J_1 \sqcup J_2$ de $\{1, \dots, n\}$.
2. Construire les listes L_1 et L_2 comme définies plus haut.
3. Si $L_1 \cap L_2 = \varnothing$, revenir à l'étape 1.
4. Sinon, retourner l'ensemble des $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ provenant de $L_1 \cap L_2$.

Théorème ($q = 2$). Soit $N = \binom{n/2}{w/2}$ et $M = N^2/2^{n-k}$. La méthode de Dumer fournit en moyenne M solutions au problème de décodage par syndrome, en temps $M + N$ (à un facteur polynomial près).

Preuve. Une bonne partition est trouvée avec probabilité $p = \binom{n/2}{w/2}^2 / \binom{n}{w}$. On doit donc effectuer $1/p$ itérations, et on note que $1/p = \mathrm{poly}(n, w)$.
Pour chaque itération, la complexité est $2N$.

Algorithme de Dumer : complexité

ALGORITHME DE DUMER

Entrée : $\boldsymbol{H}$, w et $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\top$ où $|\boldsymbol{e}| = w$.
Sortie : un ensemble de vecteurs $\boldsymbol{x}$ tels que $|\boldsymbol{x}| = w$ et $\boldsymbol{x}\boldsymbol{H}^\top = \boldsymbol{s}$.

1. Choisir aléatoirement une partition $J_1 \sqcup J_2$ de $\{1, \dots, n\}$.
2. Construire les listes L_1 et L_2 comme définies plus haut.
3. Si $L_1 \cap L_2 = \varnothing$, revenir à l'étape 1.
4. Sinon, retourner l'ensemble des $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2)$ provenant de $L_1 \cap L_2$.

Théorème ($q = 2$). Soit $N = \binom{n/2}{w/2}$ et $M = N^2/2^{n-k}$. La méthode de Dumer fournit en moyenne M solutions au problème de décodage par syndrome, en temps $M + N$ (à un facteur polynomial près).

Preuve. Une bonne partition est trouvée avec probabilité $p = \binom{n/2}{w/2}^2 / \binom{n}{w}$. On doit donc effectuer $1/p$ itérations, et on note que $1/p = \mathrm{poly}(n, w)$.
Pour chaque itération, la complexité est $2N$.
Pour la dernière itération, on obtient M solutions, donc la complexité est $M + 2N$.

Plan

État de l'art pour le décodage générique

Si vous souhaitez **aller plus loin** :

- Autres algorithmes de type ISD (*information set decoding*)
- Algorithme de Wagner
- Algorithme MMT (May, Meurer, Thomae)
- Algorithme BJMM (Becker, Joux, May, Maurer)
- Algorithme de May et Ozerov
- et beaucoup d'optimisations...

État de l'art pour le décodage générique

Si vous souhaitez **aller plus loin** :

- Autres algorithmes de type ISD (*information set decoding*)
- Algorithme de Wagner
- Algorithme MMT (May, Meurer, Thomae)
- Algorithme BJMM (Becker, Joux, May, Maurer)
- Algorithme de May et Ozerov
- et beaucoup d'optimisations...

En théorie, le meilleur exposant (en 2018) était obtenu par une optimisation de BJMM par Both et May

$\Longrightarrow$ complexité en $O(2^{2n/21})$.

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Le schéma de McEliece originel proposait $n = 1024$, beaucoup trop petit actuellement (attaques en $\simeq 2^{50}$ opérations).

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Le schéma de McEliece originel proposait $n = 1024$, beaucoup trop petit actuellement (attaques en $\simeq 2^{50}$ opérations).

Dans la récente proposition de standardisation au NIST (`https://classic.mceliece.org`), les valeurs $n = 6960$ et $t = 119$ ont été choisies (parmi d'autres), pour une sécurité annoncée de 256 bits.

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Le schéma de McEliece originel proposait $n = 1024$, beaucoup trop petit actuellement (attaques en $\simeq 2^{50}$ opérations).

Dans la récente proposition de standardisation au NIST (`https://classic.mceliece.org`), les valeurs $n = 6960$ et $t = 119$ ont été choisies (parmi d'autres), pour une sécurité annoncée de 256 bits.

Taille de clefs :

sécurité (bits)	128	192	256
clé publique	243 ko	425 ko	1.34 Mo

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Le schéma de McEliece originel proposait $n = 1024$, beaucoup trop petit actuellement (attaques en $\simeq 2^{50}$ opérations).

Dans la récente proposition de standardisation au NIST (`https://classic.mceliece.org`), les valeurs $n = 6960$ et $t = 119$ ont été choisies (parmi d'autres), pour une sécurité annoncée de 256 bits.

Taille de clefs :

sécurité (bits)	128	192	256
clé publique	243 ko	425 ko	1.34 Mo

Pour l'instant, aucun algorithme quantique ne permet de résoudre très efficacement le problème SD.

Choix des paramètres

En pratique, on choisit un rendement $R = \frac{k}{n} \simeq 0{,}8$; le poids correspondant à un décodage possible pour Alice, et le plus difficile pour un attaquant, est alors $w \simeq n/5 \log_2 n$.

Le schéma de McEliece originel proposait $n = 1024$, beaucoup trop petit actuellement (attaques en $\simeq 2^{50}$ opérations).

Dans la récente proposition de standardisation au NIST (`https://classic.mceliece.org`), les valeurs $n = 6960$ et $t = 119$ ont été choisies (parmi d'autres), pour une sécurité annoncée de 256 bits.

Taille de clefs :

sécurité (bits)	128	192	256
clé publique	243 ko	425 ko	1.34 Mo

Pour l'instant, aucun algorithme quantique ne permet de résoudre très efficacement le problème SD.

Liens :

1. `https://decodingchallenge.org`, un site web pour estimer la difficulté de problèmes de décodage
2. Cawof, un logiciel pour calculer le coût pratique des meilleurs algorithmes pour résoudre SD.

Plan

Plan

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

On construit une **fonction à sens unique et à trappe** $f : \mathcal{S} \to \mathcal{A}$ (*one-way trapdoor function*), où $\mathcal{S}$ est l'ensemble des signatures. Plus précisément, la fonction f doit satisfaire :

1. f s'évalue facilement sur $\mathcal{S}$,
2. sans la connaissance de la trappe, il est calculatoirement difficile de calculer une pré-image $f^{-1}(\boldsymbol{a})$, pour $\boldsymbol{a} = f(\boldsymbol{s})$,
3. avec la connaissance de la trappe, calculer $f^{-1}(f(\boldsymbol{s}))$ est aisé.

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

On construit une **fonction à sens unique et à trappe** $f : \mathcal{S} \to \mathcal{A}$ (*one-way trapdoor function*), où $\mathcal{S}$ est l'ensemble des signatures. Plus précisément, la fonction f doit satisfaire :

1. f s'évalue facilement sur $\mathcal{S}$,
2. sans la connaissance de la trappe, il est calculatoirement difficile de calculer une pré-image $f^{-1}(\boldsymbol{a})$, pour $\boldsymbol{a} = f(\boldsymbol{s})$,
3. avec la connaissance de la trappe, calculer $f^{-1}(f(\boldsymbol{s}))$ est aisé.

On obtient une signature de la manière suivante :

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

On construit une **fonction à sens unique et à trappe** $f : \mathcal{S} \to \mathcal{A}$ (*one-way trapdoor function*), où $\mathcal{S}$ est l'ensemble des signatures. Plus précisément, la fonction f doit satisfaire :

1. f s'évalue facilement sur $\mathcal{S}$,
2. sans la connaissance de la trappe, il est calculatoirement difficile de calculer une pré-image $f^{-1}(\boldsymbol{a})$, pour $\boldsymbol{a} = f(\boldsymbol{s})$,
3. avec la connaissance de la trappe, calculer $f^{-1}(f(\boldsymbol{s}))$ est aisé.

On obtient une signature de la manière suivante :

– la **clé privée** est la trappe (autrement dit, f^{-1}), la **clé publique** est f ;

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

On construit une **fonction à sens unique et à trappe** $f : \mathcal{S} \to \mathcal{A}$ (*one-way trapdoor function*), où $\mathcal{S}$ est l'ensemble des signatures. Plus précisément, la fonction f doit satisfaire :

1. f s'évalue facilement sur $\mathcal{S}$,
2. sans la connaissance de la trappe, il est calculatoirement difficile de calculer une pré-image $f^{-1}(\boldsymbol{a})$, pour $\boldsymbol{a} = f(\boldsymbol{s})$,
3. avec la connaissance de la trappe, calculer $f^{-1}(f(\boldsymbol{s}))$ est aisé.

On obtient une signature de la manière suivante :

- la **clé privée** est la trappe (autrement dit, f^{-1}), la **clé publique** est f ;
- $\mathsf{Sign}(\boldsymbol{m}, \mathtt{sk}) = f^{-1}(\mathcal{H}(\boldsymbol{m}))$;

Paradigme hash-and-sign

La **signature CFS** (Courtois, Finiasz, Sendrier, 2001) est une signature à base de codes reposant sur le paradigme *hash-and-sign*.

Hash-and-sign. On se donne une fonction de hachage cryptographique vers un ensemble $\mathcal{A}$:

$$\begin{array}{rccc} \mathcal{H}: & \{0,1\}^* & \to & \mathcal{A} \\ & \boldsymbol{m} & \mapsto & \boldsymbol{a} = \mathcal{H}(\boldsymbol{m})\,. \end{array}$$

On construit une **fonction à sens unique et à trappe** $f : \mathcal{S} \to \mathcal{A}$ (*one-way trapdoor function*), où $\mathcal{S}$ est l'ensemble des signatures. Plus précisément, la fonction f doit satisfaire :

1. f s'évalue facilement sur $\mathcal{S}$,
2. sans la connaissance de la trappe, il est calculatoirement difficile de calculer une pré-image $f^{-1}(\boldsymbol{a})$, pour $\boldsymbol{a} = f(\boldsymbol{s})$,
3. avec la connaissance de la trappe, calculer $f^{-1}(f(\boldsymbol{s}))$ est aisé.

On obtient une signature de la manière suivante :

- la **clé privée** est la trappe (autrement dit, f^{-1}), la **clé publique** est f ;
- $\mathsf{Sign}(\boldsymbol{m}, \mathtt{sk}) = f^{-1}(\mathcal{H}(\boldsymbol{m}))$;
- $\mathsf{Verify}(\boldsymbol{s}, \boldsymbol{m}, \mathtt{pk})$ teste si $\boldsymbol{s} \in \mathcal{S}$ et $f(\boldsymbol{s}) = \mathcal{H}(\boldsymbol{m})$.

CFS – idée

Instantiation de f dans CFS. On considère $\mathcal{F}$, l'ensemble des codes de Goppa binaires irréductibles, avec $q = 2^m$, $n \leq q$ et $k = n - wm$. On note $\mathcal{S} = S_w := \{\boldsymbol{e} \in \mathbb{F}_2^n \mid |\boldsymbol{e}| = w\}$.

Si $\boldsymbol{H}$ est une matrice de contrôle aléatoire d'un code $\mathcal{C} \in \mathcal{F}$ tel code de Goppa, alors

$$\begin{array}{rccc} f_{\boldsymbol{H}} : & \mathcal{S} = S_w & \to & \mathcal{A} \subseteq \mathbb{F}_2^{n-k} \\ & \boldsymbol{e} & \mapsto & \boldsymbol{e}\boldsymbol{H}^\top. \end{array}$$

CFS – idée

Instantiation de f dans CFS. On considère $\mathcal{F}$, l'ensemble des codes de Goppa binaires irréductibles, avec $q = 2^m$, $n \leq q$ et $k = n - wm$. On note $\mathcal{S} = S_w := \{\boldsymbol{e} \in \mathbb{F}_2^n \mid |\boldsymbol{e}| = w\}$.

Si $\boldsymbol{H}$ est une matrice de contrôle aléatoire d'un code $\mathcal{C} \in \mathcal{F}$ tel code de Goppa, alors

$$\begin{array}{rccc} f_{\boldsymbol{H}} : & \mathcal{S} = S_w & \to & \mathcal{A} \subseteq \mathbb{F}_2^{n-k} \\ & \boldsymbol{e} & \mapsto & \boldsymbol{e}\boldsymbol{H}^\top. \end{array}$$

Idée : la sécurité de la signature devrait reposer sur les mêmes problèmes que ceux associés au chiffrement de McEliece (décodage générique + distinction de code) .

CFS – idée

Instantiation de f dans CFS. On considère $\mathcal{F}$, l'ensemble des codes de Goppa binaires irréductibles, avec $q = 2^m$, $n \leq q$ et $k = n - wm$. On note $\mathcal{S} = S_w := \{\boldsymbol{e} \in \mathbb{F}_2^n \mid |\boldsymbol{e}| = w\}$.

Si $\boldsymbol{H}$ est une matrice de contrôle aléatoire d'un code $\mathcal{C} \in \mathcal{F}$ tel code de Goppa, alors

$$\begin{array}{rccc} f_{\boldsymbol{H}} : & \mathcal{S} = S_w & \to & \mathcal{A} \subseteq \mathbb{F}_2^{n-k} \\ & \boldsymbol{e} & \mapsto & \boldsymbol{e}\boldsymbol{H}^\top. \end{array}$$

Idée : la sécurité de la signature devrait reposer sur les mêmes problèmes que ceux associés au chiffrement de McEliece (décodage générique + distinction de code) .

CFS – idée

CFS (IDÉE) : GÉNÉRATION DE CLEFS

1. Alice choisit aléatoirement $\mathcal{C} = \Gamma(g, \boldsymbol{a}) \in \mathcal{F}$
2. Alice calcule une matrice de parité (aléatoire) $\boldsymbol{H} \in \mathbb{F}_2^{(n-k)\times n}$ de $\mathcal{C}$
3. Les clefs sont :
   - clé privée : g et $\boldsymbol{a}$
   - clé publique : $\boldsymbol{H}$.

CFS (IDÉE) : SIGNATURE

1. Alice hache son message $\boldsymbol{m} \in \{0,1\}^*$ et obtient $\boldsymbol{s} = \mathcal{H}(\boldsymbol{m}) \in \mathbb{F}_q^{n-k}$.
2. Alice calcule une erreur $\boldsymbol{e}$ de poids w correspondant au syndrome $\boldsymbol{s}$.
3. La signature est $\boldsymbol{e}$.

CFS (IDÉE) : VÉRIFICATION

1. Bob vérifie que $\boldsymbol{e}$ est de poids w, et que $\boldsymbol{e}\boldsymbol{H}^\top = \mathcal{H}(\boldsymbol{m})$.

CFS – problème de la surjectivité

Problème : surjectivité de la fonction $f_{\boldsymbol{H}}$?

1. Vue comme $f_{\boldsymbol{H}} : S_w \to \mathbb{F}_q^{n-k}$, elle est loin d'être surjective.
2. On ne sait pas (a priori) déterminer son image (et même en donner une approximation).

CFS – problème de la surjectivité

Problème : surjectivité de la fonction $f_{\boldsymbol{H}}$?

1. Vue comme $f_{\boldsymbol{H}} : S_w \to \mathbb{F}_q^{n-k}$, elle est loin d'être surjective.
2. On ne sait pas (a priori) déterminer son image (et même en donner une approximation).

En quoi est-ce important ? Il existe beaucoup de fichiers hachés qu'Alice ne pourra pas signer (car il n'existera pas de $\boldsymbol{e}$ associé à $\boldsymbol{s} = \mathcal{H}(\boldsymbol{m})$).

CFS – problème de la surjectivité

Problème : surjectivité de la fonction $f_{\boldsymbol{H}}$?

1. Vue comme $f_{\boldsymbol{H}} : S_w \to \mathbb{F}_q^{n-k}$, elle est loin d'être surjective.
2. On ne sait pas (a priori) déterminer son image (et même en donner une approximation).

En quoi est-ce important ? Il existe beaucoup de fichiers hachés qu'Alice ne pourra pas signer (car il n'existera pas de $\boldsymbol{e}$ associé à $\boldsymbol{s} = \mathcal{H}(\boldsymbol{m})$).

Solutions.

1. Ajouter des **erreurs artificielles** pour faire « grossir » S_w et augmenter les chances d'avoir une préimage (il faudra ajouter les indices de ces erreurs artificielles à la signature).
2. **Itérer des hachés** $\boldsymbol{s}^{(0)} = \mathcal{H}(\boldsymbol{m} \,\|\, 0)$, $\boldsymbol{s}^{(i)} = \mathcal{H}(\boldsymbol{m} \,\|\, 1)$, ..., jusqu'à obtenir un $\boldsymbol{s}^{(i)} = \mathcal{H}(\boldsymbol{m} \,\|\, i)$ qui est dans l'image de $f_{\boldsymbol{H}}$ (il faudra ajouter i à la signature).
3. Se placer dans un régime de paramètres où la proportion d'éléments $\mathcal{A} \subseteq \mathbb{F}_2^{n-k}$ qui admettent une pré-image par $f_{\boldsymbol{H}}$ est $\simeq$ constante.
   - On choisit donc des codes de Goppa **à haut rendement** $k/n = 1 - o(1)$.
   - On obtient alors une proportion de signatures valides $\binom{n}{w}/2^{n-k} \sim 1/w!$.
   - **Avantage supplémentaire :** on obtient des signatures plus courtes.

CFS – problème de la surjectivité

Problème : surjectivité de la fonction $f_{\boldsymbol{H}}$?

1. Vue comme $f_{\boldsymbol{H}} : S_w \to \mathbb{F}_q^{n-k}$, elle est loin d'être surjective.
2. On ne sait pas (a priori) déterminer son image (et même en donner une approximation).

En quoi est-ce important ? Il existe beaucoup de fichiers hachés qu'Alice ne pourra pas signer (car il n'existera pas de $\boldsymbol{e}$ associé à $\boldsymbol{s} = \mathcal{H}(\boldsymbol{m})$).

Solutions.

1. Ajouter des **erreurs artificielles** pour faire « grossir » S_w et augmenter les chances d'avoir une préimage (il faudra ajouter les indices de ces erreurs artificielles à la signature).
2. **Itérer des hachés** $\boldsymbol{s}^{(0)} = \mathcal{H}(\boldsymbol{m} \,\|\, 0)$, $\boldsymbol{s}^{(i)} = \mathcal{H}(\boldsymbol{m} \,\|\, 1)$, ..., jusqu'à obtenir un $\boldsymbol{s}^{(i)} = \mathcal{H}(\boldsymbol{m} \,\|\, i)$ qui est dans l'image de $f_{\boldsymbol{H}}$ (il faudra ajouter i à la signature).
3. Se placer dans un régime de paramètres où la proportion d'éléments $\mathcal{A} \subseteq \mathbb{F}_2^{n-k}$ qui admettent une pré-image par $f_{\boldsymbol{H}}$ est $\simeq$ constante.
   - On choisit donc des codes de Goppa **à haut rendement** $k/n = 1 - o(1)$.
   - On obtient alors une proportion de signatures valides $\binom{n}{w}/2^{n-k} \sim 1/w!$.
   - **Avantage supplémentaire :** on obtient des signatures plus courtes.

En pratique on peut faire les trois...

CFS : d'autres inconvénients... et en pratique ?

Problème : en régime $k/n = 1 - o(1)$, la méthode de Dumer permet de trouver des solutions au problème SD en temps relativement faible :

$$T = 2^{\frac{n-k+o(n)}{2}} \simeq 2^{mw/2} = n^{w/2}$$

CFS : d'autres inconvénients... et en pratique ?

Problème : en régime $k/n = 1 - o(1)$, la méthode de Dumer permet de trouver des solutions au problème SD en temps relativement faible :

$$T = 2^{\frac{n-k+o(n)}{2}} \simeq 2^{mw/2} = n^{w/2}$$

Autrement, dit, T est polynomial en la taille $K \simeq nw\log(n)$ de la clé publique, et l'exposant est borné par $w/2$.

CFS : d'autres inconvénients... et en pratique ?

Problème : en régime $k/n = 1 - o(1)$, la méthode de Dumer permet de trouver des solutions au problème SD en temps relativement faible :

$$T = 2^{\frac{n-k+o(n)}{2}} \simeq 2^{mw/2} = n^{w/2}$$

Autrement, dit, T est polynomial en la taille $K \simeq nw\log(n)$ de la clé publique, et l'exposant est borné par $w/2$.

En pratique, il existe certaines méthodes pour renforcer la sécurité du schéma CFS (ex : par exemple Parallel-CFS).

CFS : d'autres inconvénients... et en pratique ?

Problème : en régime $k/n = 1 - o(1)$, la méthode de Dumer permet de trouver des solutions au problème SD en temps relativement faible :

$$T = 2^{\frac{n-k+o(n)}{2}} \simeq 2^{mw/2} = n^{w/2}$$

Autrement, dit, T est polynomial en la taille $K \simeq nw\log(n)$ de la clé publique, et l'exposant est borné par $w/2$.

En pratique, il existe certaines méthodes pour renforcer la sécurité du schéma CFS (ex : par exemple Parallel-CFS).

On peut également choisir w une constante « assez grande ». Cependant, les tailles de clé restent exorbitantes pour les applications usuelles (plusieurs Go pour 128bits de sécurité, avec des temps de signature largement supérieurs à la seconde).

CFS : d'autres inconvénients... et en pratique ?

Problème : en régime $k/n = 1 - o(1)$, la méthode de Dumer permet de trouver des solutions au problème SD en temps relativement faible :

$$T = 2^{\frac{n-k+o(n)}{2}} \simeq 2^{mw/2} = n^{w/2}$$

Autrement, dit, T est polynomial en la taille $K \simeq nw\log(n)$ de la clé publique, et l'exposant est borné par $w/2$.

En pratique, il existe certaines méthodes pour renforcer la sécurité du schéma CFS (ex : par exemple Parallel-CFS).

On peut également choisir w une constante « assez grande ». Cependant, les tailles de clé restent exorbitantes pour les applications usuelles (plusieurs Go pour 128bits de sécurité, avec des temps de signature largement supérieurs à la seconde).

Plus récemment, **d'autres signatures** à base de codes ont été construites et sont plus prometteuses :

1. WAVE, une signature se basant sur le problème SD en poids très haut, sur le corps $\mathbb{F}_3$.
2. DURANDAL, une signature à base de **codes en métrique rang**.

Plan

Schéma d'identification de Stern

(au tableau... ou en exercice la semaine prochaine)

Références

 *A Public-Key System Based on Algebraic Coding Theory*. R. McEliece. DSN Progress Report. **1978**.

 *The use of information sets in decoding cyclic code*. E. Prange. IRE TIT. **1962**.

 *On minimum distance decoding of linear codes*. I. Dumer. 5th Soviet-Swedish Int. Workshop Inform. Theory. **1991**.

 *Decoding random linear codes in* $O(2^{0.054n})$. A. May, A. Maurer, E. Thomae. ASIACRYPT. **2011**.

 *Decoding Random Binary Linear Codes in* $2^{n/20}$ *: How 1 + 1 = 0 Improves Information Set Decoding*. A. Becker, A. Joux, A. May, A. Maurer. EUROCRYPT. **2012**.

Une thèse en français avec un joli état de l'art :

 *Cryptographie fondée sur les codes*. T. Debris-Alazard. Thèse de doctorat. **2019**.

Fin de la séance

Questions ?