Codes with locality : constructions and applications to cryptographic protocols

Julien Lavauzelle

université PARIS-SACLAY

NNT: 2018SACLX082

Thèse de doctorat

Codes with locality: constructions and applications to cryptographic protocols

Thèse de doctorat de l'Université Paris-Saclay
préparée à l'École Polytechnique

École doctorale n°580, Sciences et Technologies
de l'Information et de la Communication (STIC)
Spécialité Informatique

Thèse présentée et soutenue à Palaiseau, le 30 novembre 2018, par

Julien Lavauzelle

Composition du jury:

Gilles Zémor Professeur, Institut de mathématiques de Bordeaux	Président
Maura Paterson *Reader*, Université Birbeck de Londres	Rapporteure
Christophe Ritzenthaler Professeur, Institut de recherche mathématique de Rennes	Rapporteur
Alain Couvreur Chargé de recherche, Inria Saclay	Examinateur
Camilla Hollanti *Professor*, Université d'Aalto	Examinatrice
Françoise Levy-dit-Vehel Maître de conférence HDR, ENSTA, Palaiseau	Directrice de thèse
Daniel Augot Directeur de recherche, Inria Saclay	Co-encadrant de thèse

Remerciements

En premier lieu, je voudrais remercier ma directrice de thèse Françoise Levy-dit-Vehel, sans qui ce travail aurait été impossible. Françoise, merci de m'avoir introduit au monde de la recherche il y a maintenant plus de quatre ans. Merci pour ta confiance, pour ton implication et pour tes conseils précis. Merci enfin pour ce temps que je sais si précieux, et que tu m'as généreusement donné.

Je ne manquerai pas de remercier tout autant Daniel Augot, pour avoir co-encadré cette thèse. Tes remarques pertinentes (et ta relecture du manuscrit sous la chaleur de juillet !) ont sans aucun doute rendu ce travail bien meilleur qu'il n'aurait été sans ton aide. Merci pour ta bonne humeur quotidienne, et pour ton accompagnement tout au long de cette thèse.

Je remercie également Maura Paterson et Christophe Ritzenthaler d'avoir accepté d'être rapporteur(e) de ce travail. *Thank you Maura, for writing a report on a PhD dissertation from someone you have probably never heard of before. I am very glad that you take part in my committee.* Je te remercie également, Christophe, pour tes ultimes conseils et corrections qui ont notablement amélioré la qualité du manuscrit.

J'adresse aussi un merci chaleureux à Alain Couvreur, Camilla Hollanti et Gilles Zémor pour avoir accepté de compléter le jury de cette thèse.

I am very grateful to Leo Storme for hosting me in Gent University for a few weeks in 2017. Our discussions on designs and finite geometries helped me a lot to clarify my ideas. Thank you also Marteen for sharing your office during these days. Finally, many thanks to Antonia Wachter-Zeh and Camilla Hollanti for inviting me last winter in Munich. I hope it will lead to fruitful collaborations in the future.

Tout au long de ma thèse j'ai eu le plaisir d'assister au groupe de travail sur la cryptographie à base de codes organisé par Jean-Pierre Tillich. Au-delà de l'intérêt scientifique, ce rendez-vous mensuel m'a permis de rencontrer nombre de chercheurs et doctorants de mon domaine ; je pense immédiatement à Kévin et à Thomas, mais aussi aux autres que je ne pourrais malheureusement pas citer de manière exhaustive. Pour rester dans l'environnement de l'équipe Secret, j'adresse aussi une pensée à Yann que j'ai rencontré au MPRI et avec qui j'ai toujours plaisir à échanger.

Pour sa disponibilité et son aide en géométrie au début de cette thèse (moi qui étais alors novice complet), je souhaite remercier une nouvelle fois Alain Couvreur.

Cette thèse a aussi été rythmée par les « retraites » du projet ANR Manta que j'ai trouvées aussi plaisantes que passionnantes. Merci notamment à Régis, Marc, Emmanuel, et une nouvelle fois à Alain et Christophe ainsi qu'à ceux que j'oublie, pour vos précieuses connaissances en géométrie. Merci aussi à Jade pour son enthousiasme lors de la recherche de propriétés locales pour ses codes.

Un projet aussi long qu'une thèse ne peut aboutir sans un environnement de travail idéal. Il y a plus de quatre ans, j'intégrais l'équipe GRACE/Crypto pour un premier stage. La convivialité, la générosité et la compétence des personnes que j'y ai rencontrées m'ont séduit immédiatement. Je voudrais donc remercier les membres permanents de l'équipe, Alain, Ben, Daniel, François et Françoise, d'avoir su instaurer une si bonne ambiance de travail !

Je voudrais aussi remercier les assistantes d'équipes qui se sont succédées durant ces quatre années, et qui ont toutes effectué un travail remarquable. Merci donc à Evelyne et Vanessa pour le LIX, et côté Inria, à Valérie, Tien, Emmanuelle, Jessica et Agustina.

J'ai été heureux de partager un premier bureau (et de belles discussions) avec Irene et Cécile. Ce premier stage a aussi été marqué par la bonne humeur communicative et la sympathie de Julia et Aurore. Je n'oublierai pas non plus Razvan, Cuong et Nicolas. Merci à David et Johan, autant pour les discussions autour de Sage que pour les quelques soirées jeux. Je pense aussi à Virgile, avec qui j'ai partagé un tennis au fin fond du Lot-et-Garonne, et qui m'a permis de reconsidérer autant le catch que la médecine chinoise. Enfin, bon courage à Matthieu pour son nouveau poste.

I would also like to thank Nick for these three years. By listening again and again your explanations on the rules of those nice Australian sports, I think I improved my English (a bit). Thank you also for reading my introduction during the last week of July.

D'autres personnes ont su apporter, lors de leur passage dans l'équipe, leur contribution à cette bonne ambiance ; je pense ici à Christian, Pierre, Luca, Philippe et William, et j'espère que ceux que j'ai une nouvelle fois oubliés m'excuseront.

Chaque été a vu de nouvelles têtes rafraîchir l'équipe. Merci donc à l'ensemble des stagiaires rencontrés, notamment Christopher, Benoît, Jean et Rémi, à qui je souhaite une bonne continuation. Une pensée plus particulière pour la dernière « promotion » avec qui je crois avoir noué des liens plus précieux : Mattia que j'ai hélas trop peu vu, Lucas avec qui j'ai pu partager quelques bières, Christophe pour ses chaussettes toujours idéalement assorties, Sarah que j'ai adoré exaspérer, ainsi qu'Isabella dont je suis peut-être le pire étudiant en Italien. Et puis, bon courage à Mathilde, nouvelle venue, et merci pour ton enthousiasme débordant. Je vous souhaite sincèrement à tous une bonne continuation.

Enfin, je ne sais comment remercier Élise avec qui j'ai partagé durant trois années le bureau 2039, ainsi que d'interminables discussions mathématiques, culinaires ou cinématographiques. Le chemin fut parfois rude (surtout celui menant à Palaiseau...) mais je crois que nous en voyons le bout. Merci pour ton soutien et ton amitié.

Plus loin d'ici, je voudrais remercier les charentais Anaïs, Amandine, Audrey, Kévin, Florian, et plus particulièrement Aurélie, que je vois si peu mais dont je sais l'affection et le support. Je n'oublie pas d'autres amitiés plus récentes : merci à Amandine, Christelle et Alex, à Corentin, Édouard, Florian et Tristan. Une dernière mais non moindre pensée pour Hugo et Léa, soutiens indispensables durant ces dernières années, et à Abel, leur tout nouveau rayon de soleil.

Mes derniers mots seront pour les membres de ma famille. Non seulement vous m'avez permis et encouragé à faire ce que je souhaitais, mais vous avez aussi été une source d'inspiration et de motivation inépuisable. Je vous remercie du fond du cœur pour votre amour et votre soutien, sans quoi je n'aurais pu réussir cette thèse. Merci infiniment.

Contents

Résumé

Nous présentons ici un résumé en français de la thèse. Par souci d'intelligibilité, nous proposons quelques éléments contextuels avant d'introduire nos différentes contributions scientifiques.

Contexte applicatif de la thèse

Le stockage de données en ligne, aussi connu sous le nom de « stockage sur le *cloud* », a connu un essor spectaculaire ces dernières années. Les utilisateurs de ce type de service trouvent pratique de pouvoir déposer, partager ou accéder à des fichiers, et ce de manière facile et rapide. Néanmoins, des problèmes de confidentialité sont récemment apparus concernant son usage massif : par exemple, on peut citer l'utilisation commerciale de données personnelles telles que les détails d'accès à des fichiers spécifiques. C'est dans ce contexte cryptographique que se place cette thèse. En un sens, le problème soulevé paraît ardu : on désire garder un contrôle fort sur des fichiers dont le stockage a été délégué à un système distant. Néanmoins, des modèles de sécurité et des protocoles ont été déployés au cours des années, afin de proposer certaines fonctionnalités. En voici deux exemples.

Supposons qu'un système de stockage distant détienne une série de fichiers $M = (M_1, \ldots, M_k)$. On pourrait se demander s'il est possible de récupérer un certain fichier M_i, pour $i \in \{1, \ldots, k\}$, en ne révélant aucune information sur i au système de stockage. Ce problème est connu sous la terminologie anglaise de *private information retrieval (PIR)*, que l'on traduira par *récupération confidentielle d'information*. Des protocoles efficaces ont été conçus dans le but de résoudre ce problème, bien qu'ils requièrent de borner le système de stockage d'une certaine manière (par exemple dans sa capacité de calcul, ou dans son contrôle de l'ensemble des fichiers).

Dans un autre cas d'application, supposons qu'un utilisateur veuille vérifier si le système de stockage détient véritablement les fichiers déposés. Si ces fichiers sont très volumineux, il est légitime de supposer que l'utilisateur voudrait effectuer cette vérification en utilisant peu de bande passante ; par exemple, une vérification consistant à télécharger les fichiers et en tester l'intégrité est considérée comme trop coûteuse. Des protocoles apportant une solution à cette problématique sont des *preuves de récupérabilité*, ou en anglais *proofs of retrievability (PoR)*.

Les protocoles résolvant les problèmes présentés ci-dessus peuvent être qualifiés de *cryptographiques*, dans le sens où ils supposent un adversaire potentiellement malicieux. En effet, dans un protocole de PIR, le système de stockage voudrait obtenir de l'information sur le fichier voulu par l'utilisateur. De façon similaire, dans un protocole de PoR, le serveur aimerait diminuer ses coûts de stockage en effaçant des parties de fichier, tout en laissant croire à l'utilisateur qu'il les détient encore.

Durant l'exécution des protocoles, il est aussi intéressant d'exiger qu'une faible quantité de bits soient échangée entre le client et le serveur. Plus généralement, la *complexité de communication* nécessaire à la réalisation d'une tâche a été beaucoup abordée en informatique. Par exemple, en théorie des codes, la correction d'une partie d'un mot en accédant à seulement quelques symboles de celui-ci est un problème étudié depuis plus de deux décennies. Des codes correcteurs proposant une telle fonctionnalité sont qualifiés de *localement corrigibles* (*locally correctable code*, ou *LCC* en anglais).

Cette thèse a pour objectif d'étudier ces codes à propriétés locales ainsi que leur application à la construction de protocoles cryptographiques présentés ci-dessus.

Codes localement corrigibles

Présentation. Les codes localement décodables et corrigibles ont été formellement définis par Katz et Trevisan [KT00] au début des années 2000. Un code $\mathcal{C} \subseteq \mathbb{F}_q^n$ est dit localement corrigible s'il existe un algorithme probabiliste LC, acceptant comme entrée un indice $i \in [1, n]$ où corriger un mot bruité $\boldsymbol{y} \in \mathbb{F}_q^n$, et ayant les propriétés suivantes. Premièrement, l'algorithme LC doit effectuer au plus ℓ requêtes à $\boldsymbol{y}$, où $\ell \ll n$ est un paramètre appelé la localité du code. Deuxièmement, pour un certain $\delta > 0$ fixé, lorsqu'il existe un mot $\boldsymbol{c} \in \mathcal{C}$ tel que la distance de Hamming entre $\boldsymbol{c}$ et $\boldsymbol{y}$ est plus petite que δn, l'algorithme LC doit renvoyer c_i avec grande probabilité. La quantité δ est appelée la *fraction d'erreurs admissible*.

Les codes localement corrigibles (LCC) ont été étudiés sous différents régimes de paramètres. La plupart des résultats se placent le régime à *localité constante*, typiquement $\ell = 2$ ou 3. Étant fixée une valeur de ℓ, le but est alors de construire un LCC de localité ℓ avec un taux d'information $R = \dim(\mathcal{C})/n$ aussi grand que possible, ou de donner des bornes sur R. D'autres travaux se sont focalisés sur le régime à *taux (d'information) constant*. Dans ce cas, on cherche des familles de LCCs dont la valeur asymptotique du taux est non nulle, et le rapport ℓ/n est aussi petit que possible. Ce deuxième régime est le plus pertinent pour des applications pratiques, c'est donc celui sur lequel nous nous sommes concentrés dans cette thèse.

Les codes de Reed-Muller à bas degré représentent une famille typique de LCCs à taux constant. Rappelons que ces codes sont constitués de mots d'évaluation sur $\mathbb{F}_q^m$ de polynômes multivariés $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ dont le degré total est borné par un certain entier $d \geq 0$. Si $d < q - 1$, alors la restriction d'un tel polynôme f à une droite affine L de $\mathbb{F}_q^m$ définit un polynôme *univarié* $f_{|L}$, lui aussi de degré $\leq d$. On peut ensuite remarquer que l'ensemble des mots d'évaluation de polynômes univariés de degré $\leq d$ constitue un code que l'on sait corriger (précisément un code de Reed-Solomon). Ainsi, il est possible de définir un algorithme de correction locale pour un code de Reed-Muller de bas degré d. Il se définit informellement comme suit : (i) tirer aléatoirement et uniformément une droite affine L passant par la coordonnée i du mot à corriger ; (ii) corriger $\boldsymbol{y}_{|L}$, la restriction du mot bruité aux coordonnées correspondant à la droite ; (iii) renvoyer la version corrigée de y_i.

Lorsque le paramètre m reste fixe et lorsque d et q croissent de manière proportionnelle (toujours sous la contrainte $d < q - 1$), la famille de codes de Reed-Muller obtenue possède asymptotiquement un taux d'information R non nul. Néanmoins, ce taux est borné par $1/m!$. Quelques travaux récents [KSY14, GKS13, HOW15] ont conduit à des LCCs à localité sous-linéaire en leur longueur, et dont le taux asymptotique peut être fixé arbitrairement proche de 1. Dans cette thèse, nous nous intéressons en particulier à la seconde construction [GKS13], introduite par Guo, Kopparty et Sudan.

L'idée de la construction de Guo *et al.* [GKS13] se base sur l'observation suivante : sous certaines contraintes arithmétiques, il existe des polynômes multivariés $f \in \mathbb{F}_q[\boldsymbol{X}]$ de degré strictement plus grand qu'un certain entier d, dont la restriction à n'importe quelle droite affine peut être décrite par un polynôme univarié de degré plus petit que d. En réunissant ces polynômes et ceux du code de Reed-Muller de degré d, on obtient un code $\mathcal{C}$ sur lequel l'algorithme de décodage local décrit ci-dessus fonctionne toujours. Le code $\mathcal{C}$ est précisément appelé *relèvement du code de Reed-Solomon de degré d*, que l'on abrégera par *relèvement de code* ou *code relevé*. Bien entendu, il contient le code de Reed-Muller de même degré, mais de manière surprenante, il atteint un taux arbitrairement grand pour certaines plages de paramètres. Une des contributions principales de la thèse réside dans la construction et l'analyse détaillée d'analogues de ces codes définis sur des espaces projectifs. Mais présentons dans un premier temps un travail concernant la construction de codes localement corrigibles à partir d'objets combinatoires appelés *block designs*.

Contribution : codes localement corrigibles à travers une généralisation des codes basés sur les designs. Un (block) design $\mathcal{D}$ est un couple $(X, \mathcal{B})$ d'ensembles finis non-vides, X étant appelé l'ensemble des *points*, tandis que $\mathcal{B}$ est un ensemble de sous-ensembles $\varnothing \neq B \subset X$ appelés *blocs*. Si tout ensemble de points de taille t appartient à un même nombre de blocs, alors on parle de t-design. Il est bien connu que l'on peut construire un code à partir de $\mathcal{D}$. Précisément, le code $\mathcal{C} \subseteq \mathbb{F}_q^X$ fondé sur le design $\mathcal{D} = (X, \mathcal{B})$ est constitué des mots $\boldsymbol{c} \in \mathbb{F}_q^X$ tels que pour tout $B \in \mathcal{B}$, l'équation de parité $\sum_{x \in X} c_x = 0$ est satisfaite. En d'autres termes, on impose que $\boldsymbol{c}_{|B}$ appartienne au code de parité, pour tout $B \in \mathcal{B}$.

Si $\mathcal{D}$ est un 2-design dont la taille de tous les blocs est ℓ, nous démontrons qu'un algorithme de correction locale *lisse*[1] de localité $\ell - 1$ peut être défini. Simplement, l'idée est de tirer aléatoirement un bloc B passant par $x \in X$, la position à corriger. Puis, on utilise l'équation de parité dont le support est $B \ni x$ afin de retrouver c_x.

Néanmoins, un tel algorithme ne résiste pas à la présence d'un nombre important d'erreurs sur le mot de code, car il requiert qu'aucun des $\ell - 1$ symboles lus ne soient erronés. Pour pallier cet inconvénient, nous proposons la généralisation suivante des codes basés sur les designs. Soit $\mathcal{D} = (X, \mathcal{B})$ un design ; on se donne également $\mathcal{L} = (\mathcal{L}_B, B \in \mathcal{B})$ une famille de codes $\mathcal{L}_B \subseteq \mathbb{F}_q^B$ indexés par $B \in \mathcal{B}$. Le code basé sur la paire $(\mathcal{D}, \mathcal{L})$ est alors défini comme :

$$\mathrm{Code}_q(\mathcal{D}, \mathcal{L}) = \{\boldsymbol{c} \in \mathbb{F}_q^X \mid \forall B \in \mathcal{B}, \boldsymbol{c}_{|B} \in \mathcal{L}_B\}\,.$$

On peut observer que, si $\mathcal{L}$ contient uniquement des codes de parité, on retrouve la définition initiale d'un code basé sur un design. En revanche, si chacun des codes $\mathcal{L}_B$ corrige une fraction constante d'erreurs, alors nous démontrons qu'il est possible d'obtenir des codes localement corrigibles admettant une grande fraction admissible d'erreurs δ.

Contribution : construction de relèvements de codes projectifs. Rappelons que les relèvement de codes de Guo *et al.* [GKS13] permettent d'obtenir une famille de codes localement corrigibles dont le taux d'information dépasse $1/2$. Ils sont formellement définis ainsi :

$$\mathrm{Lift}(\mathrm{RS}_q(k), m) = \{\mathrm{ev}_{\mathbb{A}^m}(f) \mid \text{pour tout droite affine } L \subset \mathbb{A}^m, \mathrm{ev}_{\mathbb{A}^1}(f_{|L}) \in \mathrm{RS}_q(k)\}\,,$$

où $\mathrm{RS}_q(k)$ représente le code de Reed-Solomon de dimension $k+1$, et $\mathrm{ev}_{\mathbb{A}^m}$ est l'application d'évaluation sur l'espace affine tout entier. Dans la thèse, nous étudions l'extension de

[1] c'est-à-dire, dont la répartition des requêtes est uniforme

cette construction à l'espace projectif $\mathbb{P}^m$. Pour cela, nous définissons proprement $\mathrm{ev}_{\mathbb{P}^m}$, une application d'évaluation de polynômes homogènes sur $\mathbb{P}^m$, ainsi qu'une famille de plongements $\mathbb{P}^1 \to \mathbb{P}^m$, notée $\mathrm{Emb}_{\mathbb{P}}(m)$, qui respecte le choix de l'évaluation effectué pour $\mathrm{ev}_{\mathbb{P}^m}$. Un *relèvement de code projectif* peut ainsi être défini :

$$\mathrm{Lift}(\mathrm{PRS}_q(k), m) \coloneqq \{\mathrm{ev}_{\mathbb{P}^m}(f) \mid \forall L \in \mathrm{Emb}_{\mathbb{P}}(m), \mathrm{ev}_{\mathbb{P}^1}(f \circ L) \in \mathrm{PRS}_q(k)\},$$

où $\mathrm{PRS}_q(k)$ désigne le code de Reed-Solomon projectif (aussi appelé code de Reed-Solomon doublement étendu) de dimension $k+1$.

Nous démontrons ensuite que les relèvements de codes projectifs admettent un algorithme de décodage analogue à celui des relèvements affines. En voici une version simplifiée, étant donnés un mot corrompu $\boldsymbol{y} \in \mathbb{F}_q^{\mathbb{P}^m}$ et une position de correction $\boldsymbol{x} \in \mathbb{P}^m$. D'abord, un plongement $L : \mathbb{P}^1 \to \mathbb{P}^m$ tel que $\boldsymbol{x} \in L(\mathbb{P}^1)$ est tiré aléatoirement. Puis, on effectue les $\ell = q+1$ requêtes sur les symboles $y_{\boldsymbol{i}}$, $\boldsymbol{i} \in L(\mathbb{P}^1)$. Enfin, on corrige le mot $\boldsymbol{y}_{|L(\mathbb{P}^1)}$ qui, par définition du relèvement $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$, appartient au code $\mathrm{PRS}_q(k)$ (ou à un code équivalant à celui-ci, et que l'on sait parfaitement expliciter). Une version formelle de cet algorithme est présentée dans le Chapitre 3, Algorithme 10.

Nous explorons ensuite les propriétés algébriques et combinatoires des relèvements de codes projectifs. Nous prouvons qu'ils admettent une base constituée d'évaluations de monômes, et nous étudions en détail la structure de l'ensemble des exposants de ces monômes, appelé *ensemble des degrés*. Un de nos résultats principaux réside dans la démonstration d'une relation récursive liant les ensembles des degrés de relèvements affines et projectifs. Plus précisément, si $\mathrm{PDeg}(m,k)$ (resp. $\mathrm{ADeg}(m,k)$) représente l'ensemble des degrés de $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ (resp. $\mathrm{Lift}(\mathrm{RS}_q(k), m)$), nous donnons une bijection explicite entre $\mathrm{PDeg}(m,k)$ et l'union disjointe $\mathrm{PDeg}(m-1,k) \cup \mathrm{ADeg}(m,k-1)$.

Cette caractérisation inductive est d'un intérêt premier. D'abord, elle fournit une manière pratique de calculer une base — et *a fortiori* la dimension — des relèvements de codes projectifs. Ensuite, elle permet d'obtenir des relations explicites entre les codes relevés (affines et projectifs) à travers les opérations de poinçonnement et de raccourcissement de code. Par exemple, le raccourcissement de $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ sur n'importe quel hyperplan projectif donne un code isomorphe à $\mathrm{Lift}(\mathrm{RS}_q(k-1), m)$. Plus généralement, on démontre l'existence de la suite exacte

$$0 \to \mathrm{Lift}(\mathrm{RS}_q(k-1), m) \to \mathrm{Lift}(\mathrm{PRS}_q(k), m) \xrightarrow{\pi} \mathrm{Lift}(\mathrm{PRS}_q(k), m-1) \to 0$$

où π est l'application de restriction sur n'importe quel hyperplan projectif de $\mathbb{P}^m$. Ce résultat n'est pas anodin, car des suites exactes très similaires sont connues pour les codes de Reed-Muller et pour les codes basés sur des designs géométriques.

Ainsi, en un certain sens nos résultats suggèrent que les relèvements de codes forment un lien entre des codes purement algébriques de type Reed-Muller et des codes purement combinatoires basés sur les designs. Nous appuyons cette idée en prouvant que les relèvements de codes peuvent être vus comme des codes $\mathrm{Code}_q(\mathcal{D}, \mathcal{L})$, où $\mathcal{D}$ est un design géométrique et $\mathcal{L}$ une famille de codes de Reed-Solomon (affine ou projectif selon le cas).

Nous justifions ensuite l'aspect pratique des relèvements de codes en étudiant certains de leurs grandeurs et objets caractéristiques. Notamment, nous prouvons leur caractère quasi-cyclique (ce qui permet de réduire les besoins de stockage de ces codes), et nous fournissons une famille d'ensembles d'information explicites de taille importante.

Pour terminer cette étude, nous proposons une analyse fine de l'ensemble des degrés d'un relèvement de code affine dans le cas $m = 2$, qui correspond aux codes de plus fort taux d'information. Nous soulignons l'aspect fractal d'une représentation de l'ensemble de leur

degrés, et nous procédons à un calcul précis de leur cardinal. Cela nous permet en particulier d'améliorer les connaissances sur la dimension K de $\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2^{e-c} - 1), 2)$, donnée uniquement comme une borne et un fait dans [GKS13], en prouvant qu'elle vaut *exactement*, pour $1 \leq c \leq e$,

$$K = 4^e \left(1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c + \frac{1}{2^e}\left(\frac{3^c - 1}{2^{c+2}}\right)\right).$$

Précisons que cette plage de paramètres est d'un intérêt particulier, car elle permet d'obtenir une famille de codes localement corrigibles asymptotiquement bons. Enfin, dans un contexte plus général de corps finis $\mathbb{F}_{p^e}$ de caractéristique quelconque, nous démontrons que lorsque $e \to \infty$, le taux asymptotique $\rho_\infty^{(p)}(c)$ de la famille de codes $\mathrm{Lift}(\mathrm{RS}_{p^e}(p^e - p^{e-c} - 1), 2)$ est :

$$\rho_\infty^{(p)}(c) = 1 - \left(1 + \frac{1}{p+2}\right)\left(\frac{1 + 1/p}{2}\right)^c + \frac{1}{p+2}\left(\frac{1}{p^2}\right)^c.$$

Récupération confidentielle d'information

Présentation. La problématique du PIR (récupération confidentielle d'information) a été formellement introduite par Chor, Goldreich, Kushilevitz et Sudan [CGKS95] en 1995, c'est-à-dire plusieurs années en amont du déploiement effectif des services de stockage en ligne. Ces auteurs se placent dans le contexte d'une base de données M constituée de k entrées (ou fichiers) de taille 1 bit. Ils prouvent que, lorsqu'un seul serveur stocke la base de données, un protocole de PIR qui ne laisse fuiter *aucune*[2] information sur l'indice i de l'entrée M_i désirée doit utiliser $\Omega(k)$ bits de communication. Bien entendu, il est impensable d'utiliser un protocole aussi inefficace, et deux solutions sont alors apparues. Dans un premier temps, Chor *et al.* [CGKS95, CKGS98] ont proposé l'utilisation de plusieurs serveurs qui ne communiquent pas tous entre eux, en préservant une sécurité du point de vue de la théorie de l'information. Une autre solution consiste à n'utiliser qu'un seul serveur, mais dégrade la sécurité en imposant « seulement » qu'il soit calculatoirement difficile de retrouver un bit d'information sur i. Dans un certain sens, la première solution restreint l'adversaire (serveur malicieux) dans sa connaissance des requêtes, tandis que la seconde le borne dans sa capacité de calcul. Dans toute la suite, on s'intéressera uniquement à la première approche.

Dans leur travail précurseur, Chor *et al.* [CGKS95] ont proposé une première construction de protocoles de PIR dont la complexité de communication est sous-linéaire en la taille de la base de données. Quelques années plus tard, Katz et Trevisan [KT00] ont démontré que l'algorithme de décodage d'un LCC *lisse* fournit un protocole de PIR dont la localité est proportionnelle à celle du code. De plus, au prix d'un précalcul de l'encodage de la base de données, les serveurs n'auront aucun calcul à effectuer durant la phase de récupération. Néanmoins, la redondance de stockage induite par l'encodage de la base de données peut être rédhibitoire, notamment lorsque le taux d'information du LCC est faible.

Dans le but de réduire les besoins de stockage de ce type de protocoles, Augot, Levy-dit-Vehel et Shikfa [ALS14] ont suggéré un partage de l'encodage de la base de données sur les différents serveurs, qui respecte la propriété de confidentialité de la phase de récupération. Leur construction se focalise sur la famille des codes à multiplicité [KSY14] (*multiplicity codes* en anglais), et a permis d'obtenir les premiers protocoles de PIR à communication sous-linéaire et redondance de stockage inférieure à 2. En réalité, l'idée d'Augot *et al.* peut être étendue à n'importe quel LCC dont le support peut être partitionné de telle sorte que les requêtes de l'algorithme de décodage local soient « transverses » à la partition. Dans cette thèse, nous

[2] un tel protocole étant qualifié de *sûr du point de vue de la théorie de l'information*

poursuivons cette approche en adoptant un point de vue plus général, par l'intermédiaire de *designs transversaux*.

Contribution : protocoles de PIR à base de designs transversaux. La plupart des constructions récentes de protocoles de PIR se concentrent sur l'obtention d'une complexité de communication optimale dans le sens serveur vers client. Cependant, la question du coût calculatoire de la réponse du serveur est souvent négligée, bien qu'elle représente un obstacle pour le déploiement effectif des protocoles de PIR. En particulier, il semble déraisonnable pour un serveur de devoir lire la base de donnée entière, lorsque la réponse à renvoyer est de la taille d'une entrée.

Dans cette thèse, nous proposons un schéma générique pour la construction de protocoles de PIR, qui prend particulièrement en compte la problématique de la complexité calculatoire. Plus précisément, les protocoles que nous construisons sont optimaux en rapport à leur complexité de communication, dans le sens où chaque serveur doit simplement lire et renvoyer une entrée de la base de données qu'il détient.

Notre construction est fondée sur des structures combinatoires appelées designs transversaux (en anglais, *transversal designs*). Ce sont des designs $(X, \mathcal{B})$ munis d'un ensemble de *groupes* $\mathcal{G}$ qui forme une partition de X, et tel que chaque paire non ordonnée $\{x_1, x_2\} \subset X$ est, ou bien contenue dans un groupe, ou bien contenue dans exactement λ blocs, pour un certain $\lambda \geq 1$. On impose aussi que tous les blocs aient même taille ℓ, et que tous les groupes aient même taille s. Un tel design est ainsi noté $\mathrm{TD}_\lambda(\ell, s)$. Le code $\mathcal{C} \subseteq \mathbb{F}_q^X$ issu du design $\mathrm{TD}_\lambda(s, \ell)$ admet donc un support X de taille $s\ell$, qui peut être partitionné en ℓ groupes $G_1, \ldots, G_\ell$, et tel que pour toute paire d'indices $\{x_1, x_2\} \subset X$ n'appartenant pas au même groupe, il existe une équation de parité pour $\mathcal{C}$ dont le support est de taille ℓ et contient $\{x_1, x_2\}$.

Donnons maintenant un aperçu de notre construction de protocole de PIR. Soit $(X, \mathcal{B}, \mathcal{G})$ un design transversal $\mathrm{TD}_\lambda(\ell, s)$, et $\mathcal{C} \subseteq \mathbb{F}_q^X$ le code associé, de dimension k. On considère ℓ serveurs $S_1, \ldots, S_\ell$ qui stockent un encodage $\boldsymbol{c} \in \mathcal{C}$ de la base de données M à k entrées de la manière suivante : pour $1 \leq i \leq \ell$, le serveur S_i détient $\boldsymbol{c}_{|G_i}$. La phase de récupération de l'entrée c_x, pour $x \in X$, se présente alors comme suit. D'abord, l'utilisateur choisit un bloc $B \in \mathcal{B}$ tel que $x \in \mathcal{B}$. Décrivons maintenant la requête adressée au serveur S_i, pour $i \in [1, \ell]$. Si $x \in G_i$, alors le serveur doit renvoyer aléatoirement l'un des symboles qu'il détient. Sinon, il doit renvoyer c_{y_i}, où y_i est l'unique point de $G_i \cap B$. À la réception des réponses des serveurs, l'utilisateur peut collecter le vecteur $(c_b : b \in B \setminus \{x\})$. Grâce à l'équation de parité de $\mathcal{C}$ dont le support est B, il peut donc retrouver c_x. Enfin, notons que la valeur de x reste confidentielle (du point de vue d'un seul serveur), car pour tout point $y_i \in X$ n'appartenant pas au même groupe que x, il existe un nombre constant λ de blocs contenant $\{x, y_i\}$.

La construction présentée ci-dessus admet, comme escompté, une complexité de calcul optimale pour chacun des serveurs S_i, $1 \leq i \leq \ell$. En effet, le serveur S_i ne doit lire et renvoyer qu'un unique symbole parmi les données $\boldsymbol{c}_{|G_i}$ qu'il détient. Par ailleurs, la complexité de communication du protocole est égale à ℓ fois la taille de l'entrée désirée, où l'on rappelle que ℓ est la taille des blocs du design $\mathrm{TD}_\lambda(\ell, s)$.

Pour un ℓ fixé, un objectif reste de trouver des designs $\mathrm{TD}_\lambda(\ell, s)$ dont le code associé $\mathcal{C}$ admet un taux d'information $\dim(\mathcal{C})/s\ell$ important. Ainsi, on pourra atténuer la redondance de stockage sur les serveurs. Dans cette optique, nous proposons plusieurs instances de designs transversaux qui fournissent des codes à haut taux d'information. Deux premières familles proviennent des relations d'incidence entre points, droites et hyperplans dans les espaces affine et projectifs. Une troisième famille provient d'une construction classique de designs

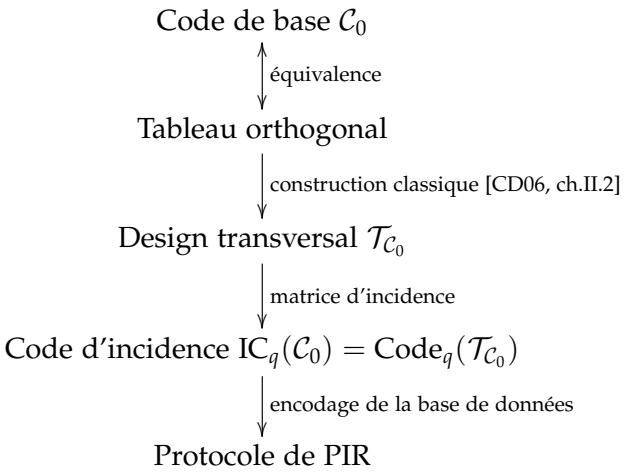


FIGURE 1 – Résumé de la construction d'un protocole de PIR à base de codes d'incidences.

transversaux à base de *tableaux orthogonaux* de force 2. Cela nous conduit à introduire la notion nouvelle de code d'incidences, dont les étapes de construction sont représentées en Figure 1. Par la suite, nous proposons une étude complète des codes d'incidence $\mathrm{IC}_q(\mathcal{C}_0) \coloneqq \mathrm{Code}_q(\mathcal{T}_{\mathcal{C}_0})$ provenant de codes $\mathcal{C}_0$ MDS et de dimension 2.

Finalement, une dernière famille de designs transversaux donnant des codes de grande dimension provient de codes $\mathcal{C}_0$ *divisibles*, c'est-à-dire, dont tous les mots ont un poids de Hamming divisible par un même entier. Nous démontrons que dans ce cas, le code $\mathcal{C} = \mathrm{IC}_q(\mathcal{C}_0)$ contient un sous-code de codimension 1 de son code dual ; son taux d'information est donc minoré approximativement par $1/2$.

Pour terminer, l'utilisation de tableaux orthogonaux de force $t = 2$ pour la construction de protocoles de PIR nous a amené à examiner le cas des forces $t > 2$. Nous montrons que dans cette situation, les protocoles obtenus restent confidentiels, même si jusqu'à $t - 1$ serveurs échangent l'information contenue dans leurs requêtes respectives. Pour clore cette partie, nous proposons et analysons les protocoles de PIR issus d'une famille de tableaux orthogonaux à force $t > 2$ quelconque.

Preuves de récupérabilité

Présentation. Sans rentrer dans les détails, une preuve de récupérabilité (PoR) fonctionne comme suit. Elle implique deux agents : le propriétaire d'un fichier M (aussi nommé le *client*, ou le *vérifieur*), et le système de stockage (que l'on appelle aussi le *serveur* ou le *prouveur*).

- Dans une première phase d'initialisation, le vérifieur effectue deux opérations : (i) il engendre, ou dérive de M, des données K gardées secrètes, et (ii) il encode son fichier M en un autre fichier W. Ensuite, le vérifieur transmet W au prouveur, supprime M localement et garde K secrètement.
- La seconde phase, dite de vérification, est le cœur de la PoR. C'est un protocole interactif dans lequel le vérifieur engendre un défi c puis l'envoie au prouveur, qui produit une réponse r en accord avec c et ce qu'il détient de W. Enfin, sur la connaissance de r, c et

K, le vérifieur estime s'il est toujours possible d'extraire M du prouveur. Notons que les données échangées durant cette phase doivent être de petite taille.
– Enfin, si le vérifieur est convaincu que son fichier est extractible grâce à la phase précédente, il peut lancer un protocole interactif d'extraction qui doit renvoyer M avec grande probabilité.

La première définition des preuves de récupérabilité (PoR) a été proposée par Juels et Kaliski [JK07]. Les auteurs donnent aussi une construction à base de *sentinelles*, c'est-à-dire de petites parties du fichier gardées secrètement et dont l'intégrité est vérifiée lors de la phase de vérification. D'autres schémas ont ensuite été introduits (par exemple [SW13, WWR$^+$11]), et Paterson, Stinson et Upadhyay ont fourni un cadre pour en analyser la sécurité [PSU13]. Plus précisément, leur idée est de modéliser par un mot (de taille potentiellement importante) la collection des réponses du prouveur à tous des défis possibles du vérifieur, de manière à ce que la procédure d'extraction corresponde à la correction d'un mot de code bruité.

Contribution : preuves de récupérabilité à partir de codes à propriétés locales. Dans cette thèse, nous proposons une construction générique de PoR dont la sécurité repose sur la vérification d'équations de parité sur un mot de code. Donnons ici l'essence de notre construction. Soit $\mathcal{C} \subseteq \mathbb{F}_q^n$ un code, et $\mathcal{H}$ un ensemble d'équations de parité pour $\mathcal{C}$ de poids $\ell \ll n$. Dans une phase d'initialisation, le vérifieur encode son fichier M en un mot de code $C \in \mathcal{C}$ et chiffre ensuite C symbole par symbole, en un mot $W \in \mathbb{F}_q^n$ à déposer sur le serveur. La phase de vérification consiste ensuite à tester des relations de parité (chiffrées) sur W. Précisément, le vérifieur choisit aléatoirement une équation de parité $h \in \mathcal{H}$, et demande au prouveur de lui fournir les symboles de W correspondant au support de h. Ensuite, le vérifieur déchiffre les symboles renvoyés par le prouveur, résultant en un mot a, et vérifie si l'équation $\sum_i h_i a_i = 0$ est satisfaite.

Dans notre travail, nous formalisons une version plus générale de l'idée présentée ci-dessus. Pour cela, nous introduisons la notion de *structure de vérification* pour un code, qui généralise l'utilisation d'une équation de parité pour la phase de vérification de la PoR. Nous aboutissons à un protocole de PoR dont les avantages premiers sont sa généralité (plusieurs familles de codes $\mathcal{C}$ peuvent d'être utilisées) ainsi que la faible complexité de calcul lors de la phase de vérification. Nous procédons ensuite à une analyse fine de la sécurité de notre construction, en utilisant le cadre proposé par Paterson *et al.* [PSU13]. Nous dégageons enfin des instanciations possibles de notre construction, par exemple à travers des produits tensoriels de codes, des codes basés sur des designs, ou des relèvements de codes.

Introduction

Context

Informal context. Outsourced data storage, also known as *cloud storage*, is a service adopted by a vast majority of Internet users. People desire to upload, share or access files stored on servers in a fast and convenient manner. Quite recently, privacy issues also emerged when using such online services. For instance, numerous scandals broke concerning the commercial use of personal data, such as details on customers' access to specific files. This thesis takes place in this cryptographic context. In a sense, the problem seems hard: one wants to keep control on files whose storage has been delegated to remote systems. Still, models and protocols have been deployed to provide specific features. Let us give two examples.

Given a collection of files stored on a remote storage system, one could first question whether it is possible to download one specific file M_i of a database $M = (M_1, \ldots, M_k)$, without leaking any information about i, the index of the desired file. This problem is known as *private information retrieval* (PIR). Efficient protocols have been designed to tackle it, but they require the storage system to be bounded in a certain way (*e.g.* its computational power, or its control on the files).

Customers may also want to verify whether the remote server actually stores the files they have uploaded. Assuming files are very large, they would like to run this verification efficiently in terms of bandwidth. For instance, trying to download the entire files is considered as inefficient. Protocols proposing a solution to this problem are called *proofs of retrievability* (PoRs).

Both previous problems can be qualified as *cryptographic*, as they may involve a malicious adversary: in private information retrieval, the remote system wants to obtain some information about the requested file; in proofs of retrievability, the server wants the customer to believe that it still holds the database, while it has maliciously erased (part of) it in order to reduce its storage costs.

We have also seen that both problems require that only a few bits are exchanged between the user and the storage system. More generally speaking, the *communication complexity* necessary to perform a given task has been studied a lot in computer science. For instance in coding theory, correcting pieces of a codeword with only a few queries to the codeword symbols is an issue addressed for almost two decades. Codes allowing such a correction are called *locally correctable codes* (LCCs).

In this thesis, we study the application of LCCs (or similar codes) in the cryptographic protocols presented above. Very informally, assume uploaded files are encoded according to an LCC. Then, the requirements necessary to the local correction (*e.g.* the number and structure of

queries) allows to retrieve information on the remote file, in an efficient and (sort of) uniform way. Let us give more details in the following paragraphs.

Locally correctable codes. Locally decodable and correctable codes were formally defined by Katz and Trevisan [KT00] in the early 2000's. A code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is said to be locally correctable if there exists a probabilistic algorithm LC, taking as input $i \in [1, n]$, the coordinate of the symbol to correct, with the following properties. First, LC makes at most ℓ queries to a noisy word $\boldsymbol{y} \in \mathbb{F}_q^n$; the parameter ℓ is called the locality and should be thought as of being much smaller than n. Second, LC must return c_i with high probability, if the Hamming distance between $\boldsymbol{y}$ and $\boldsymbol{c} \in \mathcal{C}$ is smaller than δn, where $\delta > 0$ is the admissible fraction of errors.

Locally correctable codes (LCCs) are studied under different regimes of parameters. Most works focus on the *constant locality* regime, typically $\ell = 2, 3$. For a given locality, the goal is to build an LCC with maximum information rate $R = \dim(\mathcal{C})/n$, or to give bounds on R — for instance, we refer to [Yek08, Efr12] for constructions and [KdW04, Woo12] for bounds.

Other results appeared in the *constant rate* regime. In this case we look for families of LCCs whose asymptotic rate is positive, and whose ratio locality to length is as low as possible. Since it is the most interesting for practical applications, we focus on this regime in the present thesis.

Reed-Muller (RM) codes certainly define the most typical family of constant rate LCCs. They consist in the evaluation of polynomials $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ over $\mathbb{F}_q^m$, where the total degree of f is bounded by some integer $d \geq 0$. If $d < q - 1$, then the restriction of every polynomial f to any affine line L defines a univariate polynomial $f_{|L}$ of degree $\leq d$. This univariate polynomial can be corrected with a few queries to its evaluations. Thus, a local correcting algorithm for RM codes can be defined as follows: (i) pick uniformly at random a line through the position of the symbol to correct, and (ii) correct the noisy univariate polynomial along the line.

However, RM codes of length $n = q^m$ and degree $d < q - 1$ have rate bounded by $1/m!$, when their locality is $n^{1/m}$. A few recent works [KSY14, GKS13, HOW15] lead to LCCs whose asymptotic rate R can be set arbitrarily close to 1, and whose locality remains sublinear in the length. In this thesis we mostly focus on the second construction, introduced by Guo, Kopparty and Sudan, and shortly named *lifted codes*.

Lifted codes generalise RM codes, based on the following evidence: under some arithmetic conditions, there exists polynomials f of total degree $> d$, such that the restriction of f to any affine line L describes a univariate polynomial of degree $\leq d$. Gathering polynomials satisfying this property and polynomials of degree less than d, we get a code on which the local correcting algorithm of RM codes still works. This code, called the *lifting of the Reed-Solomon code* (in short, *lifted code* since we mostly lift Reed-Solomon codes), contains the RM code and reaches arbitrarily high rate for a certain range of parameters. One purpose of the thesis is to provide and analyse an analogue of lifted codes over projective spaces.

Finally, one must report that when $d = q - 2$, a lifted code can be seen as the code based on $\mathrm{AG}_1(m, q)$, the design of points and lines in the affine space (also known as the *classical affine geometry design of* 1*-flats*). Hence, in some sense the lifting process generalise the construction of codes from designs.

Private information retrieval. Private information retrieval (PIR) was formally introduced in a work of Chor, Goldreich, Kushilevitz and Sudan [CGKS95] in 1995, that is, years before the actual deployment of cloud storage services. The authors considered the case of a database

M consisting in k files (or entries) of size 1. They proved that, when a single server stores the database, a PIR protocol leaking no information on the index i of the entry M_i (such a protocol being called *information-theoretically secure*) must use $\Omega(k)$ bits of communication. Of course, such an expensive protocol is not affordable, and two solutions have since appeared. The first one consists in restricting the definition to computational privacy; the second one allows the use of several servers, not all of them colluding, but remains information-theoretically secure. In some ways, the first solution bounds the adversary (the remote storage system) in its computational power, while the second one bounds it in its knowledge of the queries. Hereafter we only focus on the second model that makes use of several servers.

In their seminal work, Chor *et al.* [CGKS95] proposed a first construction of PIR protocols with sublinear communication complexity, which besides decreases as the number of servers used in the protocol grows. A few years later, Katz and Trevisan [KT00] proved that so-called *smooth* LCCs (that is, LCCs whose queries are close to uniform) provide PIR protocols whose communication complexity is essentially the locality of the code. Moreover, if the encoding of the database M is precomputed, then each server's answer only consists in reading one entry of the codeword it holds. Therefore, the *computational complexity* of the protocol is low. Nevertheless, the storage overhead induced by the encoding can be prohibitive, especially if the LCC has vanishing rate.

Hence, aiming at reducing the storage needs of PIR protocols, Augot, Levy-dit-Vehel and Shikfa [ALS14] proposed to *split* the encoding of the database among the servers. In this setting, servers hold different pieces of the codeword instead of copies of it, and the overall storage cost is drastically reduced. Using high-rate LCCs (the *multiplicity code* construction [KSY14]), they obtained the first PIR protocols with sublinear communication and redundancy < 2. In fact, Augot *et al.*'s idea can be extended to any LCC whose support can be split in a way that most queries of the local correcting algorithm are *transversal* to the partition. In this thesis we develop this approach with a more generic perspective, making use of so-called *transversal designs*.

Proofs of retrievability. Without going into the technicalities, proofs of retrievability (PoRs) can be introduced as follows. A PoR involves two entities: the owner of a file M that we name Alice, also called the *client* or the *verifier*, and Bob who stores of the file (also referred to as the *server* or the *prover*). In a first initialisation phase, Alice performs two operations: (i) she generates some secret material K and (ii) she encodes the file M into a string W according to K. Then, the string W is sent to Bob, and Alice erases M and keeps K secretly. The core of a PoR is the second phase, where Alice checks whether her file is extractable from Bob. This step consists in an interactive protocol where Alice defines a random challenge c, sends it to Bob who answers a response r according to c and to what he actually stores. Typically, the challenge can consist in some coordinates of the string W that Bob must send back. Then, based on c, r and her secret material K, Alice appraises whether she can retrieve M from Bob. Notice that it is highly desirable that this verification phase admits a low communication cost (compared to the size of M). Finally, if Alice is convinced that her file is retrievable (*e.g.* after several successful verification protocols), then she can run an extraction protocol whose output shall be M with high probability.

With the increasing use of decentralised storage systems, PoRs have found multiple recent applications. For instance, Storj [WBB+16] is a practical and implemented decentralised cloud storage network which uses PoRs to maintain data integrity. Though they slightly differ from PoRs, one should also notice that other kinds of schemes proving storage have been introduced during the last decade, such as provable data possession (PDP), proofs of replication (PoRep). They also were deployed in very practical applications, such as Filecoin [com17] for proofs of

replication. In this thesis, we however focus on PoRs and give a generic construction whose verification step is based on local dependencies between codewords' symbols.

Summary of contributions

Generalised design-based codes. An initial contribution concerns the construction of error-resilient locally correctable codes *via* block designs. In short, a block design $\mathcal{D}$ is a pair $(X, \mathcal{B})$ of non-empty sets, X being a finite set of *points*, and $\mathcal{B}$ being a set of subsets $B \subset X$ called blocks. The code based on a block design $(X, \mathcal{B})$ is the linear code $\mathcal{C} \subseteq \mathbb{F}_q^X$, such that every $\boldsymbol{c} \in \mathcal{C}$ satisfies $\sum_{x \in B} c_x = 0$ for every block $B \in \mathcal{B}$. In other words, we require that $\boldsymbol{c}_{|B}$ belongs to the parity code for every $B \in \mathcal{B}$. If $\mathcal{D}$ is a 2-design of block size ℓ, we prove that a very natural local correcting algorithm of locality $\ell - 1$ can be defined for $\mathcal{C}$. First, one picks at random a block B passing through the index x where to decode. Second, one uses the parity-check equation induced by the block B to recover c_x.

However, such local correction algorithms are not resilient to many errors, since they require that absolutely no error occur on symbols supported by B. To tackle this problem, we propose to consider *generalised design-based codes*, defined as follows. Let $\mathcal{D} = (X, \mathcal{B})$ be a block design, and $\mathcal{L} = (\mathcal{L}_B : B \in \mathcal{B})$ be a family of *local codes* $\mathcal{L}_B \subseteq \mathbb{F}_q^B$, indexed by blocks in $B \in \mathcal{B}$. The code based on $\mathcal{D}$ and $\mathcal{L}$, called *generalised design-based code*, is

$$\mathrm{Code}_q(\mathcal{D}, \mathcal{L}) = \{\boldsymbol{c} \in \mathbb{F}_q^X \mid \forall B \in \mathcal{B}, \boldsymbol{c}_{|B} \in \mathcal{L}_B\}.$$

We see that, if $\mathcal{L}$ consists only in parity-check codes, we end up with the usual definition of design-based codes. But, if every code $\mathcal{L}_B \in \mathcal{L}$ corrects a constant fraction of errors, we show that we improve the error resilience of the local correcting algorithm presented above. Details are given more formally in Section 2.4.

Projective lifted codes. As we have seen in a previous paragraph, lifted Reed-Solomon codes [GKS13] were initially built in order to provide an asymptotic family of LCCs achieving rate $> 1/2$. They can be formally defined as:

$$\mathrm{Lift}(\mathrm{RS}_q(k), m) = \{\mathrm{ev}_{\mathbb{A}^m}(f) \mid \text{for all affine lines } L \subset \mathbb{A}^m, \mathrm{ev}_{\mathbb{A}^1}(f_{|L}) \in \mathrm{RS}_q(k)\},$$

where $\mathrm{RS}_q(k)$ denotes a Reed-Solomon code. In this thesis, we propose to extend this construction to the evaluation of polynomials over projective spaces. To this end, we require appropriate definitions for an evaluation map of homogeneous polynomials over projective spaces $\mathrm{ev}_{\mathbb{P}^m}$, and for a space of embeddings $\mathbb{P}^1 \to \mathbb{P}^m$ denoted $\mathrm{Emb}_{\mathbb{P}}(m)$ — see Chapter 3 for details. *Projective lifted codes* can then be defined as follows:

$$\mathrm{Lift}(\mathrm{PRS}_q(k), m) \coloneqq \{\mathrm{ev}_{\mathbb{P}^m}(f) \mid \forall L \in \mathrm{Emb}_{\mathbb{P}}(m), \mathrm{ev}_{\mathbb{P}^1}(f \circ L) \in \mathrm{PRS}_q(k)\},$$

where $\mathrm{PRS}_q(k)$ denotes a projective (or doubly-extended) Reed-Solomon code.

We then show that projective lifted codes admit a local correcting algorithm analogous to the one used for lifted codes defined over affine spaces (that we name *affine* lifted codes). Let us sketch how it corrects a symbol $y_{\boldsymbol{x}}$ of a noisy codeword $\boldsymbol{y}$, for $\boldsymbol{x} \in \mathbb{P}^m$. First, one picks uniformly at random an embedding $L : \mathbb{P}^1 \to \mathbb{P}^m$ such that $\boldsymbol{x} \in L(\mathbb{P}^1)$. Then, one corrects the restricted word $\boldsymbol{y}_{|L(\mathbb{P}^1)}$, since it should lie in $\mathrm{PRS}_q(k)$. A more formal presentation is given in Algorithm 10, Chapter 3.

We then explore properties of projective lifted codes. We first prove they admit a basis made of evaluation vectors of monomials, and we then study the set of exponents of this monomial basis, called the *degree set*. A main result is a recursive relation between degree sets of affine and projective lifted codes. More precisely, denoting by $\mathrm{PDeg}(m,k)$ (resp. $\mathrm{ADeg}(m,k)$) the degree set of $\mathrm{Lift}(\mathrm{PRS}_q(k),m)$ (resp. $\mathrm{Lift}(\mathrm{RS}_q(k),m)$), we give an explicit bijection between $\mathrm{PDeg}(m,k)$ and $\mathrm{PDeg}(m-1,k)\cup\mathrm{ADeg}(m,k-1)$, see Theorem 3.23.

This recursive characterisation has multiple consequences. First, it is a convenient way to compute a basis — and *a fortiori* the dimension — of projective lifted codes. Second, it allows us to exhibit links between lifted codes *via* puncturing and shortening operations: for instance, shortening $\mathrm{Lift}(\mathrm{PRS}_q(k),m)$ on any projective hyperplane gives a code isomorphic to $\mathrm{Lift}(\mathrm{RS}_q(k-1),m)$. More precisely, we prove that the following exact sequence

$$0\to\mathrm{Lift}(\mathrm{RS}_q(k-1),m)\to\mathrm{Lift}(\mathrm{PRS}_q(k),m)\xrightarrow{\pi}\mathrm{Lift}(\mathrm{PRS}_q(k),m-1)\to 0$$

holds, where π is the restriction map on a fixed projective hyperplane of $\mathbb{P}^m$. In fact, a similar exact sequence was already known for Reed-Muller codes and codes based on geometric designs.

In some ways, lifted codes bridge algebraic constructions of codes such as Reed-Muller codes and combinatorial constructions based on block designs. For low degrees $k\leq q(1-1/p)$ where $p=\mathrm{char}(\mathbb{F}_q)$, Kaufman and Ron [KR06] proved that $\mathrm{Lift}(\mathrm{RS}_q(k),m)=\mathrm{RM}_q(k,m)$. On the opposite, if $k=q-2$, then $\mathrm{Lift}(\mathrm{RS}_q(k),m)$ is the code based on the design $\mathrm{AG}_1(m,q)$. More generally, we prove that projective and affine lifted codes are generalised design-based codes.

We also study several properties of projective lifted codes that emphasize their practicality. We show they admit quasi-cyclic automorphisms that can help to reduce their storage cost. We also give a large and explicit family of information sets for these codes.

Finally, we analyse in details the degree sets of affine lifted codes in the case $m=2$, which corresponds to the codes with highest rates. We highlight their fractal representation and provide a precise computation of their dimension. We notably improve upon a bound given as a claim in [GKS13], proving that the dimension K of $\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e-2^{e-c}-1),2)$, for $1\leq c\leq e-1$, is exactly

$$K=4^e\left(1-\frac{5}{4}\left(\frac{3}{4}\right)^c+\frac{1}{4}\left(\frac{1}{4}\right)^c+\frac{1}{2^e}\left(\frac{3^c-1}{2^{c+2}}\right)\right).$$

This particular setting is of main interest, since for fixed c and increasing e, it provides an asymptotically good family of locally correctable codes. More generally, in any characteristic $p\geq 2$ we prove that for $e\to\infty$, the asymptotic rate $\rho_\infty^{(p)}(c)$ of the family of lifted codes $\mathrm{Lift}(\mathrm{RS}_{p^e}(p^e-p^{e-c}-1),2)$ is

$$\rho_\infty^{(p)}(c)=1-\left(1+\frac{1}{p+2}\right)\left(\frac{1+1/p}{2}\right)^c+\frac{1}{p+2}\left(\frac{1}{p^2}\right)^c,$$

and can thus be set arbitrarily close to 1.

Private information retrieval based on transversal designs. Most recent constructions of PIR protocols mainly focus on obtaining an optimal download communication complexity during the retrieval process. However, the computational cost of servers' answers is neglected even though it sometimes represents a barrier to PIR practicality. For instance, it seems unreasonable for a server to read a whole database in order to give an answer whose size is roughly the size of an entry.

In this thesis, we propose a new generic framework for the construction of PIR protocols which takes into account the computational complexity issue. More precisely, the protocols we give are computationally optimal with respect to the communication complexity of the protocol, in the sense that each server only needs to read one entry in the database it holds.

Our construction is based on combinatorial structures called *transversal designs*. They are block designs $(X, \mathcal{B})$ equipped with a set of *so-called* groups $\mathcal{G}$ that forms a partition of X, and such that every pair $\{x_1, x_2\} \subset X$ is either contained in a group, or in a fixed number λ of blocks. If the block size is ℓ and the group size is s, such a transversal design is denoted $\mathrm{TD}_\lambda(\ell, s)$. The code $\mathcal{C}$ based on a $\mathrm{TD}_\lambda(\ell, s)$ therefore admits a support X, $|X| = s\ell$, that can be split into ℓ groups $G_1, \ldots, G_\ell$. Furthermore, for every pair of coordinates $\{x_1, x_2\}$ that do not lie in the same group, there exists a parity-check equation for $\mathcal{C}$ whose support contains $\{x_1, x_2\}$.

Let us now give a quick overview of the construction of our ℓ-server PIR protocol — more details are given in Chapter 4. Given an encoding $\boldsymbol{c} \in \mathcal{C}$ of the database, the user first uploads to the i-th server S_i the restriction $\boldsymbol{c}_{|G_i}$ of $\boldsymbol{c}$, for $1 \leq i \leq \ell$. Now assume the user wants to privately retrieve an entry c_x for $x \in X$. Then, they pick at random a block $B \in \mathcal{B}$ such that $x \in B$. Let us now describe queries to server S_i, for $1 \leq i \leq \ell$. If i is such that $x \notin G_i$, then the user queries c_{y_i}, where y_i is the only point in $G_i \cap B$. Otherwise, the user query c_y, for some uniformly random $y \in G_i$.

Upon reception of servers' answers, we see that the user collects the tuple $(c_b \in \mathbb{F}_q : b \in B \setminus \{x\})$. Hence, the user can retrieve c_x thanks to the parity-check equation having B as support. The privacy of the protocol comes from the fact that for every pair $\{x, y_i\}$ that do not lie in the same group, there exists a constant number of blocks B that contain $\{x, y_i\}$.

The PIR construction we propose features, as expected, low computational complexity for each server S_i, $1 \leq i \leq \ell$: it only needs to read one symbol in $\boldsymbol{c}_{|G_i}$. The overall communication complexity is proportional to the block size ℓ. In fact, the only parameter that depends on the chosen instance of transversal design is the storage overhead of the scheme. In practice, low storage overhead are desirable, so we look for transversal designs $\mathrm{TD}_\lambda(\ell, s)$ whose associated linear code $\mathcal{C}$ have dimension as large as possible compared to their length $s\ell$.

Therefore, we propose several instances of transversal designs that lead us to codes with large rate. The first two families come from incidences between points and lines in the affine (resp. projective) space. They are closely related to the classical geometric designs of 1-flats. The third family of instances makes use of a classical transformation of so-called *orthogonal arrays* of strength 2 into transversal designs. This leads us to introduce PIR protocols based on *incidence codes*, that arise from the steps depicted in Figure 2.

We then proceed to a thorough study of the dimension of incidence codes coming from MDS codes $\mathcal{C}_0$ of dimension 2. Finally, a last family of practical instances appears when showing that *divisible codes* $\mathcal{C}_0$ admit incidence codes of rate roughly greater than one half. However, the existence of such codes over large alphabets remains an open question.

The use of orthogonal arrays of strength $t = 2$ for the construction of PIR protocols leads us to examine the case of higher strengths $t > 2$. We prove that such orthogonal arrays allow the construction of PIR protocols which remain private, even if up to $t - 1$ servers collude (*i.e.* they exchange information about their respective queries). Finally, we exhibit and analysed instances of orthogonal arrays with large strength to conclude this chapter.

Codes with locality for proofs of retrievability. A formal definition of PoRs was proposed by Juels and Kaliski in [JK07], where the authors also give a first construction based on *sen-*

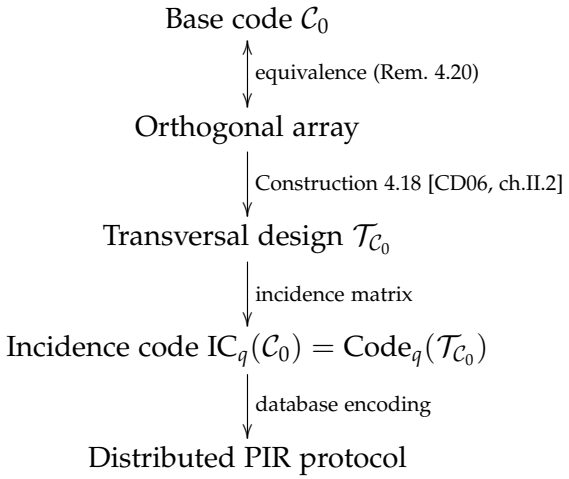


Figure 2 – Outline of the construction of a distributed PIR protocol using incidence codes.

tinels (small pieces of W that Alice keeps secretly and whose integrity is checked during the verification phase). Many other schemes were then introduced (notably [SW13, WWR+11]), and Paterson, Stinson and Upadhyay provided a coding-theoretic framework to analyse their security [PSU13]. Precisely, the idea of Paterson *et al.* is to model the collection of Bob's responses to challenges as a (very long) word, so that the extraction can be seen as an error-correction procedure.

