Schémas de partages de secrets basés sur les codes correcteurs d'erreurs

Jean Gasnier

24/07/2021

RÉSUMÉ :

Ce rapport présente une introduction aux schémas de partages de secret (SSS) basés sur des codes correcteurs d'erreurs linéaires. Il débute par un rappel de notions de bases de théorie de l'information et par une brève présentation sur les codes correcteurs d'erreurs. Quelques exemples permettent d'appréhender la notion de structure d'accès et le fonctionnement des SSS, et sont suivis de définitions formelles. On explicite des liens entre les polymatroïdes et les SSS, puis on présente les deux constructions (de Shamir et de Massey) de SSS basés sur des codes linéaires. Le rapport se termine par la présentation d'applications des SSS, spécifiquement celui de Shamir.

Ce rapport a été écrit dans le cadre d'un stage de 2 mois dans l'équipe ATI de l'Institut de Mathématiques de Marseille (I2M), encadré par Alexis Bonnecaze, du 20/05 au 23/07. Dans le cadre du stage, une présentation a été faite au séminaire des doctorants de l'I2M (voir ce lien).

Table des matières

Introduction

Les schémas de partages de secrets (ou SSS) sont des méthodes permettant de distribuer un secret en plusieurs fragments de manière à ce que seuls des groupes spécifiques de fragments permettent de reconstruire le secret. Ils ont été introduits parallèlement par A. Shamir [26] avec un point de vue polynomial, et par G. R. Blakley [2] du point de vue linéaire en 1979. Shamir justifiait l'intérêt de la théorie en donnant l'exemple de la conservation de clé : en effet, les SSS offrent un moyen de réduire le risque de perte de clés, tout en réduisant le risque de divulgation de clé après une intrusion. Les SSS sont toujours utilisés dans les services de stockage dans le cloud à l'heure actuelle [16]. D'autres applications des SSS ont été ensuite étudiées, en particulier pour le calcul multiparti sécurisé [9].

Du point de vue théorique, en 1981, R.J. McEliece et D.V. Sarwate montrent l'existence de codes correcteurs sous-jacents dans le SSS de Shamir [21]. Ce lien est ensuite particulièrement exploré dans la littérature. Deux constructions de SSS basés sur des codes correcteurs linéaires sont étudiées : la construction de Shamir, et celle de Massey. En 1993, J.L. Massey étudie une construction différente de celle de Shamir et introduit la notion des codes minimaux, importante dans sa construction [19].

Les SSS sont étroitement liés à la notion de structure d'accès. En 1991, E.F. Brickell et D.M. Davenport explicitent le lien entre structure d'accès et les matroïdes [4]. L'utilisation des matroïdes est particulièrement utile pour travailler sur les SSS idéaux et les taux d'informations de SSS, des notions qui ne sont pas explorées ici (voir [30], [4] et [24] pour plus de détails).

Le rapport est organisé de la manière suivante :

Dans la première partie du rapport, on rappelle quelques résultats de la théorie de l'information, en particulier sur l'entropie et l'entropie conditionnelle. L'objectif de cette section est d'introduire des notions et des résultats utiles pour la définition des SSS dans la section 3, et de bâtir l'intuition autour de la notion d'entropie.

La deuxième partie présente des généralités sur les codes correcteurs d'erreurs, et présente les codes de Hamming et de Reed-Solomon. Cette partie suit le plan et reprend essentiellement les résultats du début du cours de D. Boucher [3]. Certaines parties du cours ne sont pas reprises (notamment celles sur les codes cycliques et BCH) car les notions qu'elles introduisent n'interviennent pas dans la suite du rapport. Les preuves des théorèmes 2.2 et 2.3 ont été modifiées, soit pour détailler certains aspects, soit pour mettre en exergue quelques faits. Les remarques et le corps de la section sont personnels.

La troisième partie introduit les SSS par quelques exemples (personnels sauf mention contraire). On définit ensuite les structures d'accès, ainsi que le modèle adversaire mixte [14]. On donne la définition d'un SSS (modèle entropique), puis on présente les matroïdes et le lien avec les SSS. Cette présentation est largement reprise de [24]. La preuve du théorème 3.1 a été reprise pour détailler certains aspects, et améliorer certaines formulations. On introduit les SSS linéaires à partir du SSS de Shamir, et on présente la construction de SSS linéaires de Shamir. On cite les résultats de [11] sur les structures d'accès de ces SSS. Le théorème 3.2 est une version corrigée de [11], et la preuve est personnelle, reprenant un lemme mentionné dans [24]. On introduit ensuite la construction de Massey à partir des matroïdes linéaires, et on présente les résultats de [11] sur les structures d'accès des SSS basés sur la construction de Massey. Le théorème 3.8 est une version corrigée de [11].

La dernière partie présente des applications importantes du SSS de Shamir et plus généralement des SSS, dont le calcul multiparti sécurisé et le stockage sécurisé.

1 Éléments de théorie de l'information

Dans cette partie, on rappelle quelques éléments de théorie de l'information qu'on utilisera dans la partie 3 de ce rapport pour les schémas de partage de secrets. Cette partie reprend les notions introduites par Shannon dans [28] et [27], et suit la présentation faite dans [24].

1.1 Entropie

Soit X une variable aléatoire à valeurs dans un ensemble fini $\mathscr{X}$, de loi p_X. On suppose sans perte de généralité que pour tout $x \in \mathscr{X}$, $p_X(x) > 0$.

Définition 1.1. On appelle **entropie de** X le réel

$$H(X) = \sum_{x \in \mathscr{X}} p_X(x) \log(\frac{1}{p_X(x)})$$

où log désigne le logarithme en base 2.

L'entropie de X ainsi définie permet intuitivement de quantifier la quantité d'information contenue dans X. On illustre ceci par deux exemples :

Exemple 1.1 (Illustration)**.** Posons $\mathscr{X} = \{Nord, Sud, Est, Ouest\}$. Supposons qu'un générateur aléatoire tire un élément de $\mathscr{X}$ au hasard selon une loi uniforme. On souhaite transmettre la suite aléatoire, on a donc besoin de trouver un moyen d'encoder l'information en alphabet binaire, de manière à minimiser la longueur des données à envoyer.

On voit assez rapidement qu'un encodage optimal est d'associer les valeurs 00, 01, 10, 11 et les éléments *Nord*, *Sud*, *Est*, *Ouest*. La taille moyenne attendue à être envoyée à chaque tirage est donc de 2. Parallèlement, on calcule que l'entropie d'un tirage est

$$\sum_{x \in \mathscr{X}} \frac{1}{4} \log(4) = 4\frac{1}{4} \log(4) = 2,$$

soit autant que l'entropie.

Exemple 1.2 (Cas non uniforme)**.** Posons maintenant $\mathscr{X} = \{A, B, C\}$ et supposons que le générateur aléatoire tire un A ou un B avec probabilité $\frac{1}{4}$, et C avec probabilité 1/2. $\mathscr{X}$ a trois éléments donc on aura nécessairement certains éléments encodés avec deux bits. De plus si un élément est encodé par 0 et un autre par 1, il y aura ambiguïté lors du décodage car le troisième élément sera encodé par des 1 et des 0. Une solution est donc d'encoder un élément par 0 et les autres par 10 et 11. L'élément C apparait le plus fréquemment donc c'est celui-ci qu'on encode par 0.

La taille moyenne d'encodage d'un tirage est alors

$$\frac{1}{4}.2 + \frac{1}{4}.2 + \frac{1}{2} = \frac{3}{2}.$$

On vérifie que c'est encore l'entropie d'un tirage (il s'agit du même calcul).

Remarque 1.1. Dans le second exemple, on a construit le codage de Huffman statique. Huffman a montré que ce codage est le codage le plus court en moyenne parmi les codages où chaque issue de

X est associée à un symbole statique, c'est-à-dire qui reste le même tout au long de l'encodage.

Dans ces exemples, l'entropie donne la borne inférieure du nombre de bits nécessaires pour représenter l'information. Shannon a démontré ce résultat dans le cas général. On voit donc bien l'intérêt d'utiliser l'entropie pour décrire la quantité d'information contenue dans une variable aléatoire.

Étudions quelques propriétés de l'entropie :

Propriété 1.1. Soit X une variable aléatoire à valeurs dans un ensemble fini $\mathscr{X}$, alors :

- $0 \leqslant H(X) \leqslant \log(\mathrm{card}(\mathscr{X}))$.
- $H(X) = 0$ si et seulement si X est presque surement constante.
- $H(X) = \log(\mathrm{card}(\mathscr{X}))$ si et seulement si X suit la loi uniforme sur X.

En cryptographie, l'entropie intervient souvent lorsqu'on veut vérifier la sécurité d'une primitive. Pour cela, on modélise le secret que l'on étudie par une variable aléatoire. Il peut par exemple s'agir d'une clé tirée au hasard selon une loi uniforme. L'entropie représente alors l'information de la variable aléatoire, c'est-à-dire l'information qu'il nous manque, l'incertitude sur le secret. Faire diminuer cette entropie revient à diminuer l'incertitude et donc diminuer la qualité du protocole. La propriété précédente nous montre que si l'entropie tombe à 0 le secret est révélé. On comprend également que la distribution uniforme lors du tirage aléatoire d'un secret est la distribution qui donne la meilleure incertitude sur le secret (seulement lors d'un tirage, il faut faire attention au paradoxe des anniversaires entre autres dans des protocoles complexes).

1.2 Entropie conditionnelle

Soit X une variable aléatoire à valeurs dans un ensemble fini $\mathscr{X}$, et Y une variable aléatoire à valeurs dans $\mathscr{Y}$, qui suivent la loi conjointe p_{XY} sur $\mathscr{X} \times \mathscr{Y}$. On nomme p_X et p_Y les lois marginales. Pour $y \in \mathscr{Y}$, on note $p_X(\cdot|y)$ la loi de X sachant $Y = y$, et on note $p_{X|Y}$ la loi de X sachant Y.

On notera $H(XY) = H((X,Y)) = \sum_{x \in \mathscr{X}, y \in \mathscr{Y}} p_{XY}(x,y) \log(\frac{1}{p_{XY}(x,y)})$.

Définition 1.2. Soit $y \in \mathscr{Y}$. On définit **l'entropie de X sachant Y=y** comme le réel

$$H(X|Y=y) = \sum_{x \in \mathscr{X}} p_X(x|y) \log(\frac{1}{p_X(x|y)}).$$

Définition 1.3. On définit **l'entropie de X sachant Y** comme le réel

$$H(X|Y) = \sum_{y \in \mathscr{Y}} p_Y(y) H(X|Y=y).$$

L'entropie de X sachant Y représente la quantité d'information ou d'incertitude restante pour X après détermination de Y. En particulier, la différence $H(X) - H(X|Y)$ représente la quantité d'information que Y apporte sur X.

Propriété 1.2. On a, conformément à l'intuition, les propriétés suivantes :

- $0 \leqslant H(X|Y) \leqslant H(X)$.
- $H(X|Y) = 0$ si et seulement si pour tout $y \in \mathscr{Y}$ il existe $x \in \mathscr{X}$ tel que $p_X(x|y) = 1$.
- $H(X|Y) = H(X)$ si et seulement si X et Y sont indépendantes.

— $H(XY) = H(Y) + H(X|Y) = H(X) + H(Y|X)$.

Dans le cadre de la cryptographie, dans les preuves de sécurité, l'entropie conditionnelle permet de quantifier le gain d'information qu'offre certains éléments précis, par exemple le chiffré, ou un couple clair-connu. En particulier, on ne peut retrouver X à partir de Y que si $H(X|Y) = 0$. Cependant, si $H(X|Y) \neq H(X)$ alors Y apporte de l'information sur X (on peut imaginer que la distribution de X sachant Y est modifiée), ce qui peut avoir des conséquences importantes, par exemple réduire le temps de calcul nécessaire pour casser le protocole.

Enfin, on a une dernière propriété :

Propriété 1.3. Soit Z une variable aléatoire à valeurs dans $\mathscr{Z}$ fini. Alors

$$H(X|YZ) \leqslant H(X|Y).$$

Cette propriété justifie que la connaissance de Y et Z apporte plus d'information sur X que la simple connaissance de Y.

2 Codes correcteurs d'erreurs

Avant de poursuivre sur les schémas de partages de secrets, on présente quelques éléments de la théorie des codes correcteurs. On verra dans la partie 3 que des notions introduites dans ce domaine interviennent dans les résultats concernant les schémas de partage de secrets linéaires.

Le modèle qu'on pose dans le cadre de l'étude des codes correcteurs est le suivant : deux entités A et B veulent communiquer sur un canal sujet à des turbulences. Ces turbulences peuvent créer des erreurs dans le message que B reçoit en provenance de A. L'objectif de l'utilisation de codes correcteurs d'erreurs est de détecter ces erreurs et de pouvoir les corriger sous certaines hypothèses, en limitant l'ajout de temps de calcul et la congestion du canal d'information. Pour cela, il faut disposer d'algorithme d'encodage et de décodage peu coûteux en temps, et limiter la quantité de données surnuméraires.

Plus formellement, le modèle employé est *binary symmetric channel*, ce qui signifie que la probabilité d'erreur est constante sur chaque caractère envoyé, indépendamment de la position. On pourra éventuellement mentionner en remarque le modèle *binary erasure channel*, où le receveur reçoit soit un caractère correctement, soit un indicateur d'erreur avec une certaine probabilité, encore une fois invariante. La différence est la connaissance de la position de l'erreur.

Cette partie est une reprise du cours de D. Boucher [3]. Elle s'appuie essentiellement sur les articles de recherche de R. W. Hamming [15] et de I. S. Reed et G. Solomon [25].

2.1 Codes correcteurs généraux

On se donne d'abord un alphabet $\mathscr{A}$ (i.e. un ensemble fini) de cardinal q, et deux entiers non nuls positifs k et n tels que $k \leqslant n$.

Définition 2.1. On appelle **code** $(n,k)_q$ un sous-ensemble C de $\mathscr{A}^n$ de cardinal q^k. On appelle n la **longueur** du code et k son **log-cardinal**.

Remarque 2.1. Tout code C $(n,k)_q$ peut-être vu comme l'image d'une injection ϕ de $\mathscr{A}^k$ dans $\mathscr{A}^n$.

L'idée des codes correcteurs d'erreurs est de transformer un mot sur l'alphabet $\mathscr{A}$ de longueur k (i.e. un élément de $\mathscr{A}^k$) en un mot de longueur n (i.e. un élément de $\mathscr{A}^n$) grâce à ϕ, puis de transmettre ce mot de longueur n. Certains mots de $\mathscr{A}^n$ ne sont pas dans C, donc si un tel mot est reçu, on sait qu'il y a eu une erreur lors de la transmission.

On peut transmettre un message de n'importe quelle taille en découpant le message en morceaux de taille k et en envoyant chaque morceau individuellement (en complétant éventuellement le message avec un caractère nul jusqu'à ce que sa longueur soit multiple de k).

Définition 2.2. Soit C un code $(n,k)_q$. On appelle **rendement** de C la valeur $R = \frac{k}{n}$.

Plus le rendement est faible, plus la quantité d'information à envoyer augmente, et plus le temps d'envoi du message initial augmente. Il faut faire attention au rendement lors du choix d'un code correcteur pour ne pas trop augmenter le temps d'envoi, la bande-passante et le délai de transmission.

Définition 2.3. On appelle **distance de Hamming** la fonction d de $\mathscr{A}^n \times \mathscr{A}^n \longrightarrow \mathbb{N}$ telle que

$$\forall x, y \in \mathscr{A}^n, d(x,y) = \operatorname{card}\left(\{i \in [\![1,n]\!] | x_i \neq y_i\}\right).$$

Propriété 2.1. La distance de Hamming est une distance.

Preuve

— La symétrie est immédiatement visible via la définition.

— Soit $x, y \in \mathscr{A}^n$. Si $x \neq y$, $\{i \in [\![1,n]\!] | x_i \neq y_i\}$ est non vide, donc $d(x,y) \neq 0$. Si $x = y$, $\{i \in [\![1,n]\!] | x_i \neq y_i\}$ est vide donc $d(x,y) = 0$.

— Soient $x, y, z \in \mathscr{A}^n$. $\{i \in [\![1,n]\!] | x_i \neq z_i\} \subset \{i \in [\![1,n]\!] | x_i \neq y_i\} \cup \{i \in [\![1,n]\!] | y_i \neq z_i\}$, donc

$$\begin{aligned} d(x,z) = \text{card}(\{i \in [\![1,n]\!] | x_i \neq z_i\}) &\leqslant \text{card}(\{i \in [\![1,n]\!] | x_i \neq y_i\} \cup \{i \in [\![1,n]\!] | y_i \neq z_i\}) \\ &\leqslant \text{card}(\{i \in [\![1,n]\!] | x_i \neq y_i\}) + \text{card}(\{i \in [\![1,n]\!] | y_i \neq z_i\}) \\ &= d(x,y) + d(x,z). \end{aligned}$$

Définition 2.4. On appelle **distance minimale de** C l'entier $d = \min_{x \neq y} d(x,y)$. À partir d'ici, on notera le code C $(n,k,d)_q$ au lieu de $(n,k)_q$ pour tenir compte de sa distance minimale.

Propriété 2.2. Soit C un code $(n,k,d)_q$. Soit $c \in C$ et $y \in \mathscr{A}^n$, tels que $d(c,y) \leqslant d-1$. Alors $y = c$ ou $y \notin C$.

Preuve

Supposons par l'absurde que $y \in C$ et $y \neq c$. Alors $d(c,y) < d = \min_{x \neq y} d(x,y) \leqslant d(c,y)$, ce qui est absurde.

Propriété 2.3. Soit C un code $(n,k,d)_q$. Soit $y \in \mathscr{A}^n$, alors il existe au plus un $c \in C$ tel que $d(c,y) \leqslant \lfloor \frac{d-1}{2} \rfloor$.

Preuve

Cette propriété découle simplement de l'inégalité triangulaire.

Définition 2.5. On appelle **capacité de correction** la valeur $\tau = \lfloor \frac{d-1}{2} \rfloor$.

Exemple 2.1 (Code à répétitions)**.** Soit un alphabet $\mathscr{A}$ de cardinal q et $n \leqslant 1$. On appelle le code à répétition de longueur n sur $\mathscr{A}$ le code

$$C = \{(m, ..., m) \in \mathscr{A}^n ; m \in \mathscr{A}\}.$$

La distance minimale du code est n (soit tous les caractères sont identiques, soit tous distincts). C'est donc un code $(n,1,n)_q$. Le rendement de ce code est $R = \frac{1}{n}$ et sa capacité de correction est $\tau = \lfloor \frac{n-1}{2} \rfloor$. Il est donc mauvais en rendement, mais on verra dans la suite qu'il est bon pour la capacité de correction.

Des algorithmes d'encodage et de décodage simples pour ce code sont naturels. Pour l'encodage, on découpe le message en caractères isolés, et pour chacun de ces caractères, on envoie n fois le caractère. Pour le décodage, après avoir reçu n caractères, on décode par le caractère qui apparait

le plus. Si il y a moins de τ erreurs, le décodage est juste.

Les deux dernières propriétés nous permettent de comprendre comment un code correcteur permet de corriger une erreur de transmission et sous quelle conditions. La propriété 2.3 explique que si moins de τ caractères ont été mal transmis, on peut corriger l'erreur en trouvant le mot de C le plus proche pour la distance de Hamming. La propriété 2.2 montre que s'il y a eu moins de $d-1$ erreurs de transmissions, on peut détecter qu'il y a eu des erreurs de transmission, mais pas forcément corriger correctement l'erreur.

Un des objectifs de la théorie des codes correcteurs est de trouver des codes ayant à la fois un rendement et une distance minimale élevés. On demande en plus à avoir des algorithmes de codage et décodage efficaces.

2.2 Codes correcteurs linéaires

Pour pouvoir simplifier la recherche de codes, on souhaite utiliser des structures algébriques usuelles. Pour cela, on va supposer que l'alphabet $\mathscr{A}$ est $\mathbb{F}_q$ le corps à q éléments, et on va étudier un type particulier de codes : les codes linéaires.

Définition 2.6. On dit que C est un **code linéaire** $[n,k,d]_q$ si C est un $\mathbb{F}_q$-sous-espace vectoriel de $\mathbb{F}_q^n$ de dimension k et de distance minimale d.

C est donc l'image de $\mathbb{F}_q^k$ par une application linéaire injective à valeurs dans $\mathbb{F}_q^n$. On appelle **matrice génératrice de** C toute matrice G représentant une telle fonction injective linéaire dans les bases canoniques de $\mathbb{F}_q^k$ et $\mathbb{F}_q^n$, i.e. telle que $C = \{u \cdot G; u \in \mathbb{F}_q^k\}$.

Des résultats classiques d'algèbre linéaire assurent que G est une matrice $k \times n$ sur $\mathbb{F}_q$ de rang k.

Remarque 2.2 (Conventions d'écritures). On choisit par convention de noter les vecteurs de $\mathbb{F}^l$ en ligne.

Remarque 2.3. Le code à répétitions sur $\mathbb{F}_q$ vu en exemple précédemment est un code $\mathbb{F}_q$-linéaire. Une matrice génératrice est $\begin{pmatrix}1 & 1 & \dots & 1\end{pmatrix}$.

Définition 2.7. Soit $x \in F_q^n$. On appelle **support de** x l'ensemble

$$\mathrm{Supp}(x) = \{i \in [\![1,n]\!] | x_i \neq 0\}.$$

Définition 2.8. On appelle **poids de Hamming** la fonction ω sur $\mathbb{F}_q^n$ telle que

$$\forall x \in \mathbb{F}_q^n, \omega(x) = \mathrm{card}(\mathrm{Supp}(x)).$$

Remarque 2.4. Soit $x, y \in \mathbb{F}_q^n, d(x,y) = \omega(x-y)$.

Propriété 2.4. Soit C un code linéaire $[n,k,d]_q$, alors

$$d = \min_{x \in C \setminus \{0\}} \omega(x).$$

Preuve

Soit $x, y \in \mathbb{F}_q^n$ tels que $d(x,y) = d$, alors $x - y \in C \setminus \{0\}$ car le code est linéaire, et $\omega(x-y) = d(x,y) = d$, donc $\min\limits_{x \in C \setminus \{0\}} \omega(x) \leqslant d$.

Soit $x \in C \setminus \{0\}$ alors $\omega(x) = d(x,0) \geqslant d$, car $0 \in C$. Donc $\min\limits_{x \in C \setminus \{0\}} \omega(x) \geqslant d$.

Par antisymétrie, $d = \min\limits_{x \in C \setminus \{0\}} \omega(x)$.

Théorème 2.1 (Borne de Singleton)**.** *Soit C un code linéaire $[n,k,d]_q$, alors*

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

Preuve

Supposons par l'absurde que tous les mots de C sont deux à deux distincts sur les $k-1$ premières coordonnées. On peut alors construire une bijection entre $\mathbb{F}_q^{k-1}$ et C, ce qui est absurde car $\text{card}(C) = q^k > q^{k-1} = \text{card}(\mathbb{F}_q^{k-1})$. Donc il existe deux mots de C distincts qui coïncident sur les $k-1$ premières coordonnées. Donc $d \leqslant n - (k-1) = n - k + 1$.

Définition 2.9. On appelle un **code linéaire MDS (Maximal Distance Separable)** un code linéaire $[n, k, n-k+1]_q$.

D'après ce qui a été vu dans la première sous-partie, les codes MDS offrent la meilleure capacité de correction parmi les codes linéaires de longueur et dimension fixées : ils sont donc optimaux de ce point de vue et potentiellement intéressants.

Exemple 2.2. Le code à répétitions de longueur n est MDS pour tout $n \geqslant 1$.

Faisons un récapitulatif des intérêts des codes linéaires vus pour l'instant. On a vu que les codes linéaires sont générés par des matrices. Cet aspect est intéressant car il garantit que l'objectif de facilité d'encodage est rempli. En effet, dans le cas standard, on doit stocker la table de correspondance entière entre $\mathscr{A}^k$ et C et faire une recherche dans une table de q^k éléments de longueur n. Dans le cas linéaire, on doit stocker une matrice $k \times n$ et faire un calcul matriciel en complexité $\mathscr{O}(n \times k)$. On a donc réussi à linéariser la complexité spatiale pour le paramètre k et également la complexité temporelle, cette fois au prix d'une multiplication par n. Pour les valeurs de k et n qu'on souhaite utiliser en pratique (au moins quelques centaines pour k), on y gagne systématiquement beaucoup.

De plus, on a donné une borne sur la distance minimale, dont on a vu qu'elle était atteinte. On a donc défini une forme d'optimalité de capacité de correction pour les codes linéaires. Cependant, on a toujours pas donné de manière de déterminer d efficacement, sans énumérer les $q^k - 1$ éléments non nuls de C. De plus, un moyen efficace de vérifier qu'un mot est dans le code linéaire serait également utile. On va donc étudier un outil qui a un intérêt pour répondre à ces deux attentes.

Définition 2.10. Soit C un code linéaire $[n,k,d]_q$. On définit le **code dual de** C comme

$$C^\perp = \{u \in \mathbb{F}_q^n | \forall v \in C, < u, v >= 0\}$$

où $< .,. >$ désigne le produit scalaire euclidien usuel.

Il découle de résultats classiques d'algèbre linéaire que $C^\perp$ est un code linéaire de

longueur n et de dimension $n-k$. Soit H une matrice génératrice du code $C^\perp$, on appelle H **matrice de contrôle de** C.

Propriété 2.5. Soit C un code $[n,k,d]_q$, et soit G une matrice génératrice de C.

— Soit $H \in \mathscr{M}_{n-k,n}(\mathbb{F}_q)$. Alors H est une matrice de contrôle de C si et seulement si $HG^T = 0$ et $\text{rang}(H) = n-k$.

— Soit H une matrice de contrôle de C et $c \in \mathbb{F}_q^n$, alors

$$c \in C \text{ si et seulement si } Hc^T = 0.$$

Preuve

— Soit $H \in \mathscr{M}_{n-k,n}(\mathbb{F}_q)$ telle que $HG^T = 0$ et $\text{rang}(H) = n-k$. Alors les $n-k$ lignes de H forment une base d'un sous-espace vectoriel E de $\mathbb{F}_q^n$ de dimension $n-k$, le rang de H. On va montrer que $E = C^\perp$.

Soit h un vecteur de $\mathbb{F}_q^n$ donné par une ligne de H, alors pour tout vecteur g de $\mathbb{F}_q^n$ donné par une ligne de G, on a $< h, g >= hg^T = 0$ car $HG^T = 0$. Puisque les lignes de G forment une base C et les lignes de H forment une base de E, on a par bilinéarité du produit scalaire :

$$\forall u \in E, \forall v \in C, < u, v >= 0.$$

Donc $E \subset C^\perp$, et puisque dim $E =$ dim $C^\perp$, alors $E = C^\perp$.

Réciproquement, soit H une matrice de contrôle de C, alors $\text{rang}(H) =$ dim $C^\perp = n-k$. Comme H génère $C^\perp$ et G génère C, alors comme vu plus haut, on a $HG^T = 0$.

— Soit $c \in \mathbb{F}_q^n$ alors $c \in C$ si et seulement si $c \in C^{\perp\perp}$ si et seulement si $\forall v \in C^\perp, < v, c >= 0$ si et seulement si $Hc^T = 0$, car H engendre $C^\perp$.

Définition 2.11. Soit $x \in \mathbb{F}_q^n$, on appelle **syndrome de x** la quantité $S(x) = Hx^T$.

La propriété 2.5 nous a montré que le syndrome permettait de déterminer simplement si un mot de $\mathbb{F}_q^n$ était dans un code C via un calcul matriciel, grâce aux matrices de contrôle. On peut également remarquer que les matrices de contrôles sont de nouveaux descripteurs de C, en supplément des matrices génératrices. En effet, les matrices de contrôle décrivent complètement le dual de C, qui permet de décrire entièrement C. Il est donc possible de définir un code via une matrice de contrôle, comme nous le verrons dans la prochaine sous-partie.

Il reste à voir les liens entre le dual et la distance minimale, qu'on formule dans ce théorème :

Théorème 2.2. *Soit C un code $[n,k,d]_q$, et H une matrice de contrôle de C. Alors d est le plus grand des entiers D tel que tout ensemble de $D-1$ colonnes de H soit linéairement indépendant.*

Preuve

On notera H_i la i-ème colonne de H pour $i \in [\![1, n]\!]$. On commence par montrer une équivalence.

Soit $p \in [\![1, n]\!]$ et $x \in \mathbb{F}_q^n$:

$$\left\{ \begin{array}{c} x \in C \\ \omega(x) = p \end{array} \right. ssi \left\{ \begin{array}{c} Hx^T = 0 \\ \omega(x) = p \end{array} \right. ssi \left\{ \begin{array}{c} Hx^T = 0 \\ x = (0, ..., 0, x_{i_1}, 0, .., 0, x_{i_2}, 0, ..., x_{i_p}, 0, ..., 0), x_{i_j} \neq 0 \end{array} \right.$$
$$ssi \left\{ \begin{array}{c} \left(H_{i_1} | \quad ... \quad | H_{i_p}\right) (x_{i_1}, ..., x_{i_p})^T = 0 \\ x = (0, ..., 0, x_{i_1}, 0, .., 0, x_{i_2}, 0, ..., x_{i_p}, 0, ..., 0), x_{i_j} \neq 0 \end{array} \right. .$$

Notons $D_{\max}$ le plus grand des entiers D tel que tout ensemble de $D-1$ colonnes de H soit linéairement indépendants.

Soit $x \in C$ tel que $\omega(x) = d$, alors d'après l'équivalence ci-dessus, en sélectionnant les colonnes $H_{i_1}, H_{i_2}, ..., H_{i_d}$ correspondant aux composantes non nulles de x, on trouve un ensemble de d colonnes de H non linéairement indépendantes. Donc $D_{\max} - 1 < d$, donc $D_{\max} \leqslant d$.

Supposons par l'absurde qu'il existe $H_{i_1}, H_{i_2}, ..., H_{i_{d-1}}$ un ensemble de colonnes de H linéairement dépendantes, alors il existe $x_{i_1}, x_{i_2}, ..., x_{i_{d-1}} \in \mathbb{F}_q$ tels que

$$\left(H_{i_1} | \quad ... \quad | H_{i_{d-1}}\right) (x_{i_1}, ..., x_{i_{d-1}})^T = 0.$$

D'après l'équivalence, on peut donc trouver un $x \in C$ de poids de Hamming $\omega(x) = d-1$, ce qui est contradictoire. Donc tout ensemble de $d-1$ colonnes de H est linéairement indépendant donc $d \leqslant D_{\max}$.

Donc $d = D_{\max}$.

On peut remarquer que nous n'avons pas formulé de propriétés concernant la rapidité du décodage de codes linéaires, qui était un des objectifs que nous avions formulé dans la première sous-partie. C'est parce qu'on ne connait pas d'algorithmes de décodage non exponentiels dans le cas général. Cependant, il existe plusieurs familles de codes linéaires pour lesquelles on connait un algorithme de décodage efficace. Nous allons étudier deux de ces familles dans les prochaines sous-parties.

2.3 Application : Code de Hamming binaire

Dans cette sous-partie, on montre qu'en utilisant les propriétés des codes linéaires vues dans la sous-partie précédente, on peut facilement construire une famille de codes correcteurs de capacité de correction 1. On montre également que cette famille de code correcteurs à un algorithme de décodage rapide.

On se place sur l'alphabet $\mathbb{F}_2$. On souhaite construire des codes de capacité de correction 1. On a vu que cela correspondait à avoir une matrice de contrôle dont toute paire de colonnes est linéairement indépendante. Donc les colonnes de la matrice de contrôle doivent simplement être distinctes deux à deux et non nulles. Puisqu'on sait que donner une matrice de contrôle est suffisante pour définir un code on va procéder ainsi.

Définition 2.12 ([15])**.** Soit $r > 1$ un entier. On définit **le code de Hamming binaire $\mathscr{H}_r$ de longueur** $2^r - 1$ le code dont une matrice de contrôle est :

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

c'est-à-dire dont les colonnes forment l'ensemble des vecteurs de $\mathbb{F}_2^r$, et telle que pour tout $j \in [\![1, 2^r - 1]\!]$, la j-ème colonne de H, H_j, soit la décomposition binaire de j (avec le bit de poids fort en tête de colonne).

Remarque 2.5. L'ordre des colonnes ne change pas le code construit. Cependant, cela permet d'utiliser un algorithme de décodage rapide, comme on va le voir ci-dessous.

Remarque 2.6. Les codes de Hamming sont les premiers codes historiques. Ils ont été inventés pour prévenir des erreurs lors de la transmission de données pour des calculs.

Propriété 2.6. Le code de Hamming $\mathscr{H}_r$ est un code $[2^r-1, 2^r-r-1, 3]_2$. Son rendement est $R = 1 - \frac{r}{2^r-1}$.

Preuve

Il faut d'abord montrer que le code est bien défini, i.e. que H est une matrice de contrôle. Il suffit de montrer que $\text{rang}(H) = r$. C'est évident car on peut en extraire la matrice carré de côté r :

$$\begin{pmatrix} 0 & 0 & 0 & \dots & 0 & 1 \\ 0 & 0 & 0 & \dots & 1 & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots & \vdots \\ 0 & 0 & 1 & \dots & 0 & 0 \\ 0 & 1 & 0 & \dots & 0 & 0 \\ 1 & 0 & 0 & \dots & 0 & 0 \end{pmatrix}$$

qui est évidemment de rang r.

La longueur de $\mathscr{H}_r$ est bien $2^r - 1$, donc sa dimension est bien $(2^r - 1) - \text{rang}(H) = 2^r - r - 1$. On calcule donc que $R = \frac{2^r-r-1}{2^r-1} = 1 - \frac{r}{2^r-1}$.

Enfin, $d \geqslant 3$ car toute paire de colonnes de H est linéairement indépendante. Puisque la somme des deux premières colonnes donne la troisième, on a bien $d = 3$.

On va maintenant détailler l'algorithme de décodage. Soit m le message reçu. Supposons que $m = c + e$ où $c \in C$ et $\omega(e) \leqslant 1$. Alors $Hm^T = Hc^T + He^T = He^T$. Si $e = 0$, on détecte bien qu'il n'y a pas d'erreur car le syndrome est nul. Si $e \neq 0$, alors il existe $j \in [\![1, 2^r - 1]\!]$ tel que $e = (\delta_{ij})_{i \in [\![1, 2^r-1]\!]}$. En particulier, $He^T = H_j$. On peut donc retrouver j car il s'agit de l'entier dont la décomposition binaire est donnée par He^T.

Remarque 2.7 (Binary erasure channel). Dans le modèle *binary erasure channel*, si le nombre de caractères effacés e est inférieur à $d - 1$, alors on peut retrouver le mot d'origine.

En effet, les mots que l'on peut construire en complétant ce qui a été reçu sont à distance inférieure à $d-1$ du mot envoyé. D'après la propriété 2.2, il existe donc un seul mot du code dans l'ensemble des mots candidats.

Pour décoder, on procède donc ainsi : si Hm^T est nul, on décide que $c = m$, sinon, Hm^T est la décomposition binaire d'un $j \in [\![1, 2^r - 1]\!]$, et on décide que $c = m - (\delta_{ij})_{i \in [\![1, 2^r-1]\!]} = m + (\delta_{ij})_{i \in [\![1, 2^r-1]\!]}$. Cet algorithme est correct si $\omega(e) \leqslant 1$ d'après le paragraphe précédent.

2.4 Codes de Reed-Solomon généralisés (GRS)

Dans la sous-partie précédente on a montré que les codes de Hamming sont relativement efficaces en temps, particulièrement quand r augmente. Cependant, la capacité de correction est limitée. Dans la suite, on va essayer de construire une famille de codes correcteurs MDS.

Dans la suite, on reprend comme alphabet un corps $\mathbb{F}_q$.

Définition 2.13 ([25])**.** Soit $n \in [\![1, q-1]\!]$ et $k \in [\![1, n]\!]$. Soit $\alpha_1, \alpha_2, ..., \alpha_n$ des éléments non nuls et distincts de $\mathbb{F}_q$, et $v_1, ..., v_n$ des éléments non nuls de $\mathbb{F}_q$. Un **code de Reed-Solomon généralisé (GRS)** est donné par une matrice de contrôle $H = V_\alpha^{(n-k)} \times \text{Diag}((v_1, ..., v_n))$ où

$$V_\alpha^{(n-k)} = \begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_n \\ \vdots & \vdots & ... & \vdots \\ \alpha_1^{n-k-1} & \alpha_2^{n-k-1} & ... & \alpha_n^{n-k-1} \end{pmatrix} \text{ et } \text{Diag}((v_1, ..., v_n)) = \begin{pmatrix} v_1 & 0 & ... & 0 \\ 0 & v_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & v_n \end{pmatrix}.$$

On appelle $\alpha_1, ..., \alpha_n$ les **localisateurs du code**.

Remarque 2.8. On voit qu'on demande $n \leqslant q-1$. En effet, on en a besoin pour trouver n éléments distincts dans $\mathbb{F}_q^\times$. Cela signifie que si on a besoin d'une capacité de correction τ, on doit avoir $2\tau+1 \leqslant n-k+1 \leqslant q-k$ soit $q \geqslant 2\tau+k+1$. Cela force à travailler dans des corps avec beaucoup d'éléments, qui sont moins faciles à manipuler.

Propriété 2.7. Un code GRS est $[n, k, n-k+1]_q$. Il est donc MDS.

Preuve

On peut supposer que $v_1, ..., v_n$ sont égaux à 1, car multiplier par un coefficient inversible ne change pas la dimension ou la distance minimale. On remarque que tout ensemble de $n-k$ colonnes de H forme une matrice de Vandermonde de localisateurs distincts. En particulier, cette matrice est inversible. Donc H est de rang $n-k$ et $d \geqslant n-k+1$. Donc le code est de dimension k et $d = n-k+1$.

Théorème 2.3. *Le dual d'un code GRS est GRS défini par les mêmes localisateurs.*

Preuve

Soit C un code GRS $[n, k, n-k+1]_q$ de localisateurs $\alpha_1, ..., \alpha_n \in F_q^\times$, et de multiplicateurs $v_1, ..., v_n \in \mathbb{F}_q^\times$. Ainsi, $H = V_\alpha^{(n-k)} \times \text{Diag}((v_1, ..., v_n))$ est une matrice de contrôle de C. On veut montrer qu'il existe $G = V_\alpha^{(k)} \times \text{Diag}((v'_1, ..., v'_n))$ génératrice de C où $v'_1, ..., v'_n \in \mathbb{F}_q^\times$.

On procède par analyse synthèse : on cherche G de la forme précédente telle que $GH^T = 0$ et G de rang k. On remarque que G est toujours de rang k.

Soit $i \in [\![1, k]\!]$ et $j \in [\![1, n-k]\!]$. Le coefficient à la ligne i colonne j de GH^T est :

$$\sum_{l=1}^{n} \alpha_l^{i+j-2} v_l v'_l.$$

Autrement dit, $GH^T = 0$ si et seulement si pour tout $m \in [\![1, n-1]\!]$, $\sum_{l=1}^n \alpha_l^{m-1} v_l v'_l = 0$, si et seulement si

$$V_\alpha^{(n-1)} \times \text{Diag}((v_1, ..., v_n)) \times \begin{pmatrix} v'_1 \\ \vdots \\ v'_n \end{pmatrix} = 0.$$

Or $\mathrm{Diag}((v_1,...,v_n))$ est inversible et $V_\alpha^{(n-1)}$ est une matrice à $n-1$ lignes et n colonnes, elle a donc un élément dans son noyau. Donc il existe $(v'_1,...,v'_n) \in \mathbb{F}_q^n$ non nul vérifiant l'égalité ci-dessus. Il reste à montrer que les composantes de ce vecteur sont non nulles. Mais si une composante est nulle (disons sans perte de généralité après une permutation des colonnes de $V_\alpha^{(n-1)} \times \mathrm{Diag}((v_1,...,v_n))$ la n-ième), alors

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{n-2} & \alpha_2^{n-2} & ... & \alpha_{n-1}^{n-2} \end{pmatrix} \times \begin{pmatrix} v_1 & 0 & ... & 0 \\ 0 & v_2 & ... & 0 \\ \vdots & \vdots & ... & \vdots \\ 0 & 0 & ... & v_{n-1} \end{pmatrix} \begin{pmatrix} v'_1 \\ \vdots \\ v'_{n-1} \end{pmatrix} = 0.$$

Mais la matrice de cette égalité est inversible, donc $(v'_1,...,v'_{n-1})$ est nul, donc $(v'_1,...,v'_n)$ aussi, ce qui est contradictoire avec le choix de $(v'_1,...,v'_n)$. Donc toutes les composantes sont non nulles.

Remarque 2.9 (Point de vue polynomial). Soit C un code GRS $[n,k,n-k+1]_q$. On a vu que son dual était GRS aussi, donc il est généré par une matrice G. Supposons pour l'instant que G est de la forme $G = V_\alpha^k$. Soit $c = \begin{pmatrix} c_0 & ... & c_{k-1} \end{pmatrix} \in \mathbb{F}_q^k$. On peut identifier ce vecteur avec le polynôme $P_c = \sum_{i=0}^{k-1} c_i X^i$. Le mot de C correspondant est alors

$$cG = \begin{pmatrix} P_c(\alpha_1) & ... & P_c(\alpha_n) \end{pmatrix}$$

. Décoder un mot de C (sans erreur) revient donc à faire une interpolation de Lagrange.

Remarque 2.10. Comme les codes de Hamming, les codes GRS possèdent un algorithme de décodage rapide, appelé algorithme de **Welch-Berlekamp**, que l'on ne développe pas ici.

Les codes GRS ont été très utilisés pour les appareils de stockage comme les CDs, DVDs ou disques durs.

3 Schémas de partages de secrets (SSS)

Dans cette section, on présente les schémas de partage de secrets (SSS), un champ d'étude de la cryptographie. Les SSS ont été introduits parallèlement par A. Shamir [26] et G. R. Blakley [2].Le problème que les SSS cherchent à résoudre est celui de partager un secret avec un ensemble de participants en contrôlant la manière dont les participants peuvent accéder à ce secret. Ceci a des applications pratiques, comme par exemple stocker une clé secrète de manière sécurisé en se prémunissant de défaillances techniques, ou contrôler l'accès à l'information (applications militaires, gouvernementales, et dans tout système hiérarchique). Les exemples qui suivent permettent de mieux comprendre ce que l'on cherche à faire avec les SSS.

3.1 Exemples introductifs

Dans cette partie, on va souvent devoir tirer des nombres au hasard. Par convention, on notera avec une lettre majuscule une variable aléatoire et avec une minuscule l'évaluation de cette variable aléatoire.

On va tout du long utiliser le lemme suivant :

> **Lemme 3.1.** Soit Y une variable aléatoire à valeurs dans $\mathbb{F}_q$, et U une variable aléatoire uniforme sur $\mathbb{F}_q$, indépendante de Y. Alors $Y + U$ suit une loi uniforme sur $\mathbb{F}_q$, indépendante de Y.

Exemple 3.1 (Cinq brigands)**.** On nomme cinq brigands $p_1, ..., p_5$. Ces brigands possèdent un butin qu'ils souhaitent protéger. Cependant, ils ne se font pas confiance et souhaitent donc que la protection puisse être levée uniquement si tous les brigands sont présents.

Une manière de faire ceci est d'utiliser une porte cadenassée. Les brigands utilisent cinq cadenas, chacun ayant une clé différente, et chaque brigand reçoit une des clés.

Une autre manière est d'utiliser un cadenas intelligent. Celui-ci tire au hasard cinq éléments dans un corps (de grand cardinal) $\mathbb{F}_q$, indépendamment suivant la loi uniforme, que l'on note $s_1, .., s_5$, puis envoie secrètement pour tout $i \in [\![1, 5]\!]$ s_i à p_i. Le cadenas ne se déverrouillera que si il reçoit $s := \sum_{i=1}^{5} s_i$. On peut vérifier que la variable aléatoire donnant la somme s est indépendante de tout groupe de variables aléatoires donnant les s_i, sauf le groupe les contenant toutes. Ainsi, la connaissance de certaines valeurs s_i (mais pas de toutes) n'apporte pas d'information sur s, donc on ne peut pas retrouver s (si $\mathbb{F}_q$ est suffisamment grand).

On a ici donné deux manières de protéger un secret (le butin) en divisant l'information parmi un nombre de participants. C'est l'intérêt des SSS qu'on cherche à exploiter.

On voit que les deux manières de procéder ont quelques points communs : même nombre de clés, nécessité de réunir le groupe pour ouvrir la porte. Cependant, il y a une différence fondamentale : la connaissance de certains s_i ne diminue pas la sécurité du butin, alors que posséder une clé diminue le nombre de serrures à forcer. La deuxième méthode assure donc plus de sécurité.

> *Remarque* 3.1. On peut modéliser mathématiquement la méthode des cadenas. Soient S une variable aléatoire quelconque à valeurs dans $\mathbb{F}_q$, et $C_1, ..., C_5$ des variables aléatoires uniformes sur $\mathbb{F}_q$ indépendantes de S et entre elles. Les variables C_i servent de cadenas. On expose publiquement $s + \sum_{i=1}^{5} c_i$, qui représente la porte cadenassée. Chaque brigand p_k pour $k \in [\![1, 5]\!]$ reçoit c_k, qui modélise la clé du cadenas c_k. On obtient le même fonctionnement qu'avec les clés, tout en supprimant la perte de sécurité qui vient avec la possession de clés.

Dans ce premier exemple, on a considéré un cas simple, où tous les brigands doivent être réunis pour récupérer le butin. Mais que se passe-t-il si un brigand perd sa clé, ou n'est plus en mesure de participer ? Le butin est alors irrécupérable. Pour éviter ce genre de soucis, il faut assouplir les règles pour accéder au butin, tout en les conservant suffisamment strictes pour se prémunir de trahisons.

Exemple 3.2 (n brigands avertis)**.** Soit n un entier positif, et $p_1, ..., p_n$ des brigands. Les brigands souhaitent que leur butin ne puisse être retrouvé que si au moins k brigands sont réunis, où $1 \leqslant k \leqslant n$.

Une solution utilisant des cadenas et des clés est plus difficile à mettre en place. En effet, on utilise $\binom{n}{n-(k-1)}$ cadenas. Chaque cadenas correspond à une partie $A \subset \{p_1, ..., p_n\}$ de cardinal $n-(k-1)$ et chaque participant dans A reçoit une clé de ce cadenas. Ainsi, si un groupe de k brigands se réunit, pour tout cadenas et $A \subset \{p_1, ..., p_n\}$ associé, il y a au plus $k-1$ de ces brigands dans $\{p_1, ..., p_n\} \setminus A$, donc un membre du groupe doit avoir la clé. A contrario, si un groupe de moins de $k-1$ brigands se réunit, notons le B, il existe un cadenas de partie A associée tel que $B \subset \{p_1, ..., p_n\} \setminus A$, et donc le groupe ne peut pas ouvrir ce cadenas.

Dans ce second exemple, on voit que si on souhaite que $k = \lfloor \frac{2n}{3} \rfloor$ alors le nombre de cadenas est $\binom{n}{n+1-\lfloor \frac{2n}{3} \rfloor} = \binom{n}{\lceil \frac{n}{3} \rceil + 1}$ et le nombre de clés est $(\lceil \frac{n}{3} \rceil + 1)\binom{n}{\lceil \frac{n}{3} \rceil + 1}$. Ce nombre croît rapidement avec n, et on peut voir que ce schéma est inefficace. Un autre objectif de la théorie des SSS est de trouver des schémas réduisant le nombre de clés de chaque participant.

Dans un dernier exemple, on va voir qu'on peut traiter des cas où tous les participants n'ont pas le même niveau d'autorisation.

Exemple 3.3 (Hiérarchie, d'après [30])**.** Dans une entreprise, on trouve deux types d'employés : des directeurs et de simples salariés. Le règlement de l'entreprise stipule que pour accéder à certains dossiers sécurisés, trois employés doivent collaborer, sauf dans le cas où deux directeurs voudraient accéder au dossier. Il y a donc un aspect de hiérarchie à prendre en compte. On note d le nombre de directeurs et w le nombre de simples salariés. On associe à chaque employé un nombre, entre 1 et w pour un salarié et entre $w+1$ et $w+d$ pour un directeur.

On va présenter un schéma qui permet de mettre en place ces conditions. Soit $\mathbb{F}_q$ un corps fini tel que $q \leqslant 1 + w + \frac{w(w-1)}{2} + d$. L'entreprise fixe publiquement des éléments $\alpha_1, ..., \alpha_w \in \mathbb{F}_q$ distincts non nuls, puis fixe $\alpha_{w+1}, ..., \alpha_{w+d}$, des éléments non nuls distincts entre eux, distincts des éléments $\alpha_1, ..., \alpha_w$ et distincts de tous les éléments $\alpha_i\alpha_j(\alpha_i + \alpha_j)^{-1}$ pour tout $1 \leqslant i < j \leqslant w$. On remarque qu'il est possible de trouver de tels éléments grâce à la condition sur q. Puis un tiers de confiance (le PDG de l'entreprise par exemple) tire au hasard $s, a, b \in \mathbb{F}_q$ secrètement selon une loi uniforme. Enfin, ce dernier envoie à tout directeur i sa part secrète $s_i = s + \alpha_i a$, et à tout salarié j sa part secrète $s_j = s + \alpha_j a + \alpha_j^2 b$. Enfin, les dossiers sécurisés sont chiffrés avec s.

On va vérifier que deux directeurs peuvent retrouver le secret. On remarque que si i, j désignent les numéros de deux directeurs, alors

$$\begin{pmatrix} s_i & s_j \end{pmatrix} = \begin{pmatrix} s & a \end{pmatrix} G$$

où $G = \begin{pmatrix} 1 & 1 \\ \alpha_i & \alpha_j \end{pmatrix}$ est une matrice de Vandermonde inversible car $\alpha_i \neq \alpha_j$. Donc les deux directeurs peuvent retrouver le secret en inversant G (qui est publique) et en calculant $\begin{pmatrix} s_i & s_j \end{pmatrix} G^{-1}$. Le cas de trois salariés fonctionne similairement.

Il faut vérifier qu'il est possible de retrouver le secret s si un directeur et deux salariés participent. Si i, j désignent les numéros de deux salariés et k désigne celui d'un directeur, alors leurs parts sont données par :

$$\begin{pmatrix} s_i & s_j & s_k \end{pmatrix} = \begin{pmatrix} s & a & b \end{pmatrix} G$$

où $G = \begin{pmatrix} 1 & 1 & 1 \\ \alpha_i & \alpha_j & \alpha_k \\ \alpha_i^2 & \alpha_j^2 & 0 \end{pmatrix}$. Le déterminant de G est $det(G) = (\alpha_k(\alpha_i + \alpha_j) - \alpha_i\alpha_j)(\alpha_i - \alpha_j) \neq 0$ d'après les hypothèses de construction des $(\alpha_l)_{l \in [\![1, w+d]\!]}$. Donc G est bien inversible, et le groupe $\{i, j, k\}$ peut retrouver le secret.

Il reste à montrer que un groupe qui ne contient pas trois employés ou deux directeurs ne retrouve pas d'information sur le secret. Soit i, j deux salariés, alors leurs parts sont calculées comme

$$s\begin{pmatrix}1 & 1\end{pmatrix} + \begin{pmatrix}a & b\end{pmatrix}\begin{pmatrix}\alpha_i & \alpha_j \\ \alpha_i^2 & \alpha_j^2\end{pmatrix} = s\begin{pmatrix}1 & 1\end{pmatrix} + \begin{pmatrix}a & b\end{pmatrix}\begin{pmatrix}1 & 1 \\ \alpha_i & \alpha_j\end{pmatrix}\begin{pmatrix}\alpha_i & 0 \\ 0 & \alpha_j\end{pmatrix}$$

ce qui revient au même que connaitre

$$s\begin{pmatrix}1 & 1\end{pmatrix}\begin{pmatrix}\alpha_i & 0 \\ 0 & \alpha_j\end{pmatrix}^{-1}\begin{pmatrix}1 & 1 \\ \alpha_i & \alpha_j\end{pmatrix}^{-1} + \begin{pmatrix}a & b\end{pmatrix}.$$

D'après le lemme, $\{i, j\}$ ne peut pas retrouver d'information sur le secret s. On procède deux mêmes pour les autres groupes n'ayant pas la permission de voir le secret.

On voit que dans ce cas précis, on a construit un SSS dont les parts secrètes des participants font la même taille que le secret lui-même (i.e. les participants ont une clé privée chacun), tout en mettant en place le système d'accès complexe demandé.

Pour terminer cette sous-partie introductive, on revient sur les aspects essentiels des SSS qu'on a mis en évidence dans les exemples précédents, tout en introduisant le vocabulaire technique pour faciliter le développement de l'intuition dans la suite. Au préalable, on doit déterminer un **ensemble de participants** (les brigands, les employés), ainsi qu'une **structure d'accès** (l'ensemble des groupes qui peuvent retrouver le secret, et l'ensemble de ceux qui ne peuvent rien retrouver). Un **tiers de confiance, ou croupier** (cadenas intelligent, PDG) tire un **secret** au hasard, et suit un protocole de distribution pour calculer et distribuer un **fragment** à chaque participant. Cette distribution peut éventuellement contenir la **diffusion ou le broadcast** d'éléments spécifiques à l'instance et publiques (par exemple les α_i dans l'exemple de l'entreprise, qui peuvent changer d'une application du protocole à l'autre). Enfin, les participants connaissent un protocole de recouvrement pour retrouver le secret de manière correcte en groupe si ce groupe est autorisé ou **qualifié**. Conformément au principe de Kerckhoff, seuls le secret tiré au hasard et les fragments sont supposés secrets.

3.2 Structures d'accès et modèle

Avant de définir les SSS formellement, on a besoin de définir des notions au préalable. Tout d'abord on se donne un entier $n \leqslant 0$ et un ensemble $P = \{1, ..., n\}$ de participants et p_0 que l'on nomme croupier.

On note dans cette sous-partie $Q = \{p_0\} \cup P$.

Définition 3.1. On appelle **une structure d'accès sur** P une paire (Γ, Δ) où $\Gamma, \Delta \subset 2^P$ sont des ensembles de parties de P disjoints. On appelle les éléments de Γ les **groupes qualifiés** et ceux de Δ les **groupes interdits**. On dit que cette structure est **monotone** si Γ est croissant et Δ est décroissant, i.e. :

— $\forall A \in \Gamma, \forall A' \in 2^P$, si $A \subset A'$ alors $A' \in \Gamma$.

— $\forall B \in \Delta, \forall B' \in 2^P$, si $B' \subset B$ alors $B' \in \Delta$.

Dans la suite, on supposera toujours une structure d'accès monotone. On supposera également que $\Delta \neq \emptyset$ et $\emptyset \notin \Gamma$, on dit que la structure d'accès est supposée **non-triviale**.

On note $[\Gamma^-]$ l'ensemble des éléments minimaux de Γ (i.e. tels qu'ils n'existent pas d'éléments de Γ qu'ils contiennent strictement), et $[\Delta^+]$ l'ensemble des éléments maximaux de Δ.

Remarque 3.2. L'hypothèse de monotonie de la structure d'accès correspond à l'intuition qu'on peut avoir : si a et b peuvent retrouver le secret alors a, b et c aussi. Inversement, si a et b ne

peuvent pas retrouver le secret ensemble, ils ne peuvent pas le retrouver séparément. On voit que dans les exemples, on a démontré les preuves d'accès ou de confidentialité sur les éléments de $[\Gamma^-]$ et $[\Delta^+]$ exclusivement.

Définition 3.2. On dit qu'une structure d'accès (Γ, Δ) sur P est **complète** si $\Delta = \Gamma^c$. Dans ce cas, on dénote la structure d'accès par Γ seulement.

Remarque 3.3. Dans la première sous-partie, on a toujours considéré que soit un groupe pouvait retrouver le secret, soit il n'avait accès à aucune information concernant celui-ci. Cela correspond à des contraintes de sécurité qu'on préfère imposer en pratique. Cependant, il est parfois intéressant d'étudier ce qu'il se passe dans la théorie si on lève cette règle.

Définition 3.3. Soit (Γ, Δ) une structure d'accès sur P. On appelle **noyau de** Γ

$$core\ \Gamma = \bigcup_{A \in [\Gamma^-]} A.$$

Il s'agit des participants qui participent effectivement à la structure d'accès. On dit que la structure est **connexe** si $P = core\ \Gamma$.

Définition 3.4. Soit (Γ, Δ) une structure d'accès sur P. On définit la **structure d'accès duale** $(\Gamma^\perp, \Delta^\perp)$:

— $\Gamma^\perp = \{A \in 2^P | A^c \in \Delta\}$.

— $\Delta^\perp = \{B \in 2^P | B^c \in \Gamma\}$.

Propriété 3.1. La structure d'accès duale est une structure d'accès, et $(\Gamma^\perp)^\perp = \Gamma$ et $(\Delta^\perp)^\perp = \Delta$.

Preuve

$A \in (\Gamma^\perp)^\perp$ si et seulement si $A^c \in \Delta^\perp$ si et seulement si $(A^c)^c \in \Gamma$ i.e. $A \in \Gamma$. On procède similairement pour $(\Delta^\perp)^\perp$.

On donne deux familles de structures d'accès très connues :

Définition 3.5 (Structures à seuil)**.** Soit $\mathscr{A} = (\Gamma, \Delta)$ une structure d'accès sur P et $1 \leqslant k \leqslant n$ un entier. On dit que $\mathscr{A}$ est une **structure à** (k, n)**-seuil** si $\Gamma = \{A \in 2^P | \text{card}(A) \geqslant k\}$ et $\Delta = \{B \in 2^P | \text{card}(B) \leqslant k-1\}$. C'est une structure complète.

Définition 3.6 (Structures rampes)**.** Soit $\mathscr{A} = (\Gamma, \Delta)$ une structure d'accès sur P et $1 \leqslant t < k \leqslant n$ un entier. On dit que $\mathscr{A}$ est une **structure** (t, k, n)**-rampe** si $\Gamma = \{A \in 2^P | \text{card}(A) \geqslant k\}$ et $\Delta = \{B \in 2^P | \text{card}(B) \leqslant t\}$. Notons qu'une structure à (k, n)-seuil est une structure $(k-1, k, n)$-rampe.

Remarque 3.4. On a vu les structures à seuil dans l'exemple 3.2. On verra un peu plus tard qu'il existe de bien meilleurs SSS pour ces structures.

Dans le cadre de la cryptographie, on doit également proposer un modèle contenant nos hypothèses de sécurité et une modélisation du comportement d'un éventuel adversaire.

Dans la littérature [14], on modélise souvent les adversaire par un ensemble $\Delta_A \subset \Delta$, appelé **structure adversaire**, qui représente les groupes de participants qui pourraient ne pas suivre correctement le protocole. Ces participants sont appelés **corrompus**. On représente les adversaires passifs, i.e. qui suivent le protocole mais peuvent chercher à colluder par l'ensemble Δ, qui est alors appelé **structure de confidentialité**. Ces participants sont appelés **curieux**. On peut supposer que l'adversaire est **statique**, i.e. qu'il choisit un ensemble $A \in \Delta_A$ et un ensemble $B \in \Delta$ tel que $A \cup B \in \Delta$ avant le début du partage, ou bien qu'il est **adaptatif**, i.e. qu'il peut choisir de corrompre un participant activement ou passivement à chaque étape du protocole de partage, tant que l'ensemble A des participants corrompus vérifie $A \in \Delta_A$ et que l'ensemble B de participants curieux vérifie $A \cup B \in \Delta$. Cette manière de modéliser d'éventuels opposants est appelée **modèle adversaire mixte**.

Dans la suite, on supposera que toute paire de participants possède un canal sécurisé permettant de communiquer. On supposera également qu'il existe un tel canal sécurisé entre le croupier p_0 et chaque participant (on utilise la modèle de communication *secure channel* [1][6]). Enfin, on suppose que $\Delta_A = \emptyset$, i.e. tous les participants suivent le protocole.

3.3 Matroides et SSS

Dans cette sous-partie, on présente une définition des SSS, appelée **définition entropique**, et leurs liens avec les matroïdes, en particulier les matroïdes entropiques.

On reprend les notations de la sous-partie 3.2 pour le début de la sous-partie.

Définition 3.7. Un **schéma de partage de secrets (SSS)** Σ sur Q est une collection $(S_i)_{i\in Q}$ de variables aléatoires discrètes telle que S_{p_0} est non constante. Pour $A \subset Q$, on note S_A le vecteur aléatoire $(S_i)_{i\in A}$.

Remarque 3.5. Le secret correspond à la variable S_{p_0} et les fragments correspondent aux $(S_i)_{i\in P}$.

On définit la fonction $h : 2^Q \longrightarrow \mathbb{R}$ telle que $\forall A \subset Q, h(A) = H(S_A)$ où H désigne l'entropie. On notera $h(A|B) = H(S_A|S_B)$ pour $A, B \subset Q$. On se permettra de confondre les singletons et les éléments de Q en paramètre de h.

Définition 3.8. On définit la structure d'accès d'un SSS Σ comme (Γ, Δ) où :
- $\Gamma = \{A \subset P | h(p_0|A) = 0\}$.
- $\Delta = \{B \subset P | h(p_0|B) = h(\{p_0\})\}$.

On vérifie simplement que Γ est croissant, Δ est décroissant et $\Gamma \cap \Delta = \emptyset$. Donc c'est bien une structure d'accès.

Remarque 3.6. Dans le cas où on veut rendre certaines informations publiques, on peut rajouter un participant dont le fragment est l'information publique et ne s'intéresser qu'aux parties contenant le participant rajouté.

Remarque 3.7. On voit que l'entropie conditionnelle, qu'on a introduit comme étant la quantité d'information qu'on possède sur une variable aléatoire sachant une autre, permet de formaliser la notion de "pouvoir retrouver le secret" et de "ne pas pouvoir retrouver le secret".

Remarque 3.8. Remarquons qu'en réalité, c'est la loi conjointe du vecteur aléatoire qui nous intéresse dans un SSS pour définir la structure d'accès, et pas les lois marginales. On peut trouver dans la littérature une définition utilisant la loi conjointe directement ([30]). Elle est équivalente. De même, on aurait pu définir de manière équivalente la structure d'accès en utilisant les notions d'indépendance et de probabilité conditionnelle, d'après ce qui a été énoncé dans la propriété 1.2.

Définition 3.9. On dit qu'un SSS est **parfait** si sa structure d'accès est complète.

Exemple 3.4. On pose $n = 3$. Soit $S = (S_\alpha, S_\beta)$ et $U = (U_\alpha, U_\beta)$ deux variables aléatoires de loi uniforme sur $\mathbb{F}_q^2$ où $\mathbb{F}_q$ est un corps fini. On suppose de plus que $S_\alpha, S_\beta, U_\alpha, U_\beta$ sont mutuellement indépendantes. On définit un SSS Σ sur Q par :

— $S_{p_0} = S$.
— $S_1 = S + U$.
— $S_2 = U_\alpha$.
— $S_3 = U_\beta$.

On peut calculer que :

— $h(p_0) = 2\log(q)$ car S_{p_0} suit la loi uniforme sur $\mathbb{F}_q^2$.
— $h(p_0|i) = 2\log(q), \forall i \in [\![1,3]\!]$, car S est indépendant des $S_i, \forall i \in [\![1,3]\!]$.
— $h(p_0|\{1,2\}) = h(p_0|\{1,3\}) = \log(q)$ car une coordonnée de S est fonction de la variable connue et l'autre est indépendante de la variable connue.
— $h(p_0|\{2,3\}) = 2\log(q)$ car S est indépendant de U donc de (S_1, S_2).
— $h(p_0|P) = 0$.

Donc la structure d'accès de Σ est (Γ, Δ) où $\Delta = \{\emptyset, \{1\}, \{2\}, \{3\}, \{2,3\}\}$ et $\Gamma = \{\{1,2,3\}\}$.

Exemple 3.5 (SSS pour toute structure d'accès complète)**.** Dans l'exemple introductif 3.2, on présente un SSS pour toute structure d'accès à seuil. On peut étendre le principe à toute structure d'accès complète. L'idée est d'interdire l'accès au secret à tous les groupes de Δ, en créant un "cadenas" spécifique à chacun de ces groupes, et en distribuant la "clé" à tous les autres participants. On peut se contenter des groupes de $[\Delta^+]$.

Soit (Γ, Δ) une structure d'accès. Pour tout $B \in [\Delta^+]$, on définit U_B, variable aléatoire uniforme mutuellement indépendantes sur un corps $\mathbb{F}_q$. On définit S une variable aléatoire uniforme sur $\mathbb{F}_q$, indépendante de $(U_B)_{B\in[\Delta^+]}$. On pose :

— $S_{p_0} = S$.
— Pour tout $i \in P$, $S_i = (S + \sum_{B\in[\Delta^+]} U_B, (U_A)_{A\in\{B\in[\Delta^+]|i\in B^c\}})$.

Si $B \in \Delta$, il existe $B' \in [\Delta^+]$ tel que $B \subset B'$. $U_{B'}$ est indépendante de $(S_i)_{i\in B}$, donc $h(p_0|B) = \log(q) = h(p_0)$ d'après le lemme introductif. Si $A \in \Gamma$, alors pour tout $B \in [\Delta^+]$, il existe $i \in A \backslash B$. Donc $h(p_0|A) \leqslant H(S|S + \sum_{B\in[\Delta^+]} U_B, (U_B)_{B\in[\Delta^+]}) = 0$. La structure d'accès de ce SSS est donc bien (Γ, Δ).

On remarque que ce SSS, comme le deuxième exemple introductif, est généralement peu pratique.

Exemple 3.6 (SSS pour toute structure d'accès)**.** Soit (Γ, Δ) une structure d'accès non nécessairement complète. On peut créer deux structures d'accès complètes découlant de (Γ, Δ) : $\alpha = (\Gamma, \Gamma^c)$ et $\beta = (\Delta^c, \Delta)$. On peut créer un SSS $(S_i^\alpha)_{i\in Q}$ de structure d'accès α et un autre $(S_i^\beta)_{i\in Q}$ de structure d'accès β, mutuellement indépendants l'un de l'autre. Par convention, on supposera que $S_{p_0}^\alpha$ et $S_{p_0}^\beta$ sont uniformes à valeurs dans le corps $\mathbb{F}_q$.

Le SSS $(S_i)_{i\in Q} = (S_i^\alpha, S_i^\beta)_{i\in Q}$ a pour structure d'accès (Γ, Δ). En effet, si $A \in \Gamma$ alors $A \in \Delta^c$, donc on peut retrouver les deux composantes du secret. Si $B \in \Delta$, alors $B \in \Gamma^c$, donc $H(S_{p_0}|S_B) = 2\log(q) = H(S_{p_0})$. Enfin, si $C \in 2^P \setminus (\Gamma \cup \Delta)$, le groupe C peut retrouver $S_{p_0}^\beta$ mais pas $S_{p_0}^\alpha$ donc $H(S_{p_0}|S_C) = \log(q)$.

Les schémas de partage de secrets sont définis par l'entropie, notamment à cause de ces liens avec l'indépendance aléatoire. De manière générale, la notion d'indépendance a un important rôle dans ce domaine. Il existe des objets mathématiques définis pour modéliser la notion d'indépendance : les matroïdes, introduites dans [32]. On présente ici quelques généralités sur ces objets. Ce paragraphe reprend le travail de C. Pabló [24], et utilise les résultats de [8], [23] et [31].

On abandonne désormais les notations de la sous-partie 3.2.

Définition 3.10. On appelle **polymatroïde** une paire (Q, f) où Q est un ensemble fini, appelé **fondation** de la polymatroïde, et $f : 2^Q \longrightarrow \mathbb{R}$, appelée **fonction de rang**, telle que :

- $f(\emptyset) = 0$.
- f est croissante : si $A \subset B \subset Q$, alors $f(A) \leqslant f(B)$.
- f est sous-modulaire : $f(A \cup B) + f(A \cap B) \leqslant f(A) + f(B)$.

Définition 3.11. Une **matroïde** M est une polymatroïde (Q, r) entière (i.e. r est à valeurs dans $\mathbb{N}$) telle que $r(A) \leqslant |A|$ pour tout $A \subset Q$.

On dit qu'une partie A de Q est **indépendante** si $r(A) = A$, sinon on dit qu'elle est **dépendante**. On appelle **base** une partie indépendante maximale pour l'inclusion, et **circuit** une partie dépendante minimale pour l'inclusion.

Exemple 3.7. Soit $\mathbb{F}$ un corps fini et soit E un $\mathbb{F}$-espace vectoriel. E est de cardinal fini, donc on peut définir une matroïde (E, r) où r donne le rang d'une famille de vecteurs (i.e. le rang de l'espace vectoriel engendré). On peut remarquer que les définitions de famille dépendante et de base de l'algèbre linéaire coïncide avec la définition pour les matroïdes.

On peut donner des caractérisations des matroïdes par les parties indépendantes, les bases et les circuits.

Propriété 3.2 (Caractérisation par les parties indépendantes)**.** Soit Q un ensemble fini, et $\mathscr{I} \subset 2^Q$. Alors $\mathscr{I}$ est l'ensemble des parties indépendantes d'une matroïde de fondation Q si et seulement si :

- $\emptyset \in \mathscr{I}$.
- Si $F \in \mathscr{I}$ et $F' \subset F$, alors $F' \in \mathscr{I}$.
- Si $F_1, F_2 \in \mathscr{I}$ et $|F_1| < |F_2|$, alors il existe $x \in F_2 \setminus F_1$ tel que $F_1 \cup \{x\} \in \mathscr{I}$. De plus, une telle matroïde est unique.

Propriété 3.3 (Caractérisation par les bases)**.** Soit Q un ensemble fini et $\mathscr{B} \subset 2^Q$, alors $\mathscr{B}$ est l'ensemble des bases d'une matroïde de fondation Q si et seulement si :

pour tout $B_A, b_2 \in \mathscr{B}$ et $x \in B_1 \setminus B_2$, il existe $y \in B_2 \setminus B_1$ tel que $(B_1 \setminus \{x\}) \cup y \in \mathscr{B}$.

De plus, une telle matroïde est unique.

Propriété 3.4 (Caractérisation par les circuits)**.** Soit Q un ensemble fini et $\mathscr{C} \subset 2^Q$, alors $\mathscr{C}$ est l'ensemble des circuits d'une matroïde de fondation Q si et seulement si :

- $\emptyset \notin \mathscr{C}$.
- $\mathscr{C}$ est une antichaine, i.e. $C_1 \not\subset C_2$ si $C_1, C_2 \in \mathscr{C}$ et $C_1 \neq C_2$.
- Si $C_1, C_2\mathscr{C}$ sont distincts et $x \in C_1 \cap C_2$, alors il existe $C_3 \in \mathscr{C}$ tel que $C_3 \subset (C_1 \cup C_2) \setminus \{x\}$.

De plus, une telle matroïde est unique.

Maintenant, on définit quelques opérations sur les polymatroïdes.

Définition 3.12. Soit $\mathscr{S}_1 = (Q, f_1)$ et $\mathscr{S}_2 = (Q, f_2)$ deux polymatroïdes de fondation Q, et $c \in \mathbb{R}_+^*$. On définit :

- $\mathscr{S}_1 + \mathscr{S}_2 = (Q, f_1 + f_2)$.
- $c\mathscr{S}_1 = (Q, cf_1)$. On dit que $c\mathscr{S}_1$ est un **multiple** de $\mathscr{S}_1$.

Définition 3.13. Soit M une matroïde de fondation Q et d'ensemble de bases $\mathscr{B}$. On définit M^* la **matroïde duale de** M comme la matroïde dont l'ensemble des bases est :

$$\mathscr{B}^* = \{B \subset Q | Q \setminus B \in \mathscr{B}\}.$$

Propriété 3.5. Soit $M = (Q, r)$ une matroïde. On a les propriétés suivantes :

- M^* est une matroïde.
- $(M^*)^* = M$.
- La fonction de rang de M^*, qu'on note r^*, est donnée par :

$$\forall A \subset Q, r^*(A) = |A| - r(Q) + r(Q \setminus A).$$

On voit qu'on retrouve beaucoup de propriétés connues des fonctions de rang des espaces vectoriels. Les matroïdes généralisent la notion d'indépendance linéaire. On définit ci-après la famille des matroïdes basées sur les relations d'indépendance linéaire.

Soit $\mathbb{K}$ un corps.

Définition 3.14. Soit (Q, f) une polymatroïde. On dit qu'elle est **$\mathbb{K}$-linéairement représentable, ou $\mathbb{K}$-linéaire** si il existe $(V_i)_{i \in Q}$ une famille d'espaces vectoriels telle que

$$\forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i.$$

Propriété 3.6. Soit (Q, f) une polymatroïde $\mathbb{K}$-linéaire. C'est une matroïde si et seulement si $\dim V_i \leqslant 1, \forall i \in Q$.

Remarque 3.9. On voit qu'il est équivalent de définir les matroïdes linéaires en utilisant des vecteurs comme éléments de la fondation Q.

Avec ces définitions, on fait immédiatement le lien entre les matroïdes linéaires et les notions d'indépendance linéaire de vecteurs dans un espace vectoriel.

Une autre forme d'indépendance bien connue est l'indépendance de variables aléatoires. On voit que cette notion permet encore de définir une famille de matroïdes.

Définition 3.15. Soit (Q, h) une polymatroïde, on dit qu'elle est **entropique** s'il existe $(S_i)_{i_Q}$ une famille de variables aléatoires discrètes telles que

$$h(\emptyset) = 0 \text{ et } \forall \emptyset \neq A \subset Q, h(A) = H(S_A).$$

On dit qu'une polymatroïde (Q, f) est **poly-entropique** s'il existe $c \in \mathbb{R}_+^*$ et (Q, h) une polymatroïde entropique tels que $(Q, f) = c(Q, h)$.

La propriété suivante montre l'existence d'une telle polymatroïde pour toute famille de variables aléatoires discrètes.

Propriété 3.7. Soit $(S_i)_{i \in Q}$ une famille de variables aléatoires discrètes, et $h : 2^Q \longrightarrow \mathbb{R}$ telle que :

- $h(\emptyset) = 0$.
- $\forall \emptyset \neq A \subset Q, h(A) = H(S_A)$. Alors h est la fonction de rang d'une polymatroïde sur Q.

Preuve

On a bien $h(\emptyset) = 0$, et la croissance est assurée par la propriété 1.3.

Si $A \subset B \subset Q$ alors

$$h(B) = H(S_B) = H(S_A S_{B\setminus A}) = H(S_A) + H(S_{B\setminus A}|S_A).$$

Donc :

$$\begin{aligned} h(A \cup B) &= H(S_A) + H(S_{B\setminus A}|S_A) \\ &\leqslant H(S_A) + H(S_{B\setminus A}|S_{A\cap B}) \\ &= H(S_A) + H(S_B) - H(S_{A\cap B}) \\ &= h(A) + h(B) - h(A \cap B) \end{aligned}$$

On peut remarquer que la définition des polymatroïdes entropiques fait intervenir des SSS. La propriété montre qu'en réalité à chaque SSS correspond une polymatroïde entropique. Cela a parfois de l'intérêt car cela permet d'utiliser la théorie des matroïdes pour faire des preuves d'indépendance (i.e. montrer qu'un groupe est interdit).

Pour pouvoir faire ceci, on doit définir la notion de structure d'accès d'une matroïde. On commence par définir la notion de rang conditionnel d'une polymatroïde, par analogie avec l'entropie.

Définition 3.16. Soit (Q, f) une polymatroïde, soit $X, Y \subset Q$, on définit

$$f(X|Y) = f(X \cup Y) - f(Y).$$

Propriété 3.8. La croissance du rang et sa sous-modularité implique que $f(X|Y) \geqslant f(X|Y \cup Z)$ et $f(X|Y) \geqslant 0$.

Définition 3.17. Soit $\mathscr{S} = (Q, f)$ une polymatroïde, et $P_0 \subset Q$ tel que $f(P_0) > 0$. On pose $P = Q \setminus P_0$. On définit la structure d'accès de $\mathscr{S}$ comme la paire (Γ, Δ) où :

— $\Gamma = \{A \subset P | f(P_0|A) = 0\}$.

— $\Delta = \{B \subset P | f(P_0|B) = f(P_0)\}$.

Remarque 3.10. La structure d'accès reste invariante pour les multiples.

Lorsque $Q = \{p_0\} \cup P$, la structure d'accès des polymatroïdes entropiques quand $P_0 = \{p_0\}$ coïncide avec la structure d'accès du SSS associé.

On voit maintenant un théorème qui donne un exemple de l'intérêt que peuvent avoir les matroïdes.

Théorème 3.1. *Toute polymatroïde linéaire sur un corps fini est poly-entropique.*

Preuve

Soit E un $\mathbb{K}$-espace vectoriel, (Q, f) une polymatroïde linéaire sur un corps fini et $(V_i)_{i \in Q}$ des sous-espaces vectoriels de E tels que $\forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i$. On restreint E à $\sum_{i \in Q} V_i$ pour pouvoir supposer E de dimension finie.

Définissons maintenant, pour tout $i \in Q$, $W_i = V_i^{\perp} \subset E^*$. Soit une famille d'espaces vectoriels $(E_i)_{i \in Q}$ et une application linéaire injective $\pi : E^* \longrightarrow \prod_{i \in Q} E_i$ tels que les applications induites $\pi_i : E^* \longrightarrow E_i$ soient surjectives, et que pour tout $i \in Q$, $W_i = \ker(\pi_i)$ (on peut par exemple considérer des projecteurs orthogonaux parallèles aux W_i). Soit U une variable aléatoire de loi uniforme sur E^*. Alors les $(\pi_i(U))_{i \in Q}$ forment une famille de variables aléatoires. On notera pour tout $i \in Q, S_i = \pi_i(U)$ et pour tout $A \subset Q, S_A = (S_i)_{i \in A}$ et $\forall x \in E^*, \pi_A(x) = (\pi_i(x))_{i \in A}$.

On peut facilement démontrer que

$$\forall A \subset Q, H(S_A) = \operatorname{rang}(\pi_A) \log |\mathbb{K}| = (\dim\ E^* - \dim\ \ker(\pi_A)) \log |\mathbb{K}|.$$

Soit (Q, h) la polymatroïde entropique associée aux S_i. On remarque que

$$\begin{aligned} \forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i = \dim\ E - \dim \left(\sum_{i \in A} V_i \right)^{\perp} = \dim\ E - \dim \bigcap_{i \in A} W_i = \dim\ E^* - \dim\ \ker(\pi_A) \\ = \frac{H(S_A)}{\log |\mathbb{K}|} = \frac{h(A)}{\log |\mathbb{K}|}. \end{aligned}$$

Ainsi, (Q, f) est bien poly-entropique.

Remarque 3.11. La famille de variable aléatoire construite dans la preuve est importante. On y fera référence dans la suite sous le nom de **famille de variable aléatoire $\mathbb{K}$-linéaire** associée à la polymatroïde (Q, f).

On a isolé une famille de polymatroïdes entropiques dont la structure d'accès est celle d'une polymatroïde linéaire, qui s'exprime donc à l'aide de l'algèbre linéaire. On s'intéresse donc aux SSS associés à ces matroïdes, qu'on appelle **SSS linéaires, ou LSSS**. On connait principalement deux constructions pour les LSSS, qu'on étudie dans la suite.

3.4 SSS linéaire : construction de Shamir

Cette construction est la première construction historique, introduite par Shamir [26]. On commence par présenter l'exemple original.

Soit $\mathbb{F}_q$ un corps fini et $1 \leqslant k \leqslant n \leqslant q-1$ deux entiers. On va trouver un SSS dont la structure d'accès est à (k, n)-seuil.

On tire le secret $s \in \mathbb{F}_q$ aléatoirement selon une loi uniforme. On tire également $c_1, ..., c_{k-1} \in \mathbb{F}_q$ aléatoirement selon la loi uniforme. On obtient un vecteur aléatoire dans $\mathbb{F}_q^k$ tiré selon la loi uniforme, que l'on associe au polynôme $\Pi = \sum_{i=1}^{k-1} c_1 X^i + s$ de $\mathbb{F}_q[X]$. On prend de manière quelconque $\alpha_1, ..., \alpha_n \in \mathbb{F}_q^\times$ qui sont connus de tous. Le fragment de i est alors $s_i = \Pi(\alpha_i)$.

Supposons qu'un groupe de j participants collude. Si $j \geqslant k$ alors d'après le théorème d'interpolation de Lagrange, on peut retrouver Π depuis j fragments (j évaluations d'un polynôme de degré inférieur à k). Si $j \leqslant k-1$, alors pour tout $c_0 \in F_q$, il existe q^{k-j-1} polynômes de degré inférieur à k passant par les j points et par c_0 en 0. Donc $\mathbb{P}(s = c_0) = \frac{q^{k-j-1}}{q^{k-j}} = \frac{1}{q}$. Donc l'entropie de s connaissant j fragments n'est pas modifiée. Donc la structure d'accès de ce SSS est un (k, n)-seuil.

On peut représenter le calcul des fragments de la manière suivante :

$$\begin{pmatrix} s & c_1 & ... & c_{k-1} \end{pmatrix} G = \begin{pmatrix} s_1 & s_2 & ... & s_n \end{pmatrix} \text{ où } G = \begin{pmatrix} 1 & 1 & ... & 1 \\ \alpha_1 & \alpha_2 & ... & \alpha_n \\ \vdots & \vdots & ... & \vdots \\ \alpha_1^{k-1} & \alpha_2^{k-1} & ... & \alpha_n^{k-1} \end{pmatrix}.$$

On s'intéresse à la généralisation de cette méthode où le secret s et les c_i sont tirés au sort, et les fragments sont générés par la multiplication par une matrice $G \in \mathscr{M}_{k,n}(\mathbb{F}_q)$ quelconque. On doit supposer G injective : en effet, si G n'est pas injective, plusieurs vecteurs antécédents (et donc plusieurs secrets potentiellement) peuvent générer le même vecteur de fragments.

On peut voir G comme la matrice génératrice d'un code correcteur [21]. Cela a même beaucoup de sens, puisque retrouver le vecteur $\begin{pmatrix} s & c_1 & ... & c_{k-1} \end{pmatrix}$ à partir de certains fragments $s_{i_1}, ..., s_{i_l}$ revient à essayer de retrouver le mot d'un code en connaissant certaines composantes de manière sûre, et ignorant les autres composantes. C'est le problème de *binary erasure channel*.

Remarque 3.12. Le SSS de Shamir utilise des codes de Reed-Solomon en particulier. On peut remarquer que dans ce contexte, le choix de l'alphabet borne le nombre de participants. Même s'il est possible en théorie de rajouter des participants après une distribution initiale, il reste que cette borne ne peut pas être dépassée. Le choix de q est donc important.

On présente donc la première construction de SSS basée sur les codes correcteurs d'erreurs :

Définition 3.18 (Construction de Shamir)**.** Soit C un code linéaire $[n, k, d]_q$ de matrice génératrice G où $2 \leqslant k \leqslant n$. On définit le SSS basé sur la construction de Shamir généré par G pour l'ensemble de participants $P = [\![1, n]\!]$ comme :

— S_{p_0} est une variable aléatoire uniforme sur $\mathbb{F}_q$, et est le secret du croupier.

— $C_1, ..., C_{k-1}$ sont des variables aléatoires indépendantes mutuellement et avec S_{p_0}.
— $\begin{pmatrix} S_1 & ... & S_n \end{pmatrix} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} G$ définit les fragments.

Remarque 3.13. Le SSS historique de Shamir est donc un cas particulier où C est le code dual d'un code GRS.

Remarque 3.14. On voit que le SSS a une matroide de la forme vue dans le théorème 3.1. En effet, le vecteur $\begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix}$ suit la loi uniforme sur F_q^k, et

$$\begin{pmatrix} S_{p_0} & S_1 & ... & S_n \end{pmatrix} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} (\varepsilon | G)$$

où ε est la colonne $\begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$. On peut vérifier que $(\varepsilon|G)$ est injective car G l'est. Ensuite, chaque colonne est surjective ou nulle (ce sont des formes linéaires). On peut donc voir $(\varepsilon|G)$ comme une application injective de $E = \mathbb{F}_q^k$ dans $\prod_{i \in [\![0,n]\!]} E_i$ où $E_i = \mathbb{F}_q$ si la colonne correspondante est non nulle, $E_i = \{0\}$ sinon.

On peut voir les espaces vectoriels V_i définissant la polymatroïde linéaire associée comme les $\mathbb{F}_q$-droites linéaires engendrées par les colonnes de $(\varepsilon|G)$.

Remarque 3.15. En pratique, les colonnes ne sont jamais nulles : cela correspond à avoir un fragment inutile.

Dans la suite, on notera $\varepsilon = \begin{pmatrix} 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$ et $(g_i)_{1 \leqslant i \leqslant n}$ les colonnes de G, matrice génératrice d'un code $[n, k, d]_q$. Les théorèmes présentés à partir d'ici et jusqu'à la fin de la prochaine sous partie, concernant les SSS linéaires sont repris de [11].

Théorème 3.2 (Structure d'accès de la construction de Shamir)**.** *La structure d'accès du SSS basé sur la construction de Shamir généré par G est complète et l'ensemble des groupes qualifiés est :*

$$\Gamma = \{A \subset P | \exists (x_1, ..., x_n) \in \mathbb{F}_q^n, \varepsilon = \sum_{i=1}^{n} x_i g_i \text{ et } Supp((x_1, ..., x_n)) \subset A\}.$$

Autrement dit, ce sont les sous-ensembles A de $[\![1, n]\!]$ tels qu'il existe une relation de dépendance linéaire entre ε et les $(g_i)_{i \in A}$.

On commence par montrer le lemme suivant :

Lemme 3.2. Soit $\mathbb{K}$ le corps fini de cardinal q, E un $\mathbb{K}$-espace vectoriel de dimension finie et $(V_i)_{i \in Q}$ une famille de sous-espaces vectoriels de E permettant de construire une polymatroïde linéaire (Q, f). Soit $(S_i)_{i \in Q}$ un SSS formant une famille de variables aléatoires $\mathbb{K}$-linéaire construit comme dans la preuve du théorème 3.1. Alors la structure d'accès de ce SSS est définie par :
— $\Delta = \{A \subset P | V_{p_0} \cap \left(\sum_{i \in A} V_i\right) = \{0\}\}$.

— $\Gamma = \{A \subset P | V_{p_0} \subset \sum_{i \in A} V_i\}$.

Preuve: lemme

On reprend les notations de la preuve du théorème 3.1. La structure d'accès du SSS est la même que celle de la polymatroïde entropique (Q, h). Puisque $h = \log(q)f$ où $\forall A \subset Q, f(A) = \dim \sum_{i \in A} V_i$, alors c'est également la même que celle de (Q, f).

Donc, en injectant l'expression de f dans la définition de la structure d'accès de (Q, f), on obtient :

— $\Delta = \{B \subset P | \dim \left(V_{p_0} + \sum_{i \in B} V_i\right) = \dim V_{p_0} + \dim \sum_{i \in B} V_i\}$.

— $\Gamma = \{A \subset P | \dim \left(V_{p_0} + \sum_{i \in A} V_i\right) = \dim \sum_{i \in A} V_i\}$.

Le lemme s'en déduit immédiatement.

Preuve: théorème

On applique le lemme précédent en rappelant que dans ce cas, $V_{p_0} = \text{Vect}(\varepsilon)$ et pour tout $i \in P$ $V_i = \text{Vect}(g_i)$. Puisque tous ces espaces sont de dimension 1, le lemme précédent nous assure que la structure est complète. Ainsi, pour $A \subset P$:

$$A \in \Gamma \text{ ssi } V_{p_0} \subset \sum_{i \in A} V_i \text{ ssi } \text{Vect}(\varepsilon) \subset \sum_{i \in A} \text{Vect}(g_i)$$
$$\text{ssi il existe } (x_i)_{i \in A} \text{ tel que } \varepsilon = \sum_{i \in A} x_i g_i.$$

Ceci termine la preuve du théorème.

Le théorème donne une condition nécessaire et suffisante pour retrouver le secret : si il existe $x_{i_1}, ..., x_{i_l} \in \mathbb{F}_q$ tels que $\sum_{j=1}^{l} x_{i_j} g_{i_j} = \varepsilon$, on peut retrouver le secret en remarquant que

$$S_{p_0} = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} . \varepsilon = \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} \left(\sum_{j=1}^{l} x_{i_j} g_{i_j}\right)$$
$$= \sum_{i=1}^{l} x_{i_j} \begin{pmatrix} S_{p_0} & C_1 & ... & C_{k-1} \end{pmatrix} . g_{i_j} = \sum_{j=1}^{l} x_{i_j} S_{i_j}.$$

Remarque 3.16. On voit, d'après le théorème, que la structure d'accès dépend du choix de G.

Dans la suite, on souhaite étudier les structures d'accès des SSS basés sur la construction de Shamir. On reprend les résultats de [12]. Pour cela on va devoir supposer que les codes sont MDS.

Théorème 3.3. *Supposons que C est MDS. Alors toute partie de k éléments au moins est qualifiée.*

Théorème 3.4. *Supposons que C est MDS, et qu'il existe $i_0 \in [\![1, n]\!], a \in \mathbb{F}_q^\times$ tel que $\varepsilon = a g_{i_0}$. Alors une partie est qualifiée si et seulement si elle est de cardinal au moins k ou elle contient i_0.*

On voit que ce type de SSS est utile pour instaurer un système hiérarchique, pour lequel un directeur doit connaître le secret et qu'on veut un seuil pour les autres participants. On peut formuler un théorème similaire, où deux fragments sont particuliers :

Théorème 3.5. *Supposons que C est un code MDS, et que ε est une combinaison linéaire de g_1 et g_2, mais pas de seulement g_1 ou g_2. Soit $A = \{i_1, ..., i_l\} \subset P$, alors A est qualifié si et seulement si un de ces cas se produit :*

- $l \geqslant k$.
- $l \leqslant k-1$ *et* $1, 2 \in A$.
- $l = k-1$, $rang(\varepsilon, g_{i_1}, ..., g_{i_l}) = k-1$ *et* $1, 2 \notin A$.

Encore une fois, on voit qu'on peut mettre en place deux hiérarchies de fragments. On a un dernier théorème en rapport avec les colonnes de G.

Théorème 3.6. *Supposons que C est un code MDS. Alors la structure d'accès du SSS basé sur G est un (k, n)-seuil si et seulement si ε n'est pas une combinaison linéaire de tout ensemble de $k-1$ colonnes de G.*

Remarque 3.17. C'est en particulier le cas dans le SSS historique de Shamir utilisant les matrices de Vandermonde.

On a supposé dans tous les théorèmes précédents que les codes étaient MDS : c'est parce que les duaux des codes MDS sont MDS d'après [10], ce qui permet de contrôler les relations d'indépendance linéaire des colonnes de G. On voit un théorème mettant ceci en évidence.

Théorème 3.7. *Soit G matrice génératrice d'un code C $[n, k, d]_q$. Notons $d^\perp$ la distance minimale de $C^\perp$. Si $A_1 \in [\Gamma^-]$ et $A_2 \in [\Gamma^-]$ alors*

$$|A_1 \cup A_2| - |A_1 \cap A_2| \geqslant d^\perp.$$

Il semble que la construction de Shamir soit utile pour construire des SSS dont la structure d'accès crée des hiérarchies. De plus, cette construction offre l'intérêt de pouvoir choisir le secret. Cependant, les structures d'accès sont difficiles à décrire, les théorèmes demandent des contraintes qui les rendent peu pratiques. La littérature s'est principalement orientée vers la deuxième construction.

3.5 SSS linéaire : construction de Massey

Cette construction est motivée par la construction des familles de variables aléatoires linéaires (vue dans la preuve du théorème 3.1). Elle a été développée par J.L. Massey dans [19] et [20]. On conserve toujours les mêmes notations pour n, k, Q et P.

Définition 3.19. Soit C un code $[n+1, k, d]_q$ de matrice génératrice $G = \begin{pmatrix} g_0 & g_1 & ... & g_n \end{pmatrix}$. Soit $U = \begin{pmatrix} U_1 & ... & U_k \end{pmatrix}$ une variable aléatoire de loi uniforme sur $\mathbb{F}_q^k$, alors

$$\begin{pmatrix} S_{p_0} & S_1 & ... & S_n \end{pmatrix} = U \cdot G.$$

On dit que $(S_i)_{i \in Q}$ est le SSS basé sur la construction de Massey généré par G.

Remarque 3.18. On voit d'après la remarque 3.14 que la construction de Shamir ressemble à celle de Massey dans le cas particulier où $g_0 = \varepsilon$. On voit le lien entre la construction et les polymatroïdes linéaires de la même manière.

On précise la structure d'accès liée à cette construction.

Théorème 3.8. *Soit $(S_i)_{i\in Q}$ un SSS basé sur la construction de Massey généré par une matrice G génératrice d'un code C. Sa structure d'accès est complète, et l'ensemble des groupes qualifiés est*

$$\Gamma = \{A \subset P | \exists (1, x_1, ..., x_n) \in C^\perp, Supp((x_1, ..., x_n)) \subset A\}.$$

Preuve

La preuve est presque identique à celle pour la construction de Shamir.

Remarque 3.19. Le théorème donne l'algorithme pour trouver le secret. Si $(1, x_1, ..., x_n) \in C^\perp$ alors :

$$g_0 + \sum_{i=1}^{n} x_i g_i = 0 \text{ i.e. } g_0 = \sum_{i=1}^{n} (-x_i) g_i.$$

Si on trouve $x_{i_1}, ..., x_{i_l}$ tels que $g_0 = \sum_{j=1}^{l} (-x_{i_j}) g_{i_j}$ alors

$$s = u.g_0 = \sum_{j=1}^{l} (-x_{i_j}) u.g_{i_j} = \sum_{j=1}^{l} (-x_{i_j}) s_{i_j}.$$

Remarque 3.20. Le théorème montre que le choix de G pour générer C n'importe pas, contrairement à la construction de Shamir. On pourra donc parler de SSS basé sur C pour la construction de Massey.

On fixe à partir d'ici les notations C et G introduites précédemment.
On peut voir que la distance minimale du code dual intervient toujours pour cette construction.

Théorème 3.9 ([12])**.** *Soit Γ l'ensemble des groupes qualifiés du SSS basé sur la construction de Massey basé sur C. Soit $d^\perp$ la distance minimale de $C^\perp$.*

- *Soit $A \subset P$, si $card(A) \leqslant d^\perp - 2$, alors $A \notin \Gamma$.*
- *Il existe $A \subset P$ tel que $A \in \Gamma$ et $card(A) = d^\perp$.*

Pour introduire le dernier théorème, on a besoin d'introduire des notions supplémentaires sur les codes.

Définition 3.20. On dit qu'un vecteur $u \in \mathbb{F}_q^n$ recouvre un autre vecteur $v \in \mathbb{F}_q^n$ et on note $v \preceq u$ si $\text{Supp}(v) \subset \text{Supp}(u)$. On note $v \prec u$ si $\text{Supp}(v) \subsetneq \text{Supp}(u)$.

Définition 3.21. Soit C un code linéaire.

- On dit que $u \in C$ est **minimal** si il ne recouvre que ses multiples parmi les mots de C (i.e. les αu pour $\alpha \in \mathbb{F}_q$).
- On dit que C est **minimal** si tous les mots de C sont minimaux.
- **Le problème de couverture d'un code correcteur d'erreurs linéaire** est le problème de trouver l'ensemble des mots minimaux de C.

On peut maintenant donner le dernier théorème concernant les structures d'accès des SSS basés sur la construction de Massey.

Théorème 3.10 ([13]). *Soit C un code $[n+1, k, d]_q$ et H une de ses matrices de contrôle dont on note les colonnes $h_0, ..., h_n$. Si $C^\perp$ est minimal, alors le SSS basé sur C pour la construction de Massey a en tout q^{n-k+1} éléments dans $[\Gamma^-]$. De plus :*

- *Si $d = 2$, alors pour tout $i \in [\![1, n]\!]$: soit h_i est multiple de h_0, auquel cas i est dans tous les groupes qualifiés minimaux, soit h_i n'est pas un multiple de h_0, auquel cas i est dans $(q-1)q^{n-k-2}$ groupes qualifiés minimaux.*
- *Si $d \geqslant 3$, pour tout $1 \leqslant t \leqslant d-2$ alors tout groupe de t participants est inclus dans $(q-1)q^{n-k-(t+1)}$ des groupes qualifiés minimaux.*

4 Applications

Dans cette partie, on présente quelques applications des schémas de partage de secrets. Un des intérêts principaux des SSS et notamment de celui de Shamir est l'application au calcul multiparti sécurisé, et c'est dans cette direction que la recherche est la plus active à l'heure actuelle. Les SSS sont également utilisés en pratique pour le stockage sécurisé. Enfin, on verra que les SSS permettent de se passer d'un tiers de confiance, notamment grâce aux liens avec le calcul multiparti sécurisé, ce qui permet d'utiliser les SSS dans des problèmes comme celui des généraux byzantins. On détaille ces éléments dans la suite.

4.1 Retour sur le modèle adversaire

Avant de présenter les applications, on revient sur la notion de modèle adversaire qui a été introduite dans la partie 3 mais pas réutilisée ensuite, et dont l'importance intervient lorsqu'on réfléchit à l'application d'un protocole. En effet, un objectif de la cryptographie est de poser des conditions et des garanties sur la sécurité de primitives. En pratique, un participant est techniquement libre et peut suivre ou ne pas suivre un protocole. Il est donc primordial d'anticiper les actions possibles des participants, et de poser des contraintes sur celles-ci pour garantir la sécurité du protocole. Cela peut passer par la vérification des informations envoyées par chaque participant par exemple. Le modèle adversaire sert à modéliser le comportement des participants, et exprimer clairement les hypothèses sur ce comportement, et permet d'éviter des incidents comme la mauvaise utilisation d'un protocole.

En guise d'exemple, on propose une attaque sur le SSS de Shamir, introduite dans [29]. On demande à un SSS de prendre en compte deux contraintes (au moins) de sécurité en pratique : la confidentialité, i.e. les groupes dans Δ ne retrouve pas le secret, et la correction, i.e. les groupes dans Γ retrouve le secret. L'attaque porte sur la deuxième contrainte.

Soit $2 \leqslant k \leqslant n \leqslant q-1$ des entiers. Le croupier tire au hasard un secret $s \in \mathbb{F}_q$ et un polynôme $\pi \in \mathbb{F}_q[X]$ de degré inférieur à $k-1$ et tel que $\pi(0) = s$. Il fixe publiquement n localisateurs distincts non nuls $\alpha_1, ..., \alpha_n \in \mathbb{F}_q$. Il distribue les fragments $s_i = \pi(\alpha_i)$ aux participants. Supposons désormais que k participants essayent de retrouver le secret avec un seul individu malveillant. On peut sans perte de généralité supposer qu'il s'agit de $1, ..., k$ et que k est malveillant. Le participant k peut calculer par interpolation de Lagrange un polynôme δ de degré inférieur à $k-1$, tel que $\delta(\alpha_i) = 0$ pour i inférieur à $k-1$, et de coefficient constant choisi égal à ϵ. Enfin, k partage $\delta(\alpha_k) + s_k$. Les groupe retrouve alors le polynôme $\pi + \delta$ de coefficient constant $s + \epsilon$. Les participants honnêtes ont donc un secret incorrect, et pire encore, k peut retrouver le secret car il connait ϵ. Le participant malhonnête est donc en position de stopper l'accès au secret, d'utiliser le secret pour son intérêt ou d'utiliser le secret à des fins d'extorsion ou de chantage (dans le cas où un autre participant ne peut pas intervenir). De manière générale, $n-k+1$ participants malhonnêtes sont suffisants pour compromettre totalement l'efficacité du protocole.

On peut proposer des SSS alternatifs dans le cas où on suppose que l'adversaire peut ne pas suivre le protocole [7]. Pour la suite, on continue de supposer, comme jusqu'à présent, que l'adversaire est passif et qu'il suit le protocole. Ainsi, le protocole de Shamir est bien sécurisé comme on l'a démontré.

4.2 Calcul multiparti sécurisé

On commence par définir le concept de calcul multiparti sécurisé, puis on montre comment les SSS peuvent être employés pour cette application. Cette sous-partie reprend la présentation [17].

Un des premiers problèmes pouvant relever du calcul multiparti sécurisé est le problème du millionnaire de Yao [33] : deux millionnaires veulent savoir qui possède la plus grande fortune des deux, sans révéler leur propre fortune. Plus généralement, l'objectif du calcul multiparti sécurisé est de calculer des fonctions booléennes ou arithmétiques dépendant de paramètres apportés par plusieurs participants de façon à trouver des résultats corrects pour chacun des participants tout en conservant les paramètres secrets. En général, on demande que cinq contraintes soient vérifiées :

— **Confidentialité** : aucun participant n'apprend plus que son entrée et son résultat.

— **Correction** : le résultat de chacun des participant est correct.

— **Indépendance des entrées** : chaque participant choisit une entrée indépendante de celles des autres participant.

— **Garantie de service**(optionnelle) : chaque participant reçoit son résultat.

— **Justice** : les participants malhonnêtes ne peuvent pas recevoir un résultat si les participants honnêtes ne reçoivent rien.

Le protocole doit agir comme une boîte noire imperméable dans laquelle chaque participant envoie son paramètre indépendamment des autres, et qui renvoie les résultats correspondants. En particulier, le protocole ne protège pas d'attaques portant sur le choix préalable de son paramètre, car il a lieu en dehors de la boîte. Le protocole doit juste servir de module sécurisé.

La définition standard actuelle [5] est la suivante : on définit un protocole idéal faisant intervenir un tiers de confiance incorruptible lié à tous les participant par un canal sécurisé (pour la confidentialité et la capacité de service). Chaque participant envoie son paramètre au tiers de confiance. Le tiers de confiance calcule les résultats et les envoie à chaque participant respectivement. On dit qu'un protocole est **sécurisé** si pour tout ensemble d'entrées, il envoie les mêmes résultats que le protocole idéal et si pour toute attaque contre ce protocole, il existe une attaque contre le protocole idéal ayant les mêmes effets. On appelle ce modèle le *paradigme idéal/réel*. En particulier, un protocole sécurisé respecte les cinq contraintes précédentes. La définition de sécurité dépend évidemment du modèle adversaire choisi.

Canetti a montré dans [5] que la notion de sécurité est modulable : si un sous-protocole d'un protocole plus globale est sécurisé (en le prenant comme protocole principal) alors il est sécurisé comme sous-protocole (i.e. tout se passe comme si le tiers de confiance remplaçait l'exécution du sous-protocole). En particulier, si on trouve des protocoles sécurisés pour les fonctions élémentaires arithmétiques et booléennes, on peut en déduire des protocoles sécurisés pour toutes les fonctions arithmétiques et booléennes.

> *Remarque* 4.1. Lorsqu'on compose les modules, le résultat ne doit pas être évalué sinon tout le monde pourra connaître les étapes intermédiaire du calcul. Par exemple, pour trois paramètres x_1, x_2 et x_3, le calcul sécurisé de $x_1 + x_2 + x_3$ peut se décomposer comme calculer $x_1 + x_2$ puis ajouter x_3 au résultat. Cependant, la manière de calculer de façon sécurisée $x_1 + x_2$ ne doit pas permettre aux participants de retrouver immédiatement la valeur de $x_1 + x_2$, sinon le participant connaissant x_2 peut retrouver x_1, puis à la fin du calcul x_3, alors qu'il n'aurait pas dû en être capable. On verra dans la suite un manière de pouvoir manipuler $x_1 + x_2$ dans des calculs sans connaître la valeur de $x_1 + x_2$ en utilisant les SSS.

On va maintenant montrer comment on peut faire du calcul multiparti arithmétique sécurisé en utilisant le SSS de Shamir. Pour cela, on va avoir besoin de supposer que parmi n participants, strictement plus de $\frac{n}{2}$ participants sont honnêtes. On note $t = \frac{n-1}{2}$. Dans le cas des SSS, un calcul multiparti arithmétique sécurisé se déroule en trois étapes :

— *Partage initial* : chaque participant partage son secret avec les autres en utilisant le protocole de Shamir avec un polynôme de degré t π_i et des localisateurs $\alpha_1, ..., \alpha_n$ publiquement prédéterminés et communs à tous les participants. Remarquons que le secret ne peut être retrouvé par l'adversaire, car le seuil est de $t + 1 \geqslant \frac{n}{2}$.

— *Évaluation par étape* : à chaque étape du calcul sécurisé, les participants calculent de nouveaux fragments, de sorte que le nouveau polynôme retrouvé après l'interpolation de ces fragments (si elle était faite) soit de degré inférieur à t et de coefficient constant égal à une somme ou un produit de secrets déjà obtenus. On détaille ci-dessous les calculs nécessaires.

— *Reconstruction* : les participants interpolent les derniers fragments calculés, et obtiennent le résultat souhaité.

Remarque 4.2. Puisque les polynômes de degrés inférieurs à t ont leurs coefficients non constants aléatoires, l'adversaire ne peut pas prédire qu'un polynôme est de degré inférieur strictement à t. L'hypothèse que plus de $\frac{n}{2}$ participants sont honnêtes est donc suffisante pour garantir la confidentialité.

Il reste à détailler les processus à réaliser pour sommer ou multiplier des secrets.

Pour la somme, on suppose que les participants possèdent des fragments $a(\alpha_i)$ et $b(\alpha_i)$ permettant de reconstruire a et b deux polynômes de degrés inférieurs à t, dont les coefficients constants $s_a = a(0)$ et $s_b = b(0)$ sont secrets. Le participant i peut calculer $c(\alpha_i) = a(\alpha_i) + b(\alpha_i)$ un nouveau fragment d'un polynôme $c = a + b$, dont le coefficient constant $c(0) = a(0) + b(0) = s_a + s_b$. De plus, le degré de c est bien inférieur à t.

Pour le produit, on suppose encore que les participants possèdent des fragments $a(\alpha_i)$ et $b(\alpha_i)$ permettant de reconstruire a et b deux polynômes de degrés inférieurs à t, dont les coefficients constants sont $s_a = a(0)$ et $s_b = b(0)$. Le participant i peut alors calculer $c(\alpha_i) = a(\alpha_i)b(\alpha_i)$. Cette fois, le polynôme c est égal à ab. Il a donc bien comme coefficient constant $c(0) = a(0)b(0) = s_a s_b$ mais il est de degré inférieur à $2t$. Il va donc falloir utiliser une technique de réduction de degré introduite dans [9].

On suppose momentanément que les participants peuvent obtenir les fragments de deux polynômes R_t et R_{2t}, de degrés respectifs t et $2t$, dont les coefficients sont tous tirés aléatoirement et secrètement, et sont mutuellement indépendants à l'exception de la relation $r := R_t(0) = R_{2t}(0)$. Le participant i peut donc calculer $d(\alpha_i) = c(\alpha_i) - R_{2t}(\alpha_i)$ le fragment d'un polynôme d de degré $2t$ et tel que $d(0) = s_a s_b - r$. Les participants peuvent alors interpoler les fragments de d pour retrouver le polynôme d, car $2t + 1 \leqslant n$. Remarquons que c demeure entièrement secret car ses coefficients sont cachés par ceux de R_{2t}. Le participant i peut alors calculer un fragment $c'(\alpha_i) = R_t(\alpha_i) + d(0)$. Le polynôme c' est bien de degré inférieur à t, et de coefficient constant $c'(0) = R_t(0) + d(0) = s_a s_b$.

Il reste à expliquer comment calculer les fragments $R_t(\alpha_i)$ et $R_{2t}(\alpha_i)$. Pour tout $i \in [\![1, n]\!]$, le participant i tire au hasard $r_i \in \mathbb{F}_q$, puis tire au hasard et indépendamment les coefficients non constants d'un polynôme R_t^i de degré t tels que $R_t^i(0) = r_i$, puis de la même manière ceux d'un polynôme R_{2t}^i de degré $2t$ tel que $R_{2t}^i(0) = r_i$. Pour tout $j \in [\![1, n]\!]$, i envoie à j $R_t^i(\alpha_j)$ $R_{2t}^i(\alpha_j)$. Le participant j peut alors calculer $R_t(\alpha_j) = \sum_{i=1}^n R_t^i(\alpha_j)$ et $R_{2t}(\alpha_j) = \sum_{i=1}^n R_{2t}^i(\alpha_j)$, où $R_t = \sum_{i=1}^n R_t^i$ et $R_{2t} = \sum_{i=1}^n R_{2t}^i$. Au final, chaque participant à des fragments de la forme souhaitée.

4.3 Stockage sécurisé

La première application historique des SSS, suggérée par Shamir dans [26], est celle de la gestion de clés. Supposons qu'on utilise une clé secrète K pour des applications cryptographiques. Il se pose la question de comment conserver la clé. Simplement conserver la clé sans modification sur un disque dur ou dans un autre appareil de sauvegarde est problématique : en effet, ces appareils peuvent être détruits ou avoir des problèmes de fonctionnement, et la clé risque d'être perdue. Enregistrer la clé à plusieurs endroits n'est pas une bonne solution, car cela augmente les chances qu'un attaquant accède à la clé. Enregistrer la clé sans modification est dangereux en soi car un attaquant pourrait accéder à la zone mémoire où la clé est sauvegardée, et pourrait alors simplement lire la clé K, puisqu'elle n'est pas protégée. Il n'est pas utile de chiffrer K et d'enregistrer le chiffré, car le problème est transposé sur la clé utilisée pour chiffrer K.

La solution proposée par A. Shamir est la suivante : en utilisant un SSS de Shamir de paramètres n et k, on crée n fragments $(s_i)_{i \in [\![1,n]\!]}$ de K. On enregistre alors chacun des fragments sur des zones mémoires indépendantes (par exemple des serveurs différents). Ainsi, tant que moins de $n-k$ fragments sont perdus, on peut toujours retrouver K. D'autre part, un attaquant doit accéder à k fragments pour pouvoir retrouver K. Le SSS permet donc à la fois de diminuer les risques de perte et de vol de la clé. Cette technique est utilisée dans des services clouds [16].

En combinant cette manière de sauvegarder une clé et le calcul sécurisé, on peut créer des systèmes capables d'utiliser une clé qui n'apparaît nulle part dans la mémoire, ni lors des calculs, ni lors du

stockage.

4.4 Autres mentions

S. Micali [22] a proposé d'utiliser les SSS pour l'appliquer à la *cryptographie juste*, un pan de la cryptographie visant à développer des systèmes cryptographiques garantissant la confidentialité, tout en permettant à la justice d'accéder aux données chiffrées. Sa proposition est de rendre obligatoire le partage de clés via un SSS avec la justice. L'utilisation d'un SSS offre des garanties de sécurité à l'utilisateur du protocole, tandis que le partage permet à la justice d'accéder aux données chiffrées au besoin.

On a pu voir dans la sous-partie consacrée au calcul multiparti que les participants peuvent construire un polynôme de degré arbitraire aléatoire uniforme dont les coefficients sont secrets pour chacun des participants. Donc les participants sont capables de se passer de croupier pour la distribution de fragments. De manière générale, les SSS peuvent être utilisés pour se passer d'un tiers de confiance (dans le cas spécifique d'un calcul, c'est le principe du calcul multiparti sécurisé). On peut donc trouver des applications des SSS pour les problèmes liés à la décentralisation d'opérations, notamment au problème de l'accord byzantin (par exemple [18]) et à la technologie de la blockchain.

Références

[1] M. Ben-Or, S. Goldwasser et A. Wigderson : Completeness theorems for Non-Cryptographic Fault-Tolerant Distributed Computation. STOC'88, 1988.

[2] G. R. Blakley : Safeguarding cryptographic keys. *In Proc. National Computer Conference*, AFIPS, page 313–317, 1979.

[3] D. Boucher : Codes Correcteurs d'Erreurs, Aspects Algébriques. Cours de M1 à l'Université de Rennes 1,2020.

[4] E.F. Brickell et D.M. Davenport : On the Classification of Ideal Secret Sharing Schemes. *J. of Cryptology*, 4:123–134, 1991.

[5] R Canetti : Security and composition of multiparty cryptographic protocols. *J. Cryptol.*, 13(1), 2000.

[6] D. Chaum, C. Crepeau et I. Damgard : Multi-Party Unconditionally Secure Protokols. STOC'88, 1988.

[7] B. Chor, S. Goldwasser, S. Micali et B. Awerbuch : Verifiable secret sharing and achieving simultaneity in the presence of faults. *In Proceedings of the 26th IEEE Symposium on the Foundations of Computer Science (FOCS)*, pages 383 –395, 1985.

[8] T.M. Cover et J.A. Thomas : *Elements of Information Theory*. John Wiley & Sons, 1991.

[9] I. Damgård et J. Nielsen : Scalable and unconditionally secure multiparty computation. *In CRYPTO 2007*, pages 572–590. Springer.

[10] Philippe Delsarte : An algebraic approach to the association schemes of coding theory. *Philips Research Reports Supplement*, 1973.

[11] C. Ding : *Concise Encyclopedia of Coding Theory*, chapitre Secret Sharing with Linear Codes. Chapman and Hall/CRC, 2021.

[12] C. Ding et A. Renvall : The access structure of some secret-sharing schemes. *In Information Security and Privacy*, LNCS 1172, page 67–78. Springer, 1993.

[13] C. Ding et J. Yuan : Covering and secret sharing with linear codes. *In Discrete Mathematics and Theoretical Computer Science*, pages 11–25, LNCS 2732, 2003. Springer.

[14] S. Fehr et U. Maurer : Linear VSS and Distributed Commitments Based on Secret Sharing and Pairwise Checks. *In LNCS 2442*. CRYPTO'02, 2002.

[15] R. W. Hamming : Error Detecting and Error Correcting Codes. *The Bell System Technical Journal*, 29:147–160, 1950.

[16] T. Hansen et L. Larsen : Securing and managing secrets with hashicorp vault enterprise. AWS Blog, 2018. https ://aws.amazon.com/fr/blogs/apn/securing-and-managing-secrets-with-hashicorp-vault-enterprise/.

[17] Y. Lindell : Secure multiparty computation. *Communnications of the ACM*, 64(1):86–96, 2021.

[18] Jian Liu, Wenting Li, Ghassan O. Karame et N. Asokan : Scalable byzantine consensus via hardware-assisted secret sharing. *IEEE Transactions on Computers*, 68(1):139–151, Jan 2019.

[19] J.L. Massey : Minimal codewords and secret sharing. *Proc. 6th Joint Swedish-Russian Workshop on Information Theory*, pages 276–279, 1993.

[20] J.L. Massey : Some applications of coding theory in cryptography. *Cryptography, Codes and Ciphers : Cryptography and Coding IV*, pages 33–45, 1995.

[21] R.J. McEliece et D.V. Sarwate : On sharing secrets and reed-solomon codes. *Comm. ACM*, 1981.

[22] S. Micali : Fair public-key cryptosystems. *In Proc. of CRYPTO '92, Lecture Notes in Comput. Sci.*, volume 740, pages 113–138, 1993.

[23] J.G. Oxley : *Matroid Theory*. Oxford Science Publications. The Clarendon Press, Oxford University Press, New-York, 1992.

[24] C. Padró : Lecture Notes in Secret Sharing. Éléments du cours Applications of Combinatorics to Information-Theoretic Cryptography, Nanyang Technological University, Singapore, 2013.

[25] I. S. Reed et G. Solomon : Polynomial Codes Over Certain Finite Fields. *SIAM Journal of Applied Math.*, 8:300–304, 1960.

[26] A. Shamir : How to share a secret. *Communications of the ACM*, 1979.

[27] C. E. Shannon : A mathematical theory of communication. *The Bell System Technical Journal*, 27(3):379–423, 1948.

[28] C. E. Shannon : Communication theory of secrecy systems. *The Bell System Technical Journal*, 28(4):656–715, 1949.

[29] M. Tompa et H. Woll : How To Share a Secret with Cheaters. *Journal of Cryptography*, pages 133–138, 1988.

[30] M. E. van Dijk : *Secret key sharing and secret key generation*. Thèse de doctorat, Technische Universiteit Eindhoven, 1997.

[31] D.J.A Welsh : *Matroid Theory*. Academic Press, London, 1976.

[32] H. Whitney : On the Abstract Properties of Linear Dependence. *American J. of Math.*, 57:509–533, 1935.

[33] A. C. Yao : Protocols for secure computations. *In 23rd annual symposium on foundations of computer science (Chicago, Ill., 1982)*. IEEE, New York.