Building on this framework, we propose a new construction of proofs of retrievability based on linear codes. Our idea is essentially the following. Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a code and $\mathcal{H}$ be a set of parity-check equations for $\mathcal{C}$ of low weight $\ell \ll n$. Alice encodes her file M into a codeword $C \in \mathcal{C}$ and encrypts C into W (using a suitable cipher and a key K). The verification phase then consists in testing random parity-check equations on W. Precisely, Alice picks at random an $h \in \mathcal{H}$, and asks Bob to send back symbols of W supported by h. Then Alice decrypts the symbols returned by Bob, resulting in a plaintext a, and she checks whether $\sum_i h_i a_i = 0$.

In Chapter 5, we firstly define a security model for PoRs, in order to give a formal version of our construction. We then introduce *verification structures* for codes that generalise the use of parity-check equations that we depicted above. We then proceed to give a detailed and generic analysis of the security of our PoR schemes, using the framework of Paterson *et al.* [PSU13] which particularly suits our work. The main advantages of our construction are its generality (many different codes can be used, with various parameters) and its low computation complexity, once the initialisation is performed.

In practice, the communication complexity of our PoRs depends on the weight ℓ of the parity-check equations that are used (or more generally, the locality parameter ℓ of the verification structure for $\mathcal{C}$). Therefore, for practical reasons we look for codes with a quite elaborate and structured set of parity-check equations of small weight. This leads us to propose several instantiations of our PoR constructions, using well-known codes with locality that were introduced previously in the thesis (*e.g.* Reed-Muller codes or design-based codes).

Outline of the thesis

The present document is composed of two main parts. Chapters 1–3 are devoted to codes with locality, mostly locally correctable codes. Their application into two cryptographic protocols — namely private information retrieval and proofs of retrievability — are proposed, respectively in Chapters 4 and 5.

In the first chapter, we recall notions of coding and design theory that will be used throughout the thesis. This allows us to fix convenient but less common notation for codes, and to recall well-known constructions and results.

Chapter 2 is mostly devoted to a review of locally decodable and correctable codes. We first give formal definitions of these codes, and we explain some important existing constructions and bounds. We also take the opportunity to define very properly a *smooth* version of their local correcting algorithms. Finally, we propose in Section 2.4 a new perspective for the construction of locally correctable codes, through the generalisation of design-based codes presented earlier.

The third chapter focuses on the construction and analysis of projective lifted codes. After an introduction of the algebraic background necessary to the construction, we provide formal definitions of projective lifted codes, and we give their first properties. A local correcting algorithm is presented in Section 3.3, matching the parameters of the affine lifted codes of Guo *et al.* [GKS13]. Then we present recursive relations between affine and projective lifted codes, *via* shortening and puncturing. Some practical features (such as explicit information sets and automorphisms) are given in Section 3.4, and we end this chapter with a precise computation and analysis of the degree sets and the dimension of lifted codes of order $m = 2$.

In Chapter 4, we consider the application of codes based on transversal design in the construction of private information retrieval protocols. A quick review of existing schemes is proposed in Section 4.1, and motivates our construction given in the next section. We then suggest and analyse several families of instances that yield practical parameters, notably the ones based on geometric designs. Finally, in Section 4.4 we propose a generalisation of our construction using orthogonal arrays, in order to resist collusions between some of servers.

Finally, Chapter 5 examines the construction of proofs of retrievability using codes with locality. After referencing some important existing PoR schemes, we define a security model matching the framework due to Paterson *et al.* [PSU13]. According to this model, we present and analyse our PoR construction in Section 5.3. Its performance is described in the next section, and we propose several instantiations based on well-known codes and designs in Section 5.5.

A conclusion including further avenues of research completes the document.

Publications

Journal articles

- **Private Information Retrieval from Transversal Designs.** Julien Lavauzelle. *IEEE Trans. on Information Theory.* Accepted 08-2018, to appear. DOI: 10.1109/TIT.2018.2861747.
- **Lifted Projective Reed–Solomon Codes.** Julien Lavauzelle. *Designs, Codes and Cryptography.* Accepted 09-2018, to appear. DOI: 10.1007/s10623-018-0552-8.

International conference with proceedings

- **New Proofs of Retrievability Using Locally Decodable Codes.** Julien Lavauzelle and Françoise Levy-dit-Vehel. In *IEEE International Symposium on Information Theory,* ISIT 2016, Barcelona, Spain, July 10-15, 2016, pp.1809–1813. DOI: 10.1109/ISIT.2016.7541611.

International conference without proceedings

- **Constructions for Efficient Private Information Retrieval Protocols.** Julien Lavauzelle. *International Workshop on Coding and Cryptography,* Saint-Petersburg, Russia, 2017.

Prepublication

- **Generic Constructions of PoRs from Codes and Instantiations.** Julien Lavauzelle and Françoise Levy-dit-Vehel. Submitted 04-2018 to *Journal of Mathematical Cryptology* (under review).

Part I

Codes with locality

Chapter 1

Basics of coding and design theory

Contents

Overall notation. Lowercase bold characters represent tuples, codewords or vectors, while uppercase ones are used for higher-dimensional data structures (such as matrices or tensors). Sets are usually denoted in classical uppercase characters, and their elements in lowercase.

We denote by $[1,n] \coloneqq \{1,2,\dots,n\}$. A k-subset of a finite set S, $k \leq |S|$, is a subset of S of cardinality k. We also denote by $\mathfrak{S}(S)$ the set of permutations on S.

For any set A, the power set A^S represents the set of maps $f : S \to A$. Since S is finite, a function $f \in A^S$ can also be seen as a tuple $\boldsymbol{a}$ over A indexed by S, through its *evaluation tuple* $\boldsymbol{a} = (f(x) : x \in S) \in A^S$. In the tuple $\boldsymbol{a} \in A^S$, the symbol indexed by $x \in S$ is written $a_x \in A$. In our context, the correspondence between evaluation tuples and functions is very convenient and will be used regularly. In order to distinguish the two representations, recall that tuples are written in bold characters. We also identify $A^{[1,n]}$ with A^n.

If R is a ring, the ideal generated by a finite subset U of an R-module is written $\mathrm{Span}_R(U)$, or in short $\mathrm{Span}(U)$ if the base ring is implicit.

If K is a field, $K^\times$ represents the non-zero elements of K. We denote by $K[X]$ the ring of univariate polynomials over K with indeterminate X, and by $K[\boldsymbol{X}] = K[X_1,\dots,X_m]$ the ring of polynomials with m variables over K. The subspace of polynomials of total degree bounded by d is denoted by $K[\boldsymbol{X}]_{\leq d}$.

The coordinate-wise product (also known as Schur, or Hadamard product) of two vectors $\boldsymbol{a}, \boldsymbol{b} \in K^S$ is $\boldsymbol{a} \star \boldsymbol{b} \coloneqq (a_x b_x : x \in S) \in K^S$. Notice it corresponds to the classical product fg of two maps $f, g : S \to K$.

The m-dimensional affine space over K is simply K^m, and it is sometimes denoted $\mathbb{A}^m(K)$, or $\mathbb{A}^m$. We can also define the projective space $\mathbb{P}^m(K)$ as the quotient $K^{m+1}/\sim$, where the relation $\sim$ is defined by:

$$\boldsymbol{u} \sim \boldsymbol{u}' \iff \exists \lambda \in K^\times, \boldsymbol{u} = \lambda \boldsymbol{u}'\,.$$

Given two K-vector spaces U and V, the space of linear maps $U \to V$ is denoted $\mathrm{Hom}(U,V)$, and the invertible ones $\mathrm{Iso}(U,V)$. If $U = V$ we shortly write $\mathrm{Hom}(U)$ and $\mathrm{Iso}(U)$. The set of maps $\phi : K^t \to K^m$, $\phi(\boldsymbol{x}) = \psi(\boldsymbol{x}) + \boldsymbol{b}$, where $\boldsymbol{b} \in K^m$ and $\psi \in \mathrm{Hom}(K^t, K^m)$ is injective, is called the set of *affine transformations* from K^t to K^m. One must discern affine transformations from generic affine *maps*, from the fact that we require affine transformations to be injective. We will mostly deal with affine transformations, that we denote by $\mathrm{Aff}(K^t, K^m)$, or $\mathrm{Aff}(K^t)$ when $t = m$.

Given a set S, the notation $s \leftarrow_{\mathrm{R}} S$ means that s is picked *uniformly at random* from the set S. We also denote by $\mathscr{B}(p)$ the Bernoulli distribution with parameter p, and we recall that if a random variable X follows $\mathscr{B}(p)$, then its mean value is $\mathbb{E}(X) = p$ and its variance is $\mathbb{D}(X) \coloneqq \mathbb{E}((X - \mathbb{E}(X))^2) = p(1-p)$. Let X and Y be two discrete random variables with finite support. We say that X and Y are *independent* if we have

$$\Pr(X = x, Y = y) = \Pr(X = x)\Pr(Y = y)$$

for every x, y lying in the respective support of X and Y. When X and Y are real-valued, we say that X and Y are *uncorrelated* if $\mathbb{E}(XY) = \mathbb{E}(X)\mathbb{E}(Y)$. As a consequence, two uncorrelated variables X, Y satisfy $\mathbb{D}(X+Y) = \mathbb{D}(X) + \mathbb{D}(Y)$. Two independent random variables are uncorrelated. Finally, a family of random variables $\{X_i\}_{i\in I}$ is *pairwise independent* (resp. *pairwise uncorrelated*) if for all $i \neq j \in I$, X_i and X_j are independent (resp. uncorrelated).

Let us also recall that Markov's inequality states that, for a non-negative random variable X, we have for every $a > 0$:

$$\Pr(X \geq a) \leq \frac{\mathbb{E}(X)}{a}\,.$$

Furthermore, the deviation from the mean value is bounded by Chebychev's inequality. This states that, for any real-valued random variable X and for every $b > 0$, we have:

$$\Pr(|X - \mathbb{E}(X)| \geq b) \leq \frac{\mathbb{D}(X)}{b^2}\,.$$

Given two positive functions $f, g : \mathbb{N} \to \mathbb{R}^+$, we say that $f = \mathcal{O}(g)$ if there exists $\alpha > 0$ such that $f(n) \leq \alpha g(n)$ for all large enough $n \in \mathbb{N}$. We write $f = \widetilde{\mathcal{O}}(g)$ if $f = \mathcal{O}(g\log(g)^c)$ for some constant $c > 0$.

If for every $\varepsilon > 0$, $f(n) \leq \varepsilon g(n)$ holds for large enough $n \in \mathbb{N}$, then we write $f = o(g)$. Finally, the notation $f = \Omega(g)$ is equivalent to $g = \mathcal{O}(f)$, and we write $f = \Theta(g)$ when $f = \Omega(g)$ and $f = \mathcal{O}(g)$.

1.1 Elementary notions in coding theory

This section is devoted to recall basic notions related to coding theory. Here we choose to use the tuple representation for elements in Σ^S since it is the most commonly used in the literature. When meaningful, we also give correspondence of the definitions and results in the functional representation. In the whole section, we refer to [HP10] for details and proofs.

Generic notions. Given an alphabet Σ and a finite set S, an *error-correcting code over* Σ is a subset $\mathcal{C} \subseteq \Sigma^S$. The set S is the *set of coordinates* of the code, and its size $n = |S|$ is the *length* of the code. Elements $\boldsymbol{c} \in \mathcal{C}$ are called *codewords*. As its name suggests, an error-correcting code is usually employed to correct errors and erasures that can occur in messages, for instance during transmission. Given a codeword $\boldsymbol{c} \in \mathcal{C} \subseteq \Sigma^S$ and a noisy version $\boldsymbol{y} \in (\Sigma \cup \{\perp\})^S$ of $\boldsymbol{c}$, where $\perp$ denotes the *erasure symbol*, an *error* is a symbol $y_i \notin \{c_i, \perp\}$ for some $i \in S$, while an *erasure* is a symbol $y_j = \perp$ for some $j \in S$. If not stated otherwise, we assume in this thesis that only errors can occur in words. A well-suited metric for modelling errors is the *Hamming distance*, defined by $\mathrm{d}(\boldsymbol{y}, \boldsymbol{z}) := |\{i \in S, y_i \neq z_i\}|$ for any two words $\boldsymbol{y}, \boldsymbol{z} \in \Sigma^S$. The distance of $\boldsymbol{y} \in \Sigma^S$ to a code $\mathcal{C}$ is then $\mathrm{d}(\boldsymbol{y}, \mathcal{C}) := \min\{\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}), \boldsymbol{c} \in \mathcal{C}\}$, and the *minimum distance* of the code $\mathcal{C}$ is $\mathrm{d}_{\min}(\mathcal{C}) := \min\{\mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') \mid (\boldsymbol{c}, \boldsymbol{c}') \in \mathcal{C}^2, \boldsymbol{c} \neq \boldsymbol{c}'\}$. It is also convenient to consider the *relative distance* between two words $\boldsymbol{y}, \boldsymbol{z} \in \Sigma^S$, defined by $\delta(\boldsymbol{y}, \boldsymbol{z}) := \mathrm{d}(\boldsymbol{y}, \boldsymbol{z})/|S|$. Similarly one can define the relative distance of a word to a code, and the relative minimum distance of a code.

A *nearest-neighbour correcting algorithm* for $\mathcal{C}$ is an algorithm which takes as input *any* word $\boldsymbol{y} \in \Sigma^S$, and outputs a word $\boldsymbol{c} \in \mathcal{C}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) = \mathrm{d}(\boldsymbol{y}, \mathcal{C})$. Let $t := \left\lfloor \frac{\mathrm{d}_{\min}(\mathcal{C})-1}{2} \right\rfloor$ be the *packing radius* of $\mathcal{C}$. Clearly, if $\mathrm{d}(\boldsymbol{y}, \mathcal{C}) \leq t$, then the output is unique. Therefore, a *half-distance correcting algorithm* for $\mathcal{C}$ is an algorithm that outputs the unique $\boldsymbol{c} \in \mathcal{C}$ nearest to $\boldsymbol{y}$, provided $\boldsymbol{y} \in \Sigma^S$ satisfies $\mathrm{d}(\boldsymbol{y}, \mathcal{C}) \leq t$. More generally, one can also consider *T-bounded-distance correcting algorithms* when $\mathrm{d}(\boldsymbol{y}, \mathcal{C})$ is bounded by some integer $T \leq t$.

Linear codes. We have defined codes $\mathcal{C} \subseteq \Sigma^S$ with a general perspective, but they are mostly studied in the context $S = [1, n]$ and $\Sigma = \mathbb{F}_q$, where $\mathbb{F}_q$ is the finite field with q elements. If $\mathcal{C}$ has a $\mathbb{F}_q$-vector space structure, then $\mathcal{C}$ is said to be ($\mathbb{F}_q$-)*linear*. **In this thesis, every code is assumed to be $\mathbb{F}_q$-linear with alphabet $\Sigma = \mathbb{F}_q$, except stated otherwise.** Thus, linear codes over $\mathbb{F}_q$ are referred to as *codes* for short. We call *dimension of the code* the integer $k = \dim_{\mathbb{F}_q} \mathcal{C}$. The *support* of a word $\boldsymbol{y} \in \mathbb{F}_q^X$ is $\mathrm{supp}(\boldsymbol{y}) := \{i \in X, y_i \neq 0\}$. Its *weight* $\mathrm{wt}(\boldsymbol{y}) := \mathrm{d}(\boldsymbol{0}, \boldsymbol{y})$ is also the cardinality of its support. The minimal distance of a linear code $\mathcal{C} \subseteq \mathbb{F}_q^X$ can therefore be seen as the minimum weight $\mathrm{wt}(\boldsymbol{c})$ of a non-zero codeword $\boldsymbol{c} \in \mathcal{C}$.

Any injective map $E : \mathbb{F}_q^k \to \mathcal{C}$ is an *encoder* for $\mathcal{C}$, and elements $\boldsymbol{m} \in \mathbb{F}_q^k$ are called *messages* and usually written in rows $\boldsymbol{m} = (m_1, \ldots, m_k)$. Given an encoder $E : \mathbb{F}_q^k \to \mathcal{C}$, one can define a *nearest neighbour decoding algorithm* for $(\mathcal{C}, E)$, as an algorithm which takes as input any word $\boldsymbol{y} \in \Sigma^S$, and outputs a message $\boldsymbol{m} \in \mathbb{F}_q^k$ such that $\mathrm{d}(\boldsymbol{y}, E(\boldsymbol{m})) = \mathrm{d}(\boldsymbol{y}, \mathcal{C})$. Similarly, *half-distance* and *T-bounded-distance decoding algorithm* can be defined.

For $\boldsymbol{y} \in \mathbb{F}_q^S$ and $I \subseteq S$, we denote by $\boldsymbol{y}_{|I} = (y_i : i \in I)$. This corresponds to the restriction to I in the functional representation. The code $\mathrm{Short}(\mathcal{C}, I) := \{\boldsymbol{c}_{|S\setminus I}, \boldsymbol{c} \in \mathcal{C}, \boldsymbol{c}_{|I} = \boldsymbol{0}\}$ is called the *shortening of $\mathcal{C}$ on I*. The *puncturing of $\mathcal{C}$ on I* (or its *restriction on $S \setminus I$*) is $\mathrm{Punct}(\mathcal{C}, I) := \mathcal{C}_{|S\setminus I} = \{\boldsymbol{c}_{|S\setminus I}, \boldsymbol{c} \in \mathcal{C}\}$. Both shortening and puncturing operations give linear codes over $\mathbb{F}_q$, and $\mathrm{Short}(\mathcal{C}, I) \subseteq \mathrm{Punct}(\mathcal{C}, I)$.

If $|I| = \dim(\mathcal{C}_{|I}) = \dim(\mathcal{C})$ holds, then we say that I is an *information set* for $\mathcal{C}$. Given any information set $I \subseteq S$ and any encoder E, there exists a linear map $\phi_I : \mathbb{F}_q^I \to \mathbb{F}_q^k$ such that $\phi_I \circ E = \mathrm{id}_{\mathbb{F}_q^k}$. Denote by $\phi_I = (\phi_1, \ldots, \phi_k) : \mathbb{F}_q^I \to \mathbb{F}_q^k$. If moreover, there exists a one-to-one map $\sigma : I \to [1, k]$ such that, $\phi_i(\boldsymbol{x}) = x_{\sigma(i)}$ for every $\boldsymbol{x} \in \mathbb{F}_q^I$, then we say that E is a *systematic encoder with respect to the information set I*. In simple words, a systematic encoder is a linear map $\mathbb{F}_q^k \to \mathcal{C}$ such that, by selecting k coordinates $\{i_1, \ldots, i_k\} = I$, one can read any message $\boldsymbol{m}$ from the coordinates indexed by I in its encoded version $E(\boldsymbol{m})$.

Let $\boldsymbol{G} \in \mathbb{F}_q^{[1,k]\times S}$ be a matrix whose rows and columns are respectively indexed by $[1,k]$ and S. The matrix $\boldsymbol{G}$ is a *generator matrix* for $\mathcal{C}$ if the map $\boldsymbol{m} \mapsto \boldsymbol{mG}$ is an encoder for $\mathcal{C}$. For such a matrix $\boldsymbol{G}$, an information set I for $\mathcal{C}$ corresponds to a set of columns for which the minor $\boldsymbol{G}_{|[1,k]\times I}$ is invertible. Furthermore, the matrix $\boldsymbol{G}$ gives a natural systematic encoder if its submatrix $\boldsymbol{G}_{|[1,k]\times I}$ is the $k \times k$ identity matrix (up to a permutation of the columns).

Duality. We denote by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \sum_{i\in S} x_i y_i$ the classical inner product between two elements $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{F}_q^S$. Every linear code $\mathcal{C}$ admits an orthogonal space with respect the classical inner product, denoted $\mathcal{C}^\perp$ and historically but confusingly called the *dual code* of $\mathcal{C}$. This is a linear space of dimension $|S| - \dim(\mathcal{C})$. Elements of $\mathcal{C}^\perp$ are called *parity-check equations* for $\mathcal{C}$. The *dual distance* of $\mathcal{C}$ is simply $\mathrm{d}^\perp(\mathcal{C}) := \mathrm{d}_{\min}(\mathcal{C}^\perp)$.

Any matrix $\boldsymbol{H} \in \mathbb{F}_q^{[1,r]\times S}$ satisfying $\boldsymbol{Hc} = \boldsymbol{0}$ if and only if $\boldsymbol{c} \in \mathcal{C}$ is called a *parity-check matrix* for $\mathcal{C}$. Notice we do not restrict ourselves to $r = |S| - \dim(\mathcal{C})$. For $\boldsymbol{y} \in \mathbb{F}_q^S$, the vector $\boldsymbol{Hy} \in \mathbb{F}_q^r$ is called a *syndrome*.

It is clear that, if $\boldsymbol{G}$ is a generator matrix for $\mathcal{C}$, then we have $\boldsymbol{HG} = \boldsymbol{0}$. Moreover, the rows of $\boldsymbol{H}$ generates the dual code $\mathcal{C}^\perp$. Therefore, $\mathrm{rank}(\boldsymbol{H}) = |S| - \dim(\mathcal{C})$, and a row-reduced form of $\boldsymbol{H}$ defines a generator matrix for $\mathcal{C}^\perp$.

Permutations, automorphisms. Any permutation $\sigma \in \mathfrak{S}(S)$ induces a map $\mathbb{F}_q^S \to \mathbb{F}_q^S$, defined by $\sigma^*(\boldsymbol{y}) := (y_{\sigma(x)} : x \in S)$. In the functional representation, this corresponds to the composition of functions, *i.e.* $\sigma^* : f \mapsto f \circ \sigma$. A code $\mathcal{C} \subseteq \mathbb{F}_q^S$ is said to be *invariant* by σ if $\sigma^*(\mathcal{C}) = \mathcal{C}$. For a given $\mathcal{C}$, such permutations form a group under the composition law; it is called the *permutation group* of $\mathcal{C}$ and denoted $\mathrm{Perm}(\mathcal{C})$.

More generally, the semi-direct product $(\mathbb{F}_q^\times)^S \rtimes \mathfrak{S}(S)$ acts on $\mathbb{F}_q^S$ by

$$(\boldsymbol{a}, \sigma) : \quad \boldsymbol{y} \mapsto \boldsymbol{a} \star \sigma^*(\boldsymbol{y}) .$$

The group of elements $(\boldsymbol{a}, \sigma) \in (\mathbb{F}_q^\times)^S \rtimes \mathfrak{S}(S)$ which leave a code $\mathcal{C}$ invariant is called the *automorphism group* of $\mathcal{C}$ and denoted $\mathrm{Aut}(\mathcal{C})$.

Isomorphisms between codes. Given a one-to-one map $\phi : A \to B$, one can also define a map $\phi^* : \mathbb{F}_q^A \to \mathbb{F}_q^B$ by $\phi^*(\boldsymbol{y}) := (y_{\phi(x)} : x \in A)$. Then, we say that two codes $\mathcal{C} \subseteq \mathbb{F}_q^A$ and $\mathcal{C}' \subseteq \mathbb{F}_q^B$ are *permutation-equivalent* if $\phi^*(\mathcal{C}) = \mathcal{C}'$. Similarly, if there exists $\boldsymbol{b} \in (\mathbb{F}_q^\times)^B$ such that $\boldsymbol{b} \star \phi^*(\mathcal{C}) = \mathcal{C}'$, then $\mathcal{C}$ and $\mathcal{C}'$ are *isomorphic*. Notice that any pair $(\boldsymbol{b}, \phi)$ defines an isometry between metric spaces $\mathbb{F}_q^A$ and $\mathbb{F}_q^B$, where the metric is the Hamming distance.

The last remark is crucial regarding correction issues. It notably implies that, if a code $\mathcal{C}$ admits a t-bounded-distance correcting algorithm Corr, then one can also correct any code $\mathcal{C}'$ isomorphic to $\mathcal{C}$ up to t errors. Indeed, let $\boldsymbol{y}' \in \mathbb{F}_q^B$ be a noisy word, $\boldsymbol{y}' = \boldsymbol{c}' + \boldsymbol{e}'$, where $\boldsymbol{c}' \in \mathcal{C}'$ and $\boldsymbol{e}'$ is the error. We first build

$$(\phi^*)^{-1}(\boldsymbol{b}^{-1} \star \boldsymbol{y}') = \underbrace{(\phi^*)^{-1}(\boldsymbol{b}^{-1} \star \boldsymbol{c}')}_{=\boldsymbol{c}\in\mathcal{C}} + (\phi^*)^{-1}(\boldsymbol{b}^{-1} \star \boldsymbol{e}') \in \mathbb{F}_q^A ,$$

and $\boldsymbol{e} := (\phi^*)^{-1}(\boldsymbol{b}^{-1} \star \boldsymbol{e}')$ has the same weight properties as $\boldsymbol{e}'$. Therefore, if Corr is able to recover the correct codeword $\boldsymbol{c} \in \mathcal{C}$, then we get $\boldsymbol{b} \star \phi^*(\boldsymbol{c}) = \boldsymbol{c}'$.

Parameters and bounds. Linear codes admit three significant parameters, being their length n, dimension k and minimum distance d. One usually records these parameters between brackets: $[n, k, d]$, or $[n, k]$ if the minimum distance is unknown. For a given length n, it is meaningful to look for codes with large dimension (*i.e.* large amount of data stored in a single codeword) and large minimum distance (that potentially induces a large correction capability). However, there must be a trade-off between these two quantities: informally, one can not correct many errors on codewords if they do not contain enough redundancy symbols. More precisely, the famous Singleton bound states that

$$k + d \leq n + 1\,.$$

Codes attaining the Singleton bound are called *maximum distance separable* (MDS). If $\mathcal{C} \subseteq \mathbb{F}_q^S$ is MDS, then any k-subset of S is an information set for $\mathcal{C}$. As well, the dual code of an MDS code is an MDS code.

Codes with extreme parameters. Codes with minimum distance 1 or dual distance 1 are considered as *degenerate*, since they admit bad correction properties. The *zero code* $\{\mathbf{0}\}$ and the *full code* $\mathbb{F}_q^S$ are examples of degenerate codes. Codes of dimension and codimension 1 are more relevant. The $[n, 1, n]$ code generated by $\boldsymbol{a} \in (\mathbb{F}_q^\times)^S$, $|S| = n$, is the ***a*-repetition code** $\mathrm{Rep}(n, \boldsymbol{a})$; it can be seen as the code of constant functions in the functional notation. If $\boldsymbol{a} = \mathbf{1} := (1, \ldots, 1)$, then $\mathrm{Rep}(n) := \mathrm{Rep}(n, \mathbf{1})$ is simply referred to as the *repetition code*. Its dual code is the so-called ***a*-parity-check** code $\mathrm{Par}(n, \boldsymbol{a})$, with parameters $[n, n-1, 2]$, and it consists in words $\boldsymbol{c} \in \mathbb{F}_q^S$ satisfying $\sum_{i \in S} c_i a_i = 0$. Similarly one can define $\mathrm{Par}(n) := \mathrm{Par}(n, \mathbf{1})$. Notice that repetition codes and parity-check codes are MDS.

1.2 Algebraic constructions of codes

A convenient way to build good and non-extreme codes $\mathcal{C} \subseteq \mathbb{F}_q^S$ is to use subspaces of polynomial functions $S \to \mathbb{F}_q$. Codes can be defined as such, but the correspondence between polynomials and polynomial functions is sometimes tricky: over $\mathbb{F}_q$, several polynomials may correspond to the same function.

Therefore, it is sometimes more convenient to see these codes as *evaluation codes*. That is, codewords are explicitly given as *evaluation vectors* of polynomials. More precisely, given a vector space of polynomials $\mathcal{R} \subset \mathbb{F}_q[X_1, \ldots, X_m]$ and a set $S \subseteq \mathbb{F}_q^m$, we define an *evaluation map*

$$\begin{array}{rccc} \mathrm{ev}_S : & \mathcal{R} & \to & \mathbb{F}_q^S \\ & f & \mapsto & (f(\boldsymbol{x}) : \boldsymbol{x} \in S) \end{array}$$

If ev_S is injective, then the associated evaluation code $\mathcal{C} = \mathrm{ev}_S(\mathcal{R})$ has length $|S|$ and dimension $\dim(\mathcal{R})$.

1.2.1 Reed-Solomon codes

Reed-Solomon codes [RS60] define one of the most famous family of evaluation codes. We here provide a definition of their generalised version in both vector and functional representations.

Definition 1.1 (Reed-Solomon codes)**.** Let $S \subseteq \mathbb{F}_q$ and $\boldsymbol{y} \in (\mathbb{F}_q^\times)^S$. Assume that $n = |S| > 0$ and let $0 \leq k \leq n-1$. The associated *generalised Reed-Solomon code* (GRS code) of degree k is

$$\mathrm{GRS}_q(k, S, \boldsymbol{y}) \coloneqq \{(y_x f(x) : x \in S), f \in \mathbb{F}_q[X], \deg f \leq k\} \subseteq \mathbb{F}_q^S.$$

Subscript q can be omitted for convenience. The set S is the set of *evaluation points*, while $\boldsymbol{y}$ consists in the *column multipliers*. If $\boldsymbol{y} = \mathbf{1}$, then the code is simply called a *Reed-Solomon code* and denoted $\mathrm{RS}(k, S)$. Furthermore, if $S = \mathbb{F}_q$, then we get a *full-length Reed-Solomon code*, denoted $\mathrm{RS}(k)$.

In the functional representation, if $g \in \mathbb{F}_q^S$ satisfies $g(x) = y_x$ for every $x \in S$, then we have $\mathrm{GRS}_q(k, S, \boldsymbol{y}) = \{g \cdot f_{|S}, f \in \mathbb{F}_q[X]_{\leq k}\} \subseteq \mathbb{F}_q^S$. Let us also list several useful properties of GRS codes:

1. If $|S| = n$, then $\mathrm{GRS}(k, S, \boldsymbol{y})$ is an $[n, k+1, n-k]$ MDS code.
2. We have $\mathrm{GRS}(k, S, \boldsymbol{y})^\perp = \mathrm{GRS}(n-k-1, S, \boldsymbol{y}')$, where $y'_x = (y_x \prod_{z \in S \setminus \{x\}} (x - z))^{-1}$.
3. $\mathrm{GRS}(0, S, \boldsymbol{y})$ is the $\boldsymbol{y}$-repetition code, while $\mathrm{GRS}(n-1, S, \boldsymbol{y})$ is the $\boldsymbol{y}'$-parity-check code, where $\boldsymbol{y}'$ is defined above.
4. When $S = \mathbb{F}_q$, we have $\mathrm{Aff}(\mathbb{F}_q) \subseteq \mathrm{Aut}(\mathrm{RS}(k))$.

GRS codes also admit efficient correction algorithms. Precisely, one can correct any pattern of e erasures and v errors, as long as $e + 2v \leq d - 1$, where d is the minimum distance of the GRS code. Also notice that, since GRS codes are MDS, a strategy could consist in (i) first correcting errors in a punctured code of length $n - e$ and (ii) recovering the e erasures by interpolation. For the sake of completeness, we give two references: [RTM79], which is one of the first efficient error-and-erasure correcting algorithm for RS codes; and [Gao03], one of the currently most efficient half-distance decoding algorithms for GRS codes.

1.2.2 Reed-Muller codes

The m-variate generalisation of Reed-Solomon codes are called *Reed-Muller codes*. They were first designed over the binary field, then generalised over any finite field.

Definition 1.2 (Reed-Muller codes)**.** Let $m \geq 1$ and $0 \leq r \leq m(q-1)$. The q-ary *Reed-Muller code* (RM code) of order m and degree r is

$$\mathrm{RM}_q(m, r) \coloneqq \{(f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{F}_q^m), f \in \mathbb{F}_q[X_1, \ldots, X_m], \deg f \leq r\}.$$

Notice that $\mathrm{RM}_q(m, r) = \mathrm{ev}_{\mathbb{F}_q^m}(\mathbb{F}_q[\boldsymbol{X}]_{\leq r})$, where $\mathbb{F}_q[\boldsymbol{X}]_{\leq r}$ denotes the space of m-variate polynomials of *total* degree bounded by r. Unfortunately one cannot derive parameters of Reed-Muller as easily as those of Reed-Solomon codes, since $\mathrm{ev}_{\mathbb{F}_q^m}$ is not injective for r larger than q.

Still, if $r \leq q-1$, then $\mathrm{ev}_{\mathbb{F}_q^m}$ is injective over $\mathbb{F}_q[\boldsymbol{X}]_{\leq r}$, hence $\dim(\mathrm{RM}_q(m, r)) = \binom{m+r}{m}$. Furthermore, in that case the minimum distance of $\mathrm{RM}_q(m, r)$ is $(q-r)q^{m-1}$. In the next chapter, we will see that this low-degree setting provides Reed-Muller codes local correcting properties.

For $r \geq q$, the evaluation map $\mathrm{ev}_{\mathbb{F}_q^m}$ is not injective anymore. Though, its kernel is generated by $\{X_i^q - X_i\}_{1 \leq i \leq m}$, which corresponds to bounding by $q-1$ the partial degrees of the polynomials to be evaluated. Then it can be proved [AK92] that

$$\dim(\mathrm{RM}_q(m, r)) = \sum_{i=0}^{r} \sum_{j=0}^{m} (-1)^j \binom{m}{j} \binom{i - jq + m - 1}{i - jq},$$

and if we write $r = a(q-1) + b$ for $0 \leq b < q-1$, we also have

$$\mathrm{d}_{\min}(\mathrm{RM}_q(m,r)) = (q-b)q^{m-1-a}\,.$$

Remark 1.3. The Reed-Muller code $\mathrm{RM}_q(m,r)$ can be seen as a *subfield subcode* of a Reed-Solomon code defined over the extension field $\mathbb{F}_{q^m}$. Precisely, up to isomorphism, $\mathrm{RM}_q(m,r)$ is included into $\mathrm{RS}_{q^m}(q^m-d) \cap \mathbb{F}_q^{\mathbb{F}_{q^m}}$, where $d = \mathrm{d}_{\min}(\mathrm{RM}_q(m,r))$. See for instance [KLP68] for details on this embedding.

A non-trivial consequence of Remark 1.3 is that one can employ decoding/correcting algorithm of Reed-Solomon codes in order to decode/correct Reed-Muller codes. Marvellously, one sees that related RM codes and RS codes *have the same minimum distance*. As a consequence, Reed-Muller codes are efficiently correctable up to half their minimum distance. Moreover, Pellikaan and Wu [PW04] used this property to obtain an efficient *list-decoding* algorithm for Reed-Muller codes.

1.2.3 Hadamard codes

Hadamard codes, also known as *simplex codes*, have been studied for a long time for their structural properties. They can be defined as evaluation codes of linear forms over the set of non-zero elements of the binary affine space.

Definition 1.4 (Hadamard code)**.** Let $m \geq 2$ and $S = \mathbb{F}_2^m \setminus \{\mathbf{0}\}$. The binary Hadamard code of order m is:

$$\mathrm{Had}_2(m) = \mathrm{Span}_{\mathbb{F}_2}\{\mathrm{ev}_S(X_i), 1 \leq i \leq m\}\,.$$

The code $\mathrm{Had}_2(m) \subseteq \mathbb{F}_2^S$ has length $|S| = 2^m - 1$, dimension m and minimum distance 2^{m-1}, since every non-zero linear form vanishes on a hyperplane. We will see later that Hadamard codes are related to combinatorial constructions and satisfy nice *local* properties.

Notice that q-ary $[\frac{q^m-1}{q-1}, m, q^{m-1}]$ Hadamard codes $\mathrm{Had}_q(m)$ can also be defined, through the evaluation of linear forms over the q-ary projective space $\mathbb{P}^{m-1}(\mathbb{F}_q)$.

1.3 Combinatorial constructions through block designs

In this section, we recall basic notions of design theory and how to build codes from block designs. Details can be found in the following books [AK92, Sti04].

1.3.1 Designs

Definition 1.5 (Block design)**.** A *block design*, or simply *design*, is a pair $\mathcal{D} = (X, \mathcal{B})$, where X is a finite set and $\mathcal{B} \neq \varnothing$ is a collection of non-empty subsets of X. Elements of X and $\mathcal{B}$ are respectively named *points* and *blocks*.

Designs are usually considered with incidence constraints between points and blocks. For instance, we can require that each pair of points appears the same number of times in the set of blocks; we then obtain *balanced incomplete block designs*.

Definition 1.6 (BIBD)**.** Let $\lambda \geq 1$, and $2 \leq k < v$. A (v, k, λ)-*balanced incomplete block design* (BIBD) is a design $(X, \mathcal{B})$ such that:

- $|X| = v$;
- for all $B \in \mathcal{B}$, $|B| = k$;
- each pair of points appears in exactly λ blocks.

Let us give some examples of BIBDs.

Example 1.7. Let $2 \leq k \leq v$ and X a set of cardinality v. If $\mathcal{B}$ is the set of all k-subsets of X, then $(X, \mathcal{B})$ defines a $(v, k, \binom{v-2}{k-2})$-BIBD.

Example 1.8 (Fano plane)**.** The *Fano plane* is a $(7, 3, 1)$-BIBD. Recall it is defined as follows. If we denote its points by $X = \{1, 2, 3, 4, 5, 6, 7\}$, then its block set is:

$$\mathcal{B} = \{\{1, 2, 3\}, \{1, 4, 5\}, \{1, 6, 7\}, \{2, 5, 6\}, \{2, 4, 7\}, \{3, 4, 6\}, \{3, 5, 7\}\}.$$

We give a representation of the Fano plane in Figure 1.1. Notice that it corresponds to the incidence structure of points and lines in the projective plane $\mathbb{P}^2(\mathbb{F}_2)$. More generally, the projective plane $\mathbb{P}^2(\mathbb{F}_q)$ gives rise to a $(q^2 + q + 1, q + 1, 1)$-BIBD.

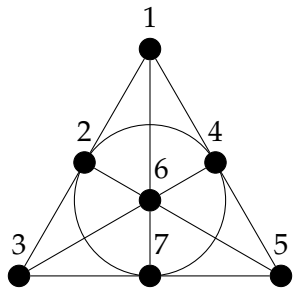


Figure 1.1 – Fano plane.

Example 1.9 (classical affine and projective space designs)**.** BIBDs can also be defined in higher dimensional ambient spaces. For instance:

- If $X = \mathbb{A}^m(\mathbb{F}_q)$ and $\mathcal{B}$ is the set of affine lines of $\mathbb{A}^m(\mathbb{F}_q)$, then $(X, \mathcal{B})$ defines a $(q^m, q, 1)$-BIBD. It is called the design of the *classical affine space* and denoted $\mathrm{AG}_1(m, q)$.
- Similarly, $X = \mathbb{P}^m(\mathbb{F}_q)$ equipped with the set $\mathcal{B}$ of its projective lines, defines a $((q^{m+1} - 1)/(q - 1), q + 1, 1)$-BIBD called the design of the *classical projective space* and denoted $\mathrm{PG}_1(m, q)$.
- The *classical affine design of s-flats* $\mathrm{AG}_s(m, q)$ can also be defined: it consists in points and subspaces of dimension s in the affine space of dimension m. It is a $(q^m, q^s, \begin{bmatrix} m-1 \\ s-1 \end{bmatrix}_q)$ BIBD, where, for any $a \geq b$, the *Gaussian coefficient* $\begin{bmatrix} a \\ b \end{bmatrix}_q$ is defined by

$$\begin{bmatrix} a \\ b \end{bmatrix}_q := \frac{(q^a - 1)(q^{a-1} - 1) \dots (q^{a-b+1} - 1)}{(q^b - 1)(q^{b-1} - 1) \dots (q - 1)}$$

  if $b \neq 0$, and by $\begin{bmatrix} a \\ 0 \end{bmatrix}_q := 1$.
- Similarly, its projective analogue $\mathrm{PG}_s(m, q)$ is the *classical projective design of s-flats*. It is a $(\theta_{m,q}, \theta_{s,q}, \begin{bmatrix} m-1 \\ s-1 \end{bmatrix}_q)$ BIBD, where $\theta_{a,q} := \frac{q^{a+1}-1}{q-1}$ is the number of points in a projective space of dimension a.

Designs of higher order. BIBDs naturally generalise by requiring that each t-subset of points appears in exactly λ blocks, for some $t \geq 1$. These constructions are termed t-(v,k,λ)-*designs*. Thus a BIBD is a 2-design. Let us report the following well-known results (see [Sti04, Theorem 9.4] and its corollaries for instance).

Theorem 1.10. *Let $\mathcal{D} = (X, \mathcal{B})$ be a t-(v,k,λ)-design for $t \geq 1$. Then, for $0 \leq s \leq t$, each s-subset appears in exactly λ_s blocks, where*

$$\lambda_s = \lambda \frac{\binom{v-s}{t-s}}{\binom{k-s}{t-s}}.$$

As a corollary,

1. *if $s \geq 1$, then $\mathcal{D}$ is also an s-(v,k,λ_s)-design,*
2. *there are exactly $\lambda\binom{v}{t}/\binom{k}{t}$ blocks in $\mathcal{B}$.*

1.3.2 Design-based codes

In this section, we present how one can build linear codes from designs, and we discuss their properties. For this purpose, we need to introduce a data structure that stores incidences between points and blocks, namely the *incidence matrix of the design*.

Definition 1.11 (incidence matrix)**.** Let $\mathcal{D} = (X, \mathcal{B})$ be a design. The *incidence matrix* of $\mathcal{D}$ over $\mathbb{F}_q$, denoted $\mathrm{Mat}_q(\mathcal{D})$, is the matrix $\boldsymbol{M}$ whose rows and columns are respectively indexed by blocks and points, and such that for all $x \in X, B \in \mathcal{B}$,

$$M_{B,x} = \begin{cases} 1 & \text{if } x \in B, \\ 0 & \text{otherwise.} \end{cases}$$

Definition 1.12 (design-based code)**.** Let $\mathcal{D} = (X, \mathcal{B})$ be a design. The *design-based code* over $\mathbb{F}_q$ associated to $\mathcal{D}$, denoted $\mathrm{Code}_q(\mathcal{D})$, is the $\mathbb{F}_q$-linear code $\mathcal{C} \subseteq \mathbb{F}_q^X$ admitting $\boldsymbol{M} = \mathrm{Mat}_q(\mathcal{D})$ as a parity-check matrix.

A crucial property of design-based codes is the following. Let $\mathcal{C} = \mathrm{Code}_q(\mathcal{D})$ where $\mathcal{D} = (X, \mathcal{B})$. Then $\boldsymbol{c} \in \mathbb{F}_q^X$ lies in $\mathcal{C}$ if and only if for every $B \in \mathcal{B}$ we have $\sum_{b \in B} c_b = 0$.

Notice that $\mathrm{Code}_q(\mathcal{D}) \subseteq \mathbb{F}_q^X$ has length $|X|$ and dimension $|X| - \mathrm{rank}_{\mathbb{F}_q}(\boldsymbol{M})$, where $\boldsymbol{M} = \mathrm{Mat}_q(\mathcal{D})$. Furthermore, if $\mathbb{F}_q$ is an extension of $\mathbb{F}_p$, then $\mathrm{Code}_q(\mathcal{D}) = \mathrm{Span}_{\mathbb{F}_q}(\mathrm{Code}_p(\mathcal{D}))$.

Here one should emphasize that $\boldsymbol{M}$ is a $\{0,1\}$-matrix, but its rank highly depends on the choice of the base field. To see this, let us first view $\boldsymbol{M}$ as a matrix over $\mathbb{Z}$, and consider $\Gamma \subset \mathbb{Z}$ the multiset of elementary divisors of the matrix $\boldsymbol{M}$, counted with multiplicity. The rank of $\boldsymbol{M}$ over $\mathbb{F}_p$ is then the number of elements $g \in \Gamma$ such that $p \nmid g$. It now clear that $\mathrm{rank}_{\mathbb{F}_p}(\boldsymbol{M})$ depends on p, but one should also remark that the rank stabilizes when $p \to \infty$.

Example 1.13 (Fano plane, continued)**.** The Fano plane $\mathcal{D}$ admits the incidence matrix

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

where columns and rows are listed according to the order of points and blocks given in Example 1.8. A computation can show that the multiset of elementary divisors of $\boldsymbol{M}$ is $\Gamma = \{1^4, 2^2, 6^1\}$, where g^i means that the element $g \in \mathbb{Z}$ is counted with multiplicity i. Hence, matrix $\boldsymbol{M}$ has rank 4 over $\mathbb{F}_2$, rank 6 over $\mathbb{F}_3$, and rank 7 over any other prime finite field $\mathbb{F}_p$. In other words, $\mathrm{Code}_p(\mathcal{D})$ is the zero code for all primes $p \notin \{2, 3\}$, and $\mathrm{Code}_3(\mathcal{D})$ is the $[7, 1, 7]$ repetition code over $\mathbb{F}_3$. Fortunately, over $\mathbb{F}_2$ (and its extensions) $\mathrm{Code}_2(\mathcal{D})$ presents more interesting properties; specifically, one can prove it is isomorphic to the binary $[7, 3, 4]$ Hadamard code $\mathrm{Had}_2(3)$ (see Proposition 2.28).

On the dimension of design-based codes. Finding the rank of a single design over a fixed field — or equivalently the dimension of its associated code — is a simple task if the parameters of the design are small enough. Indeed, it only consists in computing the rank of a matrix whose size is $|\mathcal{B}| \times |X|$. However, deriving a formula for the rank of a family of designs appears to be much harder. Especially, little is known about families of designs with low-rank incidence matrices (equivalently, families of high-rate design-based codes).

For 2-designs, one can first report negative results.

Proposition 1.14 (Hamada [Ham68])**.** *Let $\mathcal{D}$ be a 2-(v, k, λ)-design, and denote by $r = \lambda\frac{v-1}{k-1}$ its* replication number*. Let also p be a prime number. Then,* $\dim(\mathrm{Code}_p(\mathcal{D})) \geq 2$ *implies that $p \mid r - \lambda$. Precisely,*

1. *if $p \nmid r - \lambda$ and $p \mid r$, then* $\dim(\mathrm{Code}_p(\mathcal{D})) \leq 1$;
2. *if $p \nmid r(r - \lambda)$, then* $\dim(\mathrm{Code}_p(\mathcal{D})) = 0$.

Classical affine and projective designs over $\mathbb{F}_{p^e}$ satisfy $p \mid r - \lambda$. Therefore it is of interest to study them over $\mathbb{F}_p$. It is proved that the design of points and hyperplanes in the projective space satisfies:

$$\mathrm{rank}_p(\mathrm{PG}_{m-1}(m, p^e)) = \binom{p + m - 1}{m}^e + 1\,.$$

This result has been found independently by Smith [Smi69], by Goethals and Delsarte [GD68], and by MacWilliams and Mann [MM68]. Hamada generalised it for subspaces of any dimension, leading us to the famous Hamada's formula given in next theorem.

Theorem 1.15 (Hamada's formula [Ham68])**.** *The p-rank of the projective geometry design $\mathrm{PG}_t(m, p^e)$ is given by*

$$\sum_{(s_0,\ldots,s_e)\in S} \prod_{j=0}^{e-1} \sum_{i=0}^{L(s_{j+1},s_j)} (-1)^i \binom{m+1}{i} \binom{m + s_{j+1}p - s_j - ip}{m} \tag{1.1}$$

where $L(s_{j+1}, s_j) = \lfloor \frac{s_{j+1}p - s_j}{p} \rfloor$ and S is the set of tuples $\boldsymbol{s} = (s_0, \ldots, s_e) \in \mathbb{Z}^{e+1}$ such that:

$$\begin{cases} s_0 = s_e, \\ t + 1 \leq s_j \leq m + 1, \\ 0 \leq s_{j+1}p - s_j \leq (m + 1)(p - 1)\,. \end{cases}$$

Notice that ranks of affine geometry designs can also be computed thanks to another result due to Hamada [Ham68]:

$$\mathrm{rank}_p\,\mathrm{AG}_t(m, p^e) = \mathrm{rank}_p\,\mathrm{PG}_t(m, p^e) - \mathrm{rank}_p\,\mathrm{PG}_t(m - 1, p^e)\,. \tag{1.2}$$

Though Equation (1.1) is quite cumbersome, one can highlight a few important consequences.

If we fix m and t and let e grow, we get families of codes with increasing rate. Let us be more specific with the case $t = 1$ and $m = 2$. Hamada's formula gives:

$$\operatorname{rank}_p \mathrm{AG}_1(2, p^e) = \binom{p+1}{2}^e \quad \text{and} \quad \operatorname{rank}_p \mathrm{PG}_1(2, p^e) = \binom{p+1}{2}^e + 1\,.$$

Roughly, it means that when p is fixed and $e \to \infty$, the rate of the associated codes is $1 - \mathcal{O}(n^{-\log_p(e)})$ where $n = \Theta(p^{2e})$ is the code length. It means that, in the setting $(m, t) = (2, 1)$, codes based on geometric designs reach rates arbitrarily close to 1.

If we fix m and e, we remark that $p \mapsto \operatorname{rank}_p(\mathrm{AG}_1(2, p^e))$ is a polynomial in p of degree at most me. Hence, it can be computed by interpolation (or by expanding Equation (1.1)). For instance, in the case $m = 3$, $e = 2$ we can find:

$$\operatorname{rank}_p \mathrm{AG}_1(3, p^2) = \left(p^3 - \binom{p+1}{3}\right)^2 + 2\binom{p}{2}\binom{p+1}{3}\,.$$

Finally, if we fix t and $q = p^e$, the dimension of the code $\mathcal{C}_m = \mathrm{Code}_p(\mathrm{PG}_t(m, q))$ is a polynomial in m of degree less than $(q-1)t$, as it is proved by Calkin, Key and de Resmini [CKdR99]. Since its length is exponential in m, it means that the family of codes $\{\mathcal{C}_m\}_m$ has vanishing rate.

Chapter 2

Locally decodable and correctable codes

Contents

Classical correcting algorithms allow us to retrieve a codeword $\boldsymbol{c}$ from a damaged version $\boldsymbol{y}$ of $\boldsymbol{c}$. They usually take as input the entire noisy word $\boldsymbol{y}$ and return, hopefully, the entire codeword $\boldsymbol{c}$.

Assume now that one wants to retrieve only one symbol c_i of $\boldsymbol{c}$. Of course, one can use the previous algorithm which returns $\boldsymbol{c}$ entirely, and then on can extract the desired symbol c_i. However, the complexity of this method is at least linear in the length of the code. Local correcting algorithms were introduced in order to reach a better computation complexity for this problem. Explicitly, they aim at retrieving a correct symbol from a damaged codeword, with high probability (w.h.p.) and in sublinear time.

In this chapter, we recall formal definitions of *locally decodable* and *locally correctable* codes. We then present algebraic constructions in the high-rate regime, which is of main interest for our applications. We also review current bounds and issues in this area. Next, we show how

locally correctable codes (LCCs) can be built from block designs. Finally, after introducing *generalised design-based codes*, we discuss to what extent they can lead us to high-rate LCCs.

2.1 Motivation and definitions

2.1.1 Historical points

The notion of locality in codes appeared in computational theory in the early 1990s. It was initially raised in the scope of testing computations and self-correcting programs, see *e.g.* [BLR93, BFLS91, GLR$^+$91, GS92]. A typical question was to output w.h.p. the image $f(x)$ of $x \in S$ by a low-degree polynomial f, given a program P that computes $f(y)$ correctly for only a constant fraction of entries $y \in S$.

Later, codes with locality also turned out to be useful to obtain short probabilistic checkable proofs (PCPs). Very informally, assume one wants to assess w.h.p. whether a long string $\boldsymbol{x}$ belongs to some language L, with the restriction that one can make only a small number of *oracle queries*[1] to $\boldsymbol{x}$ or to another long string derived from $\boldsymbol{x}$. Generically, if we directly access $\boldsymbol{x}$, the partial information we get cannot be sufficient to determine whether $\boldsymbol{x} \in L$. Thus, a natural idea is to encode $\boldsymbol{x}$ into a longer *proof* $\boldsymbol{\pi}$, so that a few oracle queries to $\boldsymbol{\pi}$ assert w.h.p. if $\boldsymbol{\pi}$ corresponds to an encoding of an element of L. This leads us to the notion of *locally testable codes* which were a key element for proving the well-known PCP Theorem [AS98, ALM$^+$98].

A few years later, Katz and Trevisan [KT00] formally introduced so-called *locally decodable codes* in an information theoretic perspective. This seminal work was motivated by sublinear-time data recovery in large corrupted databases. The authors proved that codes equipped with a local decoding procedure must be somewhat *smooth*, in the sense that queries should be sufficiently balanced. They also gave first lower bounds on the number of redundancy symbols needed in the codeword, and they proposed an application to private information retrieval protocols — see Chapter 4 for more details.

2.1.2 Definitions

Let us formally introduce locally correctable and decodable codes, as defined in Yekhanin's survey [Yek12]. In this thesis, an *oracle query to a string* $\boldsymbol{y} \in \Sigma^S$ refers to a map $\boldsymbol{y} \mapsto y_i$ for some $i \in S$. One usually considers that the computational cost of such a map is 1, and the index i is sometimes called a *query*. If an algorithm A is only allowed to access a string $\boldsymbol{y}$ *via* oracle queries, we say it has *oracle access* to $\boldsymbol{y}$, and we represent it by $\mathsf{A}^{(\boldsymbol{y})}$. Notice that, if the string $\boldsymbol{y} \in \Sigma^S$ is seen as a map $g_{\boldsymbol{y}} : S \to \Sigma$, then an oracle query is purely the evaluation of $g_{\boldsymbol{y}}$ at $i \in S$.

Definition 2.1 (locally correctable code, or LCC)**.** Let $\mathcal{C} \subseteq \Sigma^S$ be an $\mathbb{F}_q$-linear code, $|S| = n$. Let also $1 \leq \ell \leq n$, $\delta \in (0,1)$ and $\varepsilon < 1/2$. We say that $\mathcal{C}$ is an $(\ell, \delta, \varepsilon)$-*locally correctable code* (LCC) if there exists a randomised algorithm LC, taking as input $i \in S$ and having oracle access to words $\boldsymbol{y} \in \Sigma^S$, which satisfies the following requirements. For every $\boldsymbol{y} \in \Sigma^S$ and $\boldsymbol{c} \in \mathcal{C}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq \delta n$, and for every $i \in S$, we have:

- $\Pr(\mathsf{LC}^{(\boldsymbol{y})}(i) = c_i) \geq 1 - \varepsilon$, the probability being taken over the internal randomness of LC, and

[1] an oracle query is a map $\boldsymbol{y} \mapsto y_i$ for some index i, see Subsection 2.1.2

– $\mathsf{LC}^{(\boldsymbol{y})}(i)$ queries at most ℓ symbols of $\boldsymbol{y}$.

We refer to ℓ as the *locality* of the code $\mathcal{C}$, and to LC as the *local correcting algorithm*.

In the literature, it is usually considered sufficient to have a failure probability $\varepsilon \leq 1/3$. The idea is that, by repeating the local correcting algorithm LC several times on the same input i, one can attain values of ε exponentially small in the number of procedures we run.

Definition 2.2 (locally decodable code, or LDC)**.** Let $\mathcal{C} \subseteq \Sigma^S$ be an $\mathbb{F}_q$-linear code, $|S| = n$, equipped with an encoder $E : \mathbb{F}_q^k \to \mathcal{C}$. Let also $1 \leq \ell \leq k$, $\delta \in (0,1)$ and $\varepsilon < 1/2$. We say that $\mathcal{C}$ is an $(\ell, \delta, \varepsilon)$-*locally decodable code* (LDC) if there exists a randomised algorithm LD, taking as input $i \in [1,k]$ and having oracle access to words $\boldsymbol{y} \in \Sigma^S$, which satisfies the following requirements. For every $\boldsymbol{y} \in \Sigma^S$ and $\boldsymbol{m} \in \mathbb{F}_q^k$ such that $\mathrm{d}(\boldsymbol{y}, E(\boldsymbol{m})) \leq \delta n$, and for every $i \in [1,k]$, we have:

– $\Pr(\mathsf{LD}^{(\boldsymbol{y})}(i) = m_i) \geq 1 - \varepsilon$, the probability being taken over the internal randomness of LD, and
– $\mathsf{LD}^{(\boldsymbol{y})}(i)$ queries at most ℓ symbols of $\boldsymbol{y}$.

Once again, ℓ is the *locality* of the code $\mathcal{C}$, and LD is referred to as a *local decoding algorithm*.

Remark 2.3. Let $\mathcal{C} \subseteq \Sigma^S$ be an $(\ell, \delta, \varepsilon)$-LCC with $\ell \leq k = \dim(\mathcal{C})$. If we know a systematic encoder $E : \mathbb{F}_q^k \to \mathcal{C}$, then the code $\mathcal{C}$ is also $(\ell, \delta, \varepsilon)$-locally *decodable*. Indeed, any message symbol m_i, $1 \leq i \leq k$, is also a codeword symbol c_j for some $j \in S$, and the local correcting algorithm $\mathcal{C}$ thus becomes a local decoding algorithm.

For $\gamma \in (0,1)$, we say that a locally correctable code $\mathcal{C} \subseteq \Sigma^S$ is γ-*smooth* if, for all $i, j \in S$, algorithm $\mathsf{LC}(i)$ queries y_j with probability at most γ. Moreover, $\mathcal{C}$ is called *perfectly smooth* if symbols y_j are queried with equal probability. Hence, a perfectly smooth LCC is (ℓ/n)-smooth, but the converse may be false when some vectors of queries have size strictly less than ℓ. Smooth LDCs can be defined very similarly to smooth LCCs, the only difference being that $i \in [1,k]$ is a message coordinate instead of a codeword coordinate.

In fact, any LCC intrinsically admits a certain smoothness, as we show in the following result inspired by [KT00, Theorem 1].

Proposition 2.4. *Let $\mathcal{C} \subseteq \Sigma^S$ be an $(\ell, \delta, \varepsilon)$-LCC of length n. For every $\gamma > \frac{\ell}{\delta n}$, the code $\mathcal{C}$ is also a γ-smooth $(\ell, \delta - \frac{\ell}{\gamma n}, \varepsilon)$-LCC.*

Proof. Let $\gamma > \frac{\ell}{\delta n}$ and $\delta' = \delta - \frac{\ell}{\gamma n}$, and denote by LC the local correcting algorithm for $\mathcal{C}$. Relying on LC, we will describe another local correcting algorithm LC_γ for $\mathcal{C}$, with the additional γ-smooth property.

Let $\boldsymbol{y} \in \Sigma^S$ and $\boldsymbol{c} \in \mathcal{C}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq \delta' n$. On input $i \in S$, define

$$S_i := \{j \in S \mid \Pr(\mathsf{LC}^{(\boldsymbol{y})}(i) \text{ reads } y_j) > \gamma\}$$

as the set of coordinates which are queried too often. Let us now consider a new word $\boldsymbol{y}' \in \Sigma^S$, defined for all $j \in S$ as follows:

$$y'_j = \begin{cases} 0 & \text{if } j \in S_i \\ y_j & \text{otherwise.} \end{cases}$$

The new algorithm LC_γ is simply defined by $\mathsf{LC}_\gamma^{(\boldsymbol{y})}(i) := \mathsf{LC}^{(\boldsymbol{y}')}(i)$. Let us analyse its properties.

As we removed all queries that LC makes with probability larger than γ, the algorithm LC_γ is γ-smooth. Moreover, we can easily check that $|S_i| \leq \ell/\gamma$. Hence $\mathrm{d}(\boldsymbol{y}', \boldsymbol{y}) \leq \ell/\gamma$, and $\mathrm{d}(\boldsymbol{y}', \boldsymbol{c}) \leq \ell/\gamma + \delta' n = \delta n$ follows. Therefore $\mathsf{LC}_\gamma^{(\boldsymbol{y})}(i)$ outputs c_i with probability $\geq 1 - \varepsilon$ as expected. □

The previous proposition may question the relevance of focusing on the smoothness of LCCs. However, we see that we cannot make any LCC *perfectly* smooth, since δ is strictly less than 1. Therefore, searching for perfectly smooth LCCs remains a very relevant goal.

2.1.3 A first example: the binary Hadamard code

The binary Hadamard code $\mathrm{Had}_2(m)$ is a classical example of locally correctable code — see Subsection 1.2.3 for a definition. Since the code consists in evaluation vectors of linear forms over $\mathbb{F}_2^m$, the 3 non-zero points of any plane give rise to a parity-check equation for $\mathrm{Had}_2(m)$. Such equations can then be exploited to recover any symbol with only 2 queries.

For convenience we here adopt the functional representation. Formally, denote by $S = \mathbb{F}_2^m \setminus \{\mathbf{0}\}$ and let $f \in \mathcal{C}$ (that is, f is a linear form over $\mathbb{F}_2^m$). For all $\boldsymbol{u} \neq \boldsymbol{v} \in S$, we see that $\boldsymbol{u} + \boldsymbol{v} \in S$, and the fact that $f(\boldsymbol{u}) + f(\boldsymbol{v}) = f(\boldsymbol{u} + \boldsymbol{v})$ leads us to the following local correcting algorithm.

Algorithm 1: A smooth $(2, \delta, 2\delta)$-local correcting algorithm for $\mathrm{Had}_2(m)$.

Input: a point $\boldsymbol{u} \in S := \mathbb{F}_q^m \setminus \{\mathbf{0}\}$, and an oracle access to $g \in \mathbb{F}_2^S$ such that $\mathrm{d}(g, f) \leq \delta n$ for some $f \in \mathrm{Had}(m)$, where $n = |S|$.
Output: $f(\boldsymbol{u})$ with high probability.
1 Pick $\boldsymbol{v} \leftarrow_{\mathrm{R}} S \setminus \{\boldsymbol{u}\}$ uniformly at random.
2 Toss a random binary coin $b \in \{0, 1\}$, following $\mathscr{B}\left(\frac{2}{n+1}\right)$.
3 **if** $b = 1$ **then**
4 Query $\{g(\boldsymbol{u})\}$ and output $g(\boldsymbol{u})$.
5 **else**
6 Query $\{g(\boldsymbol{v}), g(\boldsymbol{u} + \boldsymbol{v})\}$ and output $g(\boldsymbol{u} + \boldsymbol{v}) - g(\boldsymbol{v})$.

Algorithm 1 is often presented with steps 1 and 6 only. In that case, coordinate $\boldsymbol{u}$ is never queried. We here add the tossing trick in order to make the algorithm perfectly smooth, as we can see in the following proposition.

Proposition 2.5. *Let $\delta < 1/4$, and $m \geq 2$. By using Algorithm 1, the Hadamard code $\mathrm{Had}_2(m)$ is a perfectly smooth $(2, \delta, 2\delta)$-locally correctable code.*

Proof. We first check that evaluations of g are queried with equal probability. We see that $g(\boldsymbol{u})$ is read with probability $2/(n+1)$, and $g(\boldsymbol{v})$, $\boldsymbol{v} \in S \setminus \{\boldsymbol{u}\}$, is read with probability

$$\left(1 - \frac{2}{n+1}\right) \times 2 \times \frac{1}{n-1} = \frac{2}{n+1}.$$

Let us now prove that the output is correct with probability $\geq 1 - 2\delta$, and denote by $E = \mathrm{supp}(f - g)$ the support of the errors (we have $|E| \leq \delta n$). We also define $S_{\boldsymbol{u}}$ as the (random) support of the query made by $\mathsf{LC}^{(g)}(\boldsymbol{u})$. Notice that $\Pr(\boldsymbol{v} \in S_{\boldsymbol{u}}) = 2/(n+1)$ for every $\boldsymbol{v} \in S$. From Algorithm 1 we see that $S_{\boldsymbol{u}}$ can be either $\{\boldsymbol{u}\}$ or $\{\boldsymbol{v}, \boldsymbol{u} + \boldsymbol{v}\}$ for some $\boldsymbol{v} \neq \boldsymbol{u}$. If

$S_{\boldsymbol{u}} = \{\boldsymbol{u}\}$, then the local correcting algorithm $\mathsf{LC}^{(g)}(\boldsymbol{u})$ outputs $f(\boldsymbol{u})$ if and only if $\boldsymbol{u} \notin E$. If $S_{\boldsymbol{u}} = \{\boldsymbol{v}, \boldsymbol{v}+\boldsymbol{u}\}$, then $\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})$ if and only if $|E \cap S_{\boldsymbol{u}}| \in \{0,2\}$, in other words, if and only if $E \cap S_{\boldsymbol{u}} = \varnothing$ or S_u.

Thus, the probability of success of $\mathsf{LC}^{(g)}(\boldsymbol{u})$ is given by:

$$\Pr(\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})) = 1 - \Pr(|E \cap S_{\boldsymbol{u}}| = 1) \geq 1 - \Pr(|E \cap S_{\boldsymbol{u}}| \geq 1) \geq 1 - \mathbb{E}(|E \cap S_{\boldsymbol{u}}|)\,.$$

Furthermore, by linearity we obtain

$$\mathbb{E}(|E \cap S_{\boldsymbol{u}}|) = \sum_{\boldsymbol{v} \in E} \Pr(\boldsymbol{v} \in S_{\boldsymbol{u}}) = \sum_{\boldsymbol{v} \in E} \frac{2}{n+1} = |E|\frac{2}{n+1} \leq \delta n \frac{2}{n+1} < 2\delta\,. \qquad \square$$

Recall that $\mathrm{Had}_2(m)$ has dimension m. Denote by $\{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\}$ (resp. $\{X_1, \ldots, X_m\}$) the canonical basis of the affine space $\mathbb{F}_2^m$ (resp. the space of linear forms over $\mathbb{F}_2^m$). The information set $I = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_m\} \subset S$ induces a systematic encoder $E : \mathbb{F}_2^m \to \mathbb{F}_2^S$ for $\mathrm{Had}_2(m)$, given by $E(m_i) = \mathrm{ev}_S(X_i)$ for every $\boldsymbol{m} \in \mathbb{F}_2^m$. Therefore $\mathrm{Had}_2(m)$ is also a $(2, \delta, 2\delta)$-LDC.

Quantitatively, Hadamard codes define a family of binary LCCs with increasing length $2^m - 1$ and constant query size 2. However, the dimension m of these codes is only logarithmic in their length. In the following section, we will see that a certain class of Reed-Muller codes defines another family of LCCs, achieving constant rate provided an increase of the query size.

2.2 Some algebraic constructions of LCCs

This section is devoted to presenting families of locally correctable codes that are built by evaluating polynomials over vector spaces. In Subsection 2.2.1, we detail two smooth local correcting algorithms for Reed-Muller codes. We show how perfect smoothness can be achieved, and we depict qualitative relations between the three parameters of LCCs. Next subsections are then devoted to more evolved algebraic constructions leading to *high-rate* LCCs, namely *multiplicity codes* and *lifted codes*.

2.2.1 Reed-Muller codes

Reed-Muller codes have been defined in Subsection 1.2.2. In this subsection, we let $\mathcal{C} = \mathrm{RM}_q(m, r)$ for $0 \leq r \leq m(q-1)$. Informally, we see that if r is small enough, then restrictions of $\mathcal{C}$ to subspaces $A \subseteq \mathbb{F}_q^m$ define non-degenerate codes, which consist in the evaluation vectors of low degree polynomials over A. This key property will be used for local correcting purposes. Before going deeper into details, we state a few results which are straightforward to prove.

Lemma 2.6. *Let $1 \leq t \leq m$ and $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$ be an injective affine map.*

1. *If $f \in \mathbb{F}_q[X_1, \ldots, X_m]$, $\deg(f) \leq r$, then there exists $g \in \mathbb{F}_q[X_1, \ldots, X_t]$, $\deg(g) \leq r$, such that*
$$\forall \boldsymbol{u} \in \mathbb{F}_q^t, (f \circ \phi)(\boldsymbol{u}) = g(\boldsymbol{u})\,. \tag{2.1}$$
2. *Conversely, if $g \in \mathbb{F}_q[X_1, \ldots, X_t]$, $\deg(g) \leq r$, then there exists $f \in \mathbb{F}_q[X_1, \ldots, X_m]$ with $\deg(f) \leq r$, such that (2.1) holds.*

In terms of codes, Lemma 2.6 implies the next corollary.

Corollary 2.7. *Let $\mathcal{C} = \mathrm{RM}_q(m,r)$ with $r \leq t(q-1)$ for some $t \geq 1$, and $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$. Then $\phi^*(\mathcal{C}) = \mathrm{RM}_q(t,r)$. In particular, if we denote by $A = \mathrm{im}(\phi)$ the t-dimensional affine space that ϕ defines, then $\mathcal{C}_{|A} \subseteq \mathbb{F}_q^A$ is permutation-equivalent to $\mathrm{RM}_q(t,r)$, the permutation being given by $\phi : \mathbb{F}_q^t \to A$.*

A first local correcting algorithm. Corollary 2.7 induces a local correcting procedure for $\mathrm{RM}_q(m,r)$, $r < t(q-1)$. It consists in picking at random $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$, and correcting the noisy codeword restricted to the image of ϕ, thanks to an efficient decoding algorithm for $\mathrm{RM}_q(t,r)$.

Thus, we now assume to have at our disposal a correcting algorithm Corr for $\mathrm{RM}_q(t,r)$, which corrects 1 erasure and up to w errors, where $2+2w = \mathrm{d}_{\min}(\mathrm{RM}_q(t,r))$. Such an algorithm can be derived from an efficient half-distance 1-erasure correcting algorithm for Reed-Solomon codes, as those we have mentioned in Subsection 1.2.2.

Algorithm 2: A perfectly smooth local correcting algorithm of locality $\ell = q^t - 1$ for the Reed-Muller code $\mathrm{RM}_q(m,r)$, where $r < t(q-1)$.

Input: a coordinate $\boldsymbol{u} \in S := \mathbb{F}_q^m$, and an oracle access to $g \in \mathbb{F}_q^S$ such that $\mathrm{d}(f,g) \leq \delta n$, for some $f \in \mathrm{RM}_q(m,r)$, where $n = |S|$.
Output: $f(\boldsymbol{u})$ with high probability.
/* Corr_A denotes a half-distance one-erasure correcting algorithm for the code $\mathrm{RM}_q(m,r)_{|A}$, isomorphic to $\mathrm{RM}_q(t,r)$. */
Pick uniformly at random $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$ such that $\boldsymbol{u} \in A := \mathrm{im}(\phi)$.
Toss a random binary coin $b \in \{0,1\}$, following $\mathscr{B}(p)$ with $p := \frac{q^t-1}{q^m}$.
if $b = 1$ **then**
    Pick $\overline{\boldsymbol{v}}$ uniformly at random in $A \setminus \{\boldsymbol{u}\}$.
else
    Define $\overline{\boldsymbol{v}} \leftarrow \boldsymbol{u}$.
Define $A' = A \setminus \{\overline{\boldsymbol{v}}\}$ and query $\{g(\boldsymbol{v}) : \boldsymbol{v} \in A'\}$.
Define $g' \in (\mathbb{F}_q \cup \{\perp\})^A$ by $g'_{|A'} = g_{|A'}$ and $g'(\overline{\boldsymbol{v}}) = \perp$. Run Corr_A on input g'.
if Corr_A *fails* **then**
    Abort with fail.
else
    Denote by $h \in \mathbb{F}_q^A$ the output of Corr_A. Output $h(\boldsymbol{u})$.

Let us give additional notation to make the analysis of Algorithm 2 simpler. We write $r = a(q-1)+b$, with $0 \leq b \leq q-1$ and $a \leq m-1$, the degree of the Reed-Muller code $\mathrm{RM}_q(m,r)$, and we recall that $\mathrm{d}_{\min}(\mathrm{RM}_q(m,r)) = (q-b)q^{m-a-1}$. We also denote by

$$w = \left\lfloor \frac{\mathrm{d}_{\min}(\mathrm{RM}_q(t,r)) - 2}{2} \right\rfloor = \left\lfloor \frac{(q-b)q^{t-a-1} - 2}{2} \right\rfloor$$

the number of errors that Corr can correct, and by $\tau := w/(q^t-1)$ its ratio compared to the locality parameter.

Proposition 2.8. *Let $m \geq 2$ and $r = a(q-1)+b$ such that $a \leq m-1$ and $0 \leq b < (q-1)$. Let also τ be the relative error correcting capability, defined as above. Then, for every $\delta < \tau/2$ and every*

$a+1 \leq t \leq m-1$, the Reed-Muller code $\mathrm{RM}_q(m,r)$ is a perfectly smooth $(q^t-1, \delta, \delta/\tau)$-locally correctable code, using Algorithm 2.

Proof. We use the same arguments as for Hadamard codes. First, we notice that Algorithm 2 is smooth: every point is chosen with probability $p = (q^t-1)/q^m$. Let us now bound its probability of success.

Denote by $S = \mathbb{F}_q^m$, $|S| = n$. Let $g = f + e \in \mathbb{F}_q^S$, where $f \in \mathrm{RM}_q(m,r)$, $e \in \mathbb{F}_q^S$ with $E = \mathrm{supp}(e)$ and $|E| \leq \delta n$. Let also $A'_{\boldsymbol{u}}$ represent the random queries made by $\mathsf{LC}^{(g)}(\boldsymbol{u})$. We know the algorithm succeeds if $|E \cap A'_{\boldsymbol{u}}| \leq w$, where $1 + 2w = \mathrm{d}_{\min}(\mathrm{RM}_q(t,r)) - 1$. Hence,

$$\Pr(\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})) \geq 1 - \Pr(|E \cap A'_{\boldsymbol{u}}| \geq w+1) \geq 1 - \frac{\mathbb{E}(|E \cap A'_{\boldsymbol{u}}|)}{w+1}$$

by Markov's inequality. Let us now estimate $\mathbb{E}(|E \cap A'_{\boldsymbol{u}}|)$. By linearity

$$\mathbb{E}(|E \cap A'_{\boldsymbol{u}}|) = \sum_{\boldsymbol{v} \in E} \Pr(\boldsymbol{v} \in A'_{\boldsymbol{u}}) = \sum_{\boldsymbol{v} \in E} p \leq \delta q^m \frac{q^t - 1}{q^m} = \delta(q^t - 1)\,.$$

Finally, we get

$$\Pr(\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})) \geq 1 - \frac{\delta(q^t-1)}{\tau(q^t-1)+1} \geq 1 - \frac{\delta}{\tau}\,. \qquad \square$$

Many parameters are involved in Proposition 2.8. If we fix the degree r of the Reed-Muller code, there remains freedom for the choice of t. We see that t affects exponentially the locality, but its influence on τ is moderate, since we have approximately $\tau \simeq (q-b)q^{-(a+1)}/2$. Therefore, choosing the minimum $t = a+1$ appears to be the most relevant choice since the locality is a crucial parameter.

A second local correcting algorithm for RM codes. In Algorithm 2, the strategy was to use the full correcting capability of the *local code* $\mathcal{C}_{|A} \simeq \mathrm{RM}_q(t,r)$, hoping that the number of errors on the queried symbols does not exceed the packing radius w of $\mathcal{C}_{|A}$. One can reduce the locality at the expense of an increase of the failure probability. Simply, the idea is to pick at random a map $\phi \in \mathrm{Iso}(\mathbb{F}_q^t, A)$ and an *information set* $I \subset A$ for the local code $\mathcal{C}_{|A}$, and then to query the noisy codeword only on I. Hoping for *no* corrupted symbols on I, we get an LCC of locality $\ell = \dim(\mathcal{C}_{|A}) = \dim(\mathrm{RM}_q(t,r))$.

A minor difficulty appears if we require smoothness. Indeed we need that each individual query is uniformly distributed over the support $\mathbb{F}_q^m$, when information sets I are picked at random. Fortunately, Reed-Muller codes admit an automorphism group which contains the doubly-transitive group $\mathrm{Aff}(\mathbb{F}_q^t)$ of affine transformations. Therefore, if I is a fixed information set for $\mathrm{RM}_q(t,r)$, then every elements of its orbit under $\mathrm{Aff}(\mathbb{F}_q^t)$ is also an information set for $\mathrm{RM}_q(t,r)$. Hence, it allows us to define the smooth query generator given in Algorithm 3.

Let us first explain why our choice of $p = \frac{q^t-\ell}{q^m-\ell}$ makes Algorithm 3 smooth.

Lemma 2.9. *In Algorithm 3, for every $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}_q^m$ and every information set I for $\mathrm{RM}_q(t,r)$ we have:*

$$\Pr(\boldsymbol{v} \in \mathsf{Query}(\boldsymbol{u}, I)) = \frac{\ell}{q^m}\,,$$

where $\ell = |I|$.

Algorithm 3: A smooth query generator $\mathsf{Query}(\boldsymbol{u}, I)$ for Reed-Muller codes.

Input: a coordinate $\boldsymbol{u} \in \mathbb{F}_q^m$, an information set I for $\mathrm{RM}_q(t,r)$.
Output: an affine injective map $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$ such that $\boldsymbol{u} \in \mathrm{im}(\phi)$ and $J = \phi(I)$ satisfies $\Pr(\boldsymbol{v} \in J) = \ell/q^m$ for every $\boldsymbol{v} \in \mathbb{F}_q^m$, where $\ell = |I|$.

1 Pick uniformly at random $\psi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$ such that $\boldsymbol{u} \in \mathrm{im}(\psi)$.
2 **do**
3 Pick $\sigma \in \mathrm{Aff}(\mathbb{F}_q^t)$ uniformly at random, and define $J = (\psi \circ \sigma)(I)$.
4 Toss a binary coin $b \in \{0,1\}$, following $\mathscr{B}(p)$ with $p = \frac{q^t - \ell}{q^m - \ell}$.
5 **while** *($\boldsymbol{u} \in J$ and $b = 0$)*;
6 Output $\phi = \psi \circ \sigma$.

Proof. Let $\beta := \ell/q^t$ represent $\Pr(\boldsymbol{u} \in J)$ at step 5. A quick analysis shows that

$$\Pr(\boldsymbol{u} \in \mathsf{Query}(\boldsymbol{u}, I)) = \beta p + \beta(1-p)\Pr(\boldsymbol{u} \in \mathsf{Query}(\boldsymbol{u}, I)).$$

Thus we get $\Pr(\boldsymbol{u} \in \mathsf{Query}(\boldsymbol{u}, I)) = \beta p/(1-(1-p)\beta)$. If we set

$$p = \frac{\ell(1-\beta)}{(q^m - \ell)\beta} = \frac{q^t - \ell}{q^m - \ell},$$

then we get $\Pr(\boldsymbol{u} \in \mathsf{Query}(\boldsymbol{u}, I)) = \ell/q^m$ which is one of our requirements. Now, it only remains to notice that, since $\mathrm{Aff}(\mathbb{F}_q^t)$ is doubly-transitive, any other point $\boldsymbol{v} \neq \boldsymbol{u}$ has equal probability to lie in the image of the output ϕ. Hence, $\Pr(\boldsymbol{v} \in \mathsf{Query}(\boldsymbol{u}, I)) = \ell/q^m$ for every $\boldsymbol{v} \neq \boldsymbol{u}$. □

Now that we are equipped with a smooth query generator, we can define Algorithm 4, a perfectly smooth local correcting algorithm for Reed-Muller codes with smaller locality than in Algorithm 2. We give its analysis in Proposition 2.10.

Algorithm 4: A perfectly smooth local correcting algorithm of locality $\ell = \dim \mathrm{RM}_q(t,r)$ for the Reed-Muller code $\mathrm{RM}_q(m,r)$, where $r < t(q-1)$.

Input: $\boldsymbol{u} \in S := \mathbb{F}_q^m$, and an oracle access to $g \in \mathbb{F}_q^S$ such that $\mathrm{d}(f,g) \leq \delta n$ where $f \in \mathrm{RM}_q(m,r)$, where $n = |S|$.
Output: $f(\boldsymbol{u})$ with high probability.
`/* We assume that an information set` I `for` $\mathrm{RM}_q(t,r)$ `is given. */`

1 Randomly pick $\phi \leftarrow \mathsf{Query}(\boldsymbol{u}, I)$.
2 Compute $\boldsymbol{w} = \phi^{-1}(\boldsymbol{u})$.
3 Query $\{(g \circ \phi)(\boldsymbol{v}), \boldsymbol{v} \in I\}$.
4 Interpolate $h \in \mathrm{RM}_q(t,r)$ such that $h_{|I} = (g \circ \phi)_{|I}$.
5 Output $h(\boldsymbol{w})$.

Proposition 2.10. *Let $m \geq 2$ and $r = a(q-1) + b$ such that $a \leq m-1$ and $0 \leq b < q-1$. Fix some $a+1 \leq t \leq m-1$ and denote by $\ell_t = \dim \mathrm{RM}_q(t,r)$. Then, for every $\delta < 1/2\ell_t$ and every $a+1 \leq t \leq m-1$, the Reed-Muller code $\mathrm{RM}_q(m,r)$ is a perfectly smooth $(\ell_t, \delta, \delta\ell_t)$-locally correctable code, using Algorithm 4.*

Proof. Smoothness being proved in Lemma 2.9, let us give a short explanation for the probability of failure. We use the notation of Algorithm 4. Similarly to the proofs of Propositions 2.5 and 2.8, it is sufficient to show that $\Pr(\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})) \geq 1 - \mathbb{E}(|E \cap J|)$, where $J = \phi(I)$ and E is the support of errors. Furthermore, $\mathbb{E}(|E \cap J|) = \delta\ell_t$ holds by linearity of the mean value. □

A short analysis of algorithms parameters. The setting $t = 1$ might be the most interesting to study, since it provides the smallest locality. So for a moment, let us focus on $\mathrm{RM}_q(m, r)$, $r \leq q - 2$. Essentially, we learned that:

- Algorithm 2 equips Reed-Muller codes with an $(\ell, \delta, \delta/\tau)$-local correcting algorithm, for $\ell = q - 1$ and $\tau \simeq \frac{1}{2}(1 - r/q)$;
- Algorithm 4 equips Reed-Muller codes with an $(\ell, \delta, \delta\ell)$-local correcting algorithm, for $\ell = r + 1$.

Therefore, Algorithm 4 outperforms Algorithm 2 if $k < 1/\tau$, that is, if $k \lesssim \frac{q}{2}(1 - \sqrt{1 - 8/q})$. It means that Algorithm 4 should not be neglected when q is moderate and k is small. In every other case, Algorithm 2 allows us to correct much more errors for a small increase of the locality.

Now let us focus on asymptotic families of Reed-Muller codes, indexed by their length $n = q^m$. Notice that, when referring to the locality parameter, it is implicitly assumed that Algorithm 2 is used.

1. A first possible construction consists in letting m grow to infinity, all other parameters being fixed. We therefore obtain a family of LCCs with constant locality, but whose rate

$$R = \frac{1}{q^m}\binom{r+m}{m} < \frac{1}{m!}\left(\frac{r+m}{q}\right)^m \sim \exp(r)\,\frac{\exp(m)}{q^m\sqrt{2\pi m}} = \mathcal{O}\left(\frac{1}{n^{1-\log_q(\exp(1))}}\right)$$

   vanishes quickly for any $q \geq 3$.
2. We can then consider the case where both q and r both grow to infinity according to $r = \rho q$, $0 < \rho < 1$. In a sense, it means that we fix the local correction capability (see Proposition 2.8). If all other parameters are fixed, then we get a family of LCCs with sublinear locality $\ell \leq n^{1/m}$. The rate is much better than previously: we obtain

$$R = \frac{1}{q^m}\binom{r+m}{m} \sim \frac{1}{m!}\rho^m$$

   which is a non-zero constant.

Families of codes with positive asymptotic rate are important for practical issues. They lead us to introduce a precise definition of so-called *asymptotically good LCCs*.

Definition 2.11. Let $\{\mathcal{C}_i\}_i$ be a family of codes of respective length n_i, such that $n_i \to \infty$ when $i \to \infty$. The family $\{\mathcal{C}_i\}_i$ is an *asymptotically good* family of LCCs if there exists $0 < \delta, \varepsilon, R < 1$ such that for every i, the code $\mathcal{C}_i$ is $(\ell_i, \delta_i, \varepsilon_i)$-locally correctable, and if we have

$$\left(\delta_i, \varepsilon_i, \frac{\dim(\mathcal{C}_i)}{n_i}\right) \to (\delta, \varepsilon, R) \quad \text{when } i \to \infty\,.$$

From the previous discussion, we see that for $\rho \in (0, 1)$, the second family of Reed-Muller codes $\{\mathrm{RM}_q(m, \rho q)\}_q$ defined above is asymptotically good, with $R = \rho^m/m!$ and $\varepsilon \leq \frac{2\delta}{1-\rho}$.

However, the asymptotic rate of RM codes is exponentially small in m, the parameter which quantifies the locality parameter $\ell = n^{1/m}$. Moreover, the rate stays stuck below $1/2$ for its minimum value ($m = 2$). In the next two sections, we will show how this barrier has been broken with algebraic constructions, namely the *multiplicity codes* of Kopparty *et al.* [KSY14] and the *lifted Reed-Solomon codes* of Guo *et al.* [GKS13].

2.2.2 Multiplicity codes

Presentation. Multiplicity codes define the first family of asymptotically good LCCs achieving rate $> 1/2$. They were built by Kopparty, Saraf and Yekhanin in the early 2010's [KSY14] and they generalise Reed-Muller codes by evaluating low-degree multivariate polynomials *and their derivatives*. Indeed, while Reed-Muller codes $\mathrm{RM}_q(m,r)$ need to satisfy $r < q-1$ in order to remain non-trivial on lines, multiplicity codes allow higher degrees r thanks to the evaluation of the derivatives of the polynomials.

More formally, let us first introduce the *Hasse derivatives* of an m-variate polynomial $f \in \mathbb{F}_q[\boldsymbol{X}]$ as the unique collection of polynomials $\{H^{(\boldsymbol{i})}(f)(\boldsymbol{X})\}_{\boldsymbol{i}\in\mathbb{N}^m}$ satisfying

$$f(\boldsymbol{X}+\boldsymbol{Y}) = \sum_{\boldsymbol{i}\in\mathbb{N}^m} H^{(\boldsymbol{i})}(f)(\boldsymbol{X})\boldsymbol{Y}^{\boldsymbol{i}},$$

where $\boldsymbol{Y}^{\boldsymbol{i}} \coloneqq Y_1^{i_1}\dots Y_m^{i_m}$. For instance, one can check that $H^{(\boldsymbol{0})}(f) = f$. For $s \geq 0$, we define the s-order evaluation of $f \in \mathbb{F}_q[X_1,\dots,X_m]$ at point $\boldsymbol{u} \in \mathbb{F}_q^m$ as

$$\mathrm{ev}^s(f)(\boldsymbol{u}) \coloneqq \left(H^{(\boldsymbol{i})}(f)(\boldsymbol{u}) : \boldsymbol{i} \in \mathbb{N}_m, |\boldsymbol{i}| \leq s\right) \in \Sigma_{s,m} \coloneqq \mathbb{F}_q^{T_{s,m}}$$

where $T_{s,m} \coloneqq \{\boldsymbol{i} \in \mathbb{N}_m, |\boldsymbol{i}| \leq s\}$. Then, the s-order evaluation vector associated to f is

$$\mathrm{ev}^s(f) \coloneqq (\mathrm{ev}^s(f)(\boldsymbol{u}) : \boldsymbol{u} \in \mathbb{F}_s^m) \in (\Sigma_{s,m})^{\mathbb{F}_q^m}.$$

A very good point is that for every $s \geq 0$ and $m \geq 1$, $f \mapsto \mathrm{ev}^s(f)$ is injective as long as $\deg(f) < (s+1)(q-1)$.

Notation. For $\boldsymbol{c} \in (\mathbb{F}_q^{T_{s,m}})^{\mathbb{F}_q^m}$, we denote by $c_{\boldsymbol{x}}^{(\boldsymbol{i})} \in \mathbb{F}_q$ the symbol corresponding to the order of derivation $\boldsymbol{i} \in T_{s,m}$ and the coordinate $\boldsymbol{x} \in \mathbb{F}_q^m$.

We are now able to build multiplicity codes properly.

Definition 2.12 (Multiplicity code [KSY14])**.** Let $m \geq 1$, and $r, s \geq 0$. Define an alphabet $\Sigma_{s,m} = \mathbb{F}_q^{T_{s,m}}$, $T_{s,m} = \{\boldsymbol{i} \in \mathbb{N}^m, |\boldsymbol{i}| \leq s\}$. The *multiplicity code* $\mathrm{Mult}_q(m,r,s)$ is the $\mathbb{F}_q$-linear code defined over the alphabet $\Sigma_{s,m}$, which consists in s-order evaluation vectors of multivariate polynomials with total degree bounded by r. Explicitly:

$$\mathrm{Mult}_q(m,r,s) \coloneqq \{\mathrm{ev}^s(f), f \in \mathbb{F}_q[X_1,\dots,X_m], \deg(f) \leq r\} \subseteq (\Sigma_{s,m})^{\mathbb{F}_q^m}.$$

For $m = 1$, multiplicity codes $\mathrm{Mult}_q(1,r,s) \subseteq \Sigma_{s,1}^{\mathbb{F}_q}$ are also known as *derivative codes* [GW13]. They can be efficiently decoded up to half their minimum distance, thanks to an adaptation of the Berlekamp-Welch algorithm [KSY14].

In the following, we will only consider the setting $r < (s+1)(q-1)$ which is of most interest for local correcting purposes. One must take care that multiplicity codes are defined over the alphabet $\Sigma_{s,m}$, but they are $\mathbb{F}_q$-linear codes and have dimension

$$\dim_{\mathbb{F}_q}(\mathrm{Mult}_q(m,r,s)) = \binom{r+m}{m}$$

over $\mathbb{F}_q$ since the evaluation map ev^s is injective. Therefore their rate satisfies

$$R = \frac{\binom{r+m}{m}}{\binom{m+s}{m}q^m} \geq \left(\frac{r+1}{(s+1)q}\right)^m.$$

Local correction. We here give high-level ideas concerning local correction of multiplicity codes. Details are quite technical and can be found in [KSY14, Yek12]. Assume one wants to recover $\mathrm{ev}^s(f)(\boldsymbol{u})$. Basically, the idea is to pick at random a certain number of affine lines that pass through $\boldsymbol{u}$ (*e.g.* $|T_{s,m}|$ lines for Algorithm 5), and to use dependencies between the evaluations of the derivatives in order to recover $\mathrm{ev}^s(f)(\boldsymbol{u})$.

Before being more formal, let us first state a few results whose proofs are given in [KSY14, Yek12] for instance.

Claim 2.13. *For any $\boldsymbol{u} \in \mathbb{F}_q^m$ and $\boldsymbol{v} \in \mathbb{F}_q^m \setminus \{\mathbf{0}\}$ we define $\phi_{\boldsymbol{u},\boldsymbol{v}}(T) = \boldsymbol{u} + T\boldsymbol{v} \in \mathbb{F}_q[T]$. Assume $s \geq 0$ and $r < (s+1)(q-1)$, and let $f \in \mathbb{F}_q[\mathbf{X}]$ such that $\deg(f) \leq r$.*

1. *We have $\mathrm{ev}^s(f \circ \phi_{\boldsymbol{u},\boldsymbol{v}}) \in \mathrm{Mult}_q(1,r,s)$, and the following holds*
$$(f \circ \phi_{\boldsymbol{u},\boldsymbol{v}})(T) = \sum_{\boldsymbol{i}} H^{(\boldsymbol{i})}(f)(\boldsymbol{u})\boldsymbol{v}^{\boldsymbol{i}} T^{|\boldsymbol{i}|}\,.$$
2. *Let $\boldsymbol{u} \in \mathbb{F}_q^m$ and $V \subseteq \mathbb{F}_q^m \setminus \{\mathbf{0}\}$ of size $\binom{m+s}{m}$. Given the knowledge of $((f \circ \phi_{\boldsymbol{u},\boldsymbol{v}})(T) : \boldsymbol{v} \in V)$, one can retrieve $(H^{(\boldsymbol{i})}(f)(\boldsymbol{u}) : |\boldsymbol{i}| \leq s)$ by solving a linear system.*

Algorithm 5: Outline of a local correcting algorithm for $\mathrm{Mult}_q(m,r,s)$.

Input: a coordinate $\boldsymbol{u} \in \mathbb{F}_q^m$, and an oracle access to a word $\boldsymbol{y} \in (\Sigma_{s,m})^{\mathbb{F}_q^m}$, such that $\mathrm{d}(\boldsymbol{y},\boldsymbol{c}) \leq \delta q^m$ for some $\boldsymbol{c} \in \mathrm{Mult}_q(m,r,s)$.

Output: The $\Sigma_{s,m}$-symbol $c_{\boldsymbol{u}}$.

1 Pick at random a subset $V \subseteq \mathbb{F}_q^m \setminus \{\mathbf{0}\}$ of size $\binom{m+s}{m}$.
2 Query $\{y_{\boldsymbol{u}+t\boldsymbol{v}} \in \Sigma_{s,m},\ t \in \mathbb{F}_q, \boldsymbol{v} \in V\}$.
3 **foreach** $\boldsymbol{v} \in V$ **do**
4 Build $g_{\boldsymbol{v}} = ((\sum_{|\boldsymbol{i}|=j} y^{(\boldsymbol{i})}_{\boldsymbol{u}+t\boldsymbol{v}} : 0 \leq j \leq s) : t \in \mathbb{F}_q) \in \Sigma_{s,1}^{\mathbb{F}_q}$.
5 Correct $g_{\boldsymbol{v}}$ with a correcting algorithm for $\mathrm{Mult}_q(1,r,s)$.
6 Interpolate the output, giving $h_{\boldsymbol{v}}(T) \in \mathbb{F}_q[T]$.
7 Retrieve and output $(H^{(\boldsymbol{i})}(f)(\boldsymbol{u}) : |\boldsymbol{i}| \leq s)$ from $(h_{\boldsymbol{v}}(T) : \boldsymbol{v} \in V)$, as in Claim 2.13.

Denote by $\Delta := 1 - r/(s+1)q$; informally, Δ is a lower bound on the relative distance of the derivative code $\mathrm{Mult}_q(1,r,s)$. Kopparty, Saraf and Yekhanin proposed a first local correcting algorithm [KSY14], which works if few errors occur on the codeword (small δ). This algorithm is given in Algorithm 5, and it outputs the correct symbol as long as there is no more than $\Delta/2$ errors on every queried line. Thus, after analysis of the correction failure, the authors conclude that multiplicity codes are $(\ell,\delta,\varepsilon)$-locally correctable, where

$$\ell = 1 + (q-1)\binom{m+s}{m}, \quad \delta = \mathcal{O}(\Delta/\ell), \quad \varepsilon = \mathcal{O}(1)\,,$$

and their rate R satisfies $R \geq (1-\Delta)^m$.

In order to find asymptotically good families of LCCs, it is natural to let q grow to infinity as we did for Reed-Muller codes. We see that the rate of multiplicity codes can be made arbitrarily close to 1 by choosing r close enough to $(s+1)q$. However in this setting, one can check that δ vanishes when $q \to \infty$.

Kopparty, Saraf and Yekhanin proposed to solve this issue by picking a set V of size slightly larger than $\binom{m+s}{m}$, and expecting that *most of the queried lines* are corrupted with less than $\Delta/2$ errors. Then, step 7 is replaced by solving a so-called *noisy linear system* and moreover, it can be efficiently done thanks to results of Kim and Kopparty [KK17]. We also refer to [KSY14, Yek12] for more details about this technical improvement; let us simply report their main theorem.

Theorem 2.14 (Theorem 10 of [KSY14]). *Let q, m, r, s be such that $r < (s+1)(q-1)$, and let $\Delta = 1 - r/(s+1)q$. If q is large enough*[2]*, then* $\mathrm{Mult}_q(m, r, s)$ *is $(\ell, \delta, \varepsilon)$-locally correctable, where*

$$\ell \leq q\frac{(s+m)^m}{m!}, \quad \delta = \Delta/10, \quad \varepsilon \geq 0.8\,.$$

Therefore, if we consider a sequence of multiplicity codes $\mathrm{Mult}_q(m, r, s)$ with increasing q and a parameter $\Delta = 1 - r/(s+1)q$ upper bounded by some constant $1 - \tau$, then we obtain

Theorem 2.15 (Theorem 3 of [KSY14]). *Let m, s be fixed integers and $\tau > 0$. For large enough prime powers $q = \Omega(m, s, 1/\tau)$, let r_q be such that $\tau(s+1)q \leq r_q < (s+1)(q-1)$. Then the sequence $\{\mathrm{Mult}_q(m, r_q, s)\}_q$ defines an asymptotically good family of LCCs of length n over $\Sigma_{s,m}$, with*

$$\ell = \Theta(n^{1/m}), \quad \delta = \Theta(1-\tau), \quad \varepsilon \geq 0.8$$

and rate $R \geq \tau^m$.

Notice that the locality parameter ℓ is sublinear in the code length, and the exponent of sublinearity $1/m$ can be set arbitrarily small.

2.2.3 Lifted codes

Presentation. Chronologically, lifted Reed-Solomon codes are the second class of high-rate LCCs that has been designed [GKS13]. They also generalise Reed-Muller codes, but in a very different manner. We have seen that multiplicity codes [KSY14] solved the low dimension drawback of Reed-Muller codes by a modification of the evaluation map, taking into account formal derivatives of the polynomials.

Guo, Kopparty and Sudan [GKS13] used a different approach and considered the *whole* set of multivariate polynomials that interpolate into a low-degree polynomial when restricted to any affine line. Quite surprisingly, under some arithmetic conditions it appears that this set contains more polynomials than the low-degree ones evaluated in Reed-Muller codes. Example 2.16 illustrates this property with small parameters. Notice that first appearances of the *lifting* process can be found in [BMSS11, BS11].

Example 2.16. Consider the Reed-Muller code $\mathrm{RM}_q(m, r)$ with $q = 4$, $m = 2$ and $r = 2$. Let $f(X, Y) = X^2Y^2$ and $\boldsymbol{c} = \mathrm{ev}_{\mathbb{F}_4^2}(f) \in \mathbb{F}_4^{\mathbb{F}_4^2}$. One can easily check that $\boldsymbol{c} \notin \mathrm{RM}_4(2, 2)$, since f has total degree $4 > 2$. Any restriction of $\boldsymbol{c}$ to an affine line of $\mathbb{F}_4^2$ can be written $\mathrm{ev}_{\mathbb{F}_4}(g)$, where $g \in \mathbb{F}_4[T]$ is defined by

$$g(T) = f(aT + b, cT + d)$$

for some $a, b, c, d \in \mathbb{F}_4$, such that $ac \neq 0$. Since $\mathrm{char}(\mathbb{F}_4) = 2$, we then get:

$$g(T) = (aT+b)^2(cT+d)^2 = a^2c^2 \cdot T^4 + (a^2d^2 + b^2c^2) \cdot T^2 + b^2d^2\,.$$

Let us now notice that $\mathrm{ev}_{\mathbb{F}_4}(T^4) = \mathrm{ev}_{\mathbb{F}_4}(T)$, hence $\mathrm{ev}_{\mathbb{F}_4}(g)$ is actually the evaluation of a univariate polynomial of degree at most 2. To sum up, $\boldsymbol{c}$ is the evaluation of a bivariate polynomial of degree 4, such that its restriction to any affine line of $\mathbb{F}_4^2$ corresponds to the evaluation of a univariate polynomial of degree at most 2.

[2] technically one needs $q \geq \max\{10m, \frac{d+6}{s+1}, 5(s+2)\}$

Let us now introduce the construction of Guo, Kopparty and Sudan in its full generality, where the authors define the lifting of any *affine-invariant* code. An affine-invariant code is a code $\mathcal{C} \subseteq \mathbb{F}_q^S$, $S = \mathbb{F}_Q^t$ and $\mathbb{F}_Q$ an extension of $\mathbb{F}_q$, which satisfies $\mathrm{Aff}(S) \subseteq \mathrm{Aut}(\mathcal{C})$.

Definition 2.17 (lifting of affine-invariant codes [GKS13])**.** Let $\mathcal{C} \subseteq \mathbb{F}_q^S$, be an affine-invariant code, where $S = \mathbb{F}_Q^t$ and $\mathbb{F}_Q$ is an extension of $\mathbb{F}_q$. The lifting of order $m \geq t$ of $\mathcal{C}$ is:

$$\mathrm{Lift}(\mathcal{C}, m) = \big\{ \mathrm{ev}_{\mathbb{F}_Q^m}(f), f \in \mathbb{F}_Q[\boldsymbol{X}], \text{ such that } \mathrm{ev}_{\mathbb{F}_Q^t}(f \circ \phi) \in \mathcal{C} \\ \text{for every } \phi \in \mathrm{Aff}(\mathbb{F}_Q^t, \mathbb{F}_Q^m) \big\}.$$

One can check that $\mathrm{Lift}(\mathcal{C}, m) \subseteq \mathbb{F}_q^{\mathbb{F}_Q^m}$ is an $\mathbb{F}_q$-linear code, invariant under $\mathrm{Aff}(\mathbb{F}_Q^m)$. In their work [GKS13], the authors give several constructions by letting different parameters vary (*e.g.* $m, t, \log_q(Q)$, etc.). Here we will mainly focus on settings that give rise to high-rate LCCs, namely $Q = q$ and t small, and particularly $t = 1$.

Full-length Reed-Solomon codes define a typical family of affine-invariant codes supported by the affine line $\mathbb{F}_q$. By definition, their lift consists in the evaluation of every m-variate polynomial which corresponds to a low-degree univariate polynomial when restricted to any affine line of the space $\mathbb{F}_q^m$. We have seen previously that Reed-Muller codes satisfy this property; therefore lifted Reed-Solomon codes contain Reed-Muller codes.

Proposition 2.18. *For every $0 \leq r \leq q-1$, we have*

$$\mathrm{RM}_q(m, r) \subseteq \mathrm{Lift}(\mathrm{RS}_q(r), m).$$

Proof. Thanks to Lemma 2.6, for all $\boldsymbol{c} = \mathrm{ev}_{\mathbb{F}_q^m}(f) \in \mathrm{RM}_q(m, r)$ and $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$, there exists $g \in \mathbb{F}_q[X_1, \ldots, X_t]$, such that $\mathrm{ev}_{\mathbb{F}_q^t}(g) = \mathrm{ev}_{\mathbb{F}_q^t}(f \circ \phi)$ and $\deg(g) \leq r$. □

Kaufman and Ron proved in a paper [KR06] anterior to Guo *et al.*'s work, that $\mathrm{RM}_q(m, r) = \mathrm{Lift}(\mathrm{RS}_q(r), m)$ as long as $r < q - \frac{q}{p}$, where $p = \mathrm{char}(\mathbb{F}_q)$. But for higher degrees r, the inclusion can be strict, as we have seen in Example 2.16.

More details concerning the construction of lifted Reed-Solomon codes will be given in Chapter 3. Here we only report one of Guo *et al.*'s results, asserting for any order m the existence of lifted Reed-Solomon codes of rate arbitrary close to 1 and constant relative distance. For readability we give a simplified version where m is a power of 2.

Theorem 2.19 (High-rate high-error construction of [GKS13], simplified)**.** *Let $\alpha \geq 1$ and $m = 2^\mu$. Define $c = 2(1+\mu)m2^m\alpha$, and assume $q = 2^s$ where $s \geq c$. Finally, let $\mathcal{C} = \mathrm{RS}_q((1-2^{-c})q - 1)$. Then, the lifting $\mathrm{Lift}(\mathcal{C}, m)$ has rate $R \geq 1 - 2^{-\alpha}$.*

Local correction. Lifted Reed-Solomon codes have the same properties as Reed-Muller codes when restricted to affine lines. Therefore, when specified with $t = 1$, Algorithms 2 and 4 provide perfectly smooth local correction to lifted Reed-Solomon codes. We here propose another local correcting algorithm, which (i) generalises both previous algorithms in a sense that the locality ℓ can be *any* integer between k and $q-1$ (ii) is simpler than Algorithm 4, since Reed-Solomon codes are MDS and therefore admit any k-subset of $\mathbb{F}_q$ as an information set. This generic local correcting algorithm is given in Algorithm 6. It is analysed in Proposition 2.20.

Proposition 2.20. *Let $k+1 \leq \ell \leq q-1$, and denote by $t = \lfloor \frac{\ell-(k+1)}{2} \rfloor$. Then, using Algorithm 6, for every $\delta \leq \frac{t+1}{2\ell}$ the code $\mathrm{Lift}(\mathrm{RS}_q(k), m)$ is a smooth $(\ell, \delta, \frac{\delta\ell}{t+1})$-locally correctable code.*

Algorithm 6: A perfectly smooth local correcting algorithm of locality $\ell \in [k+1, q-1]$ for the lifted Reed-Solomon code $\mathcal{C} = \mathrm{Lift}(\mathrm{RS}_q(k), m)$.

Input: a point $\boldsymbol{u} \in S := \mathbb{F}_q^m$, and an oracle access to $g \in \mathbb{F}_q^S$ such that $\mathrm{d}(f, g) \leq \delta n$, for some $f \in \mathcal{C}$, where $n = |S|$.
Output: $f(\boldsymbol{u})$ with high probability.
/* Corr_A denotes a half-distance $(q-\ell)$-erasure correcting algorithm for the code $\mathcal{C}_{|A}$ isomorphic to $\mathrm{RS}_q(k)$. */
Pick uniformly at random $\phi \in \mathrm{Aff}(\mathbb{F}_q^t, \mathbb{F}_q^m)$ such that $\boldsymbol{u} \in A := \mathrm{im}(\phi)$.
Toss a random binary coin $b \in \{0,1\}$ following $\mathscr{B}(p)$, where $p := \ell / q^m$.
if $b = 0$ **then**
    Pick uniformly at random a subset $A' \subset A \setminus \{\boldsymbol{u}\}$ of size ℓ.
else
    Pick uniformly at random a subset $A' \subset A \setminus \{\boldsymbol{u}\}$ of size $\ell - 1$.
    Add $\boldsymbol{u}$ in A'.
Query $\{g(\boldsymbol{v}) : \boldsymbol{v} \in A'\}$.
Define $g' \in (\mathbb{F}_q \cup \{\perp\})^A$ by $g'_{|A'} = g_{|A'}$ and $g'(\boldsymbol{v}) = \perp$ for every $\boldsymbol{v} \in A \setminus A'$.
Run Corr_A on input g'.
if Corr_A *fails* **then**
    Abort.
else
    Denote by $h \in \mathbb{F}_q^A$ the output of Corr_A. Output $h(\boldsymbol{u})$.

Proof. First, we can easily check that the algorithm is smooth. Concerning the extraction success, the method is similar to previous proofs. Denoting by E the support of the errors, Markov's inequality gives:

$$\Pr(\mathsf{LC}^{(g)}(\boldsymbol{u}) = f(\boldsymbol{u})) = 1 - \Pr(|E \cap A'| \geq t+1) \geq 1 - \frac{\mathbb{E}(|E \cap A'|)}{t+1},$$

and as usual we have

$$\mathbb{E}(|E \cap A'|) \leq \delta \ell .$$

□

As a corollary, when $q \to \infty$ and the parameters t and ℓ remain linearly dependent in q, we get asymptotically good LCCs.

Corollary 2.21. *Let $0 < \tau, \rho < 1$ and $m \geq 2$ be fixed. Let also (ℓ_q) and (k_q) be sequences of integers such that $\rho q \leq k_q + 1 \leq \ell_q \leq q - 1$ and $t_q = \lfloor \frac{\ell_q - k_q - 1}{2} \rfloor$ satisfies $\frac{t_q+1}{\ell_q} \geq \tau$. Then, the sequence of codes $\{\mathrm{Lift}(\mathrm{RS}_q(k_q), m)\}_q$ defines an asymptotically good family of LCCs with $\varepsilon \leq \delta/\tau$ and rate $R = \dim(\mathrm{Lift}(\mathrm{RS}_q(k_q), m))/q^m$. Moreover, their rate R is larger than $\rho^m/m!$, and their locality satisfies $\ell = \Theta(n^{1/m})$.*

From Corollary 2.21, one sees that we can obtain asymptotically good *high-rate* LCCs if we manage to find specific sub-sequences of q and k_q such that $\mathrm{Lift}(\mathrm{RS}_q(k_q), m)$ has large dimension. Such instances have been found by Guo, Kopparty and Sudan, as we reported in Theorem 2.19. Therefore, we get the following asymptotic result.

Theorem 2.22. *For fixed $m = 2^\mu$ and $\alpha \geq 1$, and choosing maximum locality $\ell = q - 1$, the sequence of codes indexed by q given in Theorem 2.19 is an asymptotically good family of LCCs, with*

$$R \geq 1 - 2^{-\alpha} \quad \text{and} \quad \varepsilon \leq 2\delta \left(\frac{1}{1-R}\right)^{2(1+\mu)m2^m}.$$

Proof. Applying Theorem 2.19 and Corollary 2.21 leads to $\varepsilon \leq 2^{c+1}\delta$, where c is defined in Theorem 2.19. □

2.2.4 A short comparison between multiplicity and lifted codes

Guo proposed in [Guo16] a comparison between the asymptotic parameters of lifted Reed-Solomon codes and multiplicity codes. This comparison is mostly based on the bound on the rate R of $\mathrm{Lift}(\mathrm{RS}_q((1-2^{-c})q-1), m)$ given in Theorem 2.19. However, this bound is far from being tight, especially for small values of m. For instance, in the case $m = 2$ the bound states that $R \geq 1 - ((1/2)^{1/32})^c$, while the actual asymptotic rate, given in Proposition 3.63 is

$$R_\infty = 1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c.$$

For this reason, we choose to focus our comparison on codes of finite length, whose rate can be explicitly computed. In Figure 2.1, we give a few diagrams presenting high-rate multiplicity and lifted codes. Let us give a few explanations. Given two reals $0 < \delta, \lambda < 1$ — for instance $\delta = 1/16$ and $\lambda = 0.01$ for the first subfigure — we plot all lifted codes and multiplicity codes with relative distance approximately δ and relative locality ℓ/n less than λ. Red dots represent multiplicity codes $\mathrm{Mult}_q(m, r, s)$ for various values of s, r and m. Blue dots represent lifted codes of order 2 (in other words, the codes $\mathrm{Lift}(\mathrm{RS}_q((1-\delta)q, 2))$ satisfying the previous requirements). We also remove every code with rate < 0.2 for clarity.

Figure 2.1 confirms a usual observation (see [GKS13, Guo16] for instance): lifted codes attain rates above $1/2$ with shorter lengths than multiplicity codes. However, multiplicity codes allow us to reach higher rates.

We end this section by reporting a construction due to Wu [Wu15], which crosses lifted codes and multiplicity codes. More precisely, the author considers the space of multivariate polynomials $f \in \mathbb{F}_q[\boldsymbol{X}]$, whose s-order evaluation vector $\mathrm{ev}^{(s)}(f)$, when restricted to any affine line, lies in $\mathrm{Mult}_q(1, r, s)$. Furthermore, for some range of parameters, this space strictly contains the multiplicity code $\mathrm{Mult}_q(m, r, s)$. Notice that for $s = 0$, one retrieves the lifted Reed-Solomon code construction of Guo *et al.* [GKS13].

2.3 Other constructions and bounds

In this section we quickly report recent constructions and current bounds related to LDCs and LCCs. Notice that, when their dimension is small, it is not clear that LDCs and LCCs can achieve the same parameters since we require local correction of much more symbols in LCCs. Different regimes must be taken into account:

- The *positive rate*, or *high-rate* regime consists in studying families of codes whose asymptotic rate is non-zero. We then look for codes admitting the lowest possible locality ℓ as a function of the dimension k (or equivalently the length n).
- In the *constant locality* regime, we consider families of LCCs with fixed locality ℓ, and we focus on finding codes with lowest length n compared to the dimension k. Usually, the settings $\ell = 2$, $\ell = 3$ and $\ell > 3$ are studied separately.
- The *sublinear locality* regime consists in seeing ℓ as a sublinear function of the dimension k (or of the length n if the code rate is non-zero). In this context we can write $\ell = k^\beta$ for some constant $0 < \beta < 1$ (that can be made arbitrary close to 0), and we try to

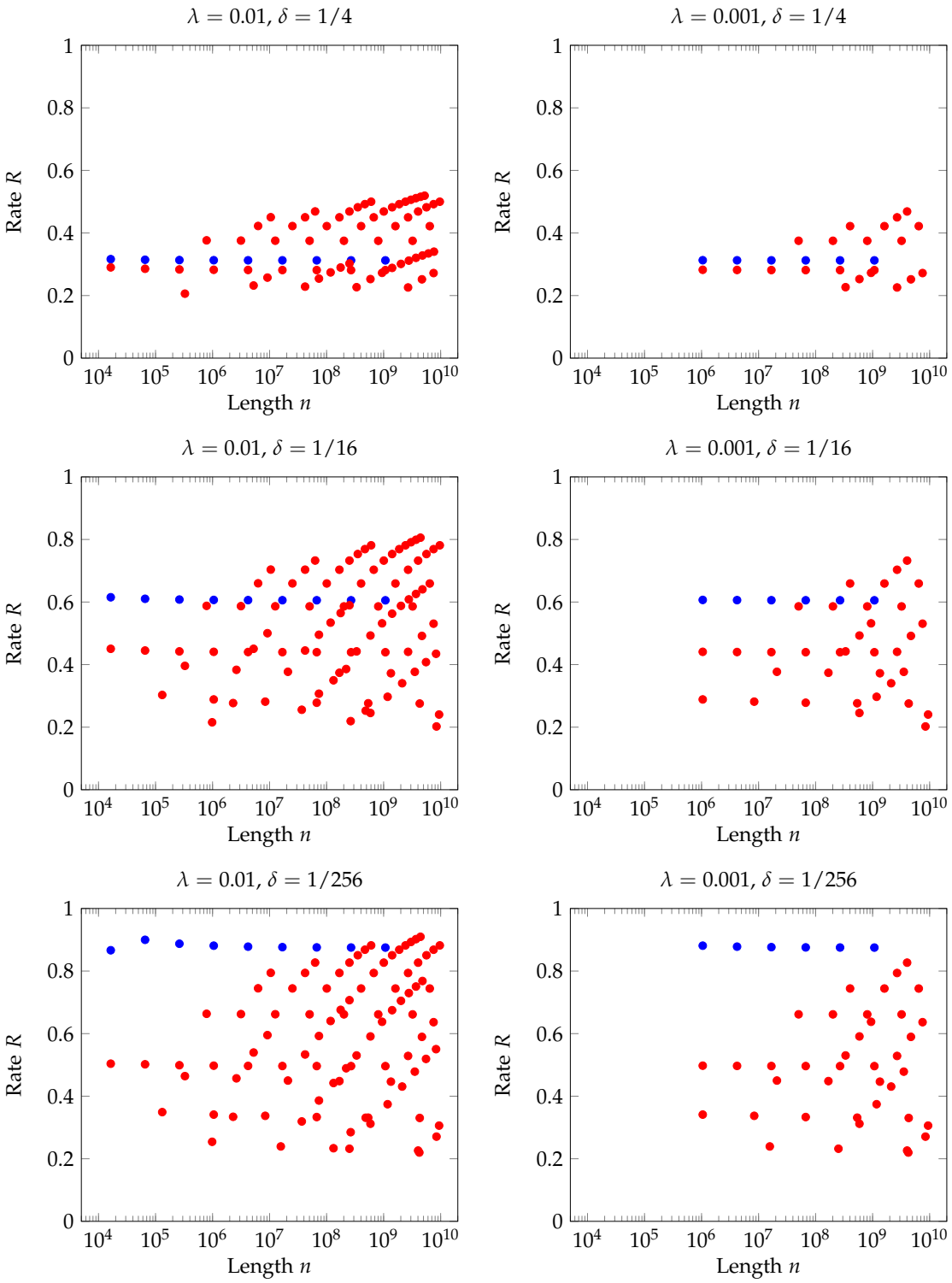


Figure 2.1 – High-rate lifted and multiplicity codes correcting a constant fraction δ of errors, with relative locality at most λ. Red dots represent multiplicity codes; blue dot represents lifted codes of order 2.

find codes with highest possible rate, or approaching well-known bounds such as the Singleton bound.

We will sometimes use the binary L-notation to easily represent asymptotic complexities. Recall it is defined by:

$$L_k(a,b) = 2^{(b+o(1))(\log k)^a(\log\log k)^{1-a}}.$$

Parameter a is the most relevant: $L_k(1,b)$ represent complexities exponential in $\log k$, while $L_k(0,b)$ complexities polynomial in $\log k$, with degree bounded by b.

2.3.1 Constructions

Matching vector codes. Matching vector codes (MVCs) define a broad class of LDCs performing well in the *constant locality* regime. They were introduced in a seminal paper of Yekhanin [Yek08], and then developed in a large series of works, notably [Efr12, DGY11, BDL14, SY11].

MVCs are based on the existence of so-called *matching families*. Denote by $\mathbb{Z}_r = \mathbb{Z}/r\mathbb{Z}$ and fix $A \subset \mathbb{Z}_r \setminus \{0\}$. Also recall that $\langle\cdot,\cdot\rangle$ represents the usual inner product. An *A-matching family* of size k is a pair of ordered lists $U = (\boldsymbol{u}^{(1)},\dots,\boldsymbol{u}^{(k)})$, $V = (\boldsymbol{v}^{(1)},\dots,\boldsymbol{v}^{(k)})$ of elements in $\mathbb{Z}_r^m$, satisfying $\langle \boldsymbol{u}^{(i)}, \boldsymbol{v}^{(i)}\rangle = 0$ and $\langle \boldsymbol{u}^{(i)}, \boldsymbol{v}^{(j)}\rangle \in A$ for all $1 \le i,j \le k$, $i \ne j$.

Now, assume that $\mathbb{F}_q^\times$ admits a multiplicative subgroup of order r, denoted D_r, generated by g. For $\boldsymbol{w} \in \mathbb{Z}_r^m$ and $\boldsymbol{v}$ an element of the list V, define a so-called *multiplicative line*

$$L_{\boldsymbol{v},\boldsymbol{w}} = \{g^{\boldsymbol{w}+\lambda\boldsymbol{v}}, \lambda \in \mathbb{Z}_r\},$$

where $g^{\boldsymbol{w}+\lambda\boldsymbol{v}}$ represents the k-tuple $(g^{w_1+\lambda v_m},\dots,g^{w_k+\lambda v_m}) \in D_r^m$.

Let now $\boldsymbol{u}$ be an element in the list U. Then it holds that

$$\forall g^{\boldsymbol{w}+\lambda\boldsymbol{v}} \in L_{\boldsymbol{v},\boldsymbol{w}},\ (g^{\boldsymbol{w}+\lambda\boldsymbol{v}})^{\boldsymbol{u}} = g^{\langle\boldsymbol{u},\boldsymbol{w}\rangle}(g^\lambda)^{\langle\boldsymbol{u},\boldsymbol{v}\rangle}. \tag{2.2}$$

In other words, the evaluation of the multivariate monomial $\boldsymbol{X}^{\boldsymbol{u}}$ on the multiplicative line $L_{\boldsymbol{v},\boldsymbol{w}}$ corresponds to the evaluation of the univariate monomial $f(Y) = g^{\langle\boldsymbol{u},\boldsymbol{w}\rangle}Y^{\langle\boldsymbol{u},\boldsymbol{v}\rangle}$ on $D_r = \{g^\lambda, \lambda \in \mathbb{Z}_r\}$. This property will be used to define codes with a local decoding procedure.

Given (U,V) an A-matching family of size k, one can define the *matching vector code* as the evaluation code over D_r^m of polynomial functions in $\mathrm{Span}\{\boldsymbol{X}^{\boldsymbol{u}}, \boldsymbol{u} \in U\}$. That is,

$$\mathrm{MVC}(U) := \left\{ \left(\sum_{j=1}^k m_j \boldsymbol{x}^{\boldsymbol{u}^{(j)}} : \boldsymbol{x} \in D_r^m \right), \boldsymbol{m} \in \mathbb{F}_q^k \right\} \subseteq \mathbb{F}_q^{D_r^m}. \tag{2.3}$$

Let now $\boldsymbol{v}^{(i)} \in V$ and $\boldsymbol{w} \in \mathbb{Z}_r^m$. If $F = \sum_{j=1}^k m_j \boldsymbol{X}^{\boldsymbol{u}^{(j)}}$, then using (2.2) and the definition of a matching family, we see that

$$F(g^{\boldsymbol{w}+\lambda\boldsymbol{v}^{(i)}}) = \sum_{j=1}^k m_j g^{\langle\boldsymbol{u}^{(j)},\boldsymbol{w}\rangle}(g^\lambda)^{\langle\boldsymbol{u}^{(j)},\boldsymbol{v}^{(i)}\rangle} = m_i g^{\langle\boldsymbol{u}^{(i)},\boldsymbol{w}\rangle} + \sum_{a\in A}\left(\sum_{\langle\boldsymbol{u}^{(j)},\boldsymbol{v}^{(i)}\rangle = a} m_j g^{\langle\boldsymbol{u}^{(j)},\boldsymbol{w}\rangle} \right)(g^\lambda)^a.$$

In other words, if $\boldsymbol{c} = \mathrm{ev}_{D_r^m}(F) \in \mathrm{MVC}(U)$, then for every $g^{\boldsymbol{w}+\lambda\boldsymbol{v}^{(i)}} \in L_{\boldsymbol{w},\boldsymbol{v}^{(i)}}$ we have $F(g^{\boldsymbol{w}+\lambda\boldsymbol{v}^{(i)}}) = f_i(g^\lambda)$ where

$$f_i(Y) = x_i g^{\langle\boldsymbol{u}^{(i)},\boldsymbol{w}\rangle} + \sum_{a\in A}\left(\sum_{\langle\boldsymbol{u}^{(j)},\boldsymbol{v}^{(i)}\rangle = a} x_j g^{\langle\boldsymbol{u}^{(j)},\boldsymbol{w}\rangle} \right) Y^a \tag{2.4}$$

is a univariate polynomial composed by monomials Y^a, for $a \in A \cup \{0\}$. Thus, the restriction of $\boldsymbol{c}$ to multiplicative lines $L_{\boldsymbol{w},\boldsymbol{v}_i}$ admit a non-trivial structure that can be used for correction purposes. Specifically, a local decoding algorithm follows, depicted in Algorithm 7.

Algorithm 7: Outline of an local decoding algorithm for $\mathcal{C} = \mathrm{MVC}(U)$ equipped with the encoder E implicit in (2.3). The pair (U, V) is an A-matching family, $|A| = \ell - 1$.

Input: a coordinate $i \in [1, k]$, and an oracle access to $h \in \mathbb{F}_q^S$, $S = D_r^m$ such that $\mathrm{d}(E(\boldsymbol{m}), h) \leq \delta r^m$, for some $\boldsymbol{m} \in \mathbb{F}_q^k$.

Output: m_i with high probability.

1 Pick uniformly at random $\boldsymbol{w} \in \mathbb{Z}_r^m$.

2 Define $Q = \{g^{\boldsymbol{w}+\lambda \boldsymbol{v}^{(i)}}, 0 \leq \lambda \leq \ell - 1\} \subset L_{\boldsymbol{w},\boldsymbol{v}^{(i)}}$.

3 Query $\{h(\boldsymbol{x}), \boldsymbol{x} \in Q\}$.

4 Interpolate the associated univariate polynomial $f_i(Y)$.

5 Output $f(0)/g^{\langle \boldsymbol{u}^{(i)}, \boldsymbol{w} \rangle}$.

We should clarify that step 4 of Algorithm 7 essentially consists in inverting a Vandermonde matrix. Hence, the output is correct as long as there is no error on the queries, since $f_i(0) = x_i g^{\langle \boldsymbol{u}^{(i)}, \boldsymbol{w} \rangle}$ according to (2.4). Therefore, after analysis of the success probability, one can prove that Algorithm 7 is a perfectly smooth $(\ell, \delta, \delta\ell)$-local decoding algorithm for $\mathcal{C} = \mathrm{MVC}(U)$, where (U, V) is an A-matching family, $|A| = \ell - 1$.

As for Reed-Muller codes, another strategy consists in querying more values on the multiplicative line $L_{\boldsymbol{w},\boldsymbol{v}^{(i)}}$ even if we need to correct a few errors on it. This correction can be performed as long as the univariate polynomial $f(Y)$ has low degree. Thus, for $b \geq 0$ we say that (U, V) is *b-bounded* if for all $a \in A$, we have $a \leq b$ (where $a \in \mathbb{Z}_r$ is seen as an integer bounded by $r - 1$).

We obtain:

Proposition 2.23. *Let $\sigma > 0$. If there exists a σm-bounded A-matching family (U, V) in $\mathbb{Z}_r^m$ of size k, then the code $\mathcal{C} = \mathrm{MVC}(U)$ is $(m, \delta, 2\delta/(1-\sigma))$-locally decodable, for all $\delta < (1-\sigma)/4$.*

MVCs with good local properties rely on the existence of A-matching families in $\mathbb{Z}_r^m$ with small m and large k. In a series of papers (*e.g.* [Gro00, Gro02]), Grolmusz provides explicit constructions of such families. Some of them are based on multilinear polynomials modulo composite integers; we refer to [Gro00, Gro02] and Yekhanin's survey [Yek12] for details. Let us only report the asymptotic families of LDCs they induce, which currently define some of the best families in the constant locality regime.

Proposition 2.24. *(based on [DGY11, Efr12] and subsequent works) For large enough $\ell > 3$, there exist asymptotic families of $(\ell, \delta, \mathcal{O}(1))$-locally decodable matching vector codes of dimension k, with the following parameters:*

1. *in the low-error regime (precisely, $\delta = \mathcal{O}(1/\ell)$ when $\ell \to \infty$), the length $n = 2^{f(k,\ell)}$ of the codes satisfies*
$$f(k, \ell) = L_k\left(1/\log \ell, 1\right);$$
2. *in the high-error regime (precisely, $\delta = 1/4 - \mathcal{O}(1/\log\log \ell)$ when $\ell \to \infty$), the length $n = 2^{f(k,\ell)}$ of the codes satisfies*
$$f(k, \ell) = L_k\left(\alpha_\ell, \log \ell\right),$$
with $\alpha_\ell = \mathcal{O}\left(\frac{\log\log \ell}{\log \ell}\right)$.

Expander codes. *Expander codes* were initially introduced by Sipser and Spielman [SS96], and have been proven to provide an asymptotically good family of codes with efficient decoding algorithms [Zém01, BZ06]. Shortly after the high-rate breakthroughs [KSY14, GKS13], Hemenway, Ostrovsky and Wootters [HOW15] showed that some families of expander codes also admit sublinear local correcting algorithms from a constant fraction of errors, while preserving a rate arbitrary close to one.

Expander codes are named after *expander graphs*. Given a graph $G = (V, E)$, we denote by $E(v) \coloneqq \{\{v, w\} \in E\}$ the edges incident to v, and the neighbourhood of a subset of vertices $W \subseteq V$ is $N(W) \coloneqq \{v \in V \mid \exists w \in W, \{v, w\} \in E\}$. Recall that a d-regular expander graph has expansion parameter λ if for every subset $W \subset V$ of size $\leq |V|/2$, we have $|N(W) \setminus W| \geq \lambda |W|$. It is important to remark that Ramanujan graphs (introduced in [LPS88]) define *extremal* d-regular expander graphs, *i.e.* they satisfy $\lambda d \leq 2\sqrt{d-1}$. From now on, we assume G is a Ramanujan graph.

Denote by $G' = (V', E')$ the double cover of $G = (V, E)$. In other words, G' is a bipartite graph where $V' = V_0 \sqcup V_1$ (V_0, V_1 being disjoint copies of V) and $(u_0, v_1) \in E'$ if $\{u_0, v_1\} \in E$. Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ be a code of relative distance δ_0 and rate r_0. Finally, let $\Phi = \{\phi_v : [1, d] \to E'(v)\}_{v \in V'}$ be a collection of bijective maps indexed by $v \in V'$.

Definition 2.25 (Expander code)**.** For $\mathcal{C}_0$, G, Φ defined as above, and a given finite field $\mathbb{F}_q$, the associated *expander code* is:

$$\mathrm{Exp}(\mathcal{C}_0, G, \Phi) = \{c \in \mathbb{F}_q^{E'} \text{ such that } \forall v \in V', c \circ \phi_v \in \mathcal{C}_0\} \subseteq \mathbb{F}_q^{E'}.$$

Code $\mathcal{C}_0$ is called the inner code of $\mathrm{Exp}(\mathcal{C}_0, G, \Phi)$.

Before getting into the main result of Hemenway *et al.*, we need a last definition. We say that (Q, R) is an (s, ℓ_0)-smooth local reconstructing algorithm for $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ if, for every $1 \leq i \leq d$,

1. $\mathsf{Q}(i)$ outputs random subsets of $[1, d]$ of size less than ℓ_0;
2. there exists S_i of size s such that $\mathsf{Q}(i)$ is uniformly distributed over S_i;
3. R is deterministic and $\mathsf{R}(\boldsymbol{c}_{|Q}, Q) = c_i$ for all $Q \leftarrow_{\mathrm{R}} \mathsf{Q}(i)$ and $\boldsymbol{c} \in \mathcal{C}_0$.

In a sense, an (s, ℓ_0)-smooth local reconstructing algorithm can be thought as a local decoding algorithm, with specific smoothness requirements for the queries, but working only without errors on the codeword.

Finally, Hemenway *et al.* [HOW15] proved the following result concerning local correctability of expander codes.

Theorem 2.26 (Theorem 5 in [HOW15])**.** *Let $\mathcal{C}_0$, G, Φ as above, and equip $\mathcal{C}_0 \subseteq \mathbb{F}_q^d$ with an (s, ℓ_0)-smooth local reconstructing algorithm. Denote by $\ell_0' = \ell_0 + d - s$. Let also $\gamma < 1/2$ and $\zeta > \gamma$ such that $8\lambda < \gamma/(e^{\zeta}\ell_0)^{1/\gamma}$. Then, for any $\delta < \gamma/(e^{\zeta}\ell_0)^{1/\gamma} - 2\lambda$, the code $\mathrm{Exp}(\mathcal{C}_0, G, \Phi)$ is an $(\ell, \delta, \varepsilon)$-LCC of length $n = |V|d$, with*

$$\ell = |V|^{\alpha}, \qquad \alpha = \left(1 + \frac{\ln(\ell_0') + 1}{\zeta - \gamma}\right) \cdot \frac{\ln(\ell_0')}{\ln(d/4)}$$

and

$$\varepsilon = \frac{1}{|V|^{1/\ln(d/4)}}.$$

Moreover, the rate of $\mathrm{Exp}(\mathcal{C}_0, G, \Phi)$ is at least $2r_0 - 1$, where r_0 is the rate of $\mathcal{C}_0$.

We just sketch the principle of the local correction (for details, see [HOW15] or Wootters' PhD thesis [Woo14]). Assume we want to retrieve symbol $c(e)$, $e = (u, v) \in E'$. The algorithm consists in three steps.

1. *Compute a subset of edges $S \subset E'$ which is merely independent from e.* We know that the map $\phi_u : [1,d] \to E'(u)$ is such that $c \circ \phi_u \in \mathcal{C}_0$. Therefore, we can use the query generator Q to generate a set $S_1 = \{(u_0 = u, s_1^{(i)}), 1 \leq i \leq \ell_0\}$ satisfying $\mathsf{R}(c \circ \phi_u, \phi_u^{-1}(e)) = c(e)$. We name T_1 the tree with root e and leaves S_1. It is possible that edges in S_1 are still not uniformly distributed over E' (they are too close to e), so let us repeat the previous procedure on every edge in S_1. The tree we get is called T_2. Inductively, there will exist an $L > 0$ such that leafs S_L of tree T_L are close to uniform over E' (a good point is that neither L nor S is too large since G is an expander code). At this time we define $S := S_L$.
2. *Recover all $c(f)$, $f \in S$, with high probability.* Now S is uniformly distributed over E', hence we know that $c(f)$, $f \in S$, is correct with probability merely $1 - \delta$. In order to correct $c(f)$ w.h.p., once again we compute trees $T_M^{(f)}$ of depth M. Without diving into details, by querying values $c(f')$, $f' \in T_M^{(f)}$ (that are possibly corrupted), and then applying in-depth majority-logic decoding, we retrieve w.h.p. each correct coordinates $c(f)$.
3. *Retrieve $c(e)$.* Assuming we get correct values $c(f)$, $f \in S$, it is now easy to go back up the initial tree T_L until we reach its root, and with the help of algorithm R we recover $c(e)$. Notice this last step outputs the desired symbol $c(e)$ as soon as all the $c(f)$, $f \in S$ are correctly recovered.

Other algebraic constructions. Table 2.1 summarises parameters of the constructions we presented above. We only give bounds or asymptotics of these parameters when exact values are unknown or hard to interpret.

Construction	LCC	$\ell(n)$	$k(n)$	$\varepsilon(\delta)$
Hadamard code $\mathrm{Had}_2(m)$	yes	2	$\log(n)$	2δ
$\mathrm{RM}_q(m, (1-2\tau)q)$ (Algo. 2)	yes	$n^{1/m}$	$\leq \frac{(1-2\tau)^m}{m!} \cdot n$	δ/τ
$\mathrm{RM}_q(m, (1-2\tau)q)$ (Algo. 4)	yes	$(1-2\tau)n^{1/m}$	$\leq \frac{1}{m!} \cdot n$	$\ell(n) \cdot \delta$
$\mathrm{Mult}_q(m, \tau(s+1)q, s)$	yes	$\geq \frac{s^m}{m!} \cdot n^{1/m}$	$\geq \tau^m \cdot n$	≥ 0.8
$\mathrm{Lift}_q(\mathrm{RS}_q((1-2^{-c})q), m)$, with $c = 2(1+\log(m))m2^m\alpha$	yes	$n^{1/m}$	$\geq (1-2^{-\alpha}) \cdot n$	$\leq 2\left(\frac{n}{n-k(n)}\right)^c$
Matching vector code (Prop. 2.24.1, asymptotics in ℓ)	no	ℓ	$\widetilde{\mathcal{O}}\left(2^{(\log\log n)^{\log \ell}}\right)$	$\mathcal{O}(1)$
Matching vector code (Prop. 2.24.2, asymptotics in ℓ)	no	ℓ	$\widetilde{\mathcal{O}}\left(2^{(\frac{\log\log n}{\log \ell})^{\frac{\log \ell}{\log\log \ell}}}\right)$	$\mathcal{O}(1)$
Expander code with a $[d, \tau d]$ local code $\mathcal{C}_0$	yes	$\left(\frac{n}{d}\right)^{\Theta(\log d)}$	$(2\tau - 1) \cdot n$	$\left(\frac{n}{d}\right)^{-\Theta(1/\log d)}$

Table 2.1 – A summary of constructions of locally decodable or correctable codes presented earlier.

Let us also report other recent constructions of high rate LCCs. First, the work of Ben-Sasson *et al.* [BGK+13] proposed to use Reed-Muller-like decoding algorithm on evaluation codes defined over other algebraic varieties than the classical affine space, namely Cartesian products H^m of Hermitian curves H. The reason of this choice is that the Hermitian function field admits a large automorphism group which can be used to define structured queries that are close to uniform over H^m. Several correction techniques are then proposed, including a direct analogue of the Reed-Muller local correction on lines (instead we get codewords

supported by the Hermitian curve), or a so-called *fractal decoding* where a tree of local Hermitian curves to be queried is computed. Notice that the notion of *degree-lifting* mentioned in [BGK+13] is different from the lifting process of Guo *et al.* [GKS13]; indeed it refers to the way of weighing monomials in order to construct the space of multivariate polynomials to be evaluated. However, codes in [BGK+13] still have rate bounded by $1/m!$ while their locality is $\ell = \mathcal{O}(n^{1/m})$; compared to Reed-Muller codes the quantitative improvement consists essentially in the size of the alphabet.

Following his work on the affine space [GKS13], Guo [Guo16] proposed to increase the rate of previous codes by computing an actual lift of Hermitian codes on the Cartesian product H^m. To this end, the author adopts a more generic framework where he considers the lifting process with respect to a set a maps. Formally, if $\mathcal{C}_0 \subseteq \mathbb{F}_q^S$ and Φ is a collection of maps $S^m \to S$, the lift of $\mathcal{C}_0$ with respect to Φ is

$$\mathrm{Lift}_\Phi(\mathcal{C}_0) \coloneqq \{c \in \mathbb{F}_q^{S^m} \mid \forall \phi \in \Phi, c \circ \phi \in \mathcal{C}_0\}\,. \tag{2.5}$$

Of course, the definition given in (2.5) can be paired with Definition 2.17. But one also should remark formal similarities with the definition of expander codes, even though they instantiate very differently (polynomials and algebraic varieties for lifted codes vs. graphs and combinatorial structures for expander codes).

Finally, we report a recent construction of high-rate LCCs with query complexity subexponential in $\log n$, given by Kopparty, Meir, Ron-Zewi and Saraf [KMRS17]. They build LCCs of length n, constant rate arbitrarily close to 1, constant relative distance and locality $L_n(1/2, 1)$. Notice this is an exponential improvement compared to other high-rate LCC (multiplicity codes, lifted codes, expander codes) whose locality is $L_n(1, \alpha)$ for arbitrary $\alpha > 0$. The construction is very elaborate, and mixes multiplicity codes in a low-error regime with the Alon-Luby distance-amplification method [AL96], by using expander graphs and concatenation of codes.

2.3.2 Bounds

We here report a few lower bounds on the length of LDCs and LCCs, depending on their dimension and locality. In the low-dimension regime, recall we must take care to distinguish locally decodable and locally correctable codes, since the last ones are much more constrained. Indeed, LCCs require local recovery of any *codeword symbol*, while in LDCs this requirement must be met only for *information symbols*. If $k \ll n$, this difference can be crucial.

Most of current bounds on codes with locality were established for LDCs. The reason is that authors mainly focused on the constant locality regime, for which LDCs might attain higher dimensions than LCCs. In this section we quickly report some of these bounds, as well as some bounds on LCCs when they exist.

When the locality is 1, Katz and Trevisan [KT00] proved that every $(1, \delta, \varepsilon)$-LDC which encodes binary strings (that is, a code $\mathcal{C} \subseteq \Sigma^n$ equipped with an encoder $E : \mathbb{F}_2^k \to \mathcal{C}$) must have constant dimension $k \leq \frac{\log(|\Sigma|)}{\delta(1-H(\varepsilon))}$, where $H(x) = -x \log x - (1-x)\log(1-x)$ is the binary entropy function. For larger but constant localities $\ell \geq 2$, they used a graph theoretic formalism to prove the following result.

Theorem 2.27 (Theorem 6 in [KT00])**.** *Let $\mathcal{C} \subseteq \Sigma^n$, equipped with an encoder $E : \mathbb{F}_2^k \to \Sigma^n$, be an $(\ell, \delta, \varepsilon)$-locally decodable code. Then:*

$$n \geq \left(\frac{(1/2-\varepsilon)\delta}{\ell^2}\right)^{\frac{1}{\ell-1}} \left(\frac{3k(1-H(\varepsilon))}{4\log(|\Sigma|)}\right)^{1+\frac{1}{\ell-1}} \tag{2.6}$$

Asymptotically, Equation (2.6) rewrites $n = \Omega((\frac{k}{\log(|\Sigma|)})^{1+\frac{1}{\ell-1}})$ when $k \to \infty$. Therefore, Theorem 2.27 shows that, for constant locality and alphabet size, there cannot exist asymptotically good families of LDCs. However, the gap between this bound and existing constructions is still exponential: for instance, for $\ell = 2$, binary Hadamard codes satisfy $n = 2^k - 1$.

In the early 2000's, Goldreich, Karloff, Schulman, and Trevisan proved that *linear* LDCs with locality 2 must have exponential length $n = 2^{\Omega(k)}$ [GKST06]. Kerenidis and de Wolf [KdW04] generalised the result to unrestricted codes, using arguments from quantum information theory. Notice that both results were stated for finite fields of bounded size; Dvir and Shpilka then proved it for any (possibly infinite) field [DS07]. Notice that this means that Hadamard codes are essentially optimal LDC for locality $\ell = 2$. Concerning LCCs, it is shown in [BDSS16] that 2-queries LCCs over $\mathbb{F}_p$ must have length $n = p^{\Omega(\delta k)}$. This induces an exponential gap between the length of optimal 2-queries LDCs over $\mathbb{F}_p$ (*e.g.* the p-ary Hadamard code) and lower bounds on the length of 2-queries LCCs over the same alphabet. For infinite fields, it is even proved in [BDYW11] that 2-queries LCCs must have bounded dimension $k = \mathcal{O}(1/\delta^9)$ regardless of their length.

For $\ell = 3$, a series of papers including [Yek08, Efr12] proved constructively that subexponential length is achievable. On the opposite, Katz and Trevisan's lower bound has been improved into a quadratic one ($n = \Omega(k^2)$) by Woodruff in [Woo12], slightly raised to $n = \Omega(k^{2+\alpha})$, $\alpha > 0$, for codes over reals in [DSW17]. Still, there remains an exponential gap for the length of 3-queries LDCs, between existing constructions and proven lower bounds.

For larger but constant localities $\ell > 3$, only a few results are known. They are mostly based on reduction from ℓ queries to 2, and then using above bounds. This way, Woodruff [Woo07] obtained a bound $n = \tilde{\Omega}(k^{1+1/(\lceil \ell/2 \rceil - 1)})$ when $k \to \infty$.

2.4 LCCs from a design theory perspective

In this section, we study the links between designs and locally correctable codes. The intuition is that, in a code $\mathcal{C}$ based on a well-structured design (say, a 2-design), blocks support parity-check equations for $\mathcal{C}$ whose coordinate are uniformly distributed over the support of the code. Hence, regardless of quantitative parameters, a design naturally induces a locally correctable code. The converse construction is however trickier. Indeed, few assumptions are made concerning the queries and the reconstruction step of a generic LCC. Notice that the link between LCCs and designs has been studied in [BIW10] where the authors show that LCCs based on secret sharing schemes give rise to *almost* 2*-designs*, in the sense that the incidence requirements of 2-designs hold for almost every points and blocks.

We should emphasize that we do not claim to give new constructions of LCCs achieving specifically good parameters in this section. The incentive is to propose a *generic* conversion from combinatorial structures such as block designs, into linear codes admitting a probabilistic local correcting algorithm. Nevertheless, we hope that this perspective can open new tracks of research for constructions (or bounds) of locally correctable codes.

2.4.1 Formulating some LCCs as design-based codes

We begin by noticing that binary Hadamard codes can be seen as design-based codes. Recall that $\mathrm{PG}_1(m, 2)$ denotes the classical design of points and lines in the projective space over $\mathbb{F}_2$.

Proposition 2.28. *Let $m \geq 2$. Denote by $S = \mathbb{F}_2^m \setminus \{\mathbf{0}\}$ and by $\mathbb{P}^{m-1}$ the projective space of order $(m-1)$ over $\mathbb{F}_2$. Then $\mathrm{Had}_2(m) \subset \mathbb{F}_2^S$ and $\mathrm{Code}_2(\mathrm{PG}_1(m-1,2)) \subset \mathbb{F}_2^{\mathbb{P}^{m-1}}$ are permutation-equivalent, through the canonical bijective map $S \to \mathbb{P}^{m-1}$.*

Proof. We identify affine points in S and projective points in $\mathbb{P}^{m-1}$ since we are over $\mathbb{F}_2$, meaning we work up to the canonical bijection $S \to \mathbb{P}^{m-1}$. Let $\mathrm{ev}_S(f) \in \mathrm{Had}_2(m)$, where $f \in \mathbb{F}_2[X_1, \ldots, X_m]$ is a linear form over $\mathbb{F}_2^m$. Then, for any projective line $L = \{\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{u}+\boldsymbol{v}\}$ of $\mathbb{P}^{m-1}$, we have:

$$\sum_{\boldsymbol{x} \in L} f(\boldsymbol{x}) = f(\boldsymbol{u}) + f(\boldsymbol{v}) + f(\boldsymbol{u}+\boldsymbol{v}) = 0$$

since f has degree 1 and $1 = -1$ over $\mathbb{F}_2$. Therefore $\mathrm{ev}_S(f) \in \mathrm{Code}_2(\mathrm{PG}_1(m-1,2))$.

Let now $\boldsymbol{c} \in \mathrm{Code}_2(\mathrm{PG}_1(m-1,2))$ be a non-zero codeword. Denote by $\boldsymbol{e}_i$ the i-th element of the canonical basis of $\mathbb{F}_2^m$. By induction on the Hamming weight of $\boldsymbol{x} \in \mathbb{P}^{m-1}$, one can prove that $c_{\boldsymbol{x}} = \sum_{i:x_i=1} c_{\boldsymbol{e}_i}$, and thus we get $c_{\boldsymbol{x}} = f_c(\boldsymbol{x})$ where

$$f(X_1, \ldots, X_m) = \sum_{i=1}^{m} c_{\boldsymbol{e}_i} X_i$$

as degree 1. This proves that $\boldsymbol{c} \in \mathrm{Had}_2(m)$. □

The same kind of relation lies between $\mathrm{Lift}(\mathrm{RS}_q(q-2), m)$ and the classical affine design $\mathrm{AG}_1(m,q)$.

Proposition 2.29. *Let $m \geq 1$. It holds:*

$$\mathrm{Lift}(\mathrm{RS}_q(q-2), m) = \mathrm{Code}_q(\mathrm{AG}_1(m,q))\,.$$

Proof. Recall that $\boldsymbol{c} = \mathrm{ev}_{\mathbb{F}_q^m}(f) \in \mathbb{F}_q^{\mathbb{F}_q^m}$ lies in $\mathrm{Lift}(\mathrm{RS}_q(q-2), m)$ if and only if for all $\phi \in \mathrm{Aff}(\mathbb{F}_q, \mathbb{F}_q^m)$, we have $\mathrm{ev}_{\mathbb{F}_q}(f \circ \phi) \in \mathrm{RS}_q(q-2)$. We have seen in the previous chapter that $\mathrm{RS}_q(q-2)$ is the parity-check code, meaning that

$$\mathrm{RS}_q(q-2) = \{\boldsymbol{a} \in \mathbb{F}_q^{\mathbb{F}_q} \mid \sum_{t \in \mathbb{F}_q} a_t = 0\}\,.$$

Since any affine line L can be written $\phi(\mathbb{F}_q)$ for $\phi \in \mathrm{Aff}(\mathbb{F}_q, \mathbb{F}_q^m)$, we get our result. □

Propositions 2.28 and 2.29 imply that some designs give rise to locally correctable codes. In the following subsection we prove a more general statement.

2.4.2 Design-based codes for low-error LCCs

Let $\mathcal{D} = (X, \mathcal{B})$ be a design, and $\mathcal{C} = \mathrm{Code}_q(\mathcal{D})$ be the associated code over $\mathbb{F}_q$. The idea for correcting $\mathcal{C}$ locally is the following. Given $\boldsymbol{c} \in \mathcal{C}$ and $x \in X$, if we want to correct symbol c_x, we pick at random a block $B \in \mathcal{B}$ such that $x \in B$ and we retrieve c_x thanks to the the parity-check equation of the code that we know to be supported by B. As long as there is no error supported by B, the algorithm outputs the right symbol.

Informally, one sees that the distribution of blocks must be uniform over X if we want to get smooth queries. Therefore, 2-designs seems to be adapted for instantiating the above idea. Let us formalise it in Algorithm 8.

Algorithm 8: An $(\ell, \delta, \delta\ell)$-local correcting algorithm for a code $\mathcal{C} \subseteq \mathbb{F}_q^X$ based on a 2-design $\mathcal{D} = (X, \mathcal{B})$ with block size $\ell + 1$.

Input: $x \in X$, and an oracle access to $\boldsymbol{y} \in \mathbb{F}_q^X$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq \delta n$ for some $\boldsymbol{c} \in \mathcal{C}$, where $n = |X|$.

Output: c_x with high probability.

Pick uniformly at random a block $B \in \mathcal{B}$ such that $x \in B$.
Toss a random binary coin b, which is 0 with probability $p = \ell/(n + \ell - 1)$.
if $b = 0$ **then**
    Query $\{y_x\}$ and output y_x.
else
    Query $\{y_b, b \in B \setminus \{x\}\}$ and output $-\sum_{b \in B \setminus \{x\}} y_b$.

Proposition 2.30. *Let $D = (X, \mathcal{B})$ be a 2-$(n, \ell + 1, \lambda)$-design, and $\mathcal{C} = \mathrm{Code}_q(D)$. Then using Algorithm 8, for every $\delta < 1/2\ell$ the code $\mathcal{C}$ is a perfectly smooth $(\ell, \delta, \delta\ell)$-locally correctable code.*

Proof. The proof relies on arguments that might now be redundant to readers. We first prove that individual queries are uniform on the support X. In Algorithm 8, the desired coordinate x is queried with probability $p = \ell/(n + \ell - 1)$. Moreover, for every $a \neq x$, there exists a constant number λ of blocks B of same size $\ell + 1$, such that the pair $\{x, a\} \subset B$. Therefore, for every $a \neq x$, we have

$$\Pr(a \text{ is queried}) = (1 - p) \Pr(a \in B) = \left(1 - \frac{\ell}{n + \ell - 1}\right) \frac{\ell}{n - 1} = \frac{\ell}{n + \ell - 1} = p\,.$$

Then, Algorithm 8 succeeds if there is no error on the queried symbols. Therefore, denoting S_x the set of queries, and $E \subset X$, $|E| \leq \delta n$, the support of the errors, we obtain:

$$\begin{aligned} \Pr(\mathsf{LC}^{(\boldsymbol{y})}(x) = c_x) &\geq 1 - \mathbb{E}(|E \cap S_x|) \geq 1 - \sum_{e \in E} \Pr(e \in S_x) \\ &\geq 1 - \frac{\delta n \ell}{(n + \ell - 1)} > 1 - \delta\ell\,. \end{aligned} \tag{2.7}$$

□

Since $\mathrm{AG}_1(m, q)$ and $\mathrm{PG}_1(m, 2)$ are 2-designs, thanks to Proposition 2.28 and 2.29 we retrieve the fact that binary Hadamard codes and lifted Reed-Solomon codes of dimension $q - 1$ are locally correctable. Also notice that in Equation (2.7), the last inequality is tight up to a constant factor (since $\ell \leq n$), thus the correction failure ε is $\Theta(\delta\ell)$ even if we are overcautious in bounding ε.

To sum up, we have seen that $(\ell, \delta, \delta\ell)$-locally correctable codes can be built from 2-designs of block size $\ell + 1$. Their local correcting algorithm is inspired by the low-error correcting algorithms of Hadamard codes or Reed-Muller codes of degree $q - 2$. We have seen in previous subsections that a larger amount of errors can be corrected if the reconstruction step is resilient to a fraction of errors. Next subsection is devoted to pushing this idea in the world of design-based codes.

2.4.3 Generalised design-based codes

We introduce a generalisation of design-based codes in order to be able to correct *locally* a fraction of errors, similarly to the high-error local correction procedures of Reed-Muller codes. Notice that, in this subsection, elements in $\mathbb{F}_q^X$ will be preferably seen as maps $X \to \mathbb{F}_q$.

Definition 2.31 (Generalised design-based code)**.** Let $\mathcal{D} = (X, \mathcal{B})$ be a design, and $\mathcal{L} = (\mathcal{L}_B \subseteq \mathbb{F}_q^B : B \in \mathcal{B})$ be a family of codes indexed by blocks $B \in \mathcal{B}$, called the *local codes*. The *generalised design-based code* with respect to $\mathcal{L}$ is:

$$\mathrm{Code}_q(\mathcal{D}, \mathcal{L}) \coloneqq \{c \in \mathbb{F}_q^X \text{ such that } \forall B \in \mathcal{B},\, c_{|B} \in \mathcal{L}_B\} \subseteq \mathbb{F}_q^X\,.$$

Our terminology and notation for *generalised* design-based codes makes sense, since $\mathrm{Code}_q(\mathcal{D})$ is actually an instance of $\mathrm{Code}(\mathcal{D}, \mathcal{L})$, where we define $\mathcal{L} = (\mathcal{L}_B : B \in \mathcal{B})$ to be such that every $\mathcal{L}_B$ is a parity-check code.

Example 2.32. Let $\mathcal{D} = \mathrm{AG}_1(m, q)$ be the classical affine design. For each affine line $B \in \mathcal{B}$, let $\phi_B : \mathbb{F}_q \to \mathbb{F}_q^m$ be an affine injective map such that $\mathrm{im}(\phi_B) = B$, and define the code $\mathcal{L}_B$ as the code isomorphic to $\mathrm{RS}_q(k) \subseteq \mathbb{F}_q^{\mathbb{F}_q}$ by the action of ϕ_B. Then, denoting $\mathcal{L} = (\mathcal{L}_B : B \in \mathcal{B})$, we have

$$\mathrm{Code}(\mathcal{D}, \mathcal{L}) = \mathrm{Lift}(\mathrm{RS}_q(k), m)\,.$$

Indeed, for every line $B \in \mathcal{B}$, $c_{|B} \in \mathcal{L}_B$ holds if and only if $\phi_B^*(c_{|B}) \in \mathrm{RS}_q(k)$. It remains to notice that $\phi_B^*(c_{|B})$ corresponds to the evaluation of $c \circ \phi_B$ on $\mathbb{F}_q$.

For a fixed design $\mathcal{D}$, let us understand the impact of the family $\mathcal{L}$ on $\mathrm{Code}(\mathcal{D}, \mathcal{L})$. For two families of local codes $\mathcal{L}, \mathcal{L}'$ indexed by B, we say that $\mathcal{L} \subseteq \mathcal{L}'$ if we have $\mathcal{L}_B \subseteq \mathcal{L}'_B$ for all $B \in \mathcal{B}$.

Definition 2.33. We say that a family of codes $\mathcal{L} = (\mathcal{L}_B \subseteq \mathbb{F}_q^B, B \in \mathcal{B})$ is $(1, w)$-correcting if every $\mathcal{L}_B$ corrects at least any 1 erasure and w errors.

In the next result, we prove that the minimum distance of generalised design-based codes can be lower-bounded, depending on the minimum of the local codes ones.

Proposition 2.34. *Let $t \geq 2$, and $\mathcal{D} = (X, \mathcal{B})$ be a t-$(n, \ell + 1, \lambda)$-design. Let $\mathcal{L}$ be a family of codes indexed by $\mathcal{B}$, and $\mathcal{C} = \mathrm{Code}(\mathcal{D}, \mathcal{L})$. Finally, define $d \coloneqq \min\{\mathrm{d}_{\min}(\mathcal{L}_B), B \in \mathcal{B}\}$. Then,*

$$\mathrm{d}_{\min}(\mathcal{C}) \geq 1 + \frac{n-1}{\ell}(d-1)\,.$$

Proof. We will prove the result for $t = 2$. Since every t-$(n, \ell + 1, \lambda)$-design is also a 2-$(n, \ell + 1, \lambda\binom{n-(t-2)}{t-2})$-design (see Lemma 1.10), the proposition will be also proved for larger $t \geq 2$.

If $\mathcal{C} = \{0\}$, the result trivially holds, so let us assume $\mathcal{C} \neq \{0\}$. Let $c \in \mathcal{C}$ be a minimum-weight codeword, $\mathrm{wt}(c) = D \neq 0$. By definition there exists $x \in X$ such that $c(x) \neq 0$. Define $\mathcal{B}_x = \{B \in \mathcal{B}, x \in B\}$; we know that $|\mathcal{B}_x| = \lambda\frac{n-1}{\ell-1}$. Furthermore,

$$\sum_{B \in \mathcal{B}_x} \mathrm{wt}(c_{|B}) = |\mathcal{B}_x| + \sum_{B \in \mathcal{B}_x} \mathrm{wt}(c_{|B \setminus \{x\}}) = |\mathcal{B}_x| + \lambda(D-1)$$

since every point of $X \setminus \{x\}$ appears exactly λ times in the summation. Therefore, there must exist a block $B_0 \in \mathcal{B}_x$ such that

$$\mathrm{wt}(c_{|B_0}) \leq \frac{1}{|\mathcal{B}_x|} \sum_{B \in \mathcal{B}_x} \mathrm{wt}(c_{|B}) = 1 + \frac{\lambda(D-1)}{|\mathcal{B}_x|} = 1 + \frac{(D-1)(\ell-1)}{n-1}\,.$$

Since $c_{|B_0} \in \mathcal{L}_{B_0}$ is non-zero, we have $\mathrm{wt}(c_{|B_0}) \geq d$, which gives the expected result. □

As a corollary, using the same notation as in Proposition 2.34, the Singleton bound implies that $\dim(\mathcal{C}) \leq n(1 - \frac{d-1}{\ell-1}) + \frac{d-1}{\ell-1}$ for codes based on 2-designs.

Local correction of generalised design-based codes. We here show how our generalisation of codes based on designs leads to locally correctable codes handling a constant fraction of errors. A generic local correcting algorithm is presented in Algorithm 9. The analysis of the fraction of errors the code can handle is given in Proposition 2.35 for the case of 2-designs, and Proposition 2.36 for the case of 3-designs.

Algorithm 9: A local correcting algorithm with locality ℓ for a code $\mathcal{C} = \mathrm{Code}(\mathcal{D}, \mathcal{L}) \subseteq \mathbb{F}_q^X$, where $\mathcal{D} = (X, \mathcal{B})$ is a design of block size $\ell + 1$, and $\mathcal{L}$ is $\lfloor \tau\ell \rfloor$-correcting for $0 < \tau < 1$.

Input: $x \in X$, and an oracle access to $g \in \mathbb{F}_q^X$, $|X| = n$, such that $\mathrm{d}(g, f) \leq \delta n$ for some $f \in \mathcal{C}$.

Output: $f(x)$ with high probability.

`/* Corr_B denotes an algorithm correcting at least ⌊τℓ⌋ errors and 1 erasure in the code L_B. */`

Pick uniformly at random a block $B \in \mathcal{B}$ such that $x \in B$.
Toss a random binary coin b, which is 0 with probability $p = \ell/n$.
if $b = 0$ **then**
    Pick $\overline{y}$ uniformly at random in $B \setminus \{x\}$.
else
    Define $\overline{y} = x$.
Define $A = B \setminus \{\overline{y}\}$.
Define $g' \in (\mathbb{F}_q \cup \{\perp\})^B$ by $g'_{|A} = g_{|A}$ and $g'(\overline{y}) = \perp$. Run Corr_B on input g'.
if Corr_B *fails* **then**
    Abort.
else
    Denote by $h \in \mathbb{F}_q^B$ the output of Corr_B. Output $h(x)$.

Proposition 2.35 (Case of 2-designs)**.** *Let $0 < \tau < 1$ and $\mathcal{D}$ be a 2-$(n, \ell + 1, \lambda)$-design. Assume $\mathcal{L}$ is a $\lfloor \tau\ell \rfloor$-correcting family of local codes for $\mathcal{D}$. Then, for every $\delta < \tau/2$, the code* $\mathrm{Code}(\mathcal{D}, \mathcal{L})$ *is a perfectly smooth $(\ell, \delta, \delta/\tau)$-LCC.*

Proof. We use the notation of Algorithm 9. Once again, the proof is similar to other proofs for high-error resilient LCCs. We easily check the algorithm is perfectly smooth. Thanks to Markov's inequality, we also get:

$$\Pr(\mathsf{LC}^{(g)}(x) = f(x)) \geq 1 - \Pr(|E \cap A| \geq \tau\ell + 1) \geq 1 - \frac{\mathbb{E}(|E \cap A|)}{\lfloor \tau\ell \rfloor + 1},$$

and the average number of errors on A is bounded by

$$\mathbb{E}(|E \cap A|) = \sum_{e \in E} \Pr(e \in A) = |E| \frac{\ell}{n} \leq \delta\ell\,.$$

We conclude easily since $\frac{\delta\ell}{\lfloor \tau\ell \rfloor + 1} \leq \frac{\delta}{\tau}$. □

Proposition 2.36 (Case of 3-designs)**.** *Let $0 < \tau < 1$ and $\mathcal{D}$ be a 3-$(n, \ell+1, \lambda)$-design. Assume $\mathcal{L}$ is a $\lfloor \tau\ell \rfloor$-correcting family of local codes for $\mathcal{D}$. Then, for every $\delta < \tau - 1/\sqrt{2\ell}$, the code $\mathrm{Code}(\mathcal{D}, \mathcal{L})$ is a perfectly smooth $(\ell, \delta, \varepsilon)$-LCC where*

$$\varepsilon = \frac{\delta(1-\delta)}{(\tau-\delta)^2} \cdot \frac{1}{\ell}\,.$$

Moreover, $\varepsilon = \mathcal{O}(1/\ell)$ when $\ell \to \infty$.

Proof. We use the notation of Algorithm 9 and we denote by E the support of the error. As always, we begin by checking that queries are smooth, and we notice that

$$\Pr(\mathsf{LC}^{(g)}(x) = f(x)) \geq 1 - \Pr(|E \cap A| \geq \lfloor \tau\ell \rfloor + 1)\,.$$

For each query A of the algorithm, let us fix an ordering $(a_1, \ldots, a_\ell)$ of points in A. For $1 \leq i \leq \ell$, we also consider the random variable $Y_i = \mathbb{1}_{a_i \in E}$ such that we have $|E \cap A| = \sum_{i=1}^{\ell} Y_i$. We see that, for every $1 \leq i \leq \ell$, Y_i is a Bernoulli variable of parameter $\frac{|E|}{n} \leq \delta$. Hence its mean value and variance satisfy respectively $\mathbb{E}(Y_i) \leq \delta$ and $\mathbb{D}(Y_i) \leq \delta(1-\delta)$.

Since $\mathcal{D}$ is a 3-design, there exists a constant number of blocks B containing any 3-subset of X. Hence, even if we only consider the blocks that pass through x, the random variables $\{Y_i\}$ are pairwise independent. Therefore we get

$$\mathbb{E}(|E \cap A|) \leq \delta\ell \quad \text{and} \quad \mathbb{D}(|E \cap A|) \leq \ell\delta(1-\delta)\,.$$

From Chebyshev's inequality, we obtain:

$$\begin{aligned}
\Pr(|E \cap A| \geq \lfloor \tau\ell \rfloor + 1) &\leq \Pr\Big(\big||E \cap A| - \mathbb{E}(|E \cap A|)\big| \geq \lfloor \tau\ell \rfloor + 1 - \mathbb{E}(|E \cap A|)\Big) \\
&\leq \frac{\mathbb{D}(|E \cap A|)}{(\lfloor \tau\ell \rfloor + 1 - \mathbb{E}(|E \cap A|))^2} \\
&\leq \frac{\ell\delta(1-\delta)}{(\lfloor \tau\ell \rfloor + 1 - \delta\ell)^2} \\
&< \frac{\delta(1-\delta)}{(\tau-\delta)^2} \cdot \frac{1}{\ell}\,.
\end{aligned}$$

It remains to notice that, if $\delta < \tau - 1/\sqrt{2\ell}$, then $\frac{\delta(1-\delta)}{\ell(\tau-\delta)^2} < 1/2$. □

We emphasize that Proposition 2.35 and 2.36 analyse the *same* local correcting algorithm, but this algorithm is applied on codes based on designs whose block structure is more or less elaborate. The difference in the result comes from the fact that blocks in 3-designs have a more refined structure than blocks in 2-designs, which allows the algorithm to succeed more frequently. Using 3-designs thus provides two improvements compared to 2-designs. First, the maximum error rate δ is now bounded by $\tau - \mathcal{O}(1/\sqrt{\ell})$ instead of $\tau/2$. Second, the failure probability is $\mathcal{O}(1/\ell)$ instead of $\mathcal{O}(1)$ when ℓ grows.

Remark 2.37. The use of highly structured set of queries can be found in other algebraic constructions of LCCs. A first occurrence might be the work of Woodruff and Yekhanin [WY05] in the perspective of resisting collusions in private information retrieval protocols. Given a multiplicity code (though the terminology did not appear yet), the authors proposed to recover symbols by generating queries supported by low-degree curves. This idea was then explicitly formalised for local correcting purposes in Yekhanin's survey [Yek12], where the

local correcting algorithm uses degree-2 curves. If we see these curves as blocks, the design we obtain is close to be a 3-design, in the sense that most of triplets of points are contained in a unique curve of degree 2. Quite similarly, we also report the local correction algorithm on curves for lifted codes given by Chee, Wu and Xing in [CWX14].

Remark 2.38. We have limited our study to the cases of t-designs with $t \in \{2,3\}$, because larger values of t do not provide significant improvements on the local correction parameters, when applying the same proof techniques as in Propositions 2.35 and 2.36.

We complete this section by recalling that 3-designs actually exist. For instance, so-called *inversive planes* provide an infinite family of 3-$(q^2+1, q+1, 1)$ designs. An inversive plane can be defined as follows. Let $X = \mathbb{F}_{q^2} \cup \{\infty\} \simeq \mathbb{P}^1(\mathbb{F}_{q^2})$. Let $B_0 = (\mathbb{F}_q \cup \{\infty\}) \subset X$, and $\mathcal{B}$ be the orbit of B_0 under $\mathrm{PGL}(2, \mathbb{F}_{q^2})$, the projective linear group consisting in homographies of the projective line $\mathbb{P}^1(\mathbb{F}_{q^2})$. Then $(X, \mathcal{B})$ is a 3-design.

However, first experimentations tend to show that inversive planes do not provide non-trivial codes. To see this, we report in Table 2.2 elementary divisors of the incidence matrices of inversive plane over $\mathbb{F}_q$, for small values of q. We observe that the associated code has dimension either 0 or 1, depending on q and the characteristic of the base field of the code. Based on the results presented in Table 2.2, we can also conjecture that the elementary divisors of the incidence matrix of the inversive plane over $\mathbb{F}_q$ are $\{1^{q^2+1}\}$ if $2 \mid q$, and $\{1^{q^2}, 2^1\}$ otherwise.

q	$\Gamma_{\mathbb{Z}}(\boldsymbol{M}_q)$	$\dim_{\mathbb{F}_2}(\mathcal{C})$	$\dim_{\mathbb{F}_p}(\mathcal{C})$, $p \neq 2$
2	$\{1^5\}$	0	0
3	$\{1^9, 2^1\}$	1	0
4	$\{1^{17}\}$	0	0
5	$\{1^{25}, 2^1\}$	1	0
7	$\{1^{49}, 2^1\}$	1	0
8	$\{1^{65}\}$	0	0
9	$\{1^{81}, 2^1\}$	1	0
11	$\{1^{121}, 2^1\}$	1	0

Table 2.2 – Elementary divisors $\Gamma_{\mathbb{Z}}(\boldsymbol{M}_q)$ of the incidence matrix $\boldsymbol{M}_q$ of the inversive plane over $\mathbb{F}_q$.

2.4.4 Further perspectives

To conclude the chapter, we open a few questions regarding the link between LCCs and (generalised) design-based codes that has been exhibited in the last section.

First, we have seen that, given a generalised design-based code $\mathcal{C} = \mathrm{Code}(\mathcal{D}, \mathcal{L})$, the parameters of a local correcting algorithm for $\mathcal{C}$ can be derived from the structure of $\mathcal{D}$ and from the error-correction capability of the family $\mathcal{L}$. Naturally, this raises the following problem: given a design $\mathcal{D}$ (or given a set of design parameters t, n, ℓ, λ), find families of codes $\mathcal{L}$ leading to the largest code $\mathcal{C}$.

For instance, Example 2.32 proved that lifted codes — which are LCCs with particularly high rate — can be seen as codes based on the affine geometry design $\mathcal{D} = \mathrm{AG}(m, q)$ and the family $\mathcal{L}$ of Reed-Solomon codes. Computations convinced us quickly that using a random family of codes $\mathcal{L}'$ (instead of the family $\mathcal{L}$ of Reed-Solomon codes) lead to very poor design-based codes $\mathcal{C} = \mathrm{Code}(\mathcal{D}, \mathcal{L}')$. A better understanding of the very reason of this irregularity should then be of primary relevance.

Secondly, one could notice that codes are mathematical structures that can be considered without any underlying decoding algorithm. However, the definition of locally correctable codes requires to attach a local and probabilistic algorithm to the code, and the parameters of the LCC (especially δ and ε) depend on the compatibility between the algorithm and the code.

The concept of generalised design-based codes allows us to relax this requirement, and to see codes with locality as purely combinatorial structures. An avenue of research could be to develop this idea and to propose a combinatorial definition for locally correctable codes, whose parameters could then be analysed *independently* of an underlying decoding algorithm.

Chapter 3

Projective lifted codes

Contents

In Chapter 2 was presented a family of high-rate and asymptotically good locally correctable codes called *lifted Reed-Solomon codes* [GKS13]. We recall they are built through the evaluation over the affine space of all the multivariate polynomials whose restriction to any affine line can be interpolated into a low-degree univariate polynomial. The first goal of this chapter is to give a formal definition of their analogues over projective spaces, that we name *projective lifted codes*. We will see they also admit perfectly smooth local correcting algorithms, whose parameters correspond quantitatively to the affine setting. As well, we exhibit interesting intertwined relations between projective and affine lifted codes, similar to those that hold for the families of Reed-Muller codes and design-based codes.

The chapter is organised as follows. A first section is devoted the introduction of the mathematical background necessary to our construction. Projective lifted codes are formally introduced in Section 3.2, where we also derive their first properties. We notably investigate their degree sets, a crucial notion introduced in Guo *et al.*'s work [GKS13]. Local correcting algorithms are proposed in Section 3.3. Last sections are devoted to further analyses of the codes. Relations with affine lifted codes *via* shortening and puncturing are given in Section 3.4; it allows us to produce recursive formulae for the dimension and some bases of projective lifted codes. Section 3.5 is devoted to features that are commonly studied in coding theory — such as the minimum distance, the automorphism group, or explicit information sets. This emphasizes the practicality of lifted codes. Finally, Section 3.6 is devoted to the practical computation of the dimension and bases of lifted codes.

3.1 Preliminaries

Let us here introduce the algebraic background for the definition of affine and projective lifted codes. Some notation and definitions are borrowed from the seminal work of Guo *et al.* [GKS13].

3.1.1 Affine and projective evaluation codes

Projective geometry. A base field $\mathbb{F}_q$ is fixed throughout the chapter. Coordinates of a representative of a projective point $\boldsymbol{a} \in \mathbb{P}^m$ are denoted $(a_0 : \ldots : a_m)$. We know that each projective point admits $(q-1)$ different representatives; we call *standard representative* the only one such that $\forall j < i, a_j = 0$ and $a_i = 1$. The projective space $\mathbb{P}^m$ contains $\theta_{m,q} \coloneqq \frac{q^{m+1}-1}{q-1}$ distinct points.

The *hyperplane at infinity* $\Pi_\infty \coloneqq \{\boldsymbol{a} \in \mathbb{P}^m, a_0 = 0\}$ is isomorphic to $\mathbb{P}^{m-1}$, and the bijective map $(a_1, \ldots, a_m) \mapsto (1 : a_1 : \ldots : a_m)$ embeds $\mathbb{A}^m$ into $\mathbb{P}^m$. A *projective line* is a $(q+1)$-subset of $\mathbb{P}^m$ of the form

$$L_{\boldsymbol{a},\boldsymbol{b}} \coloneqq \{x\boldsymbol{a} + y\boldsymbol{b}, (x : y) \in \mathbb{P}^1\},$$

where $\boldsymbol{a}, \boldsymbol{b}$ are distinct points in $\mathbb{P}^m$. The line $L_{\boldsymbol{a},\boldsymbol{b}}$ is the only one containing both $\boldsymbol{a}$ and $\boldsymbol{b}$, and there are exactly $\theta_{m-1,q} = |\mathbb{P}^{m-1}| = \frac{q^m-1}{q-1}$ projective lines on which a given point $\boldsymbol{a} \in \mathbb{P}^m$ lies.

Polynomials and degrees. Given a multivariate polynomial $f(\boldsymbol{X}) = \sum_{\boldsymbol{d}} f_{\boldsymbol{d}} \boldsymbol{X}^{\boldsymbol{d}} \in \mathbb{F}_q[\boldsymbol{X}]$, the set $\{\boldsymbol{d} \in \mathbb{N}^m, f_{\boldsymbol{d}} \neq 0\}$ is called the set of *degrees* of f and denoted $\mathrm{Deg}(f)$. This terminology is borrowed from [GKS13] for instance. If $D \subseteq \mathbb{N}^m$, we denote by $\mathrm{Poly}(D)$ the vector space of polynomials generated by monomials $\boldsymbol{X}^{\boldsymbol{d}}$ for $\boldsymbol{d} \in D$:

$$\mathrm{Poly}(D) \coloneqq \mathrm{Span}_{\mathbb{F}_q}\{\boldsymbol{X}^{\boldsymbol{d}}, \boldsymbol{d} \in D\} \subseteq \mathbb{F}_q[\boldsymbol{X}].$$

For instance, $f \in \mathrm{Poly}(\mathrm{Deg}(D))$. We write $|\boldsymbol{d}| \coloneqq \sum_i d_i$ the *weight* of a tuple of integers $\boldsymbol{d}$. Some subsets D are of particular interest. For instance, for $v \in \mathbb{N}$,

(i) the 1-norm ball $B_1^m(v) \coloneqq \{\boldsymbol{d} \in \mathbb{N}^m, |\boldsymbol{d}| \leq v\}$ generates the space $\mathbb{F}_q[\boldsymbol{X}]_{\leq v}$ of multivariate polynomials of total degree bounded by v,
(ii) the ∞-norm ball $B_\infty^m(v) \coloneqq \{\boldsymbol{d} \in \mathbb{N}^m \mid d_i \leq v, 1 \leq i \leq m\}$ generates the space of multivariate polynomials of partial degree bounded by v,
(iii) the 1-norm sphere $S^{m+1}(v) \coloneqq \{\boldsymbol{d} \in \mathbb{N}^{m+1}, |\boldsymbol{d}| = v\}$ generates the space $\mathbb{F}_q[\boldsymbol{X}]_v^H$ of homogeneous polynomials of degree v (plus the zero polynomial).

Evaluation of homogeneous polynomials on a projective point. For any homogeneous polynomial $f \in \mathbb{F}_q[\boldsymbol{X}]_v^H$, it is well-known that

$$f(\lambda \boldsymbol{X}) = \lambda^v f(\boldsymbol{X}), \quad \forall \lambda \in \mathbb{F}_q^\times .$$

It means that different representatives of a fixed projective point may result to different evaluations by f. In order to remove any ambiguity, we adopt the following convention. Let $(x_0 : \ldots : x_m)$ be the *standard* representation of a projective point $\boldsymbol{x} \in \mathbb{P}^m$. Then we define the *evaluation of f at $\boldsymbol{x}$* as:

$$\mathrm{ev}_{\boldsymbol{x}}(f) := f(x_0, \ldots, x_m) .$$

In other words, *every projective point must be written in the unique standard representation* when evaluated by homogeneous polynomials. Thanks to this definition, the following evaluation map can be defined without ambiguity for all $v \geq 0$:

$$\begin{array}{rccc} \mathrm{ev}_{\mathbb{P}^m} : & \mathbb{F}_q[\boldsymbol{X}]_v^H & \to & \mathbb{F}_q^{\theta_{m,q}} \\ & f & \mapsto & (\mathrm{ev}_{\boldsymbol{x}}(f) : \boldsymbol{x} \in \mathbb{P}^m) \end{array}$$

Recall its affine analogue is:

$$\begin{array}{rccc} \mathrm{ev}_{\mathbb{A}^m} : & \mathbb{F}_q[\boldsymbol{X}] & \to & \mathbb{F}_q^{q^m} \\ & f & \mapsto & (f(\boldsymbol{x}) : \boldsymbol{x} \in \mathbb{A}^m) \end{array}$$

Clearly, $\mathrm{ev}_{\mathbb{P}^m}$ and $\mathrm{ev}_{\mathbb{A}^m}$ are $\mathbb{F}_q$-linear maps. Since $x^q = x$ for all $x \in \mathbb{F}_q$, we have

$$\ker(\mathrm{ev}_{\mathbb{A}^m}) = \mathrm{Span}_{\mathbb{F}_q[\boldsymbol{X}]}\{X_i^q - X_i,\, 1 \leq i \leq m\} . \tag{3.1}$$

Moreover, since $\mathrm{ev}_{\mathbb{P}^m}$ evaluates homogeneous polynomials, for a fixed $v \in \mathbb{N}$ we obtain

$$\ker(\mathrm{ev}_{\mathbb{P}^m}) = \left(\mathrm{Span}_{\mathbb{F}_q[\boldsymbol{X}]}\left\{X_i^q X_j - X_i X_j^q,\, \forall i \neq j \in \{0, \ldots, m\}\right\}\right) \cap \mathbb{F}_q[\boldsymbol{X}]_v^H . \tag{3.2}$$

Proofs can be found in [RTR97] for instance.

Evaluation codes. We have seen in previous chapters that a common way to build linear codes is to evaluate polynomials over a list of points. Let us here formally define the family of *evaluation codes* we are studying.

Definition 3.1 (affine evaluation code)**.** Let $\mathcal{F}$ be a linear subspace of $\mathbb{F}_q[\boldsymbol{X}]$. The *affine evaluation code* associated to $\mathcal{F}$ is the $\mathbb{F}_q$-linear code of length $n = |\mathbb{A}^m| = q^m$ composed by the evaluation vectors of polynomials in $\mathcal{F}$:

$$\mathrm{ev}_{\mathbb{A}^m}(\mathcal{F}) = \{\mathrm{ev}_{\mathbb{A}^m}(f), f \in \mathcal{F}\} \subseteq \mathbb{F}_q^{\mathbb{A}^m} .$$

Definition 3.2 (projective evaluation code)**.** Let $v \in \mathbb{N}$ and $\mathcal{F}$ be a linear subspace of $\mathbb{F}_q[\boldsymbol{X}]_v^H$. The *projective evaluation code* associated to $\mathcal{F}$ is the $\mathbb{F}_q$-linear code of length $n = |\mathbb{P}^m| = \theta_{m,q}$ composed by the evaluation vectors of polynomials in $\mathcal{F}$:

$$\mathrm{ev}_{\mathbb{P}^m}(\mathcal{F}) = \{\mathrm{ev}_{\mathbb{P}^m}(f), f \in \mathcal{F}\} \subseteq \mathbb{F}_q^{\mathbb{P}^m} .$$

We point out a specific class of evaluation codes, generated by evaluation vectors of monomials. As we will see later, so-called *monomial codes* turn out to be very convenient to describe.

Definition 3.3 (monomial code)**.** An affine evaluation code $\mathcal{C} = \mathrm{ev}_{\mathbb{A}^m}(\mathcal{F})$ (resp. a projective evaluation code $\mathcal{C} = \mathrm{ev}_{\mathbb{P}^m}(\mathcal{F})$) is said to be *monomial* if $\mathcal{F} = \mathrm{Poly}(D)$ for some $D \subseteq \mathbb{N}^m$ (resp. $D \subseteq S^{m+1}(v)$).

Projective Reed-Solomon and Reed-Muller codes.

Definition 3.4 (Projective Reed-Solomon code)**.** Let $0 \leq k \leq q-1$. The *projective Reed-Solomon code* of degree k over $\mathbb{F}_q$ is the linear code of length $q+1 = |\mathbb{P}^1|$ consisting in the evaluation of bivariate homogeneous polynomials of degree k over $\mathbb{F}_q$:

$$\mathrm{PRS}_q(k) \coloneqq \{\mathrm{ev}_{\mathbb{P}^1}(f), f \in \mathbb{F}_q[X,Y]_k^H\}\,.$$

Projective Reed-Solomon codes are $[q+1, k+1, q-k]$-MDS codes over $\mathbb{F}_q$. In the literature, they are sometimes called *extended*, or *doubly-extended* Reed-Solomon codes. The reason is that, when puncturing $\mathrm{PRS}_q(k) \subseteq \mathbb{F}_q^{\mathbb{P}^1}$ on the point at infinity $(0:1)$, we obtain the full-length Reed-Solomon code $\mathrm{RS}_q(k)$.

Similarly, Reed-Muller codes admit analogues in projective spaces, which were firstly defined and studied by Lachaud [Lac86, Lac90] and Sørensen [Sør91].

Definition 3.5 (Projective Reed-Muller code)**.** Let $0 \leq v \leq m(q-1)$. The *projective Reed-Muller code* of order m and degree v over $\mathbb{F}_q$ is the linear code of length $|\mathbb{P}^m| = (q^{m+1}-1)/(q-1)$ consisting in evaluation vectors of $(m+1)$-variate homogeneous polynomials over $\mathbb{F}_q$ of degree v:

$$\mathrm{PRM}_q(m,v) \coloneqq \{\mathrm{ev}_{\mathbb{P}^m}(f), f \in \mathbb{F}_q[\boldsymbol{X}]_v^H\}\,.$$

The dimension of $\mathrm{PRM}_q(m,v)$ has been given by Sørensen [Sør91]:

$$\dim(\mathrm{PRM}_q(m,v)) = \sum_{t \in I_v} \left(\sum_{j=0}^{m+1} (-1)^j \binom{m+1}{j} \binom{t-jq+m}{t-jq} \right), \tag{3.3}$$

where $I_v = \{t \in [1,v],\, t \equiv v \mod q-1\}$. Notice that, for small $v \leq q-1$, Equation (3.3) simplifies to $\dim(\mathrm{PRM}_q(m,v)) = \binom{m+v}{v}$.

By definition, we see that $\mathrm{PRM}_q(1,k) = \mathrm{PRS}_q(k)$ for every $0 \leq k \leq q-1$.

3.1.2 Reduced degree sets

In the previous subsection, we have seen that well-known families of linear codes are defined as the image of polynomials by evaluation maps. For coding theoretic reasons (*e.g.* giving the dimension of the code, or computing a basis), it is interesting to find sets $D \subseteq S^{m+1}(v)$ (resp. $D \subseteq B_1^m(v)$) such that the evaluation map $\mathrm{ev}_{\mathbb{P}^m}$ (resp. $\mathrm{ev}_{\mathbb{A}^m}$) is injective over $\mathrm{Poly}(D)$.

Generally, we can group monomials according to their evaluation over the affine (resp. projective) space. Then, in each class of monomials, we can point out a specific representative $\boldsymbol{X}^{\boldsymbol{d}}$ — typically the lowest one according to a certain monomial order — and its exponent $\boldsymbol{d}$ represents its whole class of monomials. Such exponents $\boldsymbol{d}$ will be referred to as *reduced tuples*, or *reduced degrees*.

Definition 3.6 (A and P-reduced tuples)**.**

1. A tuple $\boldsymbol{d} \in \mathbb{N}^m$ is *A-reduced* if it lies in $B_\infty^m(q-1)$.
2. A tuple $\boldsymbol{d} = (d_0, \ldots, d_m) \in \mathbb{N}^{m+1}$ is *P-reduced* if, for all $0 \leq i \leq m$:

$$d_i \geq q \quad \Rightarrow \quad \begin{cases} d_j = 0 & \forall j < i\,, \\ d_j \leq q-1 & \forall j > i\,. \end{cases}$$

We see that any A-reduced tuple is also P-reduced. We also say that a set D of tuples is *A-reduced* (resp. *P-reduced*) if every tuple it contains is A-reduced (resp. P-reduced).

We denote by $(\boldsymbol{e}_1, \dots, \boldsymbol{e}_m)$ the canonical basis of $\mathbb{N}^m$. If $\boldsymbol{d} = (d_1, \dots, d_m) \in \mathbb{N}^m$ and $1 \leq j \leq m$ is such that $d_j \geq q$, then we define

$$\rho_j(\boldsymbol{d}) \coloneqq \boldsymbol{d} - (q-1)\boldsymbol{e}_j\,.$$

Similarly, if $\boldsymbol{d} \in S^{m+1}(v)$ for some $v > 0$, and $0 \leq i < j \leq m$ are such that $d_j \geq q$ and $d_i \geq 1$, we define

$$\tau_{ij}(\boldsymbol{d}) \coloneqq \boldsymbol{d} + (q-1)(\boldsymbol{e}_i - \boldsymbol{e}_j)\,.$$

Remark 3.7. Let $\boldsymbol{d} \in \mathbb{N}^m$. From (3.1) we see that $\mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\rho_j(\boldsymbol{d})}) = \mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}})$ for every $1 \leq j \leq m$ such that $d_j \geq q$. Moreover, as long as they are defined, $\rho_j \circ \rho_\ell = \rho_\ell \circ \rho_j$. Let us consider the image of $\boldsymbol{d}$ after repeatedly applying maps ρ_j until we cannot apply any of them. Then, we can see that the tuple we get is A-reduced.

Similarly, let $\boldsymbol{d} \in S^{m+1}(v)$ and $0 \leq i < j \leq m$. First notice that $|\tau_{ij}(\boldsymbol{d})| = |\boldsymbol{d}|$, hence every map τ_{ij} lets $S^{m+1}(v)$ invariant. We also have $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\tau_{ij}(\boldsymbol{d})}) = \mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}})$, and as long as they are defined, $\tau_{ij} \circ \tau_{k\ell} = \tau_{k\ell} \circ \tau_{ij}$. So once again, if we consider the image of $\boldsymbol{d}$ after applying maps τ_{ij} while it is possible, we get a P-reduced tuple.

The previous remark naturally leads to the upcoming definitions.

Definition 3.8. Let $\boldsymbol{d} \in \mathbb{N}^m$. The *$A$-reduction* of $\boldsymbol{d}$ is the tuple $\underline{\boldsymbol{d}} \in \mathbb{N}^m$ which is obtained by applying iteratively ρ_j (for $1 \leq j \leq m$) until the result lies in $B_\infty^m(q-1)$. It satisfies $\mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\underline{\boldsymbol{d}}}) = \mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}})$. The A-reduction of $D \subseteq \mathbb{N}^{m+1}$, denoted $\underline{D}$, consists in the A-reduction of the tuples in D.

Definition 3.9. Let $\boldsymbol{d} \in S^{m+1}(v)$ for some $v \geq 0$. The *P-reduction* of $\boldsymbol{d}$ is the tuple $\overline{\boldsymbol{d}} \in S^{m+1}(v)$ which is obtained by applying iteratively τ_{ij} (for $0 \leq i < j \leq m$) until the result is P-reduced. It satisfies $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\overline{\boldsymbol{d}}}) = \mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}})$. The P-reduction of $D \subseteq S^{m+1}(v)$, denoted $\overline{D}$, consists in the P-reduction of the tuples in D.

A- and P-reduction are defined in order to make the evaluation maps $\mathrm{ev}_{\mathbb{A}^m}$ and $\mathrm{ev}_{\mathbb{P}^m}$ injective over polynomial spaces of the form $\mathrm{Poly}(D)$, where D is A- or P-reduced. Next lemma details these properties.

Lemma 3.10. *Let $m \geq 1$ and $v \in \mathbb{N}$. The following properties hold:*

1. *If $D \subseteq \mathbb{N}^m$ is A-reduced, then the map $\mathrm{ev}_{\mathbb{A}^m}$ is injective over $\mathrm{Poly}(D)$.*
2. *If $D \subseteq S^{m+1}(v)$ is P-reduced, then the map $\mathrm{ev}_{\mathbb{P}^m}$ is injective over $\mathrm{Poly}(D)$.*
3. *For every $D \subseteq \mathbb{N}^m$, the A-reduction $\underline{D}$ of D is the unique A-reduced subset of $\mathbb{N}^m$ satisfying*
$$\mathrm{ev}_{\mathbb{A}^m}(\mathrm{Poly}(D)) = \mathrm{ev}_{\mathbb{A}^m}(\mathrm{Poly}(\underline{D}))\,.$$
4. *For every $D \subseteq S^{m+1}(v)$, the P-reduction $\overline{D}$ of D is the unique P-reduced subset of $S^{m+1}(v)$ satisfying*
$$\mathrm{ev}_{\mathbb{P}^m}(\mathrm{Poly}(D)) = \mathrm{ev}_{\mathbb{P}^m}(\mathrm{Poly}(\overline{D}))\,.$$

Proof.

1. By definition, if D is A-reduced, then D is a subset of $B_\infty^m(q-1)$. Therefore $\mathrm{Poly}(D) \cap \mathrm{Span}\{X_i^q - X_i, 1 \leq i \leq m\} = \{0\}$. The result follows from the fact that $\mathrm{ev}_{\mathbb{A}^m}$ vanishes only on $\mathrm{Span}\{X_i^q - X_i, 1 \leq i \leq m\}$.

2. Recall that Equation (3.2) states that

$$\ker(\mathrm{ev}_{\mathbb{P}^m}) = \mathrm{Span}\{X_i^q X_j - X_i X_j^q, 0 \le i < j \le m\} \cap \mathbb{F}_q[\boldsymbol{X}]_v^H .$$

We prove the statement by induction over m.

For $m = 1$ and $v \in \mathbb{N}$, let D be a P-reduced subset of $S^2(v)$. If $v \le q$, it is clear that $\ker(\mathrm{ev}_{\mathbb{P}^1}) \cap \mathrm{Poly}(D) = \{0\}$. So assume $v > q$ and let $f(X,Y) \in \mathrm{Poly}(D) \cap \ker(\mathrm{ev}_{\mathbb{P}^1})$. Since D is P-reduced, we can write

$$f(X,Y) \coloneqq f_v Y^v + \sum_{i=0}^{q-1} f_i X^{v-i} Y^i .$$

Then, we see that $f_v = f(0,1) = 0$, hence the polynomial $g(Y) \coloneqq f(1,Y) = \sum_{i=0}^{q-1} f_i Y^i$ lies in $\mathrm{Poly}(D')$, for some set $D' \subseteq B_\infty^1(q-1)$. Moreover $f \in \ker(\mathrm{ev}_{\mathbb{P}^1})$ implies that $g \in \ker(\mathrm{ev}_{\mathbb{A}^1})$. Hence, applying the first point of this lemma to subset D' shows that $g = 0$, and $f = 0$ follows.

For $m > 1$, let $v \in \mathbb{N}$. The proof works similarly. Let D be a P-reduced subset of $S^{m+1}(v)$, and let

$$f(\boldsymbol{X}) \coloneqq f_0(X_1, \ldots, X_m) + X_0 f_1(X_0, X_1, \ldots, X_m) \in \mathrm{Poly}(D) \cap \ker(\mathrm{ev}_{\mathbb{P}^m}) .$$

Since f_0 does not depend on X_0, we can see that $f_0 \in \ker(\mathrm{ev}_{\mathbb{P}^{m-1}})$ and $f_0 \in \mathrm{Poly}(D_0)$ where $D_0 \subset S^m(v)$. Furthermore, D_0 is P-reduced as a subset of D. Therefore, by induction $f_0 = 0$, and $f = X_0 f_1(X_0, X_1, \ldots, X_m)$ follows. Let us define $g(X_1, \ldots, X_m) \coloneqq f(1, X_1, \ldots, X_m)$; we see that $g \in \ker(\mathrm{ev}_{\mathbb{A}^m})$ and $g \in \mathrm{Poly}(D')$ where $D' \subseteq B_\infty^m(q-1)$ since D is P-reduced and every tuple in D' comes from a tuple $\boldsymbol{d} \in D$ such that $d_0 \neq 0$. Thanks again to the first point of the lemma, it follows that $g = 0$. Therefore, $f = \alpha X_0^v$ for some $\alpha \in \mathbb{F}_q$, which necessarily implies $f = 0$ (evaluate f at $(1 : 0 : \ldots : 0)$).
3. Since $\mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\underline{\boldsymbol{d}}}) = \mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}})$ for every $\boldsymbol{d} \in \mathbb{N}^m$, we have $\mathrm{ev}_{\mathbb{A}^m}(\mathrm{Poly}(\underline{D})) = \mathrm{ev}_{\mathbb{A}^m}(\mathrm{Poly}(D))$. Uniqueness comes from the injectivity of $\mathrm{ev}_{\mathbb{A}^m}$.
4. Same argument. □

Definition 3.11 (Degree set)**.** Let $\mathcal{C} = \mathrm{ev}_{\mathbb{A}^m}(\mathrm{Poly}(D))$ be an affine (resp. let $\mathcal{C} = \mathrm{ev}_{\mathbb{P}^m}(\mathrm{Poly}(D))$ be a projective) monomial code. Its *degree set* is the unique A-reduction (resp. P-reduction) of D, and is denoted $\mathrm{Deg}(\mathcal{C})$.

By definition, if $\mathcal{C}$ is monomial, then we have $\mathcal{C} = \mathrm{ev}(\mathrm{Poly}(\mathrm{Deg}(\mathcal{C})))$ where $\mathrm{ev} \in \{\mathrm{ev}_{\mathbb{A}^m}, \mathrm{ev}_{\mathbb{P}^m}\}$ depending on the context. Moreover, since the evaluation map is injective over $\mathrm{Poly}(\mathrm{Deg}(\mathcal{C}))$, it also holds that:

$$\dim(\mathcal{C}) = |\,\mathrm{Deg}(\mathcal{C})|\,.$$

Example 3.12. Reed-Solomon and Reed-Muller codes, as well as their projective analogues, are monomial codes. Table 3.1 presents their degree sets.

3.1.3 Isomorphisms and embeddings

The original definition of affine lifted codes makes use of restriction of polynomials to lines. We here give a formalism that embrace this notion for both affine and projective spaces.

Code	Degree set
Reed-Solomon code $\mathrm{RS}_q(k)$	$B_1^1(k) = \{0, 1, \ldots, k\}$
Reed-Muller code $\mathrm{RM}_q(m, v)$	$\underline{B_1^m(v)} = \{\underline{\boldsymbol{e}} \mid \boldsymbol{e} \in \mathbb{N}^m, \lvert\boldsymbol{e}\rvert \leq v\}$
projective Reed-Solomon code $\mathrm{PRS}_q(k)$	$S_1^2(k) = \{(k, 0), (k-1, 1), \ldots, (0, k)\}$
projective Reed-Muller code $\mathrm{PRM}_q(m, v)$	$\overline{S_1^{m+1}(v)} = \{\overline{\boldsymbol{d}} \mid \boldsymbol{d} \in \mathbb{N}^{m+1}, \lvert\boldsymbol{d}\rvert = v\}$

Table 3.1 – Degree sets of some classical monomial codes.

Isomorphisms. Each isomorphism $\psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$ induces a permutation of $\mathbb{P}^m$, but ψ does not necessarily preserve the standard representation of projective points. Still, for every $\boldsymbol{x} \in \mathbb{P}^m$ there exists $\lambda_{\psi,\boldsymbol{x}} \in \mathbb{F}_q^\times$ such that the standard representative of $\psi(\boldsymbol{x})$ is $\lambda_{\psi,\boldsymbol{x}}\psi(\boldsymbol{x})$. For every $f \in \mathbb{F}_q[\boldsymbol{X}]_v^H$, we then have:

$$\mathrm{ev}_{\psi(\boldsymbol{x})}(f) = f(\lambda_{\psi,\boldsymbol{x}}\psi(\boldsymbol{x})) = (\lambda_{\psi,\boldsymbol{x}})^v f(\psi(\boldsymbol{x})) = (\lambda_{\psi,\boldsymbol{x}})^v \, \mathrm{ev}_{\boldsymbol{x}}(f \circ \psi)\,,$$

and we see that $(\lambda_{\psi,\boldsymbol{x}})^v$ does not depend on f (only on its total degree). So let us denote by $\boldsymbol{w}^{(\psi,v)} := ((\lambda_{\psi,\boldsymbol{x}})^{-v} : \boldsymbol{x} \in \mathbb{P}^m) \in (\mathbb{F}_q^\times)^{\mathbb{P}^m}$. Then, we have:

$$\mathrm{ev}_{\mathbb{P}^m}(f \circ \psi) = \boldsymbol{w}^{(\psi,v)} \star \psi^*(\mathrm{ev}_{\mathbb{P}^m}(f))\,,$$

where we recall that ψ^* denotes the permutation of tuples indexed by $\mathbb{P}^m$ which is induced by ψ. Thus it is natural to introduce the group of isomorphisms $\mathrm{Proj}_v(\mathbb{P}^m) := \{(\boldsymbol{w}^{(\psi,v)}, \psi), \psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})\} \subseteq \mathbb{F}_q^{\mathbb{P}^m} \times \mathfrak{S}(\mathbb{P}^m)$. One notices that $\mathrm{Proj}_v(\mathbb{P}^m) \subseteq \mathrm{Aut}(\mathrm{PRM}_q(m, v))$, since for every $f \in \mathbb{F}_q[\boldsymbol{X}]_v^H$ and every $\psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$, the polynomial $f \circ \psi$ also lies in $\mathbb{F}_q[\boldsymbol{X}]_v^H$.

Embeddings. Let $\mathrm{Emb}_{\mathbb{P}}(m)$ be the set of full-rank (*i.e.* injective) linear maps from $\mathbb{F}_q^2$ to $\mathbb{F}_q^{m+1}$:

$$\mathrm{Emb}_{\mathbb{P}}(m) := \{L \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q^{m+1}),\ \mathrm{rank}(L) = 2\}\,.$$

Each $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ induces a *projective embedding* $\mathbb{P}^1 \to \mathbb{P}^m$ sending $(x : y) \mapsto L(x, y)$. One can easily check that this map is well-defined over projective spaces. Moreover, the set $\{L(\mathbb{P}^1), L \in \mathrm{Emb}_{\mathbb{P}}(m)\}$ describes all the projective lines of $\mathbb{P}^m$, although a projective line is obviously associated to many maps L in $\mathrm{Emb}_{\mathbb{P}}(m)$. Similarly, the set

$$\mathrm{Emb}_{\mathbb{A}}(m) := \{L' = (L_1, \ldots, L_m) \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q^m) \mid L = (L_0, \ldots, L_m) \in \mathrm{Emb}_{\mathbb{P}}(m)\}$$

defines *affine embeddings* $\mathbb{A}^1 \to \mathbb{A}^m$ by $t \mapsto L'(1, t)$. The set $\{L'(1, \mathbb{A}^1), L' \in \mathrm{Emb}_{\mathbb{A}}(m)\}$ is the set of affine lines of $\mathbb{A}^m$. Also notice that the embeddings $\mathbb{A}^1 \to \mathbb{A}^m$ we describe here are exactly those in $\mathrm{Aff}(\mathbb{F}_q, \mathbb{F}_q^m)$, but we prefer the formalism given by $\mathrm{Emb}_{\mathbb{A}}(m)$ due to its similarity with the projective setting.

Remark 3.13. For convenience and when the context is clear, we will improperly write $L'(t)$ instead of $L'(1, t)$. By using this notation, we want to emphasize that, for every $f \in \mathbb{F}_q[\boldsymbol{X}]$ and every $L' \in \mathrm{Emb}_{\mathbb{A}}(m)$, the map $t \mapsto f(L'(1, t))$ can be interpolated as a univariate polynomial denoted $f \circ L' \in \mathbb{F}_q[T]$.

Remark 3.14. For local correction purposes (see Section 3.3), it is important to notice the following points.

1. In the affine setting, for every $L' \in \mathrm{Emb}_{\mathbb{A}}(m)$ and $f \in \mathbb{F}_q[\boldsymbol{X}]$, the word $\mathrm{ev}_{\mathbb{A}^1}(f \circ L')$ is the subword of $\mathrm{ev}_{\mathbb{A}^m}(f)$ with coordinates $L'(\mathbb{A}^1) \subset \mathbb{A}^m$.
2. In the projective setting, $\mathrm{ev}_{\mathbb{P}^1}(f \circ L)$ is not necessary a subword of $\mathrm{ev}_{\mathbb{P}^m}(f)$, since nothing asserts that L preserves the standard representation of projective points. This issue is similar to the one concerning isomorphisms, and we solve it the same manner. Let $\boldsymbol{x} \in \mathbb{P}^1$ and $L \in \mathrm{Emb}_{\mathbb{P}}(m)$. We know there exists $\lambda_{L,\boldsymbol{x}} \in \mathbb{F}_q^\times$ such that the standard representative of $L(\boldsymbol{x})$ is $\lambda_{L,\boldsymbol{x}} L(\boldsymbol{x}) \in \mathbb{P}^m$. Then it holds:

$$\forall f \in \mathbb{F}_q[\boldsymbol{x}]_v^H,\ \mathrm{ev}_{L(\boldsymbol{x})}(f) = f(\lambda_{L,\boldsymbol{x}} L(\boldsymbol{x})) = (\lambda_{L,\boldsymbol{x}})^v (f \circ L)(\boldsymbol{x})\,.$$

   Therefore, as previously we define $\boldsymbol{w}^{(L,v)} \coloneqq ((\lambda_{L,\boldsymbol{x}})^{-v} : \boldsymbol{x} \in \mathbb{P}^1) \in (\mathbb{F}_q^\times)^{\mathbb{P}^1}$. Then $(\boldsymbol{w}^{(L,v)})^{-1} \star \mathrm{ev}_{\mathbb{P}^1}(f \circ L)$ is the subword of $\mathrm{ev}_{\mathbb{P}^m}(f)$ with coordinates $L(\mathbb{P}^1) \subset \mathbb{P}^m$.

Let us illustrate the second point of Remark 3.14 with a simple example.

Example 3.15. For clarity, let us fix an ordering of points in $\mathbb{P}^1(\mathbb{F}_3)$ and $\mathbb{P}^2(\mathbb{F}_3)$:

$$\begin{aligned}\mathbb{P}^1(\mathbb{F}_3) = {} & \big((1:1),(1:2),(1:0),(0:1)\big)\\ \mathbb{P}^2(\mathbb{F}_3) = {} & \big((1:1:1),(1:1:2),(1:1:0),(1:2:1),(1:2:2),(1:2:0),\\ & \quad (1:0:1),(1:0:2),(1:0:0),(0:1:1),(0:1:2),(0:1:0),(0:0:1)\big)\end{aligned}$$

Let $f = X_1 \in \mathbb{F}_3[X_0, X_1, X_2]_1^H$ and $L = (L_0, L_1, L_2) \in \mathrm{Emb}_{\mathbb{P}}(2)$ defined by

$$\begin{aligned}L_0(S,T) &= S + T,\\ L_1(S,T) &= T,\\ L_2(S,T) &= S\,.\end{aligned}$$

Denote by $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^2}(f) = (1,1,1,2,2,2,0,0,0,1,1,1,0) \in \mathbb{F}_3^{13}$. On the one hand we have

$$\begin{aligned}L(\mathbb{P}^1) &= ((2:1:1),(0:2:1),(1:0:1),(1:1:0))\\ &= ((1:2:2),(0:1:2),(1:0:1),(1:1:0))\,,\end{aligned} \tag{3.4}$$

hence $\boldsymbol{c}_{|L(\mathbb{P}^1)} = (c_5, c_{11}, c_7, c_3) = (2,1,0,1)$. On the other hand $(f \circ L)(S,T) = T \in \mathbb{F}_3[S,T]_1^H$, and we get $\mathrm{ev}_{\mathbb{P}^1}(f \circ L) = (1,2,0,1)$. So clearly $\boldsymbol{c}_{|L(\mathbb{P}^1)} \neq \mathrm{ev}_{\mathbb{P}^1}(f \circ L)$.

Nevertheless, $\boldsymbol{w}^{(L,1)}$ can be obtained through the homogenisation made in (3.4):

$$\boldsymbol{w}^{(L,1)} = (2,2,1,1)\,.$$

Therefore it gives:

$$\mathrm{ev}_{\mathbb{P}^1}(f \circ L) = (2,1,0,1) = \boldsymbol{w}^{(L,1)} \star \boldsymbol{c}_{|L(\mathbb{P}^1)}\,.$$

3.2 Definition and first properties of projective lifted codes

Before introducing the construction of *projective* lifted codes, we recall the definition of *affine* lifted codes given by Guo, Kopparty and Sudan [GKS13]. We restrict our study to the lifting of projective Reed-Solomon codes, but we are convinced that our construction can be extrapolated to the lifting of projective Reed-Muller codes. Notice that our formalism is slightly different from the paper of [GKS13], since their notion of restriction $f_{|L}$ of a polynomial f along a line L is somewhat ambiguous.

3.2.1 Affine lifted codes

We recall that, for $f \in \mathbb{F}_q[\boldsymbol{X}]$ and $L \in \mathrm{Emb}_{\mathbb{A}}(m)$, the notation $f \circ L$ represents the univariate polynomial $f(L(1,T))$. The *affine lifting* of order m of the Reed-Solomon code $\mathrm{RS}_q(k)$ is

$$\mathrm{Lift}(\mathrm{RS}_q(k),m) \coloneqq \{\mathrm{ev}_{\mathbb{A}^m}(f) \mid f \in \mathbb{F}_q[\boldsymbol{X}] \text{ such that } \forall L \in \mathrm{Emb}_{\mathbb{A}}(m), \mathrm{ev}_{\mathbb{A}^1}(f \circ L) \in \mathrm{RS}_q(k)\}\,.$$

In this chapter, $\mathrm{Lift}(\mathrm{RS}_q(k),m)$ will be referred to as an *affine lifted code* for short, since we will only deal with lifted Reed-Solomon codes.

We first introduce a few additional notation. For a non-negative integer a, we denote by $\sum_i a^{(i)} p^i$ its p-adic decomposition. One can now define a partial order $\leq_p$ over $\mathbb{N}$ by:

$$a \leq_p b \iff a^{(i)} \leq b^{(i)},\ \forall i\,. \tag{3.5}$$

Relation $\leq_p$ can be naturally extended to m-tuples by $\boldsymbol{a} \leq_p \boldsymbol{b} \iff \forall j,\ a_j \leq_p b_j$. We also extend usual binomial coefficients to m-tuple entries by $\binom{\boldsymbol{a}}{\boldsymbol{b}} \coloneqq \prod_{i=1}^m \binom{a_i}{b_i}$.

Guo *et al.* proved that every affine lifted code $\mathrm{Lift}(\mathrm{RS}_q(k),m)$ is a monomial code that satisfies

$$\mathrm{Lift}(\mathrm{RS}_q(k),m) = \mathrm{Span}_{\mathbb{F}_q}\left\{\mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \;\middle|\; \boldsymbol{d} \in B_\infty^m(q-1), \forall \boldsymbol{e} \leq_p \boldsymbol{d}, \underline{|\boldsymbol{e}|} \leq k\right\}, \tag{3.6}$$

where $p = \mathrm{char}(\mathbb{F}_q)$ and $|\boldsymbol{e}| = \sum_i e_i$. Furthermore, a careful observation of their degree sets shows that $\mathrm{Lift}(\mathrm{RS}_q(k),m)$ fits between two projective Reed-Muller codes:

$$\mathrm{RM}_q(m,k) \subseteq \mathrm{Lift}(\mathrm{RS}_q(k),m) \subseteq \mathrm{RM}_q(m, k+(m-1)(q-1))\,. \tag{3.7}$$

We also recall that the main interest of affine lifted codes appears for $k \geq q - \frac{q}{p}$, where the first inclusion is proper [KR06]. More details can be found in a previous chapter (Subsection 2.2.3), and methods for the precise computation of degree sets will be given in Section 3.6.

3.2.2 Projective lifted codes

We are now ready to define the projective analogues of lifted Reed-Solomon codes. One wants to define them as projective evaluation codes corresponding to a subspace of polynomials which satisfy a collection of local constraints. Recall we build evaluation codes over projective spaces by evaluating *homogeneous* polynomials of fixed degree v. It raises the problem of determining a meaningful expression for v. By analogy with the affine setting, Equation (3.7) suggests to set $v = v_{m,k} \coloneqq k + (m-1)(q-1)$.

Definition 3.16 (projective lifted code)**.** Let $0 \leq k \leq q$, $m \geq 1$ and $v = k + (m-1)(q-1)$. The projective lifting of order m of the projective Reed-Solomon code $\mathrm{PRS}_q(k)$ is

$$\mathrm{Lift}(\mathrm{PRS}_q(k),m) \coloneqq \{\mathrm{ev}_{\mathbb{P}^m}(f) \mid f \in \mathbb{F}_q[\boldsymbol{X}]_v^H, \forall L \in \mathrm{Emb}_{\mathbb{P}}(m), \mathrm{ev}_{\mathbb{P}^1}(f \circ L) \in \mathrm{PRS}_q(k)\}\,.$$

Such a code will be called a *projective lifted code* for short, and its length equals $\theta_{m,q} = |\mathbb{P}^m| = (q^{m+1}-1)/(q-1)$.

Let us get rid of the case $k = q$ as of now. The associated projective Reed-Solomon code $\mathrm{PRS}_q(q) = \mathbb{F}_q^{\mathbb{P}^1}$ is degenerate. Hence we get $\mathrm{Lift}(\mathrm{PRS}_q(q),m) = \mathbb{F}_q^{\mathbb{P}^m}$, which is also degenerate. Also notice that, for $k \leq q-1$, it clearly holds that $\mathrm{Lift}(\mathrm{PRS}_q(k),1) = \mathrm{PRS}_q(k)$.

So, from now on, if not stated otherwise we assume $m \geq 2$ and $0 \leq k \leq q-1$.

3.2.3 Monomiality of projective lifted codes

Similarly to the affine setting, a main issue remains to give a basis of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$. In this subsection, we prove that projective lifted codes are monomial, and then we compute their degree set. Precisely, we state the following new result, which proves the monomiality of a broader family of codes.

Theorem 3.17. *Let $\mathcal{C} = \mathrm{ev}_{\mathbb{P}^m}(\mathcal{F})$ be a projective evaluation code, where $\mathcal{F}$ is a subspace of $\mathbb{F}_q[\boldsymbol{X}]_v^H$ for some $m, v \geq 1$. Assume that $\mathrm{Proj}_v(\mathbb{P}^m) \subseteq \mathrm{Aut}(\mathcal{C})$. Then $\mathcal{C}$ is monomial.*

Before diving straight into the proof, we first observe that $\mathrm{Proj}_v(\mathbb{P}^m)$ contains elements $(\boldsymbol{w}^{(\psi,v)}, \psi)$ where $\psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$ can be:

- a diagonal isomorphism $\mathrm{diag}_{\boldsymbol{a}}$ for any $\boldsymbol{a} \in (\mathbb{F}_q^\times)^{m+1}$, where

$$\begin{array}{rccc} \mathrm{diag}_{\boldsymbol{a}} : & \mathbb{P}^m & \to & \mathbb{P}^m \\ & (x_0 : \ldots : x_m) & \mapsto & (a_0 x_0 : \ldots : a_m x_m) \end{array}$$

- an elementary transposition $s_{i,j}$ for any $0 \leq i, j \leq m$, $i \neq j$, where

$$\begin{array}{rccc} s_{i,j} : & \mathbb{P}^m & \to & \mathbb{P}^m \\ & (x_0 : \ldots : x_i : \ldots : x_j : \ldots : x_m) & \mapsto & (x_0 : \ldots : x_j : \ldots : x_i : \ldots : x_m) \end{array}$$

- an elementary transvection $t_{i,j,\beta}$ for any $0 \leq i, j \leq m$, $i \neq j$, and $\beta \in \mathbb{F}_q$, where

$$\begin{array}{rccc} t_{i,j,\beta} : & \mathbb{P}^m & \to & \mathbb{P}^m \\ & (x_0 : \ldots : x_i : \ldots : x_m) & \mapsto & (x_0 : \ldots : x_i + \beta x_j : \ldots : x_m) \end{array}$$

We first need to prove a technical lemma.

Lemma 3.18. *The following equality over bivariate polynomials holds:*

$$\sum_{\beta \in \mathbb{F}_q} (\beta X + Y)^{q-1} = -X^{q-1}\,.$$

Proof. Let $F(X, Y) = \sum_{\beta \in \mathbb{F}_q} (\beta X + Y)^{q-1} + X^{q-1}$. Since $\mathrm{ev}_{\mathbb{A}^2}$ is injective over polynomials of partial degree bounded by $q-1$, it is sufficient to prove that $\mathrm{ev}_{\mathbb{A}^2}(F) = \mathbf{0}$.

Let $(x, y) \in \mathbb{F}_q^2$. If $x = 0$, then $F(x, y) = F(0, y) = \sum_{\beta \in \mathbb{F}_q} y^{q-1} = 0$. Otherwise, $\beta \mapsto \beta x + y$ is a bijection over $\mathbb{F}_q$, hence we have:

$$F(x, y) = \sum_{\beta \in \mathbb{F}_q} (\beta x + y)^{q-1} + x^{q-1} = \sum_{\gamma \in \mathbb{F}_q} \gamma^{q-1} + 1 = q - 1 + 1 = 0\,.$$

□

Proof of Theorem 3.17. Let $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathcal{C}$, where $f = \sum_{\boldsymbol{d}} f_{\boldsymbol{d}} \boldsymbol{X}^{\boldsymbol{d}}$, and denote by $D = \mathrm{Deg}(f) = \{\boldsymbol{d}, f_{\boldsymbol{d}} \neq 0\}$. Our goal is to prove that every $\boldsymbol{d} \in D$ satisfies $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathcal{C}$. The proof will consist in three main parts:

(i) we show that $\mathrm{ev}_{\mathbb{P}^m}(Q_{\boldsymbol{d}}(\boldsymbol{X})) \in \mathcal{C}$, where $Q_{\boldsymbol{d}}(\boldsymbol{X})$ is a polynomial such that $\boldsymbol{d} \in \mathrm{Deg}(Q_{\boldsymbol{d}})$, and such that $\mathrm{Deg}(Q_{\boldsymbol{d}})$ is much smaller than $\mathrm{Deg}(\mathcal{C})$;

(ii) we analyse and rewrite $\mathrm{Deg}(Q_{\boldsymbol{d}})$, which then allows us to write the polynomial $Q_{\boldsymbol{d}}(\boldsymbol{X})$ as $X_1^{d_1} \ldots X_a^{d_a} R(X_0, X_{a+1}, \ldots, X_m)$ for some $(m-a+1)$-variate homogeneous polynomial R;

(iii) we prove that, if there exists an $(m-a+1)$-variate polynomial R satisfying some prescribed properties and such that $\mathrm{ev}_{\mathbb{P}^m}(X_1^{d_1}\dots X_a^{d_a}R(X_0,X_{a+1},\dots,X_m)) \in \mathcal{C}$, then we can compute an $(m-a)$-variate polynomial R' satisfying the same properties, and such that the vector $\mathrm{ev}_{\mathbb{P}^m}(X_1^{d_1}\dots X_{a+1}^{d_{a+1}}R'(X_0,X_{a+2},\dots,X_m)) \in \mathcal{C}$.

Reasoning inductively on the last part will conclude the proof.

Proof of part (i). Let $\boldsymbol{d} \in D$, and define

$$Q_{\boldsymbol{d}}(\boldsymbol{X}) \coloneqq (-1)^{m+1} \sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^{m+1}} \Big(\prod_{i=0}^m a_i^{-d_i}\Big)(f\circ \mathrm{diag}_{\boldsymbol{a}})(\boldsymbol{X})\,.$$

Since $\mathcal{C}$ is linear and $\mathrm{diag}_{\boldsymbol{a}} \in \mathrm{Aut}(\mathcal{C})$ for every $\boldsymbol{a} \in (\mathbb{F}_q^\times)^{m+1}$, we know that $\mathrm{ev}_{\mathbb{P}^m}(Q_{\boldsymbol{d}}(\boldsymbol{X}))$ lies in $\mathcal{C}$. Let us now rewrite $Q_{\boldsymbol{d}}(\boldsymbol{X})$:

$$\begin{aligned} Q_{\boldsymbol{d}}(\boldsymbol{X}) &= (-1)^{m+1} \sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^{m+1}} \Big(\prod_{i=0}^m a_i^{-d_i}\Big) \sum_{\boldsymbol{j}} f_{\boldsymbol{j}}\, a_0^{j_0}\dots a_m^{j_m}\, \boldsymbol{X}^{\boldsymbol{j}} \\ &= (-1)^{m+1} \sum_{\boldsymbol{j}} f_{\boldsymbol{j}} \sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^{m+1}} \Big(\prod_{i=0}^m a_i^{j_i-d_i}\Big)\boldsymbol{X}^{\boldsymbol{j}} \end{aligned}$$

One can notice that developping the product $\prod_{i=0}^m(\sum_{a_i\in\mathbb{F}_q^\times} a_i^{j_i-d_i})$ gives $\sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^{m+1}}\prod_{i=0}^m a_i^{j_i-d_i}$. Hence we get:

$$\begin{aligned} Q_{\boldsymbol{d}}(\boldsymbol{X}) &= (-1)^{m+1}\sum_{\boldsymbol{j}} f_{\boldsymbol{j}} \prod_{i=0}^m \Big(\underbrace{\sum_{a_i\in\mathbb{F}_q^\times} a_i^{j_i-d_i}}_{=0 \text{ if } j_i\not\equiv d_i \mod (q-1),\ -1 \text{ otherwise}} \Big)\boldsymbol{X}^{\boldsymbol{j}} \\ &= \sum_{\boldsymbol{j}\in E_{\boldsymbol{d}}} f_{\boldsymbol{j}}\boldsymbol{X}^{\boldsymbol{j}}\,, \end{aligned}$$

where $E_{\boldsymbol{d}} = \{\boldsymbol{j}\in D, \boldsymbol{j}\equiv\boldsymbol{d} \mod (q-1)\} \subseteq D$.

Proof of part (ii). The code $\mathcal{C}$ is invariant under the action of elementary switches of coordinates. Therefore one can assume without loss of generality that, if they exist, the only coordinates d_i of $\boldsymbol{d}$ satisfying $q-1 \mid d_i$ all lie at the end of the tuple $\boldsymbol{d}$. Furthermore, by P-reduction and by definition of $E_{\boldsymbol{d}}$, we can assume that, if $d_i \notin \{0, q-1\}$ then $d_i = j_i$, except maybe for the leftmost non-zero coordinate of $\boldsymbol{j}$ and $\boldsymbol{d}$. To sum up, without loss of generality there exists an index $a \in [1,m]$ such that every $\boldsymbol{j} \in E_{\boldsymbol{d}}$ satisfies the following three properties

$$\begin{cases} \forall 1 \le i \le a, \text{ we have } j_i = d_i < q-1 \\ \forall a < i \le m, \text{ we have } j_i \in \{0,q-1\} \text{ and } d_i \in \{0,q-1\} \\ j_0 = v - \sum_{i=1}^m j_i\,. \end{cases}$$

Therefore, $Q_{\boldsymbol{d}}(\boldsymbol{X})$ can be written as $X_1^{d_1}\dots X_a^{d_a}R(X_0,X_{a+1},\dots,X_m)$, where R is an homogeneous polynomial of degree $v-\sum_{i=1}^a d_i$, whose monomials have partial degree either 0 or $q-1$, for every coordinate X_i, $i > a$.

Proof of part (iii). Recall that we want to prove that $\mathrm{ev}_{\mathbb{P}^m}(X_0^{d_0}X_1^{d_1}\dots X_m^{d_m}) \in \mathcal{C}$, and we know that $\mathrm{ev}_{\mathbb{P}^m}(Q_{\boldsymbol{d}}(\boldsymbol{X})) \in \mathcal{C}$. Our strategy is to proceed inductively, from $i = a$ to m, by proving there exists an $(m-i+1)$-variate polynomial R_i such that $\boldsymbol{d} \in \mathrm{Deg}(X_1^{d_1}\dots X_i^{d_i}R_i(X_0,X_{i+1},\dots,X_m))$ and $\mathrm{ev}_{\mathbb{P}^m}(X_1^{d_1}\dots X_i^{d_i}R_i(X_0,X_{i+1},\dots,X_m)) \in \mathcal{C}$. Notice that step $i = a$ has been proved in part (ii), and that step $i = m$ concludes the proof. Hence there remains to prove the induction step.

Write $R_i = R'_i + X_{i+1}^{q-1} R''_i$, where polynomials R'_i and R''_i do not depend on X_{i+1}. Let $S_i = X_1^{d_1} \dots X_i^{d_i} R_i(X_0, X_{i+1}, \dots, X_m)$, and assume that $\mathrm{ev}_{\mathbb{P}^m}(S_i) \in \mathcal{C}$ and $\boldsymbol{d} \in \mathrm{Deg}(S_i)$. If $R''_i = 0$, then the induction step is proved. Otherwise:

- *1st case:* $d_{i+1} = 0$. Since $\sum_{\beta \in \mathbb{F}_q} (X_{i+1} + \beta X_0)^{q-1} = -X_0^{q-1}$ (see Lemma 3.18), we get

$$\begin{aligned}
&\sum_{\beta \in \mathbb{F}_q} S_i(X_0, \dots, X_{i+1} + \beta X_0, \dots, X_m) \\
&\quad = X_1^{d_1} \dots X_i^{d_i} \sum_{\beta \in \mathbb{F}_q} \Big(R'_i(X_0, X_{i+2}, \dots, X_m) + (X_{i+1} + \beta X_0)^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m) \Big) \\
&\quad = \Big(\sum_{\beta \in \mathbb{F}_q} X_1^{d_1} \dots X_i^{d_i} R'_i(X_0, X_{i+2}, \dots, X_m) \Big) - X_1^{d_1} \dots X_i^{d_i} X_0^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m) \\
&\quad = -X_1^{d_1} \dots X_i^{d_i} X_0^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m) \,.
\end{aligned}$$

By linearity and invariance of $\mathcal{C}$ under transvections, $\mathrm{ev}_{\mathbb{P}^m}(S_i) \in \mathcal{C}$ ensures that the word $\mathrm{ev}_{\mathbb{P}^m}(X_1^{d_1} \dots X_i^{d_i} X_0^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m)) \in \mathcal{C}$. We conclude by defining $R_{i+1} = -X_0^{q-1} R''_i$.

- *2nd case:* $d_{i+1} = q - 1$. Since $\sum_{\beta \in \mathbb{F}_q} (\beta X_{i+1} + X_0)^{q-1} = -X_{i+1}^{q-1}$, we get

$$\begin{aligned}
&\sum_{\beta \in \mathbb{F}_q} S_i(X_0, \dots, X_{i+1} + \beta X_0, \dots, X_m) \\
&\quad = X_1^{d_1} \dots X_i^{d_i} \sum_{\beta \in \mathbb{F}_q} \Big(R'_i(X_0, X_{i+2}, \dots, X_m) + (\beta X_{i+1} + X_0)^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m) \Big) \\
&\quad = -X_1^{d_1} \dots X_i^{d_i} X_{i+1}^{q-1} R''_i(X_0, X_{i+2}, \dots, X_m) \,.
\end{aligned}$$

Similarly to the first case, we can conclude by defining $R_{i+1} = -R''_i$. □

The converse of Theorem 3.17 is obviously false: for instance, if $\mathcal{F} = \mathrm{Span}_{\mathbb{F}_q}\{XY, Y^2\}$, then $\mathrm{ev}_{\mathbb{P}^2}(\mathcal{F})$ is monomial, but not invariant under $\mathrm{Proj}_2(\mathbb{P}^2)$.

A good point is that, similarly to PRM codes, projective lifted codes can be proved invariant under $\mathrm{Proj}_v(\mathbb{P}^m)$.

Lemma 3.19. *Let $k \leq q-1$, $m \geq 1$ and $\mathcal{C} = \mathrm{Lift}(\mathrm{PRS}_q(k), m)$. Then $\mathrm{Proj}_v(\mathbb{P}^m) \subseteq \mathrm{Aut}(\mathcal{C})$. Said differently,*

$$\forall \boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathcal{C},\ \forall \psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1}),\ \mathrm{ev}_{\mathbb{P}^m}(f \circ \psi) \in \mathcal{C} \,.$$

Proof. It is sufficient to notice that, for every $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ and every $\psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$, the map $\psi \circ L$ also lies in $\mathrm{Emb}_{\mathbb{P}}(m)$. □

As a corollary, Theorem 3.17 implies that

Corollary 3.20. *Every projective lifted code is monomial.*

3.2.4 Degree sets of lifted codes

A natural question is now to determine the degree set of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$. Let us first recall that affine lifted codes have the following degree sets (see Equation (3.6)):

$$\mathrm{ADeg}_q(m,k) := \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m)) = \{\boldsymbol{d} \in B_\infty^m(q-1) \mid \forall \boldsymbol{e} \leq_p \boldsymbol{d}, \underline{|\boldsymbol{e}|} \leq k\} \,.$$

Similarly, we define $\mathrm{PDeg}_q(m,k) := \mathrm{Deg}(\mathrm{Lift}(\mathrm{PRS}_q(k),m))$.

In this subsection, we state a few links between degree sets of affine and projective lifted codes. Propositions 3.21 and 3.22 show that every $\boldsymbol{d} \in \mathrm{PDeg}_q(m,k)$ can be bijectively sent either to $\mathrm{ADeg}_q(m,k-1)$ or to $\mathrm{PDeg}_q(m-1,k)$, according to the value of d_0. Thus, in Theorem 3.23 we derive a recursive formula on the degree sets of affine and projective lifted codes, which translates into another recursive formula on the dimension of these codes (Corollary 3.24).

Proposition 3.21. *Let $v = k+(m-1)(q-1)$ for $1 \le k \le q-1$ and $m \ge 2$. Let also $\boldsymbol{d} = (d_0, \boldsymbol{d}^*) \in S^{m+1}(v)$ such that $d_0 \neq 0$. Then:*

$$\mathrm{ev}_{\mathbb{P}^m}(X_0^{d_0}\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{PRS}_q(k),m) \iff \mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{RS}_q(k-1),m)\,,$$

or, equivalently,

$$\boldsymbol{d} = (d_0, \mathbf{d}^*) \in \mathrm{PDeg}_q(m,k) \iff \boldsymbol{d}^* \in \mathrm{ADeg}_q(m,k-1)\,.$$

Proof. ($\Rightarrow$). Let $\mathrm{ev}_{\mathbb{P}^m}(X_0^{d_0}\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{PRS}_q(k),m)$ with $d_0 \neq 0$, and $L' \in \mathrm{Emb}_{\mathbb{A}}(m)$. We need to prove that $\mathrm{ev}_{\mathbb{A}^1}(\boldsymbol{X}^{\boldsymbol{d}^*} \circ L') \in \mathrm{RS}_q(k-1)$. Let us define $L = (L_0,\ldots,L_m) \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q^{m+1})$ as follows:

- the last m coordinate-maps $(L_1,\ldots,L_m)$ equal L',
- the first coordinate-map L_0 is chosen to be either $L_0(S,T) = S$ or $L_0(S,T) = T$, in order to have $\mathrm{rank}(L) = 2$.

Now notice that without loss of generality we can assume that $L_0(S,T) = S$. Two points could be clarified here. First, if the linear map $(S, L'(S,T))$ has rank 1, by definition of $\mathrm{Emb}_{\mathbb{A}}(m)$ the linear map $(T, L'(S,T))$ has rank 2. Second, the particular choice $L_0(S,T) = S$ can be done since $\mathrm{PRS}_q(k)$ is invariant under $\mathrm{Proj}(\mathbb{P}^1)$.

Finally we get

$$\mathrm{ev}_{\mathbb{P}^1}(X_0^{d_0}\boldsymbol{X}^{\boldsymbol{d}^*} \circ L) = \mathrm{ev}_{\mathbb{P}^1}(X_0\boldsymbol{X}^{\boldsymbol{d}^*} \circ L) = \mathrm{ev}_{\mathbb{P}^1}(S \cdot (\boldsymbol{X}^{\boldsymbol{d}^*} \circ L')(S,T)) \in \mathrm{PRS}_q(k) \tag{3.8}$$

since $X_0^{d_0}\boldsymbol{X}^{\boldsymbol{d}^*}$ and $X_0\boldsymbol{X}^{\boldsymbol{d}^*}$ evaluate identically, and by definition of projective lifted codes. Moreover, we know that any homogeneous polynomial $P(S,T)$ satisfies

$$\mathrm{ev}_{\mathbb{P}^1}(S \cdot P(S,T)) \in \mathrm{PRS}_q(k) \iff \mathrm{ev}_{\mathbb{A}^1}(P(1,T)) \in \mathrm{RS}_q(k-1)\,. \tag{3.9}$$

Applying this to $P(S,T) = (\boldsymbol{X}^{\boldsymbol{d}^*} \circ L')(S,T)$, we get our result.

($\Leftarrow$). Let $\mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{RS}_q(k-1),m)$ and $L \in \mathrm{Emb}_{\mathbb{P}}(m)$. Let also $d_0 \neq 0$ such that $(d_0, \boldsymbol{d}^*) \in S^{m+1}(v)$. We need to prove that $\mathrm{ev}_{\mathbb{P}^1}(X^{d_0}\boldsymbol{X}^{\boldsymbol{d}^*} \circ L) \in \mathrm{PRS}_q(k)$. If $L_0 = 0$, then the result holds since $\mathbf{0} \in \mathrm{PRS}_q(k)$. Otherwise, it is worthwhile to notice that, since $\mathrm{PRS}_q(k)$ is invariant under $\mathrm{Proj}(\mathbb{P}^1)$, we can assume without loss of generality that $L_0(S,T) = S$. Define $L' = (L_1,\ldots,L_m)$, which lies in $\mathrm{Emb}_{\mathbb{A}}(m)$ by definition. Therefore $\mathrm{ev}_{\mathbb{A}^1}(\boldsymbol{X}^{\boldsymbol{d}^*} \circ L') \in \mathrm{RS}_q(k-1)$, and using (3.8) and (3.9), we prove the expected result. □

Similarly, we have:

Proposition 3.22. *Let $v = k+(m-1)(q-1)$ for $1 \le k \le q-1$ and $m \ge 2$. Let also $\boldsymbol{d} = (d_0, \boldsymbol{d}^*) \in S^{m+1}(v)$, and assume that $d_0 = 0$. Then:*

$$\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathrm{Lift}(\mathrm{PRS}_q(k),m) \iff \mathrm{ev}_{\mathbb{P}^{m-1}}(\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{PRS}_q(k),m-1)\,,$$

or equivalently,

$$\boldsymbol{d} = (0, \boldsymbol{d}^*) \in \mathrm{PDeg}_q(m,k) \iff \overline{\boldsymbol{d}^*} \in \mathrm{PDeg}_q(m-1,k)\,.$$

Proof. ($\Rightarrow$). Let $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$ where $\boldsymbol{d} = (d_0, \boldsymbol{d}^*)$ and $d_0 = 0$. Let also $L' \in \mathrm{Emb}_{\mathbb{P}}(m-1)$. We need to prove that $\mathrm{ev}_{\mathbb{P}^1}(\boldsymbol{X}^{\boldsymbol{d}^*} \circ L') \in \mathrm{PRS}_q(k)$. First, notice that any linear map $L_0 \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q)$ extends L' to $L = (L_0, L') \in \mathrm{Emb}_{\mathbb{P}}(m)$. Therefore,

$$\mathrm{ev}_{\mathbb{P}^1}(\boldsymbol{X}^{\boldsymbol{d}^*} \circ L') = \mathrm{ev}_{\mathbb{P}^1}(X_0^0 \boldsymbol{X}^{\boldsymbol{d}^*} \circ L) = \mathrm{ev}_{\mathbb{P}^1}(\boldsymbol{X}^{\boldsymbol{d}} \circ L) \tag{3.10}$$

lies in $\mathrm{PRS}_q(k)$ since $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$.

($\Leftarrow$). Let $\boldsymbol{d} = (d_0, \boldsymbol{d}^*) \in S^{m+1}(v)$ with $d_0 = 0$, and assume that $\mathrm{ev}_{\mathbb{P}^{m-1}}(\boldsymbol{X}^{\boldsymbol{d}^*}) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$. Let also $L \in \mathrm{Emb}_{\mathbb{P}}(m)$. We need to prove that $\mathrm{ev}_{\mathbb{P}^1}(\boldsymbol{X}^{\boldsymbol{d}} \circ L) \in \mathrm{PRS}_q(k)$. Write $L = (L_0, L')$. If $L' \in \mathrm{Emb}_{\mathbb{P}}(m-1)$, then the result follows using (3.10). The case $\mathrm{rank}\, L' = 1$ is a bit trickier. Since $\mathrm{Proj}_k(\mathbb{P}^1)$ lets the code $\mathrm{PRS}_q(k)$ invariant, we can assume without loss of generality that $L'(S,T)$ can be written $(\lambda_1 S, \ldots, \lambda_m S)$ with some non-zero tuple $(\lambda_1, \ldots, \lambda_m) \in \mathbb{F}_q^m$. Therefore, $(\boldsymbol{X}^{\boldsymbol{d}} \circ L)(S,T) = \alpha S^{|\boldsymbol{d}^*|}$ with $\alpha \in \mathbb{F}_q$, and $\mathrm{ev}_{\mathbb{P}^1}(\boldsymbol{X}^{\boldsymbol{d}} \circ L) = \alpha\, \mathrm{ev}_{\mathbb{P}^1}(S^{|\boldsymbol{d}^*|}) \in \mathrm{PRS}_q(k)$ since the P-reduction of $(|\boldsymbol{d}^*|, 0)$ is $(k, 0)$ which lies in $\mathrm{Deg}(\mathrm{PRS}_q(k))$. □

In Propositions 3.21 and 3.22 we avoided the case $k = 0$. Let us treat it here. $\mathrm{PRS}_q(0)$ is the repetition code of length $(q+1)$ over $\mathbb{F}_q$. Therefore, the definition of projective lifted codes impose that their codewords correspond to polynomials that evaluate as constants on every projective line of the space. Since projective lines intersects, the constant must be the same for every line. It means that $\mathrm{Lift}(\mathrm{PRS}_q(0), m)$ is the repetition code of length $|\mathbb{P}^m|$. Now, notice that this extends Propositions 3.21 and 3.22, by using the convention $\mathrm{RS}_q(-1) := \{\mathbf{0}\}$.

This leads us to the following theorem.

Theorem 3.23. *For every $m \geq 2$ and $0 \leq k \leq q-1$, there is a bijection between* $\mathrm{PDeg}(m,k)$ *and* $\mathrm{PDeg}(m-1,k) \cup \mathrm{ADeg}(m,k-1)$.

Proof. According to Propositions 3.21 and 3.22, this bijection is given by:

$$\boldsymbol{d} = (d_0, \boldsymbol{d}^*) \mapsto \begin{cases} \boldsymbol{d}^* \in \mathrm{ADeg}(m,k-1) & \text{if } d_0 \neq 0 \\ \overline{\boldsymbol{d}^*} \in \mathrm{PDeg}(m-1,k) & \text{otherwise.} \end{cases}$$

□

A recursive formula on the dimension of lifted codes can be easily derived from the last theorem.

Corollary 3.24. *Let $m \geq 2$ and $0 \leq k \leq q-1$. Then,*

$$\dim(\mathrm{Lift}(\mathrm{PRS}_q(k), m)) = \dim(\mathrm{Lift}(\mathrm{PRS}_q(k), m-1) + \dim(\mathrm{Lift}(\mathrm{RS}_q(k-1), m))\,.$$

Since $\mathrm{Lift}(\mathrm{PRS}_q(k), 1) = \mathrm{PRS}_q(k)$ and $\mathrm{Lift}(\mathrm{PRS}_q(k-1), 1) = \mathrm{RS}_q(k-1)$, we also get:

Corollary 3.25. *Let $m \geq 1$ and $0 \leq k \leq q-1$. Then,*

$$\dim(\mathrm{Lift}(\mathrm{PRS}_q(k), m)) = \sum_{j=1}^{m} \dim(\mathrm{Lift}(\mathrm{RS}_q(k-1), j)) + 1\,.$$

It could be interesting to make $\mathrm{PDeg}_q(m,k)$ explicit, as it was done for affine lifted codes. To achieve this, we use iteratively the bijective map given in Theorem 3.23 and the characterisation of $\mathrm{ADeg}_q(j, k-1)$, $1 \leq j \leq m$, given in Equation (3.6). For $\boldsymbol{d} = (d_0, \ldots, d_m) \in S^{m+1}(v)$, define $i_{\min}(\boldsymbol{d})$ the minimum i such that $d_i \neq 0$, and $\eta(\boldsymbol{d}) = (d_{i_{\min}(\boldsymbol{d})+1}, \ldots, d_m) \in S^{m-i_{\min}(\boldsymbol{d})}(v)$. We then obtain:

Corollary 3.26. *Let $m \geq 2$ and $0 \leq k \leq q-1$. Denote by $v = k + (m-1)(q-1)$. Then,*

$$\mathrm{PDeg}_q(m,k) = \left\{ \overline{\boldsymbol{d}}, \boldsymbol{d} \in S^{m+1}(v) \text{ such that } \forall \boldsymbol{e} \leq_p \eta(\overline{\boldsymbol{d}}), \underline{|\boldsymbol{e}|} \leq k-1 \right\}.$$

Notice that there exists a bijective transformation sending any degree $\boldsymbol{d} \in S^{m+1}(k)$ to a degree $\boldsymbol{d}'$ lying in $\mathrm{Deg}(\mathrm{Lift}(\mathrm{PRS}_q(k), m))$, by adding $(q-1)(m-1)$ to its leftmost non-zero coordinate. Moreover, it holds that $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}}) = \mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}'})$, even if formally, the underlying evaluation maps are different since they take as input homogeneous polynomials of distinct degree.

Hence we obtain another corollary, being the projective analogue of Equation (3.7).

Corollary 3.27. *Let $1 \leq k \leq q-1$ and $v = k + (m-1)(q-1)$. Then we have:*

$$\mathrm{PRM}_q(m,k) \subseteq \mathrm{Lift}(\mathrm{PRS}_q(k), m) \subseteq \mathrm{PRM}_q(m,v).$$

Example 3.28. We give here the smallest example of a projective lifted code that is not isomorphic to a projective Reed-Muller code. This is the analogue of Example 2.16 presented in a previous chapter. Let $q = 4$, $m = 2$ and $k = 3$, giving $v = k + (m-1)(q-1) = 6$. The projective Reed-Muller code $\mathrm{PRM}_q(m,k) = \mathrm{PRM}_4(2,3)$ has length $q^2+q+1 = 21$, dimension $\binom{m+k}{k} = 10$, and admits

$$\begin{aligned} D = \{&(3,0,0),(2,1,0),(2,0,1),(1,2,0),(1,1,1),(1,0,2),(0,3,0),\\ &(0,2,1),(0,1,2),(0,0,3)\} \end{aligned}$$

as a degree set. A tedious computation shows that $\mathrm{Lift}(\mathrm{PRS}_4(3),2)$ is given by the following degree set:

$$\begin{aligned} D_L = \{&(6,0,0),(5,1,0),(5,0,1),(4,2,0),(4,1,1),(4,0,2),(0,6,0),\\ &(0,5,1),(0,4,2),(0,0,6),(2,2,2)\}. \end{aligned}$$

One can point out that $D_L = D' \cup \{(2,2,2)\}$, where D' is obtained by adding $q-1=3$ to the leftmost non-zero coordinate of every $\boldsymbol{d} \in D$. Furthermore, the affine lifted code $\mathrm{Lift}(\mathrm{RS}_4(2),2)$ has the following degree set:

$$D_A = \{(0,0),(1,0),(0,1),(2,0),(1,1),(0,2),(2,2)\}.$$

We see that D_A corresponds to the puncturing on their first coordinate of elements $\boldsymbol{d} \in D_L$ such that $d_0 \neq 0$. We also notice that the set of remaining elements is

$$\{(0,6,0),(0,5,1),(0,4,2),(0,0,6)\}.$$

In other words, these are elements which, punctured at first on their first coordinate, and then P-reduced, give the degree set $\{(3,0),(2,1),(1,2),(0,3)\}$ of $\mathrm{PRS}_4(3)$.

Finally, notice that the extra degree $(2,2,2)$ which makes $\mathrm{Lift}(\mathrm{PRS}_4(3),2)$ larger than $\mathrm{PRM}_q(2,3)$ corresponds to the codeword $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^2}(X^2Y^2Z^2)$. Similarly to Example 2.16, one can then confirm that any embedding $(S,T) \mapsto (a_0S+b_0T, a_1S+b_1T, a_2S+b_2T)$ sends $\boldsymbol{c}$ to a projective Reed-Solomon codeword.

3.3 Local correction

After Guo *et al.*'s work [GKS13], we know that affine lifted codes are perfectly smooth locally correctable codes — see also Subsection 2.2.3. In this section, we prove that projective lifted

codes have similar local correction properties. For convenience, throughout this section we fix a projective lifted code $\mathcal{C} = \mathrm{Lift}(\mathrm{PRS}_q(k), m)$. We also denote by $n = \theta_{m,q}$ its length and by $v = k + (m-1)(q-1)$ the degree of polynomials evaluated in $\mathcal{C}$.

By definition of projective lifted codes, if $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathcal{C}$, then $\mathrm{ev}_{\mathbb{P}^1}(f \circ L) \in \mathrm{PRS}_q(k)$ for all $L \in \mathrm{Emb}_{\mathbb{P}}(m)$. In Remark 3.14 we noticed that $\mathrm{ev}_{\mathbb{P}^1}(f \circ L)$ is not a subword of $\boldsymbol{c}$. Nevertheless, there still exists $\boldsymbol{w}^{(L,v)} \in (\mathbb{F}_q^\times)^{q+1}$ such that $(\boldsymbol{w}^{(L,v)})^{-1} \star \mathrm{ev}_{\mathbb{P}^1}(f \circ L)$ is such a subword. Given L and the standard representation of points in $\mathbb{P}^1$, each $\mathbb{F}_q$-symbol in $\boldsymbol{w}^{(L,v)}$ is expressed as the v-th power of a linear combination of $\mathcal{O}(m)$ other $\mathbb{F}_q$-symbols. Moreover, a v-th power can be computed as k-th power since every $x \in \mathbb{F}_q$ satisfies $x^q = x$. Therefore, the tuple $(\boldsymbol{w}^{(L,v)})^{-1} \in \mathbb{F}_q^{\mathbb{P}^1}$ can be computed in $\mathcal{O}(mq \log k)$ operations over $\mathbb{F}_q$. To sum up we get:

Lemma 3.29. *Let $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$ and $L \in \mathrm{Emb}_{\mathbb{P}}(m)$. There exists a deterministic algorithm which computes $\mathrm{ev}_{\mathbb{P}^1}(f \circ L)$ from $\boldsymbol{c}$ and L, with $q+1$ queries to $\boldsymbol{c}$ and $\mathcal{O}(mq \log k)$ operations in $\mathbb{F}_q$.*

Let us now denote by ∞ the point $(0:1) \in \mathbb{P}^1$, and for a given $\boldsymbol{u} \in \mathbb{P}^m$,

$$\mathrm{Emb}_{\mathbb{P}}(m, \boldsymbol{u}) \coloneqq \{L \in \mathrm{Emb}_{\mathbb{P}}(m), \mathrm{ev}_\infty(L) = \boldsymbol{u}\}$$

the set of embeddings having $\boldsymbol{u}$ as image of the point at infinity.

We present in Algorithm 10 a generic local correcting algorithm for projective lifted codes. It is very similar to Algorithm 6 that locally corrects *affine* lifted codes. As well, this algorithm depends on a locality parameter $\ell \in [k+1, q]$, and it informally works as follows: (i) it picks at random ℓ points on a random projective line of $\mathbb{P}^m$ so that individual queries are uniform, (ii) it corrects the associated noisy $\mathrm{PRS}_q(k)$ codeword, and (iii) it outputs the desired corrected symbol.

For this purpose, we assume to have at our disposal an error-and-erasure correcting algorithm for $\mathrm{PRS}_q(k)$, which corrects $q+1-\ell$ erasures and up to $t = \lfloor \frac{\ell-k-1}{2} \rfloor$ errors (we recall that $\mathrm{PRS}_q(k)$ is an MDS code of dimension $k+1$). We call $\mathsf{Corr}_\ell^{\mathrm{PRS}}$ this correcting algorithm. For instance, one can use extensions of classical correcting algorithms for Reed-Solomon codes, as presented by Dür [Dür91] and Jensen [Jen95].

Proposition 3.30. *Let $k+1 \le \ell \le q$ and $t = \lfloor \frac{\ell-k-1}{2} \rfloor$. For every $\delta \le \frac{t+1}{2\ell}$, the code $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is a perfectly smooth $(\ell, \delta, \frac{\delta\ell}{t+1})$-locally correctable code using Algorithm 10.*

Proof. The proof is exactly the same as for Proposition 2.20. □

For the sake of completeness, we exhibit the two extreme instances ($\ell = k+1$ and $\ell = q$) which correspond respectively to a low-error and high-error correction setting.

Corollary 3.31 ($\ell = k+1$)**.** *For every $\delta \le \frac{1}{2(k+1)}$, the code $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is a perfectly smooth $(k+1, \delta, \delta(k+1))$-locally correctable code.*

Proof. $\ell = k+1$ implies $t = 0$. □

Corollary 3.32 ($\ell = q$)**.** *Let $\tau = \frac{1}{q}\lfloor \frac{q-k-1}{2} \rfloor$. For every $\delta \le \tau/2$, the code $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is a perfectly smooth $(q, \delta, \delta/\tau)$-locally correctable code.*

Proof. We have $\delta\ell/(t+1) \le \delta/\tau$, since $t = \tau\ell$. □

Algorithm 10: A generic local correcting algorithm of locality $\ell \in [k+1, q]$ for $\mathcal{C} = \mathrm{Lift}(\mathrm{PRS}_q(k), m)$

Input: a point $\boldsymbol{u} \in \mathbb{P}^m$, and an oracle access to $\boldsymbol{y} \in \mathbb{F}_q^{\mathbb{P}^m}$ such that $\mathrm{d}(\boldsymbol{y}, \boldsymbol{c}) \leq \delta n$, for some $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathcal{C}$, where $n = |\mathbb{P}^m|$.

Output: $\mathrm{ev}_{\boldsymbol{u}}(f)$ with high probability

/* Corr_A denotes a half-distance $(q+1-\ell)$-erasure correcting algorithm for the code $\mathcal{C}_{|A}$ isomorphic to $\mathrm{PRS}_q(k)$. */

Pick uniformly at random $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ such that $\boldsymbol{u} \in A := L(\mathbb{P}^1)$.

Toss a random binary coin $b \in \{0,1\}$, following $\mathscr{B}(p)$ with $p = \frac{\ell(q-1)}{q^{m+1}-1}$.

if $b = 0$ **then**

  Pick uniformly at random a subset $A' \subset A \setminus \{\boldsymbol{u}\}$ of size ℓ.

else

  Pick uniformly at random a subset $A' \subset A \setminus \{\boldsymbol{u}\}$ of size $\ell - 1$.

  Add $\boldsymbol{u}$ in A'.

Query $\{y_{\boldsymbol{x}} : \boldsymbol{x} \in A'\}$.

Define $\boldsymbol{y}' \in (\mathbb{F}_q \cup \{\perp\})^A$ by:

$$y'_{\boldsymbol{x}} := \begin{cases} (w^{(L,v)})_{\boldsymbol{x}} \cdot y_{\boldsymbol{x}} & \text{if } \boldsymbol{x} \in A', \\ \perp & \text{otherwise,} \end{cases}$$

where we recall that $\boldsymbol{w}^{(L,v)}$ is such that $\boldsymbol{w}^{(L,v)} \star \boldsymbol{c}_{|A} = \mathrm{ev}_{\mathbb{P}^1}(f \circ L) \in \mathrm{PRS}_q(k)$.

Run Corr_A on input $\boldsymbol{y}'$.

if Corr_A *fails* **then**

  Abort.

else

  Denote by $\boldsymbol{y} \in \mathbb{F}_q^A$ the output of Corr_A. Output $y_{\boldsymbol{u}}$.

It must be noticed that, in the minimum locality setting $\ell = k+1$, we obtain an LCC with parameters of the kind $(\ell, \delta, \delta\ell)$. These parameters were already obtained for the local correction of design-based codes (Proposition 2.30) and of Reed-Muller codes in the minimum locality case (Proposition 2.10). In the maximum locality setting $\ell = q$, we obtain parameters of the kind $(\ell, \delta, 2\delta/\delta_{\mathrm{loc}})$, where δ_{loc} is the relative minimum distance of the local PRS code. Once again, these parameters are similar to those of the first local correcting algorithm of Reed-Muller codes (Proposition 2.10), for instance.

Remark 3.33. In Algorithm 10, we can avoid to compute the tuple $\boldsymbol{w}^{(L,v)}$. Indeed, it can be proved that for every projective line $A \subset \mathbb{P}^m$ and every point $\boldsymbol{u} \in A$, there exists an $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ such that $L(\mathbb{P}^1) = A$, $\boldsymbol{u} \in \{\mathrm{ev}_{(1:0)}(L), \mathrm{ev}_{\infty}(L)\}$ and $\boldsymbol{w}^{(L,v)} = (1, \ldots, 1)$.

Let $L \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q^{m+1})$ such that $L(\mathbb{P}^1) = A$ and $L(\infty) = \boldsymbol{u}$ (for instance). The matrix of L in the canonical basis is then

$$M = \begin{pmatrix} \vdots & \vdots \\ \boldsymbol{a} & \boldsymbol{u} \\ \vdots & \vdots \end{pmatrix}.$$

Define i the minimum integer such that $(a_i, u_i) \neq \mathbf{0}$, and $j > i$ the minimum integer such that the minor

$$\begin{pmatrix} a_i & u_i \\ a_j & u_j \end{pmatrix}$$

is invertible.

If $u_i \neq 0$, then by definition of i, it must hold that $a_0 = \cdots = a_{i-1} = 0$. We then compute

$$M' = \begin{pmatrix} \vdots & \vdots \\ \boldsymbol{u} & \frac{1}{\lambda}(\boldsymbol{a} - \boldsymbol{u}) \\ \vdots & \vdots \end{pmatrix},$$

where λ is the first non-zero coordinate of $\boldsymbol{a} - \boldsymbol{u}$. Let $L' \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q^{m+1})$ with matrix M'. Then $L'(\mathbb{P}^1) = L(\mathbb{P}^1) = A$ since $\mathbb{P}^1$ is invariant under $\mathrm{Iso}(\mathbb{F}_q^2)$. Moreover, we can check that $L'((1:0)) = \boldsymbol{u}$, and that L' preserves the standard representation. Hence, $\boldsymbol{w}^{(L',v)} = \mathbf{1}$.

If $u_i = 0$, then we compute

$$M'' = \begin{pmatrix} \vdots & \vdots \\ \frac{1}{\mu}\boldsymbol{a} & \boldsymbol{u} \\ \vdots & \vdots \end{pmatrix}$$

where μ is the first non-zero coordinate of $\boldsymbol{a}$. Similarly to the previous case, $L''(\mathbb{P}^1) = A$, and points in $L''(\mathbb{P}^1)$ are written in standard representation, hence $\boldsymbol{w}^{(L'',v)} = \mathbf{1}$. Finally, we see that $L''(\infty) = \boldsymbol{u}$.

3.4 Intertwined relations between affine and projective lifted codes

In this section we state links between affine and projective lifted codes, notably after shortening and puncturing operations.

3.4.1 Motivation and similar results

The embedding of both $\mathbb{P}^{m-1}$ and $\mathbb{A}^m$ into $\mathbb{P}^m$ has raised relations between affine and projective Reed-Muller codes. Indeed, the hyperplane at infinity $\Pi_\infty := \{\boldsymbol{x} \in \mathbb{P}^m, x_0 = 0\}$ defines a restriction map

$$\begin{array}{rccc} \pi: & \mathbb{F}_q^{\mathbb{P}^m} & \to & \mathbb{F}_q^{\Pi_\infty} \\ & \boldsymbol{y} & \mapsto & \boldsymbol{y}_{|\Pi_\infty}, \end{array}$$

and π induces a map $\mathrm{PRM}_q(m,k) \to \mathrm{PRM}_q(m-1,k)$ by seeing Π_∞ as the projective space $\mathbb{P}^{m-1}$. Moreover, this map is surjective since every m-variate homogeneous polynomial of degree k can be also considered as an $(m+1)$-variate homogeneous polynomial of the same degree (in which the new variable, denoted X_0, does not appear).

Furthermore, the vector space $K := \ker\left(\mathrm{PRM}_q(m,k) \twoheadrightarrow \mathrm{PRM}_q(m-1,k)\right)$ consists in evaluation vectors of homogeneous polynomials $P \in \mathbb{F}_q[X_0, \ldots, X_m]_k^H$ such that X_0 divides P. That is,

$$K = \{\mathrm{ev}_{\mathbb{P}^m}(X_0 Q(\boldsymbol{X})), Q(\boldsymbol{X}) \in \mathbb{F}_q[\boldsymbol{X}]_{k-1}^H\}.$$

We now remark that restricting K to coordinates in $(\mathbb{P}^m \setminus \Pi_\infty) \simeq \mathbb{A}^m$ gives a vector space isomorphic to $\mathrm{RM}_q(m, k-1)$, since the monomial X_0 evaluates to 1 on every point of $\mathbb{A}^m$.

To sum up, we obtain the following short exact sequence:

$$0 \to \mathrm{RM}_q(m,k-1) \to \mathrm{PRM}_q(m,k) \xrightarrow{\pi} \mathrm{PRM}_q(m-1,k) \to 0.$$

From a coding theoretic point of view, it may be more comfortable to consider this sequence in the terminology of puncturing and shortening of codes. Indeed, up to isomorphism, the surjective map π corresponds to the puncturing of $\mathrm{PRM}_q(m,k)$ on coordinates lying in $\mathbb{A}^m \subset \mathbb{P}^m$, while the injection $\mathrm{RM}_q(m,k-1) \hookrightarrow \mathrm{PRM}_q(m,k)$ corresponds to its shortening on $\mathbb{P}^{m-1} \subset \mathbb{P}^m$.

A very similar exact sequence holds for the codes based on classical geometric designs. For instance, we have

$$0 \to \mathrm{Code}(\mathrm{AG}_t(m,q)) \to \mathrm{Code}(\mathrm{PG}_t(m,q)) \xrightarrow{\pi} \mathrm{Code}(\mathrm{PG}_t(m-1,q)) \to 0\,.$$

This result is presented by Assmus and Key in [AK92, Lemma 5.7.1] for the dual of these codes, but it remains true for the codes we consider, since duality of codes preserves such short sequences.

In this section, our goal is to prove similar results for lifted codes.

3.4.2 Shortening and puncturing projective lifted codes

We recall that Π_∞ denotes the hyperplane of $\mathbb{P}^m$ defined by $X_0 = 0$.

Theorem 3.34. *Let $m \geq 1$, $1 \leq k \leq q-1$, and $v = k + (m-1)(q-1)$. Let also $\mathcal{C} = \mathrm{Lift}(\mathrm{PRS}_q(k), m)$. Then we have*

$$\mathrm{Short}(\mathcal{C}, \Pi_\infty) = \mathrm{Lift}(\mathrm{RS}_q(k-1), m)$$

and

$$\mathrm{Punct}(\mathcal{C}, \mathbb{P}^m \setminus \Pi_\infty) = \mathrm{Lift}(\mathrm{PRS}_q(k), m-1).$$

Proof. For short, we denote by $\mathcal{S} = \mathrm{Short}(\mathcal{C}, \Pi_\infty)$ and $\mathcal{P} = \mathrm{Punct}(\mathcal{C}, \mathbb{P}^m \setminus \Pi_\infty)$.

(i) Proof of $\mathcal{S} = \mathrm{Lift}(\mathrm{RS}_q(k-1), m)$. Let $\boldsymbol{c} = \mathrm{ev}_{\mathbb{A}^m}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathrm{Lift}(\mathrm{RS}_q(k-1), m)$ and extend it to $\boldsymbol{c}' = \mathrm{ev}_{\mathbb{P}^m}(X_0^{d_0}\boldsymbol{X}^{\boldsymbol{d}})$, with $d_0 = v - |\boldsymbol{d}| > 0$. We have $\boldsymbol{c}' \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$ due to our previous study of degree sets (see Theorem 3.23). Thus, we simply notice that $\boldsymbol{c}'$ vanishes on Π_∞, and that $\boldsymbol{c}' = \boldsymbol{c}$ elsewhere, hence $\boldsymbol{c} \in \mathcal{S}$.

Conversely, let $\boldsymbol{c} \in \mathcal{S}$. There exists $f \in \mathbb{F}_q[\boldsymbol{X}]_v^H$ such that $\boldsymbol{c}' = \mathrm{ev}_{\mathbb{P}^m}(f)$ satisfies $\boldsymbol{c}' = 0$ over all coordinates of Π_∞, and $\boldsymbol{c}' = \boldsymbol{c}$ elsewhere. It means that the polynomial f vanishes on the whole projective hyperplane Π_∞ given by $X_0 = 0$. Therefore f vanishes over the *affine* hyperplane $\Pi'_\infty \subseteq \mathbb{A}^{m+1}$ given by $X_0 = 0$.

The previous remark makes sense since we can apply the Combinatorial *Nullstellensatz* proved by Alon in [Alo99]. This result asserts the following. Assume that $W = \prod_{i=0}^m W_i \subseteq \mathbb{F}_q^{m+1}$ and that $\deg(f)$ can be written as $|\boldsymbol{t}| = \sum_{i=0}^m t_i$, where each $t_i < |W_i|$. Then, $f(W) = \{0\}$ implies that $f_{\boldsymbol{t}} = 0$, where $f_{\boldsymbol{t}}$ denotes the coefficient of the monomial $\boldsymbol{X}^{\boldsymbol{t}}$ in f. In our context, let $W = \{\boldsymbol{0}\} \times \mathbb{F}_q^m$ and $\boldsymbol{t}$ satisfies $t_0 = 0$ and $t_i \leq q-1$ for all $i > 0$. The Combinatorial *Nullstellensatz* then shows that $\mathrm{Coeff}(f, \boldsymbol{X}^{\boldsymbol{t}}) = 0$. Therefore every monomial in f must be divisible by X_0. Said differently, f is a sum of monomials $\boldsymbol{X}^{\boldsymbol{d}}$ with $\boldsymbol{d}$ such that $d_0 \neq 0$, and Proposition 3.21 then shows that $\mathrm{ev}_{\mathbb{A}^m}(f) \in \mathrm{Lift}(\mathrm{RS}_q(k-1), m)$.

(ii) Proof of $\mathcal{P} = \mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$. First, we see that $\mathrm{Lift}(\mathrm{PRS}_q(k), m-1) \subseteq \mathcal{P}$. Indeed, a codeword $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^{m-1}}(\boldsymbol{X}^{\boldsymbol{d}}) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$ can be extended to $\boldsymbol{c}' = \mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}'}) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$, where we define $\boldsymbol{d}'$ by adding $q-1$ to the leftmost non-zero coordinate of $\boldsymbol{d}$.

Conversely, let $\boldsymbol{c}' = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$ such that $\boldsymbol{c}'_{|\Pi_\infty} \in \mathcal{P} \setminus \{\mathbf{0}\}$. Let $\boldsymbol{X}^{\boldsymbol{d}}$ be a monomial in f. If $d_0 \neq 0$, then $\mathrm{ev}_{\mathbb{P}^m}(\boldsymbol{X}^{\boldsymbol{d}})_{|\Pi_\infty} = \mathbf{0}$, hence one can assume that every monomial $\boldsymbol{X}^{\boldsymbol{d}}$ composing f satisfies $d_0 = 0$. Using Proposition 3.22 and linearity, we get that $\boldsymbol{c}'_{|\Pi_\infty} \in \mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$. □

Remark 3.35. For $m = 1$, we know that by definition, $\mathrm{Lift}(\mathrm{RS}_q(k-1), 1) = \mathrm{RS}_q(k-1)$ and $\mathrm{Lift}(\mathrm{PRS}_q(k), 1) = \mathrm{PRS}_q(k)$. Therefore, Theorem 3.34 rewrites the well-known result stating that the shortening at infinity of the projective Reed-Solomon code is a classical Reed-Solomon code of dimension one less.

Theorem 3.34 also translates in terms of exact sequences:

Corollary 3.36. *The following exact sequence holds for every $1 \leq k \leq q-1$ and $m \geq 1$:*

$$0 \to \mathrm{Lift}(\mathrm{RS}_q(k-1), m) \to \mathrm{Lift}(\mathrm{PRS}_q(k), m) \xrightarrow{\pi} \mathrm{Lift}(\mathrm{PRS}_q(k), m-1) \to 0\,,$$

where π is the restriction map to points at infinity.

3.4.3 Projective lifted codes as generalised design-based codes

We here prove that projective lifted codes are generalised design-based codes. Let $\mathcal{D} = (\mathbb{P}^m, \mathcal{B}) = \mathrm{PG}_1(m, q)$ be the classical design of points and projective lines.

For $B \in \mathcal{B}$, let $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ such that $L(\mathbb{P}^1) = B$. If $0 \leq k \leq q-1$, we define the code $\mathcal{L}_B^{(k)} \subseteq \mathbb{F}_q^B$, as the image of $\mathrm{PRS}_q(k)$ by the action of L on its support $\mathbb{P}^1$. In other words,

$$\mathcal{L}_B^{(k)} \coloneqq L^*(\mathrm{PRS}_q(k)) \subseteq \mathbb{F}_q^B\,.$$

Finally, we define $\mathcal{L}^{(k)} \coloneqq (\mathcal{L}_B^{(k)} : B \in \mathcal{B})$.

Proposition 3.37. *Let $\mathcal{L}^{(k)}$ defined as above. The projective lifted code* $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ *is the generalised design-based* $\mathrm{Code}_q(\mathrm{PG}_1(m, q), \mathcal{L}^{(k)})$.

Proof. For every projective line $B \in \mathcal{B}$, choose $L \in \mathrm{Emb}_{\mathbb{P}}(m)$ such that $L(\mathbb{P}^1) = B$ and $\boldsymbol{w}^{(L,v)} = \mathbf{1}$. Such an embedding exists thanks to Remark 3.33. Define the code $\mathcal{L}_B = (\boldsymbol{w}^{(L,v)})^{-1} \star \mathrm{PRS}_q(k) = \mathrm{PRS}_q(k)$. According to Remark 3.14, we know that for all $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$, we have $\boldsymbol{c}_{|B} = \mathrm{ev}_{\mathbb{P}^1}(f \circ L)$. Therefore $\boldsymbol{c}_{|B}$ lies in $\mathrm{PRS}_q(k)$. By definition, we then get

$$\mathrm{Lift}(\mathrm{PRS}_q(k), m) = \{\boldsymbol{c} \in \mathbb{F}_q^{\mathbb{P}^m} \mid \forall B \in \mathcal{B}, \boldsymbol{c}_{|B} \in \mathrm{PRS}_q(k)\} = \mathrm{Code}_q(\mathrm{PG}_1(m, q), \mathcal{L})\,.$$

□

In particular, if $k = q-1$, we see that projective lifted codes are actually codes based on the design $\mathrm{PG}_1(m, q)$. Let us now recall that the rank of such designs was given in Chapter 1. Hence, we get:

Corollary 3.38. *For any $t \geq 1$ and any prime p, we have:*

$$\dim\Big(\mathrm{Lift}(\mathrm{PRS}_{p^t}(2, p^t - 1))\Big) = p^{2t} + p^t - \Big(\frac{p(p+1)}{2}\Big)^t\,.$$

Remark 3.39. Very similarly, let $\mathrm{AG}_1(m, k) = (X, \mathcal{B})$. For $B \in \mathcal{B}$ and $0 \leq k \leq q-2$, we can define $\mathcal{L}_B^{(k)} \coloneqq L^*(\mathrm{RS}_q(k)) \subseteq \mathbb{F}_q^B$ a code isomorphic to $\mathrm{RS}_q(k)$, where $L(\mathbb{A}^1) = B$. A result similar to Proposition 3.37 then holds for affine lifted codes:

$$\mathrm{Lift}(\mathrm{RS}_q(k), m) = \mathrm{Code}(\mathrm{AG}_1(m, k), (\mathcal{L}_B^{(k)} : B \in \mathcal{B}))\,.$$

3.5 Other properties towards practicality

We here present miscellaneous results emphasizing the practicality of projective lifted codes. In Subsection 3.5.1, we prove that the storage cost of projective lifted codes can be reduced since they admit (quasi-)cyclic automorphisms. Explicit information sets are then computed in Subsection 3.5.2. We conclude this section by estimating their minimum distance (Subsection 3.5.3).

3.5.1 Automorphisms and (quasi-)cyclicity

In coding theory, automorphism groups of codes, and *a fortiori* their permutation groups, are interesting for many reasons. For instance, they can be used for reducing the practical storage cost of the codes. Cyclic codes are known to be specifically efficient in that sense.

In this section, we address the question of (quasi-)cyclicity for projective lifted codes. We prove in Proposition 3.44 that under arithmetic conditions, the code $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is quasi-cyclic up to a diagonal isomorphism. This result relies deeply on the fact that $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is invariant under the action of $\mathrm{Proj}_v(\mathbb{P}^m)$, that has been proved in Lemma 3.19. But let us first define cyclicity and quasi-cyclicity.

Definition 3.40 (Cyclicity, quasi-cyclicity)**.** A code $\mathcal{C} \subseteq \mathbb{F}_q^X$, $|X| = n$, is *cyclic* if $\mathrm{Perm}(\mathcal{C})$ contains a cyclic permutation of order n (that is, an n-cycle). It is said to be *quasi-cyclic* of index d if $\mathrm{Perm}(\mathcal{C})$ contains a permutation which is the product of d different (n/d)-cycles with disjoint orbits. In particular, a cyclic code is a quasi-cyclic code of index 1.

In all what follows, we fix an integer $m \geq 1$, and we denote by $n = \theta_{m,q}$ and by $d = \gcd(n, q-1)$. Let also $\phi : \mathbb{F}_{q^{m+1}} \to \mathbb{F}_q^{m+1}$ be an isomorphism of $\mathbb{F}_q$-vector spaces, and ω be a primitive element of $\mathbb{F}_{q^{m+1}}$. We define $\beta := \omega^{q-1}$. It is clear that β has order n in the multiplicative group $\mathbb{F}_{q^{m+1}}^\times$ since $(q-1)n = q^{m+1} - 1$. For every $0 \leq i < d$, we now define:

$$U_i = \{\phi(\omega^i \beta^d), \ldots, \phi(\omega^i (\beta^d)^{n/d})\} \subset \mathbb{F}_q^{m+1},$$

and the union $U = \cup_{j=0}^{d-1} U_j$.

Definition 3.41 (representation of $\mathbb{P}^m$)**.** Recall that $n = |\mathbb{P}^m|$. We say that an n-subset $V = \{v_1, \ldots, v_n\} \subset \mathbb{F}_q^{m+1}$ represents $\mathbb{P}^m$ if the projective points that V defines fill $\mathbb{P}^m$ entirely.

Lemma 3.42. *If n/d and $q-1$ are coprime, then U represents $\mathbb{P}^m$.*

Proof. We only need to prove that the elements $\phi(\omega^i \beta^{dj}) \in \mathbb{F}_q^{m+1}$ define distinct projective points for $0 \leq i < d$ and $1 \leq j \leq n/d$. Since ϕ is bijective, it reduces to prove that, for $0 \leq i_1, i_2 < d$ and $1 \leq j_1, j_2 \leq n/d$, if $\omega^{i_1 - i_2} \beta^{d(j_1 - j_2)} \in \mathbb{F}_q$, then $(i_1, j_1) = (i_2, j_2)$.

Assume $(\omega^{i_1 - i_2} \beta^{d(j_1 - j_2)})^{q-1} = 1$. Then $\mathrm{ord}(\omega) = (q-1)n$ divides $(q-1) \times ((i_1 - i_2) + d(q-1)(j_1 - j_2))$, that is, $n \mid (i_1 - i_2) + d(q-1)(j_1 - j_2)$.

Since $d \mid n$, we get $d \mid (i_1 - i_2)$ which implies $i_1 = i_2$ because $0 \leq i_1, i_2 < d$. Hence $n \mid d(q-1)(j_1 - j_2)$, and our assumption $\gcd(n/d, q-1) = 1$ ensures that $(n/d) \mid j_1 - j_2$. Since $1 \leq j_1, j_2 < n/d$, we finally obtain $j_1 = j_2$. □

Of course, every $\boldsymbol{u} = \phi(\omega^i \beta^{dj}) \in U$ is not necessarily represented in a standard form. Denote by $\boldsymbol{x}_{\boldsymbol{u}} \in \mathbb{F}_q^{m+1}$ its standard form. As usual we have $\boldsymbol{x}_{\boldsymbol{u}} = \lambda_{\boldsymbol{u}} \boldsymbol{u}$ for some $\lambda_{\boldsymbol{u}} \in \mathbb{F}_q^\times$, and we can define

$$\boldsymbol{w}^{(v)} \coloneqq ((\lambda_{\boldsymbol{u}})^{-v} : \boldsymbol{u} \in U) \in (\mathbb{F}_q^\times)^U .$$

Similarly to the definition of $\mathrm{ev}_{\mathbb{P}^m}$ given previously, we can define an evaluation map over $U \subset \mathbb{F}_q^{m+1}$ by $\mathrm{ev}_U : \mathbb{F}_q[X_0, \ldots, X_m] \to \mathbb{F}_q^U$, $f \mapsto (f(\boldsymbol{u}) : \boldsymbol{u} \in U)$. The proof of the following lemma is elementary.

Lemma 3.43. *Assume n/d and $q-1$ are coprime, and recall that $v = k + (m-1)(q-1)$. Let $\mathcal{C} = \mathrm{Lift}(\mathrm{PRS}_q(k), m)$, and denote by $D = \mathrm{Deg}(\mathcal{C})$. Let finally $\mathcal{C}' = \mathrm{ev}_U(\mathrm{Poly}(D))$. Then,*

$$\mathcal{C}' = \boldsymbol{w}^{(v)} \star \mathcal{C} ,$$

when coordinates $U \subset \mathbb{F}_q^{m+1} \setminus \{\mathbf{0}\}$ are seen as projective points.

Let us now introduce $\sigma : \mathbb{F}_{q^{m+1}} \to \mathbb{F}_{q^{m+1}}$ given by $x \mapsto \beta x$. We also denote by $\psi \coloneqq \phi \circ \sigma \circ \phi^{-1}$ the associated map over the vector space $\mathbb{F}_q^{m+1}$. It is clear that $\psi \in \mathrm{Hom}(\mathbb{F}_q^{m+1}, \mathbb{F}_q^{m+1})$, and since σ and ϕ are bijective maps, we get $\psi \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$. For $i \geq 0$, we finally denote by ψ^i the i-fold composition of ψ. Notice that $\psi^i(\boldsymbol{x}) = \phi(\omega^{i(q-1)} \phi^{-1}(\boldsymbol{x}))$ holds for every point $\boldsymbol{x} \in \mathbb{P}^m$.

Proposition 3.44. *If n/d and $(q-1)$ are coprime, then $\mathcal{C}' \coloneqq \boldsymbol{w}^{(v)} \star \mathrm{Lift}(\mathrm{PRS}_m(k), m)$ is quasi-cyclic of index d, through the permutation $\psi^d \in \mathfrak{S}(U)$. The orbits of ψ^d are given by the subsets U_i, $0 \leq i \leq d-1$.*

Proof. We can check that $\psi^d(U) = U$, hence $\psi^d \in \mathfrak{S}(U)$. Since $\psi^d \in \mathrm{Iso}(\mathbb{F}_q^{m+1})$, the polynomial space $\mathrm{Poly}(D)$, $D = \mathrm{Deg}(\mathcal{C})$, is invariant under ψ^d. Moreover, we have $\mathcal{C}' = \mathrm{ev}_U(\mathrm{Poly}(D))$ thanks to Lemma 3.43. Therefore $\psi^d \in \mathrm{Perm}(\mathcal{C}')$.

Let us now prove that ψ^d is an (n/d)-cycle. For $\phi(\omega^i \beta^{jd}) \in U_i$, we have

$$\psi^d(\phi(\omega^i \beta^{jd})) = \phi(\omega^i \beta^{jd} \beta^d) = \phi(\omega^i \beta^{(j+1)d}) \in U_i .$$

It remains to show that the order of ψ^d is n/d. Since ϕ is one-to-one and U represents $\mathbb{P}^m$, for every $0 \leq s \leq t < n/d$ we have:

$$\forall \boldsymbol{u} \in U_i, (\psi^d)^s(\boldsymbol{u}) = (\psi^d)^t(\boldsymbol{u}) \iff \omega^{(t-s)d(q-1)} = 1 \iff n \mid (t-s)d(q-1) .$$

Our assumption on n/d and $(q-1)$ implies that $t = s$; hence ψ^d has order n/d. □

As a corollary, when $d = 1$ we obtain the following result.

Corollary 3.45. *If n and $q-1$ are coprime, then for all $1 \leq k \leq q-1$ the code*

$$\boldsymbol{w}^{(v)} \star \mathrm{Lift}(\mathrm{PRS}_q(k), m)$$

is a cyclic code.

Remark 3.46. A very similar approach was used by Berger and de Maximy in [BdM01], in order to prove the quasi-cyclicity of codes isomorphic to our definition of projective Reed-Muller codes.

3.5.2 Explicit information sets

In this section, we give explicit information sets for projective lifted codes. Among other applications, such sets are useful to extend the local correctability of lifted codes to a local *decodability* property (see [Yek12]).

Our techniques are highly inspired by the work of Guo and Kopparty [GK16, Appendix A]. We also prove a stronger result, being that a large family of affine evaluation codes present the same information sets as affine lifted codes.

Monomiality of bounded degree affine evaluation codes. Similarly to the previous section, let $\phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$ be an $\mathbb{F}_q$-isomorphism. We denote by $\mathbb{F}_q[\boldsymbol{X}]_{q-1}^{\infty} \coloneqq \mathrm{Poly}(B_{\infty}^m(q-1))$ the space of m-variate polynomials of *partial* degree bounded by $q-1$. If $f \in \mathbb{F}_q[\boldsymbol{X}]_{q-1}^{\infty}$ is seen as a function, then the map $f \circ \phi : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$ can be interpolated uniquely as a univariate polynomial $\mathbb{F}_{q^m}[X]$ of degree less than $q^m - 1$. We denote by ϕ^* this process, which also appears to be an $\mathbb{F}_q$-isomorphism:

$$\begin{array}{rccc} \phi^* : & \mathbb{F}_q[\boldsymbol{X}]_{q-1}^{\infty} & \to & \mathbb{F}_{q^m}[X]_{q^m-1} \\ & f(\boldsymbol{X}) & \mapsto & (f \circ \phi)(X)\,. \end{array}$$

For a nonzero $a \in \mathbb{F}_{q^m}^{\times}$, we denote by $\mu_a : \mathbb{F}_{q^m} \to \mathbb{F}_{q^m}$, $x \mapsto ax$. It is well-known that $\mathrm{Iso}(\mathbb{F}_{q^m}) = \{\mu_a, a \in \mathbb{F}_{q^m}^{\times}\}$. Every map μ_a being $\mathbb{F}_q$-linear, we can define $M_a \coloneqq \phi \circ \mu_a \circ \phi^{-1} \in \mathrm{Iso}(\mathbb{F}_q^m)$. The map M_a is often known as the $\mathbb{F}_q$*-homomorphism of the multiplication by* $a \in \mathbb{F}_{q^m}$.

Let $\mathcal{F}$ be a subspace of $\mathbb{F}_q[\boldsymbol{X}]$. The group of automorphisms of $\mathcal{F}$, denoted $\mathrm{Aut}(\mathcal{F})$, is the group of invertible linear maps $\mathbb{F}_q[\boldsymbol{X}] \to \mathbb{F}_q[\boldsymbol{X}]$ which let the set $\mathcal{F}$ invariant. We also recall that every $\phi = (\phi_1, \ldots, \phi_m) \in \mathrm{Hom}(\mathbb{F}_q^m)$ can be seen as a linear map $\mathbb{F}_q[\boldsymbol{X}] \to \mathbb{F}_q[\boldsymbol{X}]$, through the action $\phi^*(f) : f(\boldsymbol{X}) \mapsto f(\phi_1(\boldsymbol{X}), \ldots, \phi_m(\boldsymbol{X}))$.

Lemma 3.47. *Let $\mathcal{F}$ be a subspace of $\mathbb{F}_q[\boldsymbol{X}]_{q-1}^{\infty}$, and assume that $\mathrm{Iso}(\mathbb{F}_q^m) \subseteq \mathrm{Aut}(\mathcal{F})$. Then $\mathrm{Iso}(\mathbb{F}_{q^m}) \subseteq \mathrm{Aut}(\phi^*(\mathcal{F}))$.*

Proof. Let $f \circ \phi \in \phi^*(\mathcal{F})$. For every $\mu_a \in \mathrm{Aut}(\mathbb{F}_{q^m})$, we have $f \circ \phi \circ \mu_a = f \circ M_a \circ \phi$ by definition of the matrix of the multiplication by a. But $M_a \in \mathrm{Aut}(\mathbb{F}_q^m)$, hence $f \circ M_a \in \mathcal{F}$ and we get $f \circ \phi \circ \mu_a \in \phi^*(\mathcal{F})$. □

Let us define the subgroup of diagonal isomorphisms

$$\mathrm{Diag}(\mathbb{F}_q^m) \coloneqq \{\mathrm{diag}_{\boldsymbol{a}} : \boldsymbol{x} \mapsto (a_1 x_1, \ldots, a_m x_m),\ \boldsymbol{a} \in (\mathbb{F}_q^{\times})^m\} \subseteq \mathrm{Iso}(\mathbb{F}_q^m)\,.$$

Proposition 3.48. *Let $\mathcal{F}$ be a subspace of m-variate polynomials of partial degree bounded by $q-2$, that is $\mathcal{F} \subseteq \mathrm{Poly}(B_{\infty}^m(q-2))$. If $\mathrm{Diag}(\mathbb{F}_q^m) \subseteq \mathrm{Aut}(\mathcal{F})$, then $\mathcal{F}$ is generated by monomials.*

Proof. Let $f \in \mathcal{F}$, such that $f(\boldsymbol{X}) = \sum_{\boldsymbol{d} \in D} f_{\boldsymbol{d}} \boldsymbol{X}^{\boldsymbol{d}}$ with $D = \{\boldsymbol{d} \in \mathbb{N}^m, f_{\boldsymbol{d}} \neq 0\}$. It is sufficient to prove that for all $\boldsymbol{d} \in D$, $\boldsymbol{X}^{\boldsymbol{d}}$ lies in $\mathcal{F}$.

Let $\boldsymbol{d} \in D$. Similarly to the proof of Theorem 3.17, we define

$$Q_{\boldsymbol{d}}(\boldsymbol{X}) \coloneqq (-1)^m \sum_{\boldsymbol{a} \in (\mathbb{F}_q^{\times})^m} \Big(\prod_{i=1}^{m} a_i^{-d_i} \Big) (f \circ \mathrm{diag}_{\boldsymbol{a}})(\boldsymbol{X})$$

Since $\mathcal{F}$ is a vector space and $\mathrm{Diag}(\mathbb{F}_q^m) \subseteq \mathrm{Aut}(\mathcal{F})$, we see that $Q_{\boldsymbol{d}}(\boldsymbol{X}) \in \mathcal{F}$.

$$\begin{aligned} Q_{\boldsymbol{d}}(\boldsymbol{X}) &= \sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^m} \Big(\prod_{i=1}^m -a_i^{-d_i}\Big) \sum_{\boldsymbol{j}} f_{\boldsymbol{j}}\, a_1^{j_1}\dots a_m^{j_m}\, \boldsymbol{X}^{\boldsymbol{j}} \\ &= \sum_{\boldsymbol{j}} f_{\boldsymbol{j}} \sum_{\boldsymbol{a}\in(\mathbb{F}_q^\times)^m} \Big(\prod_{i=1}^m -a_i^{j_i-d_i}\Big) \boldsymbol{X}^{\boldsymbol{j}} \\ &= \sum_{\boldsymbol{j}} f_{\boldsymbol{j}} \prod_{i=1}^m \Big(\underbrace{-\sum_{a_i\in\mathbb{F}_q^\times} a_i^{j_i-d_i}}_{=0 \text{ if } d_i\neq j_i,\, 1 \text{ otherwise}} \Big) \boldsymbol{X}^{\boldsymbol{j}} \quad = \quad f_{\boldsymbol{d}} \boldsymbol{X}^{\boldsymbol{d}}\,. \end{aligned}$$

We know that $\boldsymbol{d} \in D$, hence $f_{\boldsymbol{d}} \neq 0$ and by linearity we obtain $\boldsymbol{X}^{\boldsymbol{d}} = \frac{1}{f_{\boldsymbol{d}}} Q_{\boldsymbol{d}}(\boldsymbol{X}) \in \mathcal{F}$. □

Notice that the last result can be seen as a reformulation of the *monomial extraction lemma* of Kaufman and Sudan [KS08] in the case of invariance under diagonal automorphisms of the affine space.

Information sets of some affine evaluation codes.

Lemma 3.49. *Let $\mathcal{F} \subseteq \mathrm{Poly}(B_\infty^m(q-1))$ and assume that $S \subseteq \mathbb{A}^1(\mathbb{F}_{q^m})$ is an information set for $\mathrm{ev}_{\mathbb{A}^1}(\phi^*(\mathcal{F}))$. Then $\phi(S)$ is an information set for $\mathrm{ev}_{\mathbb{A}^m}(\mathcal{F})$.*

Proof. This follows from the fact that $\phi^*(\mathcal{F}) = \{f \circ \phi, f \in \mathcal{F}\}$ and ϕ is an $\mathbb{F}_q$-isomorphism. □

In the next proposition, we give a result that improves upon the theorem given by Guo and Kopparty in [GK16, Appendix A], in the specific case of codes evaluating polynomials with partial degree bounded by $q-2$ (which is the case for many interesting codes). Indeed, their result holds for affine-invariant codes while here we only need codes invariant under $\mathrm{Iso}(\mathbb{F}_q^m)$.

Proposition 3.50. *Let $\mathcal{C} = \mathrm{ev}_{\mathbb{A}^m}(\mathcal{F})$ be an affine evaluation code of dimension k over $\mathbb{F}_q$, and assume that $\mathcal{F} \subseteq \mathrm{Poly}(B_\infty^m(q-2))$ and $\mathrm{Iso}(\mathbb{F}_q^m) \subseteq \mathrm{Aut}(\mathcal{F})$. Then, for every primitive element ω of $\mathbb{F}_{q^m}$, and every isomorphism $\phi : \mathbb{F}_{q^m} \to \mathbb{F}_q^m$, the set $\{\phi(\omega), \dots, \phi(\omega^k)\}$ is an information set for $\mathcal{C}$.*

Proof. Thanks to Lemma 3.49, it is sufficient to prove that $S = \{\omega, \dots, \omega^k\}$ is an information set for $\mathcal{C}' = \mathrm{ev}_{\mathbb{A}^1}(\phi^*(\mathcal{F}))$. Moreover, since $\mathrm{Diag}(\mathbb{F}_q^m) \subseteq \mathrm{Iso}(\mathbb{F}_q^m)$, Proposition 3.48 and Lemma 3.47 ensures that $\mathcal{C}'$ is monomial. Denote by $I = \mathrm{Deg}(\mathcal{C}') = \{i_1, \dots, i_k\}$, and let $g(X) = \sum_{i\in I} a_i X^i \in \mathcal{F}$. We need to prove that:

$$g \neq 0 \quad \Longrightarrow \quad \mathrm{ev}_S(g) \neq 0\,.$$

To this end, let us remark that

$$\begin{pmatrix} \omega^{i_1} & \omega^{i_2} & \dots & \omega^{i_k} \\ \omega^{2i_1} & \omega^{2i_2} & \dots & \omega^{2i_k} \\ \vdots & \vdots & \ddots & \vdots \\ \omega^{ki_1} & \omega^{ki_2} & \dots & \omega^{ki_k} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_k \end{pmatrix} = \begin{pmatrix} g(\omega) \\ g(\omega^2) \\ \vdots \\ g(\omega^k) \end{pmatrix} = \mathrm{ev}_S(g)\,.$$

Since the left-hand square matrix is a Vandermonde matrix and ω is primitive, it is invertible and the result is proved. □

As a corollary we retrieve Guo and Kopparty's result, since $\mathrm{Iso}(\mathbb{F}_q^m)$ is a subgroup of the group of affine transformations.

Corollary 3.51 (given in [GK16]). *Let $\mathcal{C} = \mathrm{Lift}(\mathrm{RS}_q(k), m)$ for $k \leq q-2$. Then, for every ω primitive element of $\mathbb{F}_{q^m}$, and every ϕ isomorphism $\mathbb{F}_{q^m} \to \mathbb{F}_q^m$, the set $\{\phi(\omega), \ldots, \phi(\omega^{\dim \mathcal{C}})\}$ is an information set for $\mathcal{C}$.*

The case of projective evaluation codes. We would like to prove a similar result for projective lifted codes. Unfortunately, it does not make sense to define an isomorphism between $\mathbb{P}^1(\mathbb{F}_{q^m})$ and $\mathbb{P}^m(\mathbb{F}_q)$ since they do not have same cardinality. To solve this issue, our idea is to decompose $\mathbb{P}^m(\mathbb{F}_q)$ into affine parts, and to use recursively the links between projective and affine lifted codes we stated in Section 3.2.

Let $\mathbb{P}^m(\mathbb{F}_q) = \sqcup_{i=0}^m \mathbb{A}^{m,i}(\mathbb{F}_q)$, where

$$\mathbb{A}^{m,i}(\mathbb{F}_q) \coloneqq \{(0 : \ldots : 0 : 1 : x_1 : \ldots : x_i), (x_1, \ldots, x_i) \in \mathbb{A}^i(\mathbb{F}_q)\}.$$

Informally, $\mathbb{A}^{m,i}(\mathbb{F}_q)$ is the affine part of the i-dimensional projective subspace at infinity of $\mathbb{P}^m(\mathbb{F}_q)$.

Theorem 3.52. *Let $\mathcal{C} = \mathrm{Lift}(\mathrm{RS}_q(k), m)$ for $k \leq q-1$. Let us fix, for every $1 \leq i \leq m$, a primitive element ω_i of $\mathbb{F}_{q^i}$ and an isomorphism $\phi_i : \mathbb{F}_{q^i} \to \mathbb{A}^{m,i}(\mathbb{F}_q)$. Then, the set*

$$S = \sqcup_{i=0}^m \{\phi_i(\omega_i), \ldots, \phi_i(\omega_i^{\dim \mathcal{C}_i})\}$$

is an information set for $\mathcal{C}$, where $\mathcal{C}_i = \mathrm{Lift}(\mathrm{RS}_q(k-1), i)$ for $i > 0$, and by convention, $\dim(\mathcal{C}_0) = 1$ and $\phi_0(\mathbb{F}_{q^0}) \coloneqq \{(0 : \ldots : 0 : 1)\}$.

Proof. We proceed by induction on m.

- *Case $m = 1$.* Since $\mathcal{C} = \mathrm{PRS}_q(k)$ is an MDS code of dimension $k+1$, any $(k+1)$-subset of $\mathbb{P}^1$ is an information set for $\mathcal{C}$. In particular, $S = \{(0:1)\} \cup \{\phi_1(\omega_1), \ldots, \phi_1(\omega_1^k)\}$ is one of them.
- *Induction step.* Assume the result holds for step $m-1$. A basis of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ consists in evaluating monomials with exponents in $\mathrm{PDeg}_q(m, k)$. Thanks to Theorem 3.23, we know that $\mathrm{PDeg}_q(m,k)$ is in bijection with the disjoint union $\mathrm{ADeg}_q(m, k-1) \cup \mathrm{PDeg}_q(m-1, k)$, where the bijection is given in the proof of the theorem. Hence, there exists a generator matrix of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ defined as follows:

$$\begin{array}{l} \\ \\ \text{evaluation of monomials with degrees in } \mathrm{ADeg}(m, k-1) \\ \\ \text{evaluation of monomials with degrees in } \mathrm{PDeg}(m-1, k) \end{array} \begin{array}{c} \overbrace{\qquad\qquad\qquad\qquad}^{\mathbb{A}^m(\mathbb{F}_q)} \quad \overbrace{\qquad\qquad}^{\simeq \mathbb{P}^{m-1}(\mathbb{F}_q)} \\ \ldots \quad (1 : x_1 : \ldots : x_m) \quad \ldots \quad (0 : \ldots) \\ \left(\begin{array}{c|c} G_0 & 0 \\ \hline * & G_1 \end{array} \right) \end{array}$$

  where G_0 and G_1 are respective generator matrices for the codes $\mathrm{Lift}(\mathrm{RS}_q(k-1), m)$ and $\mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$.

Since G_0 and G_1 are full-rank, we know that the union of an information set S_0 of $\mathrm{Lift}(\mathrm{RS}_q(k-1), m)$ and an information set S_1 of $\mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$ gives an information set S of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$. Information sets of affine lifted codes are described in Corollary 3.51 (we just need to take care about the way we represent affine points in the projective space, whence the definition of the $\mathbb{A}^{m,i}$, $1 \leq i \leq m$). Therefore we have $S_0 = \{\phi_m(\omega_m), \ldots, \phi_m(\omega_m^{\dim \mathcal{C}_m})\}$ where ϕ_m, ω_m are defined as in the statement of the theorem. Moreover, the inductive step gives the information set of $\mathrm{Lift}(\mathrm{PRS}_q(k), m-1)$: it is exactly $S_1 = \sqcup_{i=0}^{m-1} \{\phi_i(\omega_i), \ldots, \phi_i(\omega_i^{\dim \mathcal{C}_i})\}$.
Setting $S = S_0 \cup S_1$ finally leads us to the result at step m. □

3.5.3 Estimation of the minimum distance

We give bounds on the minimum distance of a projective lifted code, depending on the minimum distance of the underlying projective Reed-Solomon code. In this section, $\mathrm{nz}_{\mathbb{P}^m}(f)$ denotes the number of zeroes of $f \in \mathbb{F}_q[X_0, \ldots, X_m]_v^H$ over the set $\mathbb{P}^m$, and recall that $\theta_{m,q} = \frac{q^{m+1}-1}{q-1}$.

Proposition 3.53 (upper bound). *Let $1 \leq k \leq q-1$ and $\mathrm{PRS}_q(k)$ be the projective Reed-Solomon code of dimension $k+1$ and distance $d = q+1-k$. Then the distance D of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ satisfies:*

$$D \leq dq^{m-1} + \theta_{m-2,q}\,.$$

As a corollary, the respective relative distances δ and Δ of $\mathrm{PRS}_q(k)$ and $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ satisfy:

$$\Delta \leq (1-b)\delta + b, \quad \text{where } 0 \leq b \leq q^{-2}\,.$$

Proof. Let $\boldsymbol{c} = \mathrm{ev}_{\mathbb{P}^1}(g) \in \mathrm{PRS}_q(k)$ be a minimum-weight codeword, *i.e.* $\mathrm{wt}(\boldsymbol{c}) = d$. Assume that $g(X_0, X_1) = \sum_{i=0}^{k} g_i X_0^i X_1^{k-i}$, and let

$$f(X_0, \ldots, X_m) \coloneqq g_0 X_1^{(q-1)(m-1)+k} + \sum_{i=1}^{k} g_i X_0^{(q-1)(m-1)+i} X_1^{k-i} \in \mathbb{F}_q[X_0, \ldots, X_m]_v^H$$

where $v = (q-1)(m-1)+k$. By studying the degrees of f, one can check that $\boldsymbol{c}' \coloneqq \mathrm{ev}_{\mathbb{P}^m}(f) \in \mathrm{Lift}(\mathrm{PRS}_q(k), m)$. Moreover, for every $(x_0 : x_1) \in \mathbb{P}^1$, we have:

$$f(x_0, x_1, x_2, \ldots, x_m) = g(x_0, x_1), \quad \forall \boldsymbol{x} = (x_2, \ldots, x_m) \in \mathbb{F}_q^{m-1}\,.$$

Hence $\boldsymbol{c}'$ is non-zero, and:

$$D \leq \mathrm{wt}(\boldsymbol{c}') = \theta_{m,q} - \mathrm{nz}_{\mathbb{P}^m}(f) \leq \theta_{m,q} - \mathrm{nz}_{\mathbb{P}^1}(g)\, q^{m-1} = \theta_{m,q} - q^{m-1}(q+1-d)\,. \tag{3.11}$$

It remains to notice that $\theta_{m,q} - (q+1)q^{m-1} = \theta_{m-2,q}$. For the bound on the relative distance, we divide both sides of Equation (3.11) by $\theta_{m,q}$ and we use the definition $d = (q+1)\delta$. Then we get:

$$\Delta \leq 1 + (\delta - 1)a\,,$$

where $a = \frac{(q+1)q^{m-1}}{\theta_{m,q}} = 1 - \frac{q^{m-1}-1}{q^{m+1}-1}$ satisfies $1 - q^{-2} \leq a \leq 1$. Denoting $b = 1-a$ concludes the proof. □

Proposition 3.54 (lower bound). *Let $1 \leq k \leq q-1$ and $\mathrm{PRS}_q(k)$ be a projective Reed-Solomon code of dimension $k+1$ and distance $d = q+1-k$. Then the distance D of $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ satisfies:*

$$D \geq 1 + (d-1)\theta_{m-1,q}$$

where $\theta_{m-1,q} = \frac{q^m-1}{q-1}$. *As a corollary, the relative distances* δ *of* $\mathrm{PRS}_q(k)$ *and* Δ *of* $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ *satisfy:*

$$\Delta \geq (1-b')\delta - b', \quad \text{where } 0 \leq b' \leq q^{-1}.$$

Proof. Proposition 3.37 shows that the code $\mathrm{Lift}(\mathrm{PRS}_q(k), m)$ is a generalised design-based code $\mathrm{Code}(\mathrm{PG}_1(m,q), \mathcal{L})$. Every $\mathcal{L}_B$ is isomorphic to a projective Reed-Solomon code $\mathrm{PRS}_q(k)$ which has minimum distance d. Therefore the bound $D \geq 1 + (d-1)\theta_{m-1,q}$ comes from Proposition 2.34. Dividing both sides by $\theta_{m,q}$ and using $\theta_{m,q} = q\theta_{m-1,q} + 1$, we finally get:

$$\Delta \geq \frac{(q+1)\theta_{m-1,q}}{\theta_{m,q}}\delta - \frac{\theta_{m-1,q}-1}{\theta_{m,q}} \geq (1-b')\delta - b',$$

where $b' = \frac{\theta_{m-1,q}-1}{\theta_{m,q}} = qb$ and b is defined in the previous proposition. □

We do not have any result concerning the sharpness of the bounds presented in Propositions 3.53 and 3.54. Yet, letting $q \to \infty$ for fixed $0 < \delta < 1$, we see that the family of lifted codes $\{\mathrm{Lift}(\mathrm{PRS}_q((q+1)(1-\delta)), m)\}_q$ reaches the same asymptotical relative distance δ as the family of PRS codes $\{\mathrm{PRS}_q((q+1)(1-\delta))\}_q$.

3.6 Rate and degree sets of lifted codes

The explicit construction of affine and projective lifted codes is not as easy as many other families of codes, since their definition is not constructive. It is still possible to build a parity-check matrix for these codes, by listing the parity-check equations given by the constraints on the lines. Then, a basis of the code can be obtained by solving a linear system. However, this technique admits two main drawbacks. First, it does not produce a basis of the code that we really understand (for instance, we have few information about which polynomials are evaluated in the basis we obtain). Second, it can be quite inefficient: the number of equations of the linear system can be quadratic in the dimension of code we look for, since for large values of m, the number of lines in $\mathbb{A}^m$ or $\mathbb{P}^m$ is almost quadratic in the number of points.

The degree set of a lifted code provides a solution to the first issue: it gives a simple and understandable basis of the code, since it represents the exponents of a monomial basis. Though its definition is not constructive either, in Subsection 3.6.1 we propose an algorithm to compute it efficiently. Subsection 3.6.2 is devoted to the analysis of the degree sets of second order lifted codes ($m = 2$).

Notice that this section mainly focuses on *affine* lifted codes, since we have seen in Theorem 3.23 that degree sets of projective lifted codes can be obtained easily from those of affine lifted codes.

3.6.1 Computation of degree sets

Let $S \subseteq [0, q-1]^m$. We know that $(S, \leq_p)$ is a partially ordered set, or *poset*. Let $\boldsymbol{d}, \boldsymbol{d}' \in S$ such that $\boldsymbol{d} \leq_p \boldsymbol{d}'$. Using classical poset terminology, we say that $\boldsymbol{d}$ is a *lower bound* for $\boldsymbol{d}'$, and that $\boldsymbol{d}'$ is an *upper bound* for $\boldsymbol{d}$. The subset of S that consists in lower (resp. upper) bounds for $\boldsymbol{d}$ is denoted $\mathrm{Low}_D(\boldsymbol{d})$ (resp. $\mathrm{Up}_D(\boldsymbol{d})$). Similarly, one can define lower and upper bounds for subsets $T \subseteq S$:

$$\mathrm{Low}_S(T) \coloneqq \{\boldsymbol{d} \in S \mid \forall \boldsymbol{t} \in T, \boldsymbol{d} \leq_p \boldsymbol{t}\},$$
$$\mathrm{Up}_S(T) \coloneqq \{\boldsymbol{d} \in S \mid \forall \boldsymbol{t} \in T, \boldsymbol{t} \leq_p \boldsymbol{d}\}.$$

The subset T is said to be $\leq_p$-closed in S if $\mathrm{Low}_S(T) = T$. We say that $\boldsymbol{d}'$ is a *cover* of $\boldsymbol{d} \neq \boldsymbol{d}'$ in S, written $\boldsymbol{d} \lessdot_p \boldsymbol{d}'$, if the existence of $\boldsymbol{e} \in S$ such that $\boldsymbol{d} \leq_p \boldsymbol{e} \leq_p \boldsymbol{d}'$ implies that $\boldsymbol{e} \in \{\boldsymbol{d}, \boldsymbol{d}'\}$. We also say that $\boldsymbol{d}$ is a *co-cover* of $\boldsymbol{d}'$. The set of covers (resp. co-covers) of $\boldsymbol{d}$ is denoted $\mathrm{Cover}_S(\boldsymbol{d}) \subseteq \mathrm{Up}_S(\boldsymbol{d})$ (resp. $\mathrm{Cocover}_S(\boldsymbol{d}) \subseteq \mathrm{Low}_S(\boldsymbol{d})$). Informally, covers and co-covers are nearest upper and lower bounds.

Assume that $S = [0,k]^m$. A rewriting of the degree set $D = \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m))$ is given by

$$D = \left\{ \boldsymbol{d} \in S \mid \forall \boldsymbol{e} \in \mathrm{Low}_S(\boldsymbol{d}), \underline{|\boldsymbol{e}|} \leq k \right\} . \tag{3.12}$$

Using Equation (3.12), the following lemma can be easily stated.

Lemma 3.55. *The degree set* $\mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m))$ *is* $\leq_p$*-closed in* $S = [0,k]^m$.

More generally, one must notice that this result can be seen as a consequence to the fact that $\mathrm{Aff}(\mathbb{F}_q^m) \subseteq \mathrm{Aut}(\mathrm{Lift}(\mathrm{RS}_q(k), m))$.

Lemma 3.56. *The degree set of every monomial affine-invariant code* $\mathcal{C}$ *is* $\leq_p$*-closed, where* p *is the characteristic of the base field.*

Proof. Original proof of this lemma can be found in a paper of Kaufman and Sudan [KS08]. □

A direct consequence of Lemma 3.55 is that, if some tuple $\boldsymbol{e} \notin \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m))$, then we have $\mathrm{Up}_S(\boldsymbol{e}) \cap \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m)) = \varnothing$. It naturally leads us to the design of Algorithm 12, which computes the degree set of affine lifted codes. Notice that Algorithm 12 makes use of a subroutine JoinUp given in Algorithm 11. This subroutine simply computes efficiently the union between $\mathrm{Up}_S(\boldsymbol{d})$, for some $\boldsymbol{d} \in S = [0,k]^m$, and a subset $C \subseteq S$ whose complementary set is $\leq_p$-closed. We also assume that:

- A function $\mathsf{Iterate}([0,k]^m)$ returns an iterator on elements $\boldsymbol{d} \in [0,k]^m$ which respects the partial order $\leq_p$. In other words, if $\boldsymbol{d}$ is returned before $\boldsymbol{d}'$, then it cannot hold that $\boldsymbol{d}' \leq_p \boldsymbol{d}$.
- A function $\mathsf{AReduce}(\boldsymbol{d})$ computes the A-reduction $\underline{|\boldsymbol{d}|}$. It can be done in time $\mathcal{O}(m \log^2 q)$.
- The set $\mathrm{Cover}_S(\boldsymbol{d})$ can be computed in $\mathcal{O}(em)$, where $q = p^e$. Since $|\mathrm{Cover}_S(\boldsymbol{d})| \leq em$, it corresponds to say that testing membership in the set S is $\mathcal{O}(1)$, which is realistic using hash tables lookup.

Algorithm 11: Procedure JoinUp which computes the union $C \cup \mathrm{Up}_S(\boldsymbol{d})$

Input: integers p, e, m, k such that $k \leq p^e - 1$, a set $C \subseteq S = [0,k]^m$, and a tuple $\boldsymbol{d} \in C$.
Output: the set $C \cup \mathrm{Up}_S(\boldsymbol{d})$.

```
T ← Cover_S(d)
for t ∈ T \ C do
    C ← C ∪ {t}
    C ← JoinUp(C, t, p, e, m, k)
return C
```

Thanks to (3.12), it is clear that Algorithm 12 computes correctly $\mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_q(k), m))$, since the algorithm essentially eliminates every tuple $\boldsymbol{d}$ that does not satisfy the needed requirements. Furthermore, Algorithm 12 has complexity $\mathcal{O}(emk^m) = \mathcal{O}(n \log(n))$, where n is the length of $\mathrm{Lift}(\mathrm{RS}_{p^e}(k), m)$, since every element $\boldsymbol{d} \in [0,k]^m$ is queried at most $\mathcal{O}(|\mathrm{Cover}_S(\boldsymbol{d})|) = \mathcal{O}(em)$ times.

Algorithm 12: Algorithm computing the degree set of $\mathrm{Lift}(\mathrm{RS}_{p^e}(k), m)$

Input: a prime p, and three integers e, m and k such that $0 \leq k \leq p^e - 1$
Output: the degree set of $\mathrm{Lift}(\mathrm{RS}_{p^e}(k), m)$

```
1  D ← ∅
2  C ← ∅                                   /* intended to store elements not in D */
3  for d ← Iterate([0,k]^m) do
4      if d ∉ C then
5          if AReduce(d) ≤ k then
6              D ← D ∪ {d}
7          else
8              C ← C ∪ {d}
9              C ← JoinUp(C, d, p, e, m, k)
10 return D
```

Remark 3.57. The study of degree sets is linked with the concept of defining set of cyclic codes. One must first notice that a code $\mathcal{C}$ invariant under $\mathrm{Aff}(\mathbb{F}_q^m)$ can be viewed as the extension of a cyclic code. Indeed, puncture $\mathcal{C}$ at $\mathbf{0} \in \mathbb{A}^m$. Then $\mathcal{C}' = \mathrm{Punct}(\mathcal{C}, \{\mathbf{0}\})$ is cyclic, since the group of affine transformation that fix $\mathbf{0}$ contains a cyclic subgroup.

Kasami, Lin and Peterson [KLP67] proved that, in the univariate case, every affine-invariant code $\mathcal{C}$ extending a primitive cyclic code can be described by a so-called *defining set* $I \subseteq [0, q-1]$. Originally, this defining set represents the exponents i of zeroes $\alpha^i \in \mathbb{F}_q$ of the polynomial representation of codewords in that lie in $\mathcal{C}$. In fact, it can be proved that, for such codes, $I = \mathrm{Deg}(\mathcal{C}^\perp)$. Without going into technicalities, using the Mattson-Solomon transform, one can write extended cyclic codes as evaluation codes (arguments can be found in Wolfmann's paper [Wol89]). Then, affine-invariance proves that these codes are monomial (or trace-monomial if the alphabet is a subfield of $\mathbb{F}_q$). Finally, analysing the degree sets with the help of Charpin's work [Cha90] provides the claim.

3.6.2 The case $m = 2$

In this subsection we focus particularly on the case $m = 2$, since it is the most simple and it gives the best code rates. More precisely, we describe the degree set of affine lifted codes $\mathrm{Lift}(\mathrm{RS}_{p^e}(k, 2))$ for prime powers p^e and $k \leq p^e - 1$.

We first need to introduce a few notation. For $0 \leq k \leq p^e - 1$, we define two particular kind of subsets of $[0, p^e - 1]^2$:

– subsets representing degree sets of affine lifted codes:

$$A_p(e, k) \coloneqq \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_{p^e}(k, 2)))\,;$$

– subsets representing degree sets of Reed-Muller codes:

$$R_p(e, k) \coloneqq \{(i, j) \in [0, p^e - 1]^2, i + j \leq k\}\,.$$

For an integer $0 \leq d < p^e$, we define

$$\Delta_p(d) \coloneqq \{d' \in [0, p^e - 1], d' \leq_p d\}\,,$$

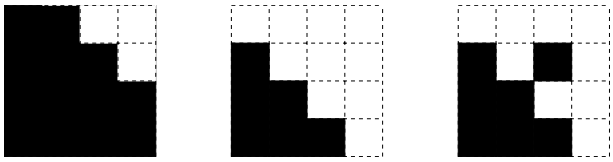


Figure 3.1 – Representation of degree sets $R_2(2,4)$, $R_2(2,2)$ and $A_2(2,2)$. The square $S_2 = [0, 2^2 - 1]^2$ is represented by a 4×4 grid. A black square at coordinates (i, j) means that (i, j) lies in the degree set.

and we extend this definition to m-tuples by $\Delta_p(\boldsymbol{d}) = \Delta_p(d_1) \times \cdots \times \Delta_p(d_m)$. For convenience, we denote by $\psi_e(\boldsymbol{d})$ the A-reduction of $|\boldsymbol{d}| \coloneqq \sum_i d_i$ when $q = p^e$, as defined in Subsection 3.1.2. Hence we can rephrase the degree set of affine lifted codes as follows:

$$A_p(e,k) = \{\boldsymbol{d} \in [0, p^e - 1]^2, \psi_e(\Delta_p(\boldsymbol{d})) \subseteq [0,k]\}\,.$$

Finally, we denote by $S_e = [0, p^e - 1]^2$. If $e \geq 2$, we partition the square S_e into p^2 subsquares $S_e^{(i,j)} = \{(ip^{e-1}, jp^{e-1})\} + S_{e-1}$, for $0 \leq i, j \leq p - 1$. Notice that here we denote by $A + B \coloneqq \{a + b, a \in A, b \in B\}$ the Minkowski sum of two subsets A and B of integer tuples.

In the $m = 2$ setting, we can have a visual representation of degree sets. It is illustrated in Figure 3.1. We see that degree sets $R_p(e,k)$ of Reed-Muller codes $\mathrm{RM}_{p^e}(2,k)$ are represented by discrete triangles if $k \leq p^e$, and generally by discrete polygons. Degree sets of strict affine lifted codes are more complicated to describe. This is the purpose of the following paragraphs.

Characteristic $p = 2$. For convenience we write $\Delta \coloneqq \Delta_2$. Our goal is to prove the following result.

Theorem 3.58. *Let $0 \leq k \leq 2^e - 1$. The degree set $A_2(e,k)$ of $\mathrm{Lift}(\mathrm{RS}_{2^e}(k), 2)$ satisfies:*

$$A_2(e,k) = R_2(e-1,k) \cup \left(\Delta((2^e, 2^e)) + A_2(e-1, k - 2^{e-1})\right).$$

In other words, we want to prove that

$$\begin{aligned}
A_2(e,k) \cap S_e^{(0,0)} &= R_2(e-1,k) \\
A_2(e,k) \cap S_e^{(1,0)} &= \{(2^{e-1}, 0)\} + A_2(e-1, k - 2^{e-1}) \\
A_2(e,k) \cap S_e^{(0,1)} &= \{(0, 2^{e-1})\} + A_2(e-1, k - 2^{e-1}) \\
A_2(e,k) \cap S_e^{(1,1)} &= \{(2^{e-1}, 2^{e-1})\} + A_2(e-1, k - 2^{e-1})
\end{aligned} \tag{3.13}$$

Figure 3.2 might give an overview of this result. Given the right-hand side degree set $A_2(4, 14)$, we see it can be decomposed according to four 8×8 subsquares $S_4^{(0,0)}$, $S_4^{(0,1)}$, $S_4^{(1,0)}$, and $S_4^{(1,1)}$. The lower-bottom one corresponds to the degree set $R_2(3, 14)$ of a Reed-Muller code, and the three others are equivalent to $A_2(3, 6)$, the degree set of a smaller lifted code (with the blue frame in Figure 3.2). This decomposition is recursive, as proves the red subsquares in the same figure.

Proof of Theorem 3.58. First notice that, if $k < 2^{e-1}$, then the theorem is proved thanks to the result of Kaufman and Ron [KR06] who essentially showed that $\mathrm{Lift}(\mathrm{RS}_q(k), m) = \mathrm{RM}(m, k)$ when $k < q - q/p$. So let us now assume that $k \geq 2^{e-1}$.

Then, we know that $(i,j) \in A_2(e,k)$ if and only if $(j,i) \in A_2(e,k)$. Hence the second and third equations in (3.13) are equivalent. We finally end our proof by stating Lemmata 3.59 and 3.61 which follow. □

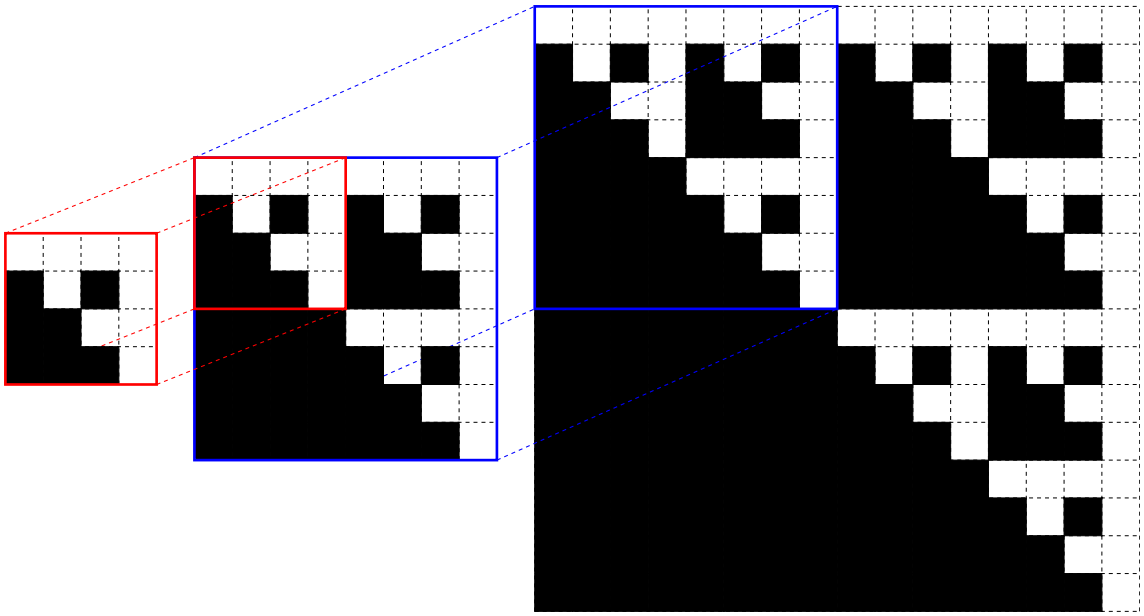


Figure 3.2 – Illustration of recursive patterns in degree sets. Here are represented $A_2(e, 2^e - 2) = \mathrm{Deg}(\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2), 2))$, for $e = 2, 3, 4$ respectively.

Lemma 3.59. *Let $2^{e-1} \leq k \leq 2^e - 1$. Then,*

$$A_2(e,k) \cap S_e^{(0,0)} = R_2(e-1,k)$$

Proof. If $\boldsymbol{u} \in A_2(e,k) \cap S_e^{(0,0)}$, then in particular $u_1 + u_2 = \psi_e(\boldsymbol{u}) \leq k$. In other words, $\boldsymbol{u} \in R_2(e-1,k)$. Conversely, if $\boldsymbol{u} \in R_2(e-1,k)$, then clearly $\boldsymbol{u} \in S_e^{(0,0)}$. Moreover, for any $\boldsymbol{y} \in \Delta(\boldsymbol{u})$, we have $\psi_e(\boldsymbol{y}) = y_1 + y_2 \leq u_1 + u_2 \leq k$. Hence $\boldsymbol{u} \in A_2(e,k)$. □

The first equality in (3.13) is therefore proved. We need a preliminary result for the three other statements.

Lemma 3.60. *Let $2^{e-1} \leq k \leq 2^e - 1$, $\boldsymbol{u} \in S_e^{(0,0)}$ and $\boldsymbol{d} \in \{(2^{e-1}, 0), (2^{e-1}, 2^{e-1})\}$. Then, the following assertions are equivalent:*

(i) $\psi_e(\Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d}))) \subseteq [0,k]$,
(ii) $\psi_{e-1}(\Delta(\boldsymbol{u})) \subseteq [0, k - 2^{e-1}]$.

Proof.

$(i) \Rightarrow (ii)$. Let $\boldsymbol{v} \in \Delta(\boldsymbol{u})$. If $v_1 + v_2 < 2^{e-1}$, then:

$$\psi_{e-1}(\boldsymbol{v}) = v_1 + v_2 = 2^{e-1} + v_1 + v_2 - 2^{e-1} = \psi_e(\underbrace{\boldsymbol{v} + (2^{e-1}, 0)}_{\in \Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d})}) - 2^{e-1} \leq k - 2^{e-1}.$$

Otherwise,

$$\begin{aligned}\psi_{e-1}(\boldsymbol{v}) &= v_1 + v_2 - (2^{e-1} - 1) \leq (2^{e-1} - 1) + v_2 - (2^{e-1} - 1) = (2^{e-1} + v_2) - 2^{e-1} \\ &= \psi_e(\underbrace{(0, v_2) + (2^{e-1}, 0)}_{\in \Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d})}) - 2^{e-1} \leq k - 2^{e-1}.\end{aligned}$$

$(ii) \Rightarrow (i)$. Let $\boldsymbol{v} + \boldsymbol{w} \in \Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d})$. If $v_1 + v_2 < 2^{e-1}$, we first claim that $\psi_e(\boldsymbol{v} + \boldsymbol{w}) \leq v_1 + v_2 + 2^{e-1}$. It is clear if $\boldsymbol{w} = \boldsymbol{0}$. If $|\boldsymbol{w}| = 2^{e-1}$, then $\psi_e(\boldsymbol{v} + \boldsymbol{w}) = v_1 + v_2 + 2^{e-1}$. Finally, if $|\boldsymbol{w}| = 2^e$, then $\psi_e(\boldsymbol{v} + \boldsymbol{w}) = v_1 + v_2 + 1 \leq v_1 + v_2 + 2^{e-1}$. Hence,

$$\psi_e(\boldsymbol{v} + \boldsymbol{w}) \leq v_1 + v_2 + 2^{e-1} = \psi_{e-1}(\boldsymbol{v}) + 2^{e-1} \leq k - 2^{e-1} + 2^{e-1} \leq k\,.$$

If $v_1 + v_2 \geq 2^{e-1}$, then the result holds if $\boldsymbol{w} = \boldsymbol{0}$. So assume $\boldsymbol{w} \neq \boldsymbol{0}$. Then,

$$\begin{aligned}\psi_e(\boldsymbol{v} + \boldsymbol{w}) &= v_1 + v_2 + w_1 + w_2 - (2^e - 1) \leq v_1 + v_2 + 1 \\ &\leq \psi_{e-1}(\boldsymbol{v}) + 2^{e-1} - 1 + 1 \leq k - 2^{e-1} + 2^{e-1} \leq k\,.\end{aligned}$$

□

Lemma 3.61. *Let* $2^{e-1} \leq k \leq 2^e - 1$. *The last three statements in* (3.13) *hold.*

Proof. Let $\boldsymbol{u} \in A_2(e-1, k-2^{e-1})$ and $\boldsymbol{d} \in \{(2^{e-1}, 0), (2^{e-1}, 2^{e-1})\}$. Lemma 3.60 proves that, for every $\boldsymbol{v} \in \Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d})$, we have $\psi_e(\boldsymbol{v}) \in [0, k]$. Hence $\boldsymbol{u} + \boldsymbol{d} \in A_2(e, k)$.

Let now $\boldsymbol{w} \in A_2(e, k)$, and assume $\boldsymbol{w} \notin S_e^{(0,0)}$. Write uniquely $\boldsymbol{w} = \boldsymbol{u} + \boldsymbol{d}$ where $\boldsymbol{u} \in S_2^{(0,0)}$ and $\boldsymbol{b} \in \Delta((2^{e-1}, 2^{e-1})) \setminus \{\boldsymbol{0}\}$. Thanks to symmetry in the problem, without loss of generality we can assume that $\boldsymbol{d} \in \{(2^{e-1}, 0), (2^{e-1}, 2^{e-1})\}$. We need to prove that $\Delta(\boldsymbol{u}) \subseteq A_2(e-1, k-2^{e-1})$. Notice that $\psi_e(\Delta(\boldsymbol{u}) + \Delta(\boldsymbol{d})) \subseteq [0, k]$ since $\boldsymbol{u} + \boldsymbol{d} = \boldsymbol{w} \in A_2(e, k)$. Hence Lemma 3.60 shows that $\psi_{e-1}(\Delta(\boldsymbol{u})) \subseteq [0, k - 2^{e-1}]$, proving the result we claim. □

Since the cardinality of the degree set of a code equals its dimension, we get the following corollary.

Corollary 3.62. *Let* $0 \leq k \leq 2^e - 1$. *The dimension* $|A_2(e, k)|$ *of* $\mathrm{Lift}(\mathrm{RS}_{2^e}(k), 2)$ *satisfies the recursive relation:*

$$|A_2(e, k)| = |R_2(e-1, k)| + 3|A_2(e-1, k-2^{e-1})|\,,$$

where by convention, $A_2(e-1, k) = \varnothing$ *if* $k < 0$.

We now consider the case $k = 2^e - 2^{e-c} - 1$, for some $1 \leq c \leq e-1$. In other words, it corresponds to setting the information rate of the underlying Reed-Solomon code to 2^{-c}. In Proposition 3.63, we state an exact formula for the dimension of the associated affine lifted code. This new result improves upon Guo *et al.*'s Claim 3.19 in [GKS13], which was only a lower bound on the dimension of the lifted code.

Proposition 3.63. *For every* $1 \leq c \leq e-1$. *Let* $k_{e,c} = 2^e - 2^{e-c} - 1$. *Then,*

$$\dim(\mathrm{Lift}(\mathrm{RS}_{2^e}(k_{e,c}), 2)) = 4^e \left(1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c + \frac{1}{2^e}\left(\frac{3^c - 1}{2^{c+2}}\right)\right).$$

Proof. Let us fix $c \leq e$. For every $0 \leq i \leq c$, define $a_i := |A_2(e-i, 2^{e-i} - 2^{e-c} - 1)|$ and $r_i := |R_2(e-1-i, 2^{e-i} - 2^{e-c} - 1)|$. Previous corollary translates into

$$a_0 - 3a_1 = r_0\,,$$

and more generally we have

$$a_i - 3a_{i+1} = r_i$$

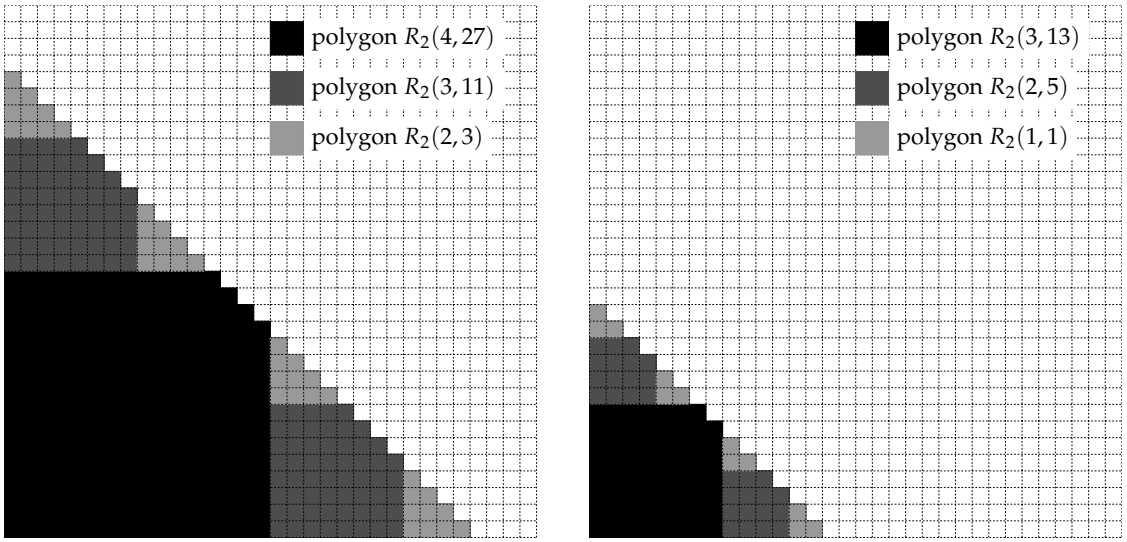


Figure 3.3 – A typical example of a triangle $R_2(e-k, 2^{e-k}-2^{e-c}-1)$ that can be partitioned into polygons of the shape $R_2(e-k-i-1, 2^{e-k-i}-2^{e-c}-1)$, for $0 \leq i \leq c-k-1$. The instances correspond to $(e,c,k) = (5,3,0)$ (on the left), and $(e,c,k) = (5,4,1)$ (on the right).

for every $0 \leq i \leq c$. Notice that $a_c = 0$ and $r_c = 0$, and recall that we look for a closed formula for a_0. Therefore, it is useful to notice that:

$$\sum_{i=0}^{c-1} 3^i r_i = \sum_{i=0}^{c-1} 3^i (a_i - 3a_{i+1}) = a_0 - 3^c a_c = a_0 \,.$$

We now claim that, for every $0 \leq k \leq c-1$, we have

$$\sum_{i=0}^{c-k-1} 2^i r_{i+k} = t_k := \binom{2^{e-k} - 2^{e-c} + 1}{2} \,.$$

This can be seen by decomposing the triangle $R_2(e-k, 2^{e-k}-2^{e-c}-1)$ into polygons of the shape $R_2(e-k-i-1, 2^{e-k-i}-2^{e-c}-1)$ (see Figure 3.3). Therefore we get

$$\begin{aligned}
\sum_{i=0}^{c-1} 3^i r_i &= \sum_{i=0}^{c-1} 2^i r_i + \sum_{i=0}^{c-1} (3^i - 2^i) r_i = t_0 + \sum_{i=1}^{c-1} \left(\sum_{k=0}^{i-1} 3^k 2^{i-1-k} \right) r_i \\
&= t_0 + \sum_{k=0}^{c-1} \sum_{i=k+1}^{c-1} 3^k 2^{i-1-k} r_i = t_0 + \sum_{k=0}^{c-1} 3^k \sum_{i=0}^{c-k-2} 2^i r_{i+k+1} \qquad (3.14) \\
&= t_0 + \sum_{k=0}^{c-1} 3^k t_{k+1} \,.
\end{aligned}$$

Then,

$$\begin{aligned}
2 \sum_{k=0}^{c-1} 3^k t_{k+1} &= \sum_{k=0}^{c-1} 3^k (2^{e-k-1} - 2^{e-c} + 1)(2^{e-k-1} - 2^{e-c}) \\
&= \sum_{k=0}^{c-1} 3^k (4^{e-k-1} + 4^{e-c} - 2^{2e-k-c} + 2^{e-k-1} - 2^{e-c}) \\
&= 4^{e-1} \sum_{k=0}^{c-1} (3/4)^k + (4^{e-c} - 2^{e-c}) \sum_{k=0}^{c-1} 3^k + (2^{e-1} - 2^{2e-c}) \sum_{k=0}^{c-1} (3/2)^k \\
&= 4^e - 3^c 4^{e-c} + \frac{(4^{e-c} - 2^{e-c})(3^c - 1)}{2} + 2^{e-c} 3^c - 2^e - 2^{2e-2c+1} 3^c + 2^{2e-c+1} \,.
\end{aligned}$$

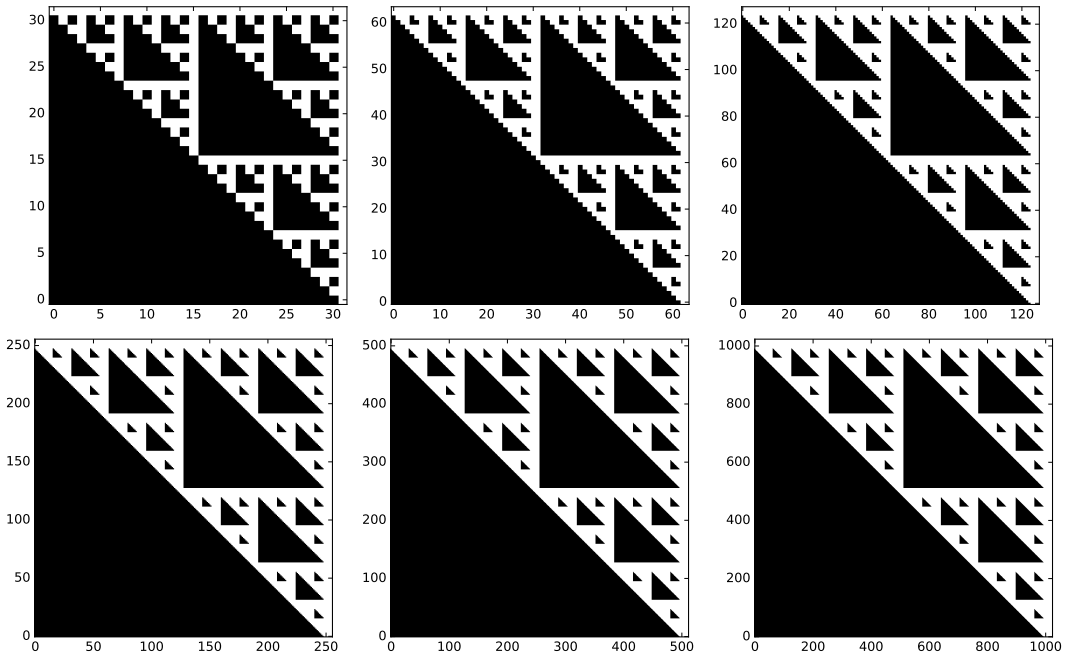


Figure 3.4 – Illustration of the sequence of degree sets $A_2(e, 2^e - 2^{e-c} - 1)$, for $c = 5$ and $e = 5, \ldots, 10$. Recall that a black square at coordinate (i, j) means that (i, j) lies is the degree set. For increasing e we observe that the black shape converges to a finite union of triangles.

Hence,

$$\begin{aligned} 2a_0 &= 2t_0 + 2\sum_{k=0}^{c-1} 3^k t_{k+1} = (2^e - 2^{e-c} + 1)(2^e - 2^{e-c}) + 2\sum_{k=0}^{c-1} 3^k t_{k+1} \\ &= 4^e + 4^{e-c} - 2^{2e-c+1} + 2^e - 2^{e-c} + 4^e - 3^c 4^{e-c} \\ &\quad + \frac{(4^{e-c} - 2^{e-c})(3^c - 1)}{2} + 2^{e-c}3^c - 2^e - 2^{2e-2c+1}3^c + 2^{2e-c+1} \\ &= 4^e \cdot (2 + 2^{-1}4^{-c} - 5 \cdot 2^{-1}3^c 4^{-c}) + 2^e \cdot (3^c 2^{-c-1} - 2^{-c-1}) \,. \end{aligned}$$

Finally,

$$a_0 = 4^e \left(1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c + \frac{1}{2^e}\left(\frac{3^c - 1}{2^{c+2}}\right)\right) . \qquad \square$$

Remark 3.64. We know that $\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2), 2)$ is equivalent to $\mathrm{Code}_{2^e}(\mathrm{AG}_1(2, 2^e))$. Taking $c = e$ in the last corollary, we get the well-known result concerning the dimension of affine geometry design-based codes: for every $e \geq 2$, we have

$$\dim(\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2), 2)) = 4^e - 3^e \,.$$

As we said previously, it is claimed in [GKS13] that

$$\dim(\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2^{e-c} - 1), 2)) \geq 4^e \left(1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c\right) .$$

We can notice their lower bound corresponds to the asymptotic dimension of the family of codes $\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2^{e-c} - 1), 2)$, when $e \to \infty$ and c is fixed.

Let us propose an interpretation to this observation. In Figure 3.4, we represent the evolution of $\dim(\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2^{e-c} - 1), 2))$ when e grows. It is reasonable to conjecture that the rate

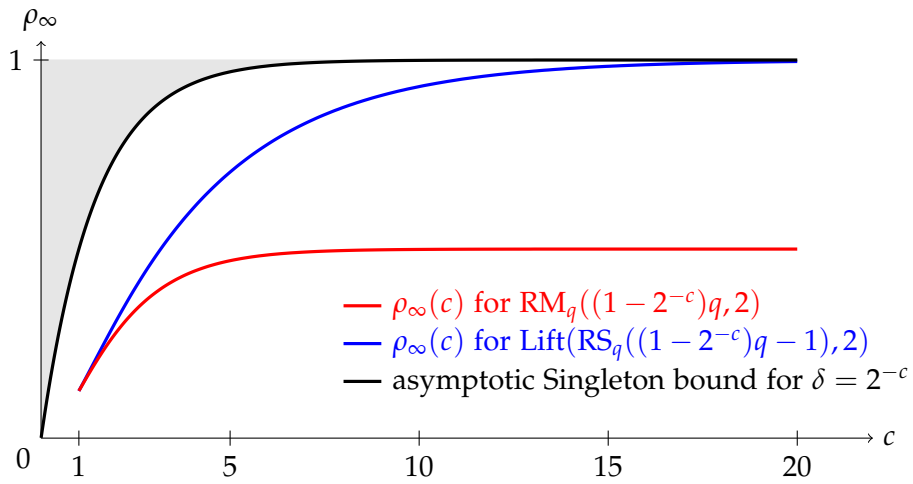


Figure 3.5 – Asymptotic rates of Reed-Muller codes $\mathrm{RM}_q((1-2^{-c})q, 2)$ and affine lifted codes $\mathrm{Lift}(\mathrm{RS}_q((1-2^{-c})q-1), 2)$, with comparable local correcting capabilities. We make $e \to \infty$, where $q = p^e$. The black curve represents the asymptotic Singleton bound for codes with relative distance 2^{-c}, corresponding approximately to the relative distance of both Reed-Muller and lifted codes.

of the code tends to the (finite) sum of the area of triangles that show up. To prove this, it is sufficient to see that $\lim_{e\to\infty}(t_k/4^{e-k}) = (1-2^{k-c})^2/2$, where $t_k := \binom{2^{e-k}-2^{e-c}+1}{2}$ is defined in the proof of Proposition 3.63. Furthermore, if c is fixed, we can make e tend to infinity in the equality

$$4^{-e}a_0 = 4^{-e}\left(t_0 + \sum_{k=0}^{c-1} 3^k t_{k+1}\right)$$

that we derive from (3.14). Formally denote by

$$\rho_\infty(c) := \lim_{e\to\infty} \frac{\dim(\mathrm{Lift}(\mathrm{RS}_{2^e}(2^e - 2^{e-c} - 1), 2))}{4^e}$$

the asymptotic rate of the family of affine lifting of Reed-Solomon codes of rate 2^{-c}. Then we get

$$\rho_\infty(c) = \lim_{e\to\infty}(4^{-e}a_0) = \frac{(1-2^{-c})^2}{2} + \frac{1}{4}\sum_{k=0}^{c-1}\left(\frac{3}{4}\right)^k \frac{(1-2^{k+1-c})^2}{2}.$$

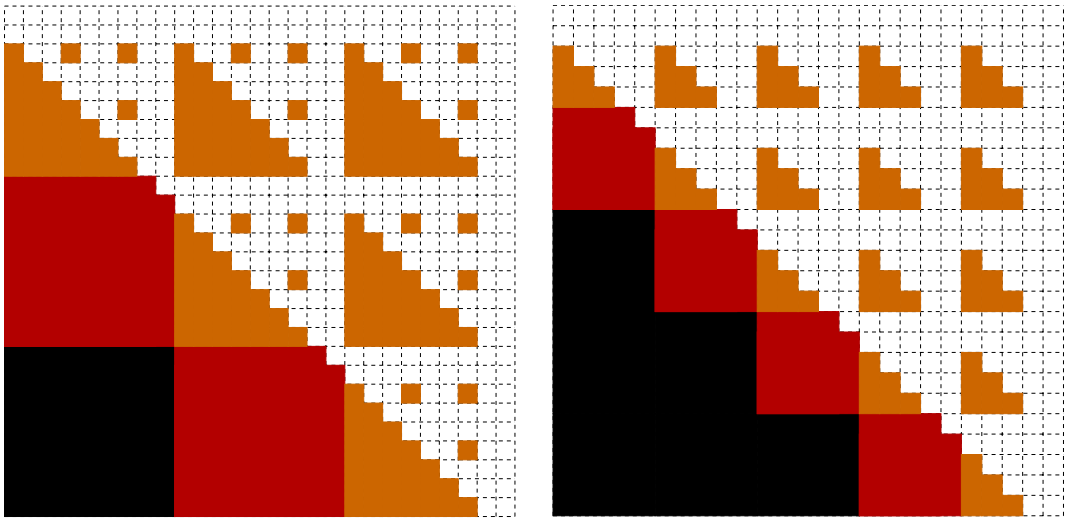
An easy computation then shows that

$$\rho_\infty(c) = 1 - \frac{5}{4}\left(\frac{3}{4}\right)^c + \frac{1}{4}\left(\frac{1}{4}\right)^c.$$

In Figure 3.5 we depict the asymptotic rate of lifted codes and Reed-Muller codes, for the setting $m = 2$, $p = 2$, $e \to \infty$. We see that lifted codes outperform Reed-Muller codes, and that their rate tends to 1 when we let their relative distance 2^{-c} tend to 0.

Characteristic $p \neq 2$. Odd characteristic makes recursive relations between $A_p(\cdot,\cdot)$ and $R_p(\cdot,\cdot)$ more complex, since $S_e = [0, p^e - 1]^2$ must be split into more subsquares $S_e^{(i,j)}$. By analogy with the even characteristic setting, we claim that

$$A_p(e,k) \cap S_e^{(i,j)} = \begin{cases} S_e^{(i,j)} & \text{if } i+j \leq p-3 \\ \{(ip^{e-1}, jp^{e-1})\} + R_p(e-1, k-(p-2)p^{e-1}) & \text{if } i+j = p-2 \\ \{(ip^{e-1}, jp^{e-1})\} + A_p(e-1, k-(p-1)p^{e-1}) & \text{if } i+j \geq p-1 \end{cases} \quad (3.15)$$



$A_3(3,24)$ $A_5(2,22)$

Figure 3.6 – Degree sets $A_p(e,k)$ of $\mathrm{Lift}(\mathrm{RS}_{p^e}(k),2)$ in odd characteristic p. On the left-hand side, $(p=3, e=3, k=24)$; on the right-hand side, $(p=5, e=2, k=22)$. In black, we plot regions $S_e^{(i,j)}$ for $i+j \leq p-3$; in red, regions of the form $R_p(e-1, k-(p-2)p^{e-1})$; in orange, regions of the form $A_p(e-1, k-(p-1)p^{e-1})$ which are defined recursively.

A full proof of this result would be very technical to state, but very similar to the one of Theorem 3.58. Instead, we give in Figure 3.6 an overview of the structure of the degree sets for small values of p, e and k.

Using recursive relations given in (3.15), we get

$$\begin{aligned} |A_p(e,k)| = &\binom{p-1}{2} \cdot |S_e^{(0,0)}| \\ &+ (p-1) \cdot |R_p(e-1, k-(p-2)p^{e-1})| \\ &+ \binom{p}{2} \cdot |A_p(e-1, k-(p-1)p^{e-1})| \end{aligned}$$

As previously, we consider the case $k = p^e - p^{e-c} - 1$, corresponding to lifting Reed-Solomon codes of rate p^{-c}. We also denote by $a_i := |A_p(e-i, p^{e-i} - p^{e-c} - 1)|$ and by $r_i := |R_p(e-1-i, 2p^{e-i-1} - p^{e-c} - 1)|$. With arguments very similar to the proof of Proposition 3.63, one can then show that

$$a_i - \binom{p}{2} a_{i+1} = \binom{p-1}{2} p^{2(e-i-1)} + (p-1) r_i, \quad 0 \leq i \leq c-1\,.$$

We then obtain

$$a_0 = \sum_{i=0}^{c-1} \binom{p}{2}^i \left(\binom{p-1}{2} p^{2(e-i-1)} + (p-1) r_i \right),$$

which leads us, after quite technical computations, to

$$|A_p(e,k)| = a_0 = t_0 + \binom{p}{2} \sum_{k=0}^{c-1} \binom{p+1}{2}^k t_{k+1}\,,$$

where $t_k := \binom{p^{e-k} - p^{e-c} + 1}{2}$.

Asymptotic analysis. Recall that $\lim_{e\to\infty}(t_k/p^{2(e-k)}) = (1-p^{k-c})^2/2$. A quite tedious computation then gives

$$\rho_\infty^{(p)}(c) = \lim_{e\to\infty}(a_0/p^{2e}) = 1 - \left(1+\frac{1}{p+2}\right)\left(\frac{1+1/p}{2}\right)^c + \frac{1}{p+2}\left(\frac{1}{p^2}\right)^c .$$

One can first check that the above formula matches with the even characteristic setting. Moreover, we see that lifted codes over fields of odd characteristic also produce locally correctable codes with arbitrarily large rate.

Part II

Cryptographic applications

Chapter 4

Private information retrieval from transversal designs

Contents

A private information retrieval (PIR) protocol aims reassuring a user that he can retrieve some entry M_i of a remote database M without revealing the index i to the server(s) holding the database. In this chapter, we propose a specific encoding of the database which yields PIR protocols with reasonable communication complexity, low storage overhead and optimal computational complexity for the servers. This encoding is based on incidence matrices of transversal designs, from which a natural and efficient recovering algorithm is derived (Section 4.2). We also present practical instances of the construction in Section 4.3, making use of finite geometries and orthogonal arrays. Furthermore, we give a generalisation of the construction in order to resist collusions of servers (Section 4.4). Finally, we review very recent constructions and bounds on PIR protocols to conclude the chapter.

4.1 Private information retrieval

4.1.1 PIR models

Two kinds of security models for PIR have been introduced so far. In [CGKS95, CKGS98], Chor, Goldreich, Kushilevitz and Sudan considered so-called *information-theoretic PIR* (IT-PIR): one requires that absolutely no information on i is leaked to the servers when querying the database. However, the authors proved that in this model, if only one server stores the database M, then one must download at least $\Omega(|M|)$ bits in order to retrieve a single entry of the database. A second security model was proposed a few years later by Chor and Gilboa [CG97], named *computational PIR (CPIR)*. Here, one requires that it is computationally hard for the server to recover any bit of information on i. Notice that, in practice, CPIR constructions are much less efficient than IT-PIR constructions.

We situate our work in the information-theoretic PIR (IT-PIR) model. Throughout the chapter, a user U is the owner of a database $M = (M_1, \ldots, M_K)$ composed of messages M_i lying in some message space $\mathcal{M}$. A system of ℓ servers $S_1, \ldots, S_\ell$ is at his disposal, and stores possibly encoded versions of the messages. The goal of U is to retrieve some message M_i, for $1 \leq i \leq K$, without revealing any information about i to the servers.

In the first model given by Chor, Goldreich, Kushilevitz and Sudan [CGKS95, CKGS98], the user simply replicates M on the servers.

Definition 4.1 (replication-based PIR protocol)**.** We assume that every server S_j, $1 \leq j \leq \ell$, stores a copy of the database $M = (M_1, \ldots, M_K)$. An ℓ-server replication-based PIR protocol is a set of three algorithms $(\mathsf{Query}, \mathsf{Ans}, \mathsf{Rec})$ running the following steps on input $i \in [1, K]$:

1. *Query generation.* The randomised algorithm Query generates ℓ queries
$$(q_1, \ldots, q_\ell) \leftarrow_{\mathrm{R}} \mathsf{Query}(i)\,.$$
Query q_j is sent to server S_j.
2. *Servers' answer:* Each server S_j computes an answer
$$a_j \leftarrow \mathsf{Ans}_j(q_j, M)$$
and sends it back to U.
3. *Reconstruction:* Denote by $\boldsymbol{a} = (a_1, \ldots, a_\ell)$ and $\boldsymbol{q} = (q_1, \ldots, q_\ell)$. User U computes and outputs
$$R \leftarrow \mathsf{Rec}(i, \boldsymbol{a}, \boldsymbol{q})\,.$$

The PIR protocol is said to be:

- *correct* if $R = M_i$ when the servers follow the protocol;
- *t-private* if, for every $1 \leq i, i' \leq K$ and every $T \subseteq [1, \ell]$ such that $|T| \leq t$, the distributions $\mathsf{Query}(i)_{|T}$ and $\mathsf{Query}(i')_{|T}$ are the same. We also say that the PIR protocol resists a collusion of t servers.

In practice, it is usual that the answer of server S_j does not depend on j; in that case we simply write $\mathsf{Ans}(q_j, M)$. We call *communication complexity* the total number of bits sent between the user and the servers during one iteration of the protocol. The *server* (resp. *user*) *computational complexity* is the maximal number of operations in the underlying field made by a server in order to compute an answer a_j (resp. made by Rec to reconstruct the desired item). The *storage rate* of the protocol is the ratio between the size $|M|$ of the database, and the size of what is actually stored on the servers. For replication-based protocols, it is then $1/\ell$.

A way to reduce the storage overhead of a PIR protocol is to preprocess the database in order to distribute it smartly among the servers. From now on, we denote by C an encoded version of M, that can be split into ℓ parts $C_1, \ldots, C_\ell$, where C_j generally lies in some set $\mathcal{C}_j$. One can consider that the map $M \mapsto C = (C_1, \ldots, C_\ell)$ is performed by the user, and the part C_j is then uploaded on server S_j. This leads us to the definition of distributed PIR protocols.

Definition 4.2 (coded, or distributed PIR protocol)**.** Assume that for $1 \leq j \leq \ell$, server S_j holds the part C_j of an encoded version C of the database $M = (M_1, \ldots, M_K)$. An ℓ-server distributed PIR protocol is a set of three algorithms $(\mathsf{Query}, \mathsf{Ans}, \mathsf{Rec})$ running the following steps on input $i \in [1, K]$:

1. *Query generation.* The randomised algorithm Query generates ℓ queries
$$(q_1, \ldots, q_\ell) \leftarrow_{\mathrm{R}} \mathsf{Query}(i)\,.$$
Query q_j is sent to server S_j.
2. *Servers' answer.* Each server S_j computes an answer $a_j \leftarrow \mathsf{Ans}_j(q_j, C_j)$ and sends it back to the user.
3. *Reconstruction.* Denote by $\boldsymbol{a} = (a_1, \ldots, a_\ell)$ and $\boldsymbol{q} = (q_1, \ldots, q_\ell)$. User U computes and outputs
$$R \leftarrow \mathcal{R}(i, \boldsymbol{a}, \boldsymbol{q})\,.$$

Correctness and privacy are defined identically to Definition 4.1.

Communication and computational complexities, as well as storage rate, are defined similarly to the replication-based model. Furthermore, we see that the definition of replication-based PIR protocols is included in Definition 4.2, by means of an encoding map
$$M \mapsto (C_1 = M, \ldots, C_\ell = M)\,.$$

There exists a basic replication-based PIR protocol that uses only $\ell = 1$ server: it simply consists in downloading M entirely. This protocol will be called the *trivial* PIR protocol. It is clear that it is private, but it also admits a huge download complexity. Unfortunately, Chor *et al.* proved that, in the case $\ell = 1$, this protocol is essentially optimal in terms of communication complexity [CKGS98]. Precisely, one cannot achieve information-theoretic privacy with only 1 server, expect by downloading $\Omega(|M|)$ bits. So from now on we assume $\ell \geq 2$.

4.1.2 First constructions, and links with locally decodable codes

In their seminal paper, Chor *et al.* [CKGS98] proposed a first PIR protocol involving ℓ servers with a total communication complexity $\mathcal{O}(\ell \log(\ell) K^{1/\log(\ell)})$ bits (each message M_i has size 1 bit). Their construction relies on a smart arrangement of bits in M in a $\log(\ell)$-dimensional binary array. Then, the user queries sum of bits supported by random Cartesian products in order to retrieve the desired bit.

A few years later, Katz and Trevisan [KT00] showed that any smooth locally decodable code $\mathcal{C} \subseteq \mathbb{F}_q^n$ of locality ℓ gives rise to an ℓ-server PIR protocol whose communication complexity is essentially $\ell \log(nq)$. Building on this idea, many PIR constructions (notably [BIKR02, Yek08, Efr12, DG16]) successively decreased the communication complexity, achieving $\mathcal{O}(K^{\sqrt{\log\log K / \log K}})$ bits with only $\ell = 2$ servers.

In fact, the definition of PIR in [KT00] allows the protocol to fail with small probability $\varepsilon > 0$, which corresponds to the failure probability of the underlying LDCs. In this thesis we choose

to use the most-common definition of PIR, which does not allow retrieval failure. Hence we need to replace LDCs by (ℓ, ℓ)-local reconstructing algorithms, as defined in the paragraph about expander codes in Subsection 2.3.1.

So, let $\mathcal{C} \subseteq \Sigma^X$ be a code of dimension K equipped with an (ℓ, ℓ)-local reconstructing algorithm (Q, R), and denote by E a systematic encoder for $\mathcal{C}$. Assume the database $M = (M_1, \ldots, M_K)$ is replicated over ℓ servers. Let also $x \in X$ such that $M_i = E(M)_x$ (we recall that M_i is the message to be retrieved privately). The PIR protocol based on (Q, R) can be described as follows:

- *Query generation.* User U calls the query generator Q on input x. It produces a subset $Q = \{q_1, \ldots, q_\ell\} \subseteq X$. Query q_j is sent to server S_j.
- *Servers' answer.* Each server S_j computes the encoding $E(M)_{q_j}$ and returns it to U. In other words,
$$\mathsf{Ans}(q_j, M) = E(M)_{q_j}\,.$$
- *Reconstruction.* User U collects $\boldsymbol{a} = (E(M)_{q_1}, \ldots, E(M)_{q_\ell})$, and feeds R with $\boldsymbol{a}$ and Q. Then U outputs $\mathsf{Rec}(i, \boldsymbol{a}, \boldsymbol{q}) = \mathsf{R}(\boldsymbol{a}, Q)$.

By definition of local reconstructing algorithms, each individual query of $\mathsf{Q}(i)$ is uniformly distributed in X. Therefore, the servers get no information about i. Moreover, the output is correct as long as the servers' answers are correct.

Remark 4.3. Most known perfectly smooth LDCs and LCCs of locality ℓ are actually (ℓ, ℓ)-local reconstructing algorithms. This is the case for all the LCCs we presented in Chapters 2 and 3. Hence, for convenience and conformity with the literature, we will say that protocols presented above are *LDC-based PIR protocols*.

Remark 4.4. The construction of Chor *et al.* [CKGS98] can also be seen as an LDC-based PIR protocol. The underlying LDC is the evaluation code of $\log(\ell)$-linear forms over the space $\mathbb{F}_2^m$ where $m = K^{1/\log(\ell)}$.

The main drawback of LDC-based PIR protocols is their computational cost: each server must compute a new symbol of $E(M)$ for each run of the protocol. The cost of this computation is usually $\Omega(|M|)$. Preprocessing the encoding can reduce this cost, but in this case, each server has to store the encoding $E(M)$. Therefore the induced PIR storage rate is $R_{\mathcal{C}}/\ell$, where $R_{\mathcal{C}}$ is the rate of the code $\mathcal{C}$. For small ℓ, this quantity is impractical since we have seen in Chapter 2 that constant locality LDCs have vanishing rate.

Aiming at reducing this storage overhead, Augot, Levy-dit-Vehel and Shikfa [ALS14] proposed to benefit from a natural partition of the evaluation support of multiplicity codes defined in [KSY14]. In a more generic setting, assume that each codeword $\boldsymbol{c} \in \mathcal{C}$ can be *split* into ℓ disjoint parts $\boldsymbol{c}^{(1)}, \ldots, \boldsymbol{c}^{(\ell)}$, such that each coordinate q_j of any possible query $\boldsymbol{q} = (q_1, \ldots, q_\ell)$ of the PIR protocol corresponds to some symbols on the piece $\boldsymbol{c}^{(j)}$. Defining $C_j := \boldsymbol{c}^{(j)}$, we obtain a distributed PIR protocol as in Definition 4.2. The PIR protocols we present in Sections 4.2 and 4.4 should be seen in the spirit of Augot *et al.*'s construction. Indeed we propose a generalisation of this *support partition* idea, which makes use of transversal designs, and achieves *small storage overhead* and *low computational complexity* for both user and servers.

To further emphasize the relevance of our work, we point out that most current PIR constructions admit a large computational complexity (see Section 4.5). In the context of a system storing a large amount of files accessed very frequently, one could question the practicality of such schemes. Especially, in a survey [Yek12] Yekhanin brings attention to this issue,

and claims that 'the overwhelming computational complexity of PIR schemes (...) currently presents the main bottleneck to their practical deployment'.

4.2 A 1-private protocol based on transversal designs

In the upcoming constructions, the database can be viewed as a message $\boldsymbol{m} \in \mathbb{F}_q^k$, and the user wants to retrieve its entry m_i for $1 \leq i \leq k$.

4.2.1 Transversal designs

We refer to Section 1.3 for basics on design-based codes. We also recall that 2-designs give rise to perfectly smooth locally correctable codes. Therefore it is natural to use them in the LDC-based PIR construction. In fact, since we want to use the *splitting* technique of Augot *et al.* [ALS14], we can relax a few constraints on the incidence between points and blocks. This leads us to introduce *transversal designs*.

Definition 4.5 (transversal design)**.** Let $s, \ell \geq 1$ and $\lambda \geq 1$ be integers. A *transversal design*, denoted $\mathrm{TD}_\lambda(\ell, s)$, is a block design $(X, \mathcal{B})$ equipped with a partition $\mathcal{G} = \{G_1, \ldots, G_\ell\}$ of X called the set of *groups*, such that:

- $|X| = \ell s$;
- every group in $\mathcal{G}$ has size s and every block in $\mathcal{B}$ has size ℓ;
- every unordered pair of elements from X is contained either in exactly one group, or in exactly λ blocks.

When $\lambda = 1$, we use the simpler notation $\mathrm{TD}(\ell, s)$.

Remark 4.6. In a transversal design, a block and a group intersect in a set of size at most 1, otherwise the third condition of the definition would be disproved. Moreover, since the block size equals the number of groups, any block must meet any group. Hence the following holds:

$$\forall (B, G) \in \mathcal{B} \times \mathcal{G},\ |B \cap G| = 1\,.$$

The definition also implies that there must lie exactly λs^2 blocks in $\mathcal{B}$. To prove this, let S be the set of unordered pairs $\{x, y\} \subseteq X$ which are contained in a block $B \in \mathcal{B}$. Notice that S contains exactly $\binom{s\ell}{2} - \ell\binom{s}{2}$ elements: there are $\binom{s\ell}{2}$ unordered pairs in X, but for each group $G \in \mathcal{G}$, we need to remove $\binom{s}{2}$ pairs contained in G. On the one hand, we have

$$N := \sum_{\{x,y\} \in S} \sum_{B \in \mathcal{B}} \mathbb{1}_{\{x,y\} \subset B} = \sum_{\{x,y\} \in S} \lambda = \lambda \left(\binom{s\ell}{2} - \ell\binom{s}{2} \right) = \lambda s^2 \binom{\ell}{2}\,.$$

by definition of transversal designs. On the other hand,

$$N = \sum_{\{x,y\} \in S} \sum_{B \in \mathcal{B}} \mathbb{1}_{\{x,y\} \subset B} = \sum_{B \in \mathcal{B}} \sum_{\{x,y\} \in S} \mathbb{1}_{\{x,y\} \subset B} = \sum_{B \in \mathcal{B}} \binom{\ell}{2} = |\mathcal{B}| \binom{\ell}{2}\,,$$

which proves our claim.

Example 4.7. Let us define a transversal design as follows. The points X are those of the affine plane $\mathbb{A}^2(\mathbb{F}_3)$. Any set of three pairwise disjoint affine lines of $\mathbb{A}^2(\mathbb{F}_3)$ define the set of groups $\mathcal{G}$. For instance, one can consider

$$\mathcal{G} = \{\, \{(0,0),(0,1),(0,2)\},\ \{(1,0),(1,1),(1,2)\},\ \{(2,0),(2,1),(2,2)\}\,\}\,.$$

The other 9 affine lines define the blocks $\mathcal{B}$. The design $\mathcal{T} = (X, \mathcal{B}, \mathcal{G})$ we obtain is a transversal design TD$(3,3)$. Indeed, $\mathcal{T}$ is composed of $\ell s = 9$ points, $\ell = 3$ groups of size $s = 3$ and $s^2 = 9$ blocks of size $\ell = 3$ each. Moreover, in the affine plane every unordered pair of points is contained in a single line, which is represented in $\mathcal{T}$ either by a group or by a block. More generally, for any prime power q, a transversal design TD(q,q) can be built with the affine plane $\mathbb{A}^2(\mathbb{F}_q)$.

Also notice that the code $\mathcal{C} = \mathrm{Code}_3(\mathcal{T})$ is a linear code over $\mathbb{F}_3$, of length 9 and dimension 3. A full-rank generator matrix of $\mathcal{C}$ is given by:

$$G = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 2 & 2 & 2 \\ 0 & 1 & 2 & 0 & 1 & 2 & 0 & 1 & 2 \end{pmatrix},$$

where the ordering of the points X we have chosen can be read in the columns of $\boldsymbol{G}$. One can check that $\mathcal{C}$ is identical to $\mathrm{Code}_3(\mathrm{AG}_1(2,3))$, the code based on the classical affine design, though the designs $\mathcal{T}$ and $\mathrm{AG}_1(2,3)$ are different. This property will be proved in a more generic setting in Proposition 4.13.

4.2.2 The protocol

In this section we present a construction of PIR protocols relying on transversal designs. The idea is that the knowledge of one point of a block of a transversal design gives (almost) no information on the other points lying on this block. This privacy property is transferred to the coordinates of words lying on the code based on a transversal design. Hence, if messages are encoded such a way, we obtain a 1-private PIR protocol. Though this protocol cannot resist collusions, we will see in Subsection 4.4.1 that a natural generalisation gives t-private PIR protocols, for $t > 1$.

Remark 4.8. One must notice that Barkol, Ishai and Weinreb consider in [BIW10] the case of LCCs arising from designs for applications to PIR. In that sense, their approach is close to ours. Nevertheless, the protocols they propose are costly in terms of storage (they use the replication-based model), and focus on small communication complexity. Moreover, they do not address the computational complexity issue.

Let $\mathcal{T}$ be a transversal design TD(ℓ, s) and $n = |X| = \ell s$. Denote by $\mathcal{C} = \mathrm{Code}_q(\mathcal{T}) \subseteq \mathbb{F}_q^X$ the associated $\mathbb{F}_q$-linear code, and let $k = \dim(\mathcal{C})$. A PIR protocol based on $\mathcal{C}$ is defined in Figure 4.1. We refer to this protocol as the *TD-based* PIR protocol. We then summarise the steps of the construction in Figure 4.2.

Theorem 4.9 states that the protocol we build in Figure 4.1 is indeed a 1-private PIR protocol.

Theorem 4.9. *If there exists a transversal design* $\mathrm{TD}_\lambda(\ell, s)$ *whose associated code has dimension* k*, then there exists a distributed* ℓ*-server* 1*-private PIR protocol for databases with* k *entries over* $\mathbb{F}_q$*, where:*

- *only one* $\mathbb{F}_q$*-symbol to read for each server,*
- $\ell - 1$ *field operations over* $\mathbb{F}_q$ *for the user,*
- $\ell \log(sq)$ *bits of communication (*$\ell \log s$ *are uploaded,* $\ell \log q$ *are downloaded),*
- *a storage rate* $R_{\mathcal{C}} = k/\ell s$*.*

Proof. Recall the PIR protocol we are dealing with is defined in Figure 4.1.

System parameters. $\mathcal{T} = (X, \mathcal{B}, \mathcal{G})$ a transversal design $\mathrm{TD}_\lambda(\ell, s)$, and $\mathcal{C} = \mathrm{Code}_q(\mathcal{T})$ its associated code of length $n = \ell s$ and dimension k.

1) Initialisation step.

1. *Encoding.* User U computes a systematic encoding of the database $\boldsymbol{m} \in \mathbb{F}_q^k$, resulting in the codeword $\boldsymbol{c} \in \mathcal{C}$. We denote by $x \in X$ the point such that $c_x = m_i$.
2. *Distribution.* Each server S_j receives $\boldsymbol{c}_{|G_j}$, for $1 \leq j \leq \ell$, where $G_j \in \mathcal{G}$.

2) Retrieval of symbol $c_x = m_i$. Denote by $j_x \in [1, \ell]$ the index of the unique group $G_{j_x} \in \mathcal{G}$ which contains i. Also denote by $\mathcal{B}_x$ the subset of blocks containing x. The three steps of the distributed PIR protocol are:

1. *Queries generation.* User U picks uniformly at random a block $B \in \mathcal{B}_x$. For $j \neq j_x$, U sends the unique index $q_j \in B \cap G_j$ to server S_j. Server S_{j_x} receives a random query q_{j_x} uniformly picked in G_{j_x}. To sum up, $Q \leftarrow_{\mathrm{R}} \mathsf{Query}(i)$ is defined by:
$$B \leftarrow_{\mathrm{R}} \mathcal{B}_x$$
and
$$\begin{cases} \mathcal{Q}(i)_{j_x} & \leftarrow_{\mathrm{R}} G_{j_x} \quad \text{where } j_x \text{ satisfies } x \in G_{j_x} \\ \mathcal{Q}(i)_j & \leftarrow B \cap G_j \quad \text{for } j \neq j_x \end{cases}$$
2. *Servers' answer.* Each server S_j (including S_{j_x}) reads $a_j := c_{q_j}$ and sends it back to the user. That is,
$$\mathsf{Ans}(q_j, \boldsymbol{c}_{|G_j}) := c_{q_j}\,.$$
3. *Reconstruction.* Denote by $\boldsymbol{a} = (a_1, \ldots, a_\ell)$ and $\boldsymbol{q} = (q_1, \ldots, q_\ell)$. User U outputs
$$\mathsf{Rec}(i, \boldsymbol{a}, \boldsymbol{q}) := -\sum_{j \neq j_x} a_{j_x} = -\sum_{j \neq j_x} c_{q_j}\,.$$

Figure 4.1 – A 1-private distributed PIR protocol based on the $\mathbb{F}_q$-linear code defined by a transversal design.

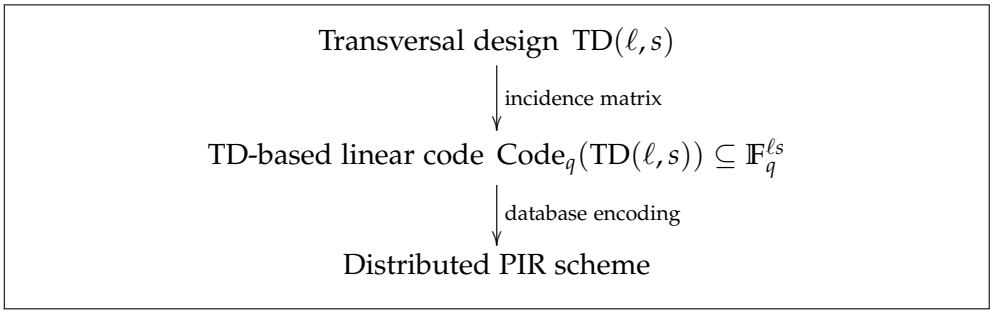


Figure 4.2 – Summary of the steps leading to the construction of a transversal-design-based PIR scheme.

Correctness. By definition of the code $\mathcal{C} = \mathrm{Code}_q(\mathcal{T})$, the incidence vector $\mathbb{1}_B$ of any block $B \in \mathcal{B}$ belongs to the dual code $\mathcal{C}^\perp$. Hence, for any $\boldsymbol{c} \in \mathcal{C}$, the inner product $\langle \mathbb{1}_B, \boldsymbol{c} \rangle = \sum_{y \in B} c_y = 0$. We recall that j_x represents the index of the group which contains x, where x satisfies $c_x = m_i$. Since the servers S_j, $j \neq j_x$, receive queries corresponding to the points of a block B which

contains x, we have $c_x = -\sum_{y \in B \setminus \{x\}} c_y = -\sum_{j \neq j_x} c_{q_j}$. Therefore, the PIR protocol is correct as long as there is no error on the symbols c_{q_j} returned by the servers.

Security (1-privacy). We need to prove that for all $1 \leq j \leq \ell$, it holds that $\mathrm{I}(i\,; q_j) = 0$, where $\mathrm{I}(\cdot\,;\cdot)$ denotes the mutual information. If $j = j_x$, then it is clear that q_j is picked independently to i, therefore $\mathrm{I}(i\,; q_{j_x}) = 0$. Now assume that $j \neq j_x$. By definition of a transversal design, the number of blocks containing both x and q_j is exactly λ, and the total number of blocks passing through x is λs. Therefore the probability $\Pr(q_j \mid i) = 1/s$, and we also get $\mathrm{I}(i\,; q_j) = 0$.

Communication complexity. Exactly one index $q_j \in [1, s]$ and one symbol $c_{q_j} \in \mathbb{F}_q$ are exchanged between each server and the user. So the overall communication complexity is $\ell \times (\log(s) + \log(q)) = \ell \log(sq)$ bits.

Storage rate. The number of bits stored on a server is $s \log q$, giving a storage rate of $k/s\ell = R$, the code rate. Equivalently, the storage overhead is $(s\ell - k) \log q$ bits.

Computational complexity. Each server S_j only needs to read the symbol defined by query q_j, and the protocol does not incur any extra computational cost. □

Theorem 4.9 shows that best practical parameters of the TD-based PIR protocol appear for small values of ℓ, the number of groups of the transversal design. However, one also observes that the dimension k of $\mathrm{Code}_q(\mathcal{T})$ strongly depends on ℓ and n, and tiny values of ℓ can lead us to degenerate or very small codes. This issue should be carefully taken into account, since for instance, codes of dimension $k \leq \ell$ represent PIR protocols which are more expensive in terms of communication than the trivial one. Hence, it is very natural to raise the main issue of the TD-based construction.

Problem 4.10. *Find codes $\mathcal{C} = \mathrm{Code}_q(\mathcal{T})$ arising from transversal designs $\mathcal{T} = \mathrm{TD}_\lambda(\ell, s)$ with few groups (small ℓ) and large dimension $k = \dim(\mathcal{C})$ compared to their length $n = \ell s$.*

A first answer comes by adapting Proposition 1.14 to transversal designs. Essentially, it shows that the characteristic of the base field $\mathbb{F}_q$ must be chosen properly in order to have a chance to get high-rate codes.

Proposition 4.11. *Let $\mathcal{T} = (X, \mathcal{B}, \mathcal{G})$ be a $\mathrm{TD}_\lambda(\ell, s)$. Let $q = p^e$, p prime. If $p \nmid \lambda s$, then*

$$\mathrm{Code}_q(\mathcal{T}) \subseteq \{\boldsymbol{c} \in \mathbb{F}_q^X \mid \forall G \in \mathcal{G}, \boldsymbol{c}_{|G} \in \mathrm{Rep}(s)\}\,,$$

where $\mathrm{Rep}(s)$ represents the repetition code of length s. In particular, if $p \nmid \lambda s$, then $\mathrm{Code}_q(\mathcal{T})$ has dimension at most ℓ.

Proof. For $x \in X$, recall that $\mathcal{B}_x = \{B \in \mathcal{B}, x \in B\}$, and denote by $\boldsymbol{a}^{(x)} = \sum_{B \in \mathcal{B}_x} \mathbb{1}_B$. We know that $\boldsymbol{a}^{(x)} \in \mathrm{Code}_q(\mathcal{T})^\perp$, since $\mathrm{Code}_q(\mathcal{T})^\perp$ is generated by $\{\mathbb{1}_B, B \in \mathcal{B}\}$. Denote by $G_x \in \mathcal{G}$ the only group that contains x. We see that:

$$\begin{cases} a_x^{(x)} = \lambda s & \\ a_i^{(x)} = 0 & \text{for all } i \in G_x \setminus \{x\} \\ a_j^{(x)} = \lambda & \text{for all } j \in X \setminus G_x\,. \end{cases}$$

Therefore $\boldsymbol{a}^{(x)} - \boldsymbol{a}^{(y)} = \lambda s(\mathbb{1}_{\{x\}} - \mathbb{1}_{\{y\}})$ if x and y lie in the same group G. If $p \nmid \lambda s$, then we get $\mathbb{1}_{\{x\}} - \mathbb{1}_{\{y\}} \in \mathrm{Code}_q(\mathcal{T})^\perp$. Let now

$$\mathcal{C} = \mathrm{Span}_{\mathbb{F}_q}\{\mathbb{1}_{\{x\}} - \mathbb{1}_{\{y\}}, \forall \{x, y\} \subset G \in \mathcal{G}\}$$

We see that $\mathcal{C}^\perp = \{\boldsymbol{c} \in \mathbb{F}_q^X \mid \forall G \in \mathcal{G}, \boldsymbol{c}_{|G} \in \mathrm{Rep}(s)\}$. Therefore we obtain the expected result. □

Section 4.3 is devoted to the construction of transversal designs leading to codes with high rate, in the perspective of providing solutions to Problem 4.10. Collusions of servers will be addressed in Section 4.4.

4.3 Instances

From now on, we denote by $\ell(k)$ the number of servers involved in a given PIR protocol running on a database with k entries, and by $n(k)$ the actual number of symbols stored by all the servers. As it is proved in Theorem 4.9, both parameters are crucial for the practicality of the TD-based PIR protocol, and they respectively correspond to the block size and the number of points of the transversal design used in the construction. In practice, we look for small values of ℓ and n as explained in Problem 4.10.

In this section, we first give two classical instances of transversal designs derived from finite geometries (Subsections 4.3.1 and 4.3.2), leading us to good PIR parameters. We then show how *orthogonal arrays* produce transversal designs, and we more deeply study a family of such arrays producing high-rate codes. Subsection 4.3.4 is finally devoted to another family of orthogonal arrays whose *divisibility* properties ensure to give an upper bound on the storage overhead of related PIR protocols.

4.3.1 Transversal designs from affine geometries

Transversal designs can be built with incidence properties between subspaces of an affine space.

Definition 4.12 (affine transversal design)**.** Let $\mathbb{A}^m(\mathbb{F}_q)$ be the affine space of dimension m over $\mathbb{F}_q$, and $\mathcal{G} = \{G_1, \ldots, G_q\}$ be a collection of q affine hyperplanes that partition $\mathbb{A}^m(\mathbb{F}_q)$. We define a transversal design $\mathcal{T}_A(m, q)$ as follows:

- the point set X consists in all the points in $\mathbb{A}^m(\mathbb{F}_q)$;
- the set of groups $\mathcal{G}$ is defined as above;
- the blocks in $\mathcal{B}$ are all the 1-dimensional affine subspaces (lines) which do not entirely lie in one of the H_j, for $1 \leq j \leq q$. We also say that such lines are *secant* to the hyperplanes in H.

This design is a $\mathrm{TD}(q, q^{m-1})$.

Let us now focus on the dimension of codes based on $\mathcal{T}_A(m, q)$. In order to obtain a large dimension, Proposition 4.11 inclines us to choose a code whose base field characteristic divides $q = p^e$, so let us choose $\mathbb{F}_p$. We see that the block set of $\mathcal{T}_A(m, q)$ is contained in the block set of $\mathrm{AG}_1(m, q)$, therefore

$$\mathrm{Code}_p(\mathrm{AG}_1(m, q)) \subseteq \mathrm{Code}_p(\mathcal{T}_A(m, q)) .$$

Let us prove that equality holds.

Proposition 4.13. *For every $q = p^e$ and $m \geq 2$, we have*

$$\mathrm{Code}_p(\mathrm{AG}_1(m, q)) = \mathrm{Code}_p(\mathcal{T}_A(m, q)) .$$

Proof. Denote by $\mathcal{B}^{(\mathrm{AG})}$ the blocks of $\mathrm{AG}_1(m, q)$, and by $\mathcal{B}^{(\mathcal{T})}$ and $\mathcal{G}^{(\mathcal{T})}$ the blocks and groups of $\mathcal{T}_A(m, q)$. Thanks to the previous discussion, we only need to show that for every block

$B \in \mathcal{B}^{(\mathrm{AG})}$ contained in a group $G \in \mathcal{G}^{(\mathcal{T})}$, it holds that $\mathbb{1}_B \in \mathrm{Code}_p(\mathcal{T}_A(m,q))^\perp$. To this end, let us first notice that $\mathrm{Code}_p(\mathcal{T}_A(m,q))^\perp = \mathrm{Span}\{\mathbb{1}_{B'}, B' \in \mathcal{B}^{(\mathcal{T})}\}$.

Let now $G \in \mathcal{G}^{(\mathcal{T})}$ and $B \in \mathcal{B}^{(\mathrm{AG})}$ such that $B \subseteq G$. Recall that G is a hyperplane of $\mathbb{A}^m(\mathbb{F}_q)$, and let P be a 2-dimensional affine plane of $\mathbb{A}^m(\mathbb{F}_q)$ such that $P \cap G = B$. We claim that $\mathbb{1}_P \in \mathrm{Span}\{\mathbb{1}_{B'}, B' \in \mathcal{B}^{(\mathcal{T})}\}$. Indeed, P admits a partition into affine lines which are secant to every hyperplane in $\mathcal{G}$, thus $\mathbb{1}_P$ is the sum of the characteristic vectors of these lines.

Now let $x \in B$, and $\mathcal{B}^{(\mathcal{T})}_{x,P} := \{B' \in \mathcal{B}^{(\mathcal{T})}, \{x\} \in B' \subset P\} \subseteq \mathcal{B}^{(\mathcal{T})}$. Define $\boldsymbol{b}^{(x)} = \sum_{B' \in \mathcal{B}^{(\mathcal{T})}_{x,P}} \mathbb{1}_{B'}$. It is clear that $\boldsymbol{b}^{(x)} \in \mathrm{Span}\{\mathbb{1}_{B'}, B' \in \mathcal{B}^{(\mathcal{T})}\}$, and we can notice that

$$\begin{cases} b^{(x)}_x = q = 0 & \\ b^{(x)}_i = 0 & \text{for all } i \in B \setminus \{x\} \\ b^{(x)}_j = 1 & \text{for all } j \in P \setminus B\,. \end{cases}$$

In other words, $\boldsymbol{b}^{(x)} = \mathbb{1}_P - \mathbb{1}_B$, therefore $\mathbb{1}_B \in \mathrm{Span}\{\mathbb{1}_{B'}, B' \in \mathcal{B}^{(\mathcal{T})}\}$. □

Now recall that the dimension of $\mathrm{Code}_p(\mathcal{T}_A(m,q))$ is known thanks to Hamada's formulae (1.1) and (1.2) that we reported in Chapter 1. However, for generic values of m and q, they do not allow us to grasp how large the dimension can be. So let us focus on the case $m = 2$.

In this case, we get $\mathrm{rank}_p(\mathrm{AG}_1(2,p^e)) = \binom{p+1}{2}^e$, hence $\mathrm{Code}_p(\mathcal{T}_A(2,p^e))$ has dimension $p^{2e} - \binom{p+1}{2}^e$. Therefore we obtain the following family of PIR protocols.

Proposition 4.14. *Let M be a database with $k = p^{2e} - \binom{p+1}{2}^e$ entries, p a prime, $e \geq 1$. There exists a distributed 1-private PIR protocol for M with:*

$$\ell(k) = p^e \quad \text{and} \quad n(k) = p^{2e}\,.$$

For fixed p and $k \to \infty$, we have

$$\ell(k) = \sqrt{k} + \Theta(k^{\frac{1}{2}+c_p}) \quad \text{and} \quad n(k)/k = \frac{1}{1 - \left(\frac{1+1/p}{2}\right)^e} = 1 + \Theta(k^{c_p}) \to 1\,, \tag{4.1}$$

where $c_p = \frac{1}{2}\log_p(\frac{1+1/p}{2}) < 0$.

Proof. The existence of the PIR protocol is a consequence of the Theorem 4.9. Let us state the asymptotics of the parameters. Recall that prime p is fixed, and let $e \to \infty$. First we have:

$$\begin{aligned} \frac{n(k)}{k} = \frac{p^{2e}}{p^{2e} - \binom{p+1}{2}^e} &= \frac{1}{1 - \left(\frac{1+1/p}{2}\right)^e} \\ &= 1 + \left(\frac{1+1/p}{2}\right)^e + \mathcal{O}\left(\left(\frac{1+1/p}{2}\right)^{2e}\right). \end{aligned} \tag{4.2}$$

Notice that

$$\log_p k = 2e + \log_p\left(1 - \left(\frac{1+1/p}{2}\right)^e\right) = 2e + \mathcal{O}\left(\left(\frac{1+1/p}{2}\right)^e\right).$$

Hence,

$$\begin{aligned}\left(\frac{1+1/p}{2}\right)^e &= \left(\frac{1+1/p}{2}\right)^{\frac{1}{2}\log_p k+\mathcal{O}\left(\left(\frac{1+1/p}{2}\right)^e\right)} \\ &= k^{\frac{1}{2}\log_p\left(\frac{1+1/p}{2}\right)} \times \left(\frac{1+1/p}{2}\right)^{\mathcal{O}\left(\left(\frac{1+1/p}{2}\right)^e\right)} \\ &= \Theta\left(k^{c_p}\right) ,\end{aligned}$$

since $\left(\frac{1+1/p}{2}\right)^{\mathcal{O}\left(\left(\frac{1+1/p}{2}\right)^e\right)} \to 1$ when $e \to \infty$. Using (4.1) we obtain the asymptotics we claimed for $n(k)/k$.

For $\ell(k)$, we see that $n(k) = \ell(k)^2$. Therefore, we get

$$\ell(k) = \sqrt{k}\sqrt{n(k)/k} = \sqrt{k}\sqrt{1+\Theta(k^{c_p})} = \sqrt{k} + \Theta(k^{\frac{1}{2}+c_p}) . \qquad \square$$

We give in Table 4.1 the dimension of codes arising from some affine transversal designs. In this table, m is not restricted to 2 but remains small. Finally, for a better understanding of the parameters we can point out two instances of PIR protocols:

- choosing $m = 2$ and $\ell = 4096$, there exists a PIR protocol on a $\simeq$ 2.0MB file with 6kB of communication and only 3.2% storage overhead;
- for a $\simeq$ 46GB database ($m = 3$, $\ell = 8192$), we obtain a PIR protocol with 39kB of communication and 27.1% storage overhead.

m	$\ell = q$	$n = s\ell = q^m$	$k = \dim(\mathcal{C})$	$R = k/n$
2	8	64	37	0.578
2	16	256	175	0.684
2	32	1024	781	0.763
2	64	4096	3367	0.822
2	1024	1 048 576	989 527	0.944
2	4096	16 777 216	16 245 775	0.968
2	16 384	268 435 456	263 652 487	0.982
2	65 536	4 294 967 296	4 251 920 575	0.990
3	8	512	139	0.271
3	16	4096	1377	0.336
3	64	262 144	118 873	0.453
3	256	16 777 216	9 263 777	0.552
3	1024	1 073 741 824	680 200 873	0.633
3	8192	549 755 813 888	400 637 408 211	0.729
4	8	4096	406	0.099
4	64	16 777 216	2 717 766	0.162
4	256	4 294 967 296	890 445 921	0.207
5	8	32 768	994	0.030
5	64	1 073 741 824	44 281 594	0.041

Table 4.1 – Dimension and rate of binary codes arising from $\mathcal{T}_A(m, q)$. Remind that the rate R of the code is the server storage rate of the PIR protocol, and that $q = \ell$ is the number of servers, which essentially corresponds to the communication complexity of the PIR protocol.

4.3.2 Transversal designs from projective geometries

Projective geometries are closely related to affine geometries, but contrary to them, there is no partition of the projective space into hyperplanes, since every pair of distinct projective hyperplanes intersects in a projective space of co-dimension 2. To tackle this problem, an idea is to consider the hyperplanes H_i which intersect on a fixed subspace of co-dimension 2 (call it Π_∞). Then, all the sets $H_i \setminus \Pi_\infty$ are disjoint, and their union gives exactly $\mathbb{P}^m(\mathbb{F}_q) \setminus \Pi_\infty$, where $\mathbb{P}^m(\mathbb{F}_q)$. Moreover, every projective line disjoint from Π_∞ is either contained in one of the H_i, or is 1-secant to all of them. It results to the following construction:

Definition 4.15 (Projective transversal design)**.** Let $\mathbb{P}^m(\mathbb{F}_q)$ and Π_∞ defined as above. Let us define

- a point set $X = \mathbb{P}^m(\mathbb{F}_q) \setminus \Pi_\infty$;
- a group set $\mathcal{G} = \{\text{projective hyperplanes } H \subset \mathbb{P}^m(\mathbb{F}_q),\ \Pi_\infty \subset H\}$;
- a block set $\mathcal{B} = \{\text{projective lines } L \subset \mathbb{P}^m(\mathbb{F}_q),\ L \cap \Pi_\infty = \varnothing \text{ and } \forall H \in \mathcal{G},\ L \not\subset H\}$.

Finally, denote by $\mathcal{T}_P(m,q) := (X, \mathcal{B}, \mathcal{G})$. Then $\mathcal{T}_P(m,q)$ is a $\mathrm{TD}(q+1, (q+1)q^{m-1})$.

The incidence matrix $\boldsymbol{M}_{\mathcal{T}_P(m,q)}$ of $\mathcal{T}_P(m,q)$ is a submatrix of $\boldsymbol{M}_{\mathrm{PG}_1(m,q)}$, from which we removed:

- the columns corresponding to the points in Π_∞,
- the rows corresponding to the lines not in $\mathcal{B}$.

Said differently, the code associated to $\mathcal{T}_P(m,q)$ contains (as a subcode) the Π_∞-shortening of the code associated to $\mathrm{PG}_1(m,q)$. Hence we have

$$\dim \mathrm{Code}(\mathcal{T}_P(m,q)) \geq \dim \mathrm{Code}(\mathrm{PG}_1(m,q)) - |\Pi_\infty|\,.$$

Once again, Hamada's formula allows us to compute $\dim \mathrm{Code}(\mathrm{PG}_1(m,q))$ efficiently, and it gets simpler for $m = 2$. We obtain the following proposition.

Proposition 4.16. *Let M be a database with $k = p^{2e} + p^e - \binom{p+1}{2}^e - 1$ entries, p a prime and $e \geq 1$. There exists a distributed 1-private PIR protocol for M with:*

$$\ell(k) = p^e + 1 \quad \text{and} \quad n(k) = p^{2e} + p^e\,.$$

Asymptotics are the same as in Equation (4.1)*.*

In order to emphasize that both previous constructions are asymptotically the same, in Figure 4.3 we draw the rates of the codes involved in these PIR protocols.

4.3.3 Orthogonal arrays and the *incidence code* construction

In this subsection, we first recall a way to produce transversal designs from another combinatorial construction called *orthogonal array*.

Definition 4.17 (orthogonal array)**.** Let $\lambda, s \geq 1$ and $1 \leq t \leq \ell$, and let $\boldsymbol{A}$ be an array with ℓ columns and λs^t rows, whose entries are elements of a set S of size s. We say that $\boldsymbol{A}$ is an orthogonal array $\mathrm{OA}_\lambda(t, \ell, s)$ if, in any subarray $\boldsymbol{A}'$ of $\boldsymbol{A}$ formed by t columns and all its rows, every row vector from S^t appears exactly λ times in the rows of $\boldsymbol{A}'$. We call λ the *index* of the orthogonal array, t its *strength* and ℓ its *degree*. If t (resp. λ) is omitted, it is understood to be 2 (resp. 1). When both these parameters are omitted we simply write $\boldsymbol{A} = \mathrm{OA}(\ell, s)$.

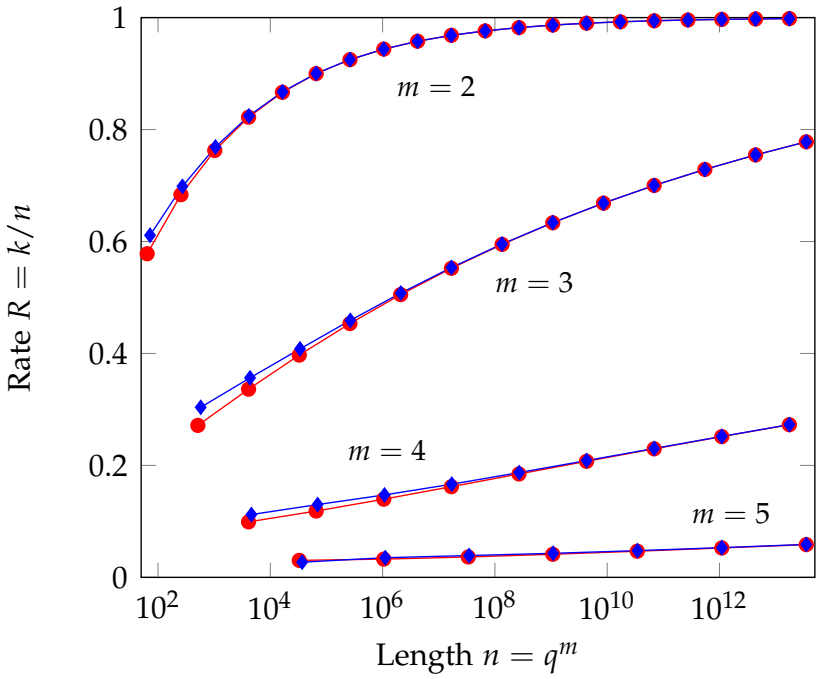


Figure 4.3 – Rate of binary codes coming from $\mathcal{T}_A(m,q)$ (in red) and $\mathcal{T}_P(m,q)$ (in blue). For every fixed m, we let q grow.

From now on, we restrict Definition 4.17 to orthogonal arrays with no repeated column and no repeated row. Next paragraph introduces a link between orthogonal arrays and transversal designs.

Construction of transversal designs from orthogonal arrays. We can build a transversal design $\mathrm{TD}(\ell, s)$ from an orthogonal array $\mathrm{OA}(\ell, s)$ with the following construction, given as a remark in [CD06, ch.II.2].

Construction 4.18 (Transversal designs from orthogonal arrays)**.** *Let $\boldsymbol{A}$ be an $\mathrm{OA}(\ell, s)$ of strength $t = 2$ and index $\lambda = 1$ with symbol set S, $|S| = s$. Denote by $\mathrm{Rows}(A)$ the set of rows of $\boldsymbol{A}$. We define the point set $X = S \times [1, \ell]$. To each row $\boldsymbol{a} \in \mathrm{Rows}(A)$ we associate a block*

$$B_{\boldsymbol{a}} := \{(a_j, j), 1 \leq j \leq \ell\},$$

so that the block set is defined as

$$\mathcal{B} := \{B_{\boldsymbol{a}}, \boldsymbol{a} \in \mathrm{Rows}(A)\}.$$

Finally, let $\mathcal{G} := \{S \times \{j\}, 1 \leq j \leq \ell\}$. Then $(X, \mathcal{B}, \mathcal{G})$ is a transversal design $\mathrm{TD}(\ell, s)$.

Example 4.19. A very simple example of this construction is given in Figure 4.4, where for clarity we use letters $\{a, b\}$ for elements of the symbol set, while the $\ell = 3$ columns are indexed by integers. On the left-hand side, $\boldsymbol{A}$ is an $\mathrm{OA}_1(2, 3, 2)$ with symbol set $S = \{a, b\}$. On the right-hand side, the associated transversal design $\mathrm{TD}(3, 2)$ is represented as a hypergraph: the nodes are the points of the design, its groups are depicted in columns, and a block consists in all nodes linked with a path of a fixed colour. One can check that every pair of nodes either belongs to the same group or is linked with one path.

Remark 4.20. If one lists in an array the codewords of a (possibly non-linear) code $\mathcal{C}_0 \subseteq S^\ell$, it gives rise to an orthogonal array whose strength t is derived from the dual distance d' of $\mathcal{C}_0$

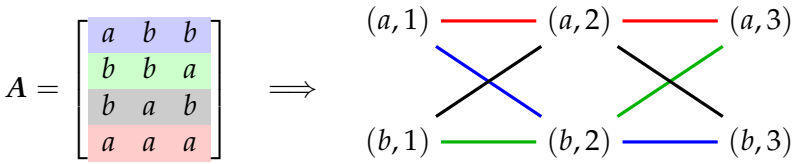


Figure 4.4 – A representation of the construction of a transversal design from an orthogonal array.

by $t = d' - 1$. Notice that for linear codes, the dual distance is simply the minimum distance of the dual code, but it can also be defined for non-linear codes (see [MS77, Ch.5.§5.]). More details about the link between orthogonal arrays and codes can also be found in [CD06]. For example, the orthogonal array of Figure 4.4 comes from the binary parity-check code $\mathrm{Par}(3)$ (replacing a by 0 and b by 1). One can check that its dual distance is 3 and its associated transversal design has strength 2.

Let us state more formally the link between the dual distance of a *linear* code, and the strength of the orthogonal array made of its codewords.

Lemma 4.21. *Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^X$ be a linear code and $T \subset X$, $|T| = t$ where $t < \mathrm{d}^\perp(\mathcal{C}_0)$. For $\boldsymbol{a} \in \mathbb{F}_q^T$, denote by $\mathcal{V}_{\boldsymbol{a}} = \{\boldsymbol{c} \in \mathcal{C}_0, \boldsymbol{c}_{|T} = \boldsymbol{a}\}$, and $N_{\boldsymbol{a}} = |\mathcal{V}_{\boldsymbol{a}}|$. Then,*

1. *$\mathcal{V}_{\mathbf{0}} = \{\boldsymbol{v} \in \mathbb{F}_q^X \mid \boldsymbol{v}_{|T} = \mathbf{0}$ and $\boldsymbol{v}_{|X\setminus T} \in \mathrm{Short}(\mathcal{C}_0, T)\}$ is a linear subcode of $\mathcal{C}_0$;*
2. *for every non-zero $\boldsymbol{a} \in \mathbb{F}_q^T$, there exists a non-zero $\boldsymbol{c}^{(\boldsymbol{a})} \in \mathcal{C}_0$ such that*

$$\mathcal{V}_{\boldsymbol{a}} = \mathcal{V}_{\mathbf{0}} + \{\boldsymbol{c}^{(\boldsymbol{a})}\};$$

3. *for every $\boldsymbol{a} \in \mathbb{F}_q^T$, $N_{\boldsymbol{a}} = q^{k-t}$ where $k = \dim \mathcal{C}_0$.*

Proof.

1. The fact that $\mathcal{V}_{\mathbf{0}} = \{\boldsymbol{v} \in \mathbb{F}_q^X \mid \boldsymbol{v}_{|T} = \mathbf{0}$ and $\boldsymbol{v}_{|X\setminus T} \in \mathrm{Short}(\mathcal{C}_0, T)\}$ is actually the definition of the shortening of a code.
2. Let $\boldsymbol{a} \in \mathbb{F}_q^T$ be non-zero, and let us first prove that there exists $\boldsymbol{c}^{(\boldsymbol{a})} \in \mathcal{C}_0$ such that $\boldsymbol{c}^{(\boldsymbol{a})}_{|T} = \boldsymbol{a}$. If it were not the case, then we would have $\mathrm{Punct}(\mathcal{C}_0, X \setminus T) \neq \mathbb{F}_q^t$ by definition, or equivalently $\mathrm{Short}(\mathcal{C}_0^\perp, X \setminus T) = \mathrm{Punct}(\mathcal{C}_0, X \setminus T)^\perp \neq \{\mathbf{0}\}$. But this is impossible since $\mathcal{C}_0^\perp$ contains no non-zero codeword of weight less that t.
   It is then easy to check that $\mathcal{V}_{\boldsymbol{a}} = \mathcal{V}_{\mathbf{0}} + \{\boldsymbol{c}^{(\boldsymbol{a})}\}$.
3. First notice that $\mathcal{V}_{\boldsymbol{a}} \cap \mathcal{V}_{\boldsymbol{b}} = \varnothing$ if $\boldsymbol{a} \neq \boldsymbol{b}$. Since

$$\mathcal{C}_0 = \bigcup_{\boldsymbol{a} \in \mathbb{F}_q^t} \mathcal{V}_{\boldsymbol{a}},$$

   we get the expected result. □

Given a code $\mathcal{C}_0$, we denote by $\boldsymbol{A}_{\mathcal{C}_0}$ the orthogonal array it defines and by $\mathcal{T}_{\mathcal{C}_0}$ the transversal design built from $\boldsymbol{A}_{\mathcal{C}_0}$ thanks to Construction 4.18.

As a straightforward corollary of Lemma 4.21, we get:

Proposition 4.22. *Let $\mathcal{C}_0 \subseteq \mathbb{F}_q^X$ be a linear code of dimension k, and denote by $t = \mathrm{d}^\perp(\mathcal{C}) - 1$. Then, the array $\boldsymbol{A}_{\mathcal{C}_0}$ whose rows are codewords of $\mathcal{C}_0$ is an orthogonal array $\mathrm{OA}_\lambda(t, |X|, q)$, where $\lambda = q^{k-t}$.*

Proof. It is clear that $\boldsymbol{A}_{\mathcal{C}_0}$ has $q^k = \lambda q^t$ rows, since the dimension of $\mathcal{C}_0$ is k. Let $T \subset X$ such that $|T| = t$, and let $\boldsymbol{a} \in \mathbb{F}_q^T$. Thanks to Lemma 4.21, there exists exactly $\lambda = s^{k-t}$ rows $\boldsymbol{r}$ of $\boldsymbol{A}_{\mathcal{C}_0}$ such that $\boldsymbol{r}_{|T} = \boldsymbol{a}$. It proves that $\boldsymbol{A}_{\mathcal{C}_0}$ has strength t. □

Example 4.23. Let S be a set of ℓ pairwise distinct elements of $\mathbb{F}_q$. We recall that $\mathrm{RS}_q(1, S) \subseteq \mathbb{F}_q^S$ denotes the Reed-Solomon code of length ℓ and dimension 2 over $\mathbb{F}_q$, with evaluation points from S. Then, $\mathrm{RS}_q(1, S)$ has dual distance 3, so its codewords form an orthogonal array $\boldsymbol{A}_{\mathrm{RS}_q(1,S)} = \mathrm{OA}(\ell, q)$ of strength 2. Columns of this array can be naturally indexed by S instead of $[1, \ell]$. Now, we use Construction 4.18 to obtain a transversal design $\mathcal{T}_{\mathrm{RS}_q(1,S)} = \mathrm{TD}(\ell, q)$. The point set is $X = \mathbb{F}_q \times S$, and the blocks are Reed-Solomon codewords, that is, sets of the form $\{(c_\alpha, \alpha), \alpha \in S\}$ with $\boldsymbol{c} \in \mathrm{RS}_q(1, S) \subseteq \mathbb{F}_q^S$. The set of groups $\mathcal{G} = \{G_\alpha, \alpha \in S\}$ correspond to the ℓ coordinates of the code: $G_\alpha = \mathbb{F}_q \times \{\alpha\}$, $\alpha \in S$.

Remark 4.24. The use of low-degree polynomials for the construction of orthogonal arrays was described as early as 1952 by Bush [Bus52], that is, even before the actual formalisation of Reed-Solomon codes [RS60].

We can finally summarise the construction by introducing the code $\mathrm{Code}_q(\mathcal{T}_{\mathcal{C}_0})$ that arises from $\mathcal{C}_0$ through Construction 4.18. We call $\mathrm{Code}_q(\mathcal{T}_{\mathcal{C}_0})$ the *incidence code* of $\mathcal{C}_0$. Our motivation is that studying the structure of $\mathcal{C}_0$ may lead us to properties of $\mathrm{Code}_q(\mathcal{T}_{\mathcal{C}_0})$ which is used in PIR protocols.

Definition 4.25 (incidence code)**.** Let $\mathcal{C}_0 \subseteq \Sigma^S$ be an unrestricted code of length $|S| = \ell$ over an alphabet Σ of size s. The *incidence code* of $\mathcal{C}_0$ over $\mathbb{F}_q$, denoted $\mathrm{IC}_q(\mathcal{C}_0)$, is the $\mathbb{F}_q$-linear code of length $n = s\ell$ built from the transversal design $\mathcal{T}_{\mathcal{C}_0}$, that is:

$$\mathrm{IC}_q(\mathcal{C}_0) := \mathrm{Code}_q(\mathcal{T}_{\mathcal{C}_0}) .$$

Notice that the field $\mathbb{F}_q$ does not need to be identical to the alphabet Σ of the code $\mathcal{C}_0$.

Incidence codes are introduced in order to design PIR protocols, as summarises Figure 4.5. From previous results, it is clear that, if $\mathcal{C}_0$ has dual distance more than 3, then the induced PIR protocol is 1-private. A generalisation will be formally proved in a next section, see Corollary 4.46.

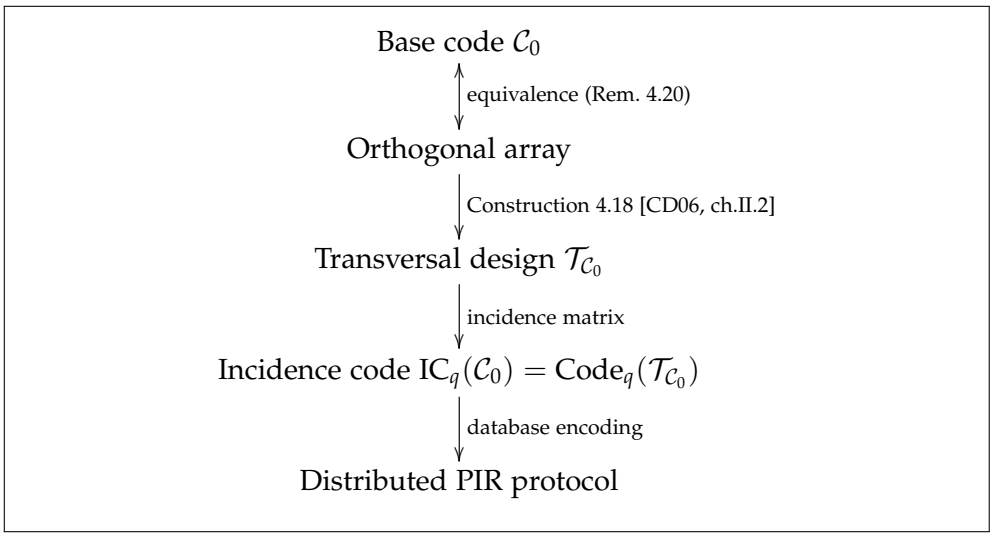


Figure 4.5 – Outline of the construction of a distributed PIR protocol using incidence codes.

Example 4.26. Here we provide a full example of the construction of an incidence code. In order to have clearer representations of matrices and tuples, we will choose a particular order of points and groups of a design, but one should keep in mind this is not necessary.

Let $\mathcal{C}_0 = \mathrm{RS}_4(1) \subseteq \mathbb{F}_4^4$ be the full-length Reed-Solomon code of dimension 2 over the field $\mathbb{F}_4$, whose elements are arbitrarily ordered as follows: $\{0, 1, \alpha, \alpha^2 = \alpha + 1\}$. The orthogonal array associated to $\mathcal{C}_0$ is composed of the following list of codewords:

$$\boldsymbol{A} = \begin{pmatrix} 0, & 0, & 0, & 0 \\ 1, & 1, & 1, & 1 \\ \alpha, & \alpha, & \alpha, & \alpha \\ \alpha^2, & \alpha^2, & \alpha^2, & \alpha^2 \\ 0, & 1, & \alpha, & \alpha^2 \\ 0, & \alpha, & \alpha^2, & 1 \\ 0, & \alpha^2, & 1, & \alpha \\ 1, & 0, & \alpha^2, & \alpha \\ 1, & \alpha^2, & \alpha, & 0 \\ 1, & \alpha, & 0, & \alpha^2 \\ \alpha, & \alpha^2, & 0, & 1 \\ \alpha, & 0, & 1, & \alpha^2 \\ \alpha, & 1, & \alpha^2, & 0 \\ \alpha^2, & \alpha, & 1, & 0 \\ \alpha^2, & 1, & 0, & \alpha \\ \alpha^2, & 0, & \alpha, & 1 \end{pmatrix}$$

Using Construction 4.18, we get a transversal design $\mathcal{T}_{\mathcal{C}_0} = (X, \mathcal{B}, \mathcal{G})$ with 16 points (4 groups made of 4 points) and 16 blocks. Let us recall how we map a row of $\boldsymbol{A}$ to a word in $\{0,1\}^{16}$. For instance, consider the fifth row of $\boldsymbol{A}$, highlighted in grey:

$$\boldsymbol{a} := (0, 1, \alpha, \alpha^2)\,.$$

We turn $\boldsymbol{a}$ into a block $B_{\boldsymbol{a}} := \{(0,1), (1,2), (\alpha,3), (\alpha,4)\} \in \mathcal{B}$, and we build the incidence vector $\mathbb{1}_{B_{\boldsymbol{a}}}$ of the block $B_{\boldsymbol{a}}$ over the point set $X = \{(\beta, i), \beta \in \mathbb{F}_4, i \in [1,4]\}$. In order to see $\mathbb{1}_{B_{\boldsymbol{a}}}$ as a word in $\{0,1\}^{16}$, we can also order elements in X, for instance as follows:

$$\begin{aligned}(\ &(0,1)\,, (1,1), (\alpha,1), (\alpha^2,1), (0,2), \\ &(1,2)\,, (\alpha,2), (\alpha^2,2), \\ &(0,3), (1,3), \\ &(\alpha,3)\,, (\alpha^2,3), (1,4), (1,4), (\alpha,4), \\ &(\alpha^2,4)\ )\,.\end{aligned}$$

Using this ordering, we get:

$$\mathbb{1}_{B_{\boldsymbol{a}}} = (1, 0,0,0, 1, 0,0,0, 1, 0,0,0, 1) \in \{0,1\}^{16}\,.$$

By computing all characteristic vectors $\mathbb{1}_{B_a}$ of every $\boldsymbol{a} \in \mathrm{Rows}(A)$, we obtain the following incidence matrix $\boldsymbol{M}$ for the transversal design $\mathcal{T}_{\mathcal{C}_0}$:

$$\boldsymbol{M} = \left(\begin{array}{c:c:c:c} 1000 & 1000 & 1000 & 1000 \\ 0100 & 0100 & 0100 & 0100 \\ 0010 & 0010 & 0010 & 0010 \\ 0001 & 0001 & 0001 & 0001 \\ \hline 1000 & 0100 & 0010 & 0001 \\ 1000 & 0010 & 0001 & 0100 \\ 1000 & 0001 & 0100 & 0010 \\ 0100 & 1000 & 0001 & 0010 \\ 0100 & 0001 & 0010 & 1000 \\ 0100 & 0010 & 1000 & 0001 \\ 0010 & 0001 & 1000 & 0100 \\ 0010 & 1000 & 0100 & 0001 \\ 0010 & 0100 & 0001 & 1000 \\ 0001 & 0010 & 0100 & 1000 \\ 0001 & 0100 & 1000 & 0010 \\ 0001 & 1000 & 0010 & 0100 \end{array}\right),$$

Notice that this matrix can be quickly obtained by respectively replacing entries 0, 1, α and α^2 in the array $\boldsymbol{A}$ by the binary 4-tuples (1000), (0100), (0010) and (0001) in the matrix $\boldsymbol{M}$ (of course this map depends on the ordering of X we choose, but another choice would lead us to a column-permutation-equivalent matrix, hence a permutation-equivalent code). Notice that in matrix $\boldsymbol{M}$, coordinates lying in the same group have been distinguished by dashed vertical lines.

Matrix $\boldsymbol{M}$ then defines, over any extension $\mathbb{F}_{2^e}$ of the prime field $\mathbb{F}_2$, the dual code of the incidence code $\mathrm{IC}_{2^e}(\mathcal{C}_0)$. For all values of e, the incidence codes $\mathrm{IC}_{2^e}(\mathcal{C}_0)$ have the same generator matrix $\boldsymbol{G}$, being:

$$\boldsymbol{G} = \left(\begin{array}{c:c:c:c} 1001 & 0000 & 0011 & 1010 \\ 0101 & 0000 & 0110 & 0011 \\ 0011 & 0000 & 0101 & 0110 \\ 0000 & 1001 & 0101 & 1100 \\ 0000 & 0101 & 0011 & 0110 \\ 0000 & 0011 & 0110 & 0101 \\ 0000 & 0000 & 1111 & 1111 \end{array}\right).$$

A deeper analysis of incidence codes coming from linear MDS codes of dimension 2. Incidence codes lead to a very large family of PIR protocols — as many as there exists codes $\mathcal{C}_0$ — but most of them are not practical, since the incidence codes are too small. To simplify their study, one can first remark that intuitively, the more blocks a transversal design, the larger its incidence matrix, and consequently, the lower the dimension of its associated code. Furthermore, the number of blocks of $\mathcal{T}_{\mathcal{C}_0}$ is the cardinality of $\mathcal{C}_0$. Hence, heuristically the smaller the code $\mathcal{C}_0$, the larger $\mathrm{IC}(\mathcal{C}_0)$.

In this paragraph we analyse the incidence codes constructed with MDS codes of dimension 2. Their interest lies in being the smallest codes with dual distance 3, which is the minimal setting for defining 1-private PIR protocols. Recall that generalised Reed-Solomon codes are MDS. The next lemma shows they are essentially the only one of dimension 2.

Lemma 4.27. *All $[\ell, 2, \ell-1]_q$ MDS linear codes with $2 \leq \ell \leq q$ are generalised Reed-Solomon codes.*

Proof. Let $\mathcal{C}$ be an $[\ell, 2, \ell-1]_q$ code with $2 \leq \ell \leq q$. Since $\mathcal{C}$ is MDS, it has dual distance $\mathrm{d}^\perp(\mathcal{C}) = 3$, and we claim there exists a codeword $\boldsymbol{c} \in \mathcal{C}$ with Hamming weight ℓ. To see this, let $\boldsymbol{G} = (\boldsymbol{g}_1, \ldots, \boldsymbol{g}_\ell)$ be a generator matrix of $\mathcal{C}$, where $\boldsymbol{g}_i \in \mathbb{F}_q^2$ is an affine point written in column. Notice that each point $\boldsymbol{g}_i$ is non-zero (otherwise $\mathrm{d}^\perp(\mathcal{C}) = 1$) and $\mathbf{0}, \boldsymbol{g}_i, \boldsymbol{g}_j$ are not collinear for $i \neq j$ (otherwise $\mathrm{d}^\perp(\mathcal{C}) = 2$). Moreover, the $\boldsymbol{g}_i$'s are not all on the same affine line (otherwise, $\dim(\mathcal{C}) \leq 1$).

An important remark is that one can see codewords in $\mathcal{C}$ as evaluations of linear maps $\mu : \mathbb{F}_q^2 \to \mathbb{F}_q$ over $(\boldsymbol{g}_1, \ldots, \boldsymbol{g}_\ell)$:

$$\mathcal{C} = \{(\mu(\boldsymbol{g}_1), \ldots, \mu(\boldsymbol{g}_\ell)), \mu \in \mathrm{Hom}(\mathbb{F}_q^2, \mathbb{F}_q)\}.$$

Since $\ell \leq q$, there exists $\boldsymbol{a} = (a_0, a_1) \in \mathbb{F}_q^2 \setminus \{\mathbf{0}\}$ such that $\boldsymbol{a}$ does not lie in the affine line through $\mathbf{0}$ and any of the $\boldsymbol{g}_i$'s (indeed there are $q+1$ affine lines through $\mathbf{0}$). Let now $\mu_{\boldsymbol{a}}(X, Y) = a_1 X - a_0 Y$: it is a non-zero linear form which must vanish on a line of $\mathbb{F}_q^2$, and since $\mu_{\boldsymbol{a}}(\boldsymbol{a}) = \boldsymbol{a}$, it vanishes on $\mathrm{Span}\{\boldsymbol{a}\}$. To sum up, for every $i \in [1, \ell]$, we have $\mu_{\boldsymbol{a}}(\boldsymbol{g}_i) \neq 0$. Hence, $\boldsymbol{c} = (\mu_{\boldsymbol{a}}(\boldsymbol{g}_1), \ldots, \mu_{\boldsymbol{a}}(\boldsymbol{g}_\ell))$ belongs to $\mathcal{C}$ and has Hamming weight ℓ.

Let now $\boldsymbol{d} \in \mathcal{C}$ such that $\{\boldsymbol{c}, \boldsymbol{d}\}$ spans $\mathcal{C}$. Then $\boldsymbol{c} = \mathbf{1} \star \boldsymbol{c}$ and $\boldsymbol{d} = \boldsymbol{c} \star (\boldsymbol{c}^{-1} \star \boldsymbol{d})$, where $\boldsymbol{c}^{-1}$ is the coordinate-wise inverse of $\boldsymbol{c}$. Hence, the code $\mathcal{C}$ can be written $\boldsymbol{c} \star \mathcal{C}'$ where $\mathcal{C}'$ is generated by $\{\mathbf{1}, \boldsymbol{c}^{-1} \star \boldsymbol{d}\}$. These two tuples are respectively the evaluation of polynomials 1 and X over the set S of symbols in $\boldsymbol{c}^{-1} \star \boldsymbol{d}$. Hence, $\mathcal{C} = \mathrm{GRS}_q(1, S, \boldsymbol{c})$. □

Let us understand the consequences of Lemma 4.27 in terms of transversal designs. We say a map $\phi : X \to X'$ is an isomorphism between transversal designs $(X, \mathcal{B}, \mathcal{G})$ and $(X', \mathcal{B}', \mathcal{G}')$ if it is one-to-one and if it preserves the incidence relation. In other words, we require that ϕ is invertible on the points, blocks and groups:

$$\phi(X) = X', \quad \phi(\mathcal{B}) = \mathcal{B}', \quad \phi(\mathcal{G}) = \mathcal{G}'.$$

Lemma 4.28. *Let $\mathcal{C}, \mathcal{C}' \subseteq \mathbb{F}_q^\ell$ be two codes such that $\mathcal{C}' = \boldsymbol{y} \star \mathcal{C}$ for some $\boldsymbol{y} \in (\mathbb{F}_q^\times)^\ell$. Denote by $\mathcal{T}_{\mathcal{C}}$, $\mathcal{T}_{\mathcal{C}'}$ the transversal designs they respectively define. Then, $\mathcal{T}_{\mathcal{C}}$ and $\mathcal{T}_{\mathcal{C}'}$ are isomorphic.*

Proof. Write $\mathcal{T}_{\mathcal{C}} = (X, \mathcal{B}, \mathcal{G})$ and $\mathcal{T}_{\mathcal{C}'} = (X', \mathcal{B}', \mathcal{G}')$. From the definition it is clear that $X = X' = \mathbb{F}_q \times [1, \ell]$ and $\mathcal{G} = \mathcal{G}' = \{\mathbb{F}_q \times \{i\}, 1 \leq i \leq \ell\}$. Now consider the block sets. We see that $\mathcal{B} = \{\{(c_i, i), 1 \leq i \leq \ell\}, \boldsymbol{c} \in \mathcal{C}\}$ and $\mathcal{B}' = \{\{(y_i c_i, i), 1 \leq i \leq \ell\}, \boldsymbol{c} \in \mathcal{C}\}$. Let:

$$\begin{array}{rccc} \phi_{\mathbf{y}} : & \mathbb{F}_q \times [1, \ell] & \to & \mathbb{F}_q \times [1, \ell] \\ & (x, i) & \mapsto & (y_i x, i) \end{array}$$

The vector $\boldsymbol{y}$ is $\star$-invertible, hence $\phi_{\mathbf{y}}$ is one-to-one on the point set X. It remains to notice that $\phi_{\boldsymbol{y}}$ maps $\mathcal{G}$ to itself since it only acts on the first coordinate, and that $\phi_{\boldsymbol{y}}(\mathcal{B})$ is exactly $\mathcal{B}'$ by definition of $\mathcal{C}$ and $\mathcal{C}'$. □

Proposition 4.29. *Let $2 \leq \ell \leq q$ and $\mathcal{C}_0$ be an $[\ell, 2, \ell-1]_q$ MDS code. Let also $\mathbb{F}_p$ be any finite field. The incidence code $\mathrm{IC}_p(\mathcal{C}_0)$ is permutation-equivalent to $\mathrm{IC}_p(\mathrm{RS}_q(1, S))$, for some $S \subseteq \mathbb{F}_q$, $|X| = \ell$.*

Proof. Lemma 4.27 shows that all $[\ell, 2, \ell-1]_q$ linear codes $\mathcal{C}_0$ are generalised Reed-Solomon codes $\mathrm{GRS}_q(1, S, \boldsymbol{y}) = \boldsymbol{y} \star \mathrm{RS}_q(1, S)$ for some $S \subseteq \mathbb{F}_q$, $|X| = \ell$. Moreover, with the previous notation $\phi_{\boldsymbol{y}}(\mathcal{T}_{\mathrm{RS}_q(1,S)}) = \mathcal{T}_{\boldsymbol{y} \star \mathrm{RS}_q(1,S)}$, so we have

$$\mathrm{IC}_p(\boldsymbol{y} \star \mathrm{RS}_q(1, S)) = \mathrm{Code}_p(\phi_{\boldsymbol{y}}(\mathcal{T}_{\mathrm{RS}_q(1,S)})).$$

Now, let:

$$\begin{array}{rccc} \tilde{\phi}_{\mathbf{y}}: & \mathbb{F}_p^X & \to & \mathbb{F}_p^X \\ & \boldsymbol{c} = (c_x)_{x\in X} & \mapsto & (c \circ \boldsymbol{\phi}_{\boldsymbol{y}}(x))_{x\in X} \end{array}.$$

Clearly $\tilde{\phi}_{\boldsymbol{y}}(\mathrm{IC}_p(\mathrm{RS}_q(1,S))) = \mathrm{Code}_p(\boldsymbol{\phi}_{\boldsymbol{y}}(\mathcal{T}_{RS_q(1,S)}))$ and $\tilde{\phi}_{\boldsymbol{y}}$ acts as a permutation of coordinates. So $\mathrm{IC}_p(\mathcal{C}_0)$ is permutation-equivalent to $\mathrm{IC}_p(\mathrm{RS}_q(1,S))$ which proves the result. □

Proposition 4.29 allows us to restrict our study to incidence codes of Reed-Solomon codes of dimension 2. Let us first consider the case of full-length Reed-Solomon codes $\mathrm{RS}_q(1) \subseteq \mathbb{F}_q^{\mathbb{F}_q}$. An interesting result is that these incidence codes are equivalent to codes based on affine geometric designs.

Proposition 4.30. *The following two codes are equal up to permutation:*

1. $\mathcal{C}_1 = \mathrm{IC}_q(\mathrm{RS}_q(1))$*, the incidence code over* $\mathbb{F}_q$ *of the q-ary full-length Reed-Solomon code of dimension* 2*;*
2. $\mathcal{C}_2$*, the code over* $\mathbb{F}_q$ *based on the transversal design* $\mathcal{T}_A(2,q)$*.*

Proof. It is sufficient to show that the transversal design defined by $\mathcal{C}_0 = \mathrm{RS}_q(1)$ is isomorphic to $\mathcal{T}_A(2,q)$. We first recall that $\mathcal{T}_{\mathcal{C}_0} = (X,\mathcal{B},\mathcal{G})$ where:

$$\begin{cases} X = \mathbb{F}_q \times \mathbb{F}_q \\ \mathcal{B} = \{\{(c_x,x), x\in\mathbb{F}_q\}, \boldsymbol{c}\in\mathcal{C}_0\} \\ \mathcal{G} = \{\{(\alpha,x), \alpha\in\mathbb{F}_q\}, x\in\mathbb{F}_q\}. \end{cases}$$

Up to an affine transformation of the plane $\mathbb{A}^2$, the groups of $\mathcal{T}_A(2,q) = (X',\mathcal{B}',\mathcal{G}')$ can be set to $\{\{(\alpha,x), \alpha\in\mathbb{F}_q\}, x\in\mathbb{F}_q\}$. Hence, up to isomorphism $\mathcal{T}_A(2,q)$ can be written as:

$$\begin{cases} X' = \mathbb{F}_q \times \mathbb{F}_q \\ \mathcal{B}' = \{\{(ax+b,x), x\in\mathbb{F}_q\}, (a,b)\in\mathbb{F}_q^2,\} \\ \mathcal{G}' = \{\{(\alpha,x), \alpha\in\mathbb{F}_q\}, x\in\mathbb{F}_q\}. \end{cases}$$

It remains to notice that a codeword $\boldsymbol{c}\in\mathcal{C}_0$ is the evaluation of a polynomial of degree ≤ 1 over $\mathbb{F}_q$. Hence for some $(a,b)\in\mathbb{F}_q^2$, we have $c_x = ax+b, \forall x\in\mathbb{F}_q$. □

We now need to study the more general case of subsets $S\subset\mathbb{F}_q$, $|S| = \ell < q$. First, it worth noticing that $\mathrm{IC}_q(\mathrm{RS}_q(1,S))$ is a shortening of $\mathrm{IC}_q(\mathrm{RS}_q(1))$. Indeed, we have the following property:

Lemma 4.31. *Let* $\mathcal{C}_0\subset\mathbb{F}_q^S$*,* $S=\ell$*, and* $\mathrm{Punct}(\mathcal{C}_0,I)$ *its puncturing on coordinates* $I\subset S$*. Then for every prime p,*

$$\mathrm{IC}_p(\mathrm{Punct}(\mathcal{C}_0,I)) = \mathrm{Short}(\mathrm{IC}_p(\mathcal{C}_0),J)$$

where $J = \cup_{i\in I} G_i$ *and* $G_i = \mathbb{F}_q\times\{i\}$ *is the group indexed by* $i\in S$ *in the transversal design* $\mathcal{T}_{\mathcal{C}_0}$*.*

Proof. Let us analyse the link between $\mathcal{T}_{\mathrm{Punct}(\mathcal{C}_0,I)} = (X',\mathcal{B}',\mathcal{G}')$ and $\mathcal{T}_{\mathcal{C}_0} = (X,\mathcal{B},\mathcal{G})$. We have:

$$\begin{array}{lll} X' & = \mathbb{F}_q\times(S\setminus I) & \subset S, \\ \mathcal{G}' & = \{\mathbb{F}_q\times\{i\},\, i\in(S\setminus I)\} & \subset \mathcal{G}, \\ \mathcal{B}' & = \{B\cap X',\, B\in\mathcal{B}\}. & \end{array}$$

Let $\mathcal{C} = \mathrm{IC}_p(\mathcal{C}_0)$ and $\mathcal{C}' = \mathrm{IC}_p(\mathrm{Punct}(\mathcal{C}_0,I))$. For $\boldsymbol{u}\in\mathbb{F}_p^{X'}$, we define an extension $\mathrm{ext}(\boldsymbol{u})\in\mathbb{F}_p^X$ of $\boldsymbol{u}$, such that $\mathrm{ext}(\boldsymbol{u})_{|X'} = \boldsymbol{u}$ and $\mathrm{ext}(\boldsymbol{u})_{|X\setminus X'} = \mathbf{0}$. By definition of the shortening operation, all we need to prove is:

$$\mathcal{C}' = \{\boldsymbol{c}\in\mathbb{F}_p^{X'},\, \mathrm{ext}(\boldsymbol{c})\in\mathcal{C}\}.$$

Let $\boldsymbol{c} \in \mathcal{C}'$, and consider $\boldsymbol{u} = \mathrm{ext}(\boldsymbol{c})$. For every $B \in \mathcal{B}$, we have $\sum_{b \in B} u_b = \sum_{b' \in B \cap X'} c_{b'} = 0$, since $\mathbb{1}_{B \cap X'}$ is a parity-check matrix for $\mathcal{C}'$. Hence $\mathrm{ext}(\boldsymbol{c}) \in \mathcal{C}$. The converse inclusion is similar. □

Despite this result, incidence codes of Reed-Solomon codes $\mathrm{RS}_q(1, S)$ remain hard to classify for $|S| = \ell < q$. Indeed, for a given length $\ell < q$, it appears that distinct subsets S may produce non-equivalent incidence codes $\mathrm{IC}(\mathrm{RS}_q(1, S))$. Even their dimension can differ, as shows for instance an exhaustive search on $\mathrm{IC}_2(\mathrm{RS}_{16}(1, S))$ with pairwise distinct $|S| = \ell = 5$: we observe that 48 of these codes have dimension 24 while the 4320 others have dimension 22. Further interesting research would then be to understand the subsets S which give the largest codes.

4.3.4 High-rate incidence codes from divisible codes

In this subsection, we prove that linear codes $\mathcal{C}_0$ satisfying a *divisibility* condition yield incidence codes whose rate is roughly greater than $1/2$. Let us first define divisible codes.

Definition 4.32 (code divisibility)**.** Let $p \geq 2$. A linear code is p-divisible if p divides the Hamming weight of any of its codewords.

When studying the incidence matrix which defines an incidence code, we exhibit the following property.

Lemma 4.33. *Let $\mathcal{C}_0 \subset S^I$ be an unrestricted code, $|I| = \ell$, and let $\mathcal{T}$ be the transversal design associated to $\mathcal{C}_0$. We denote by $\boldsymbol{M}$ the incidence matrix of $\mathcal{T}$, where rows of $\boldsymbol{M}$ are indexed by codewords from $\mathcal{C}_0$. Then we have:*

$$(\boldsymbol{M}\boldsymbol{M}^T)_{\boldsymbol{c},\boldsymbol{c}'} = \ell - \mathrm{d}(\boldsymbol{c}, \boldsymbol{c}'), \quad \forall \boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}_0 \,.$$

Proof. For clarity we adopt the notation $M[c, (\alpha, i)]$ for the entry of $\boldsymbol{M}$ indexed by the codeword $\boldsymbol{c} \in \mathcal{C}_0$ (for the row), and $(\alpha, i) \in S \times I$ (for the column). Given an assertion $\mathcal{U}$, we also denote by $\mathbb{1}_{\mathcal{U}}$ the integer $r \in \{0, 1\}$ such that $r = 1$ if and only if $\mathcal{U}$ is true. Now, let $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}_0$.

$$\begin{aligned}(\boldsymbol{M}\boldsymbol{M}^T)[\boldsymbol{c}, \boldsymbol{c}'] &= \sum_{\alpha \in S, i \in I} M[\boldsymbol{c}, (\alpha, i)] M[\boldsymbol{c}', (\alpha, i)] = \sum_{\alpha \in S, i \in I} \mathbb{1}_{c_i = \alpha} \mathbb{1}_{c'_i = \alpha} \\ &= \sum_{i \in I} \sum_{\alpha \in S} \mathbb{1}_{c_i = c'_i = \alpha} = \sum_{i \in I} \mathbb{1}_{c_i = c'_i} = \ell - \mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') \,. \quad \square\end{aligned}$$

Hence, if some prime p divides ℓ as well as the weight of all the codewords in $\mathcal{C}_0$, then the product $\boldsymbol{M}\boldsymbol{M}^T$ vanishes over any extension of $\mathbb{F}_p$, and $\boldsymbol{M}$ is a parity-check matrix of a code which contains its dual. A more general setting is analysed in the following proposition.

Proposition 4.34. *Let $\mathcal{C}_0$ be a linear code of length ℓ over S, $|S| = s$. Let also $\mathcal{C} = \mathrm{IC}_q(\mathcal{C}_0)$ with $\mathrm{char}(\mathbb{F}_q) = p$. Denote the length of $\mathcal{C}$ by $n = \ell s$, and recall that $\mathrm{Par}(n)$ is the parity-check code of length n over $\mathbb{F}_q$. If $\mathcal{C}_0$ is p-divisible, then*

$$\mathcal{C}^\perp \cap \mathrm{Par}(n) \subseteq \mathcal{C} \,.$$

In particular, we get $\dim(\mathcal{C}) \geq \frac{n-1}{2}$. Moreover, if $p \mid \ell$, then $\mathcal{C}^\perp \subseteq \mathcal{C}$ and $\dim(\mathcal{C}) \geq \frac{n}{2}$.

Proof. Let $\boldsymbol{M}$ be the incidence matrix of the transversal design $\mathcal{T}_{\mathcal{C}_0}$. Also denote by $\boldsymbol{J}$ and $\boldsymbol{J}'$ the all-one matrices of respective size $|\mathcal{C}_0| \times n$ and $|\mathcal{C}_0| \times |\mathcal{C}_0|$. An easy computation shows that

$$\boldsymbol{M}\boldsymbol{J}^T = \ell \boldsymbol{J}' \,.$$

If we assume that $\mathcal{C}_0$ is p-divisible, then Lemma 4.33 translates into

$$\boldsymbol{M}\boldsymbol{M}^T = \ell \boldsymbol{J}' \mod p\,. \tag{4.3}$$

Hence, we obtain

$$\boldsymbol{M}(\boldsymbol{M}-\boldsymbol{J})^T = \boldsymbol{0} \mod p\,. \tag{4.4}$$

It leads us to consider the code $\mathcal{A}$ of length n generated over $\mathbb{F}_q$ by the matrix $\boldsymbol{M}-\boldsymbol{J}$. Equation (4.4) ensures that $\mathcal{A} \subseteq \mathcal{C}$. Let us prove that $\mathcal{C}^\perp \cap \mathrm{Par}(n) \subseteq \mathcal{A}$.

Assume $p \nmid \ell$ and let $\boldsymbol{c} \in \mathcal{C}^\perp \cap \mathrm{Par}(n)$. Then $\boldsymbol{c} = \boldsymbol{u}\boldsymbol{M}$ for some $\boldsymbol{u} \in \mathbb{F}_q^{|\mathcal{C}_0|}$, and $\sum_{i=1}^n c_i = 0$. Hence, $\ell \boldsymbol{u}\boldsymbol{J}' = \boldsymbol{u}\boldsymbol{M}\boldsymbol{J}^T = \boldsymbol{c}\boldsymbol{J}^T = \boldsymbol{0}$. Since $\ell \neq 0$ in $\mathbb{F}_q$ of characteristic p, we get $\boldsymbol{u}\boldsymbol{J}' = \boldsymbol{0}$, which also implies that $\boldsymbol{u}\boldsymbol{J} = \boldsymbol{0}$. Therefore, $\boldsymbol{c} = \boldsymbol{u}(\boldsymbol{M}-\boldsymbol{J}) \in \mathcal{A}$, and it follows that:

$$\mathcal{C}^\perp \cap \mathrm{Par}(n) \subseteq \mathcal{A} \subseteq \mathcal{C}\,.$$

Now, if $p \mid \ell$, then equation (4.3) turns into $\boldsymbol{M}\boldsymbol{M}^T = \boldsymbol{0}$, meaning that $\mathcal{C}^\perp \subseteq \mathcal{C}$. Finally, the first bound on the dimension comes from

$$\dim(\mathcal{C}) \geq \dim(\mathcal{C}^\perp \cap \mathrm{Par}(n)) \geq \dim \mathcal{C}^\perp - 1 = n - \dim \mathcal{C} - 1\,,$$

and the second bound is straightforward. □

In terms of PIR protocols, previous result translates into the following corollary.

Corollary 4.35. *Let p be a prime, and assume there exists a p-divisible linear code $\mathcal{C}_0$ of length ℓ_0 over $\mathbb{F}_q$. Then using the incidence code $\mathrm{IC}_q(\mathcal{C}_0)$, there exists $k \geq (\ell_0 q - 1)/2$ such that we can build a distributed PIR protocol for a k-entry database over $\mathbb{F}_q$, and whose parameters are $\ell(k) = \ell_0$ and $n(k) = \ell_0 q \leq 2k+1$.*

Divisible codes over small fields have been well-studied, and contain for instance the extended Golay codes [MS77, ch.II.6], or the famous family of MDS codes of dimension 3 and length $q+2$ over $\mathbb{F}_q$ [MS77, ch.XI.6].

Example 4.36. The extended binary Golay code is a self-dual $[24, 12, 8]_2$ linear code. It produces a transversal design with 24 groups, each storing 2 points. Its associated incidence code $\mathrm{Code}_2(\mathrm{Golay})$ has length $n = 24 \times 2 = 48$ and dimension $k \geq 24$, and by computation we can show that $k = 24$.

Remark 4.37. In the application for PIR protocols, we would like to find divisible codes $\mathcal{C}_0$ defined over large alphabets (compared to the code length), but these two constraints seem to be inconsistent. For instance, the binary Golay code presented in Example 4.36 leads us to a PIR protocol with a too expensive communication cost (24 bits of communication for an original file of size... 24 bits: that is exactly the communication cost of the trivial PIR protocol where the whole database is downloaded). Nevertheless, Example 4.36 represents the worst possible case for the TD-based construction, in a sense that the rate of $\mathrm{IC}_2(\mathrm{Golay}_2)$ is exactly $1/2$ (it attains the lower bound), and that each server stores 2 bits (which is the smallest possible). Codes with better rate and/or with larger server storage capability would then give PIR protocols with relevant communication complexity. For instance, the extended ternary Golay code gives better parameters — see Example 4.47.

To conclude this section, let us point out that to the best of our knowledge, divisible codes over *large* fields does not seem to have been thoroughly studied (a reason might be that codes over small alphabets are usually considered as the most practical ones). We hope that the TD-based construction of PIR protocols based on divisible codes may encourage research in this direction.

4.4 The t-private construction

When servers are colluding, the PIR protocol based on a simple transversal design does not ensure sufficient privacy, because the knowledge of two points lying on a block gives some information on it. To solve this issue, we propose to use orthogonal arrays with higher strength $t > 2$.

4.4.1 Generic construction and analysis

In the previous section, classical ($t = 2$) orthogonal arrays were used to build transversal designs. Considering higher values of t, we naturally generalise the latter as follows.

Definition 4.38 (t-transversal design)**.** Let $\ell \geq t \geq 1$ and $\lambda \geq 1$. A t-transversal design is a block design $\mathcal{D} = (X, \mathcal{B})$ equipped with a group set $\mathcal{G} = \{G_1, \ldots, G_\ell\}$ partitioning X such that:

- $|X| = s\ell$;
- any group has size s and any block has size ℓ;
- for any $T = \{i_1, \ldots, i_t\} \subseteq [1, \ell]$ with $|T| = t$ and for any $(x_1, \ldots, x_t) \in \prod_{j=1}^{t} G_{i_j}$, there exist exactly λ distinct blocks $B \in \mathcal{B}$ such that $\{x_1, \ldots, x_t\} \subset B$.

A t-transversal design with parameters s, ℓ, t, λ is denoted t-TD$_\lambda(\ell, s)$, or t-TD(ℓ, s) for short, when $\lambda = 1$.

Lemma 4.39. *Let $\ell \geq 2$ and $\lambda \geq 1$. For every $\ell \geq t \geq t' \geq 2$, any t-TD$_\lambda(\ell, s)$ is also a t'-TD$_{\lambda'}(\ell, s)$, where $\lambda' = \lambda s^{t-t'}$.*

Proof. Let $\mathcal{T}$ be a t-TD$_\lambda(\ell, s)$. Let $T' = \{i_1, \ldots, i_t\} \subseteq [1, \ell]$, $|T| = t$, and $\boldsymbol{x} = (x_1, \ldots, x_{t'}) \in \prod_{j=1}^{t'} G_{i_j}$. For every $\boldsymbol{y} = (y_{t'+1}, \ldots, y_t) \in \prod_{j=t'+1}^{t} G_{i_j}$, we can define the tuple $(\boldsymbol{x}, \boldsymbol{y}) \in \prod_{j=1}^{t} G_{i_j}$. Since $\mathcal{T}$ is a t-TD$_\lambda(\ell, s)$, there exists exactly λ blocks B such that $\{x_1, \ldots x_{t'}, y_{t'+1}, \ldots, y_t\} \subseteq B$. Since there are $s^{t-t'}$ possible choices for $\boldsymbol{y}$, and since each choice defines pairwise distinct blocks, we get our result. □

Given a t-transversal design $\mathcal{T}$, we can build a $(t-1)$-private PIR protocol with the exactly the same steps as in Section 4.2. First, we define the design-based code $\mathcal{C} = \mathrm{Code}_q(\mathcal{T})$ according to Definition 1.12, and then we follow the algorithm given in Figure 4.1. As we see in Lemma 4.39, for $t \geq 2$ any t-transversal design is also a 2-transversal design. Hence the analysis of PIR protocols based on t-TDs is identical to the one using classical TDs. Even the security proof is very similar, as shows the following argument.

Security ($(t-1)$-privacy). Let T be a collusion of servers of size $|T| \leq t-1$. We need to prove that $\mathrm{I}(i\,; \boldsymbol{q}_{|T}) = 0$ where $i \in X$ is the coordinate of the desired item, and $\boldsymbol{q}$ is the random query. For every $x_t \in G_{i_t}$ such that $i_t \notin T$, there exist exactly λ blocks that contain the points in $\boldsymbol{q}_{|T} \cup \{x_t\}$. Hence, knowing $\boldsymbol{q}_{|T}$, the distribution of x_t remains balanced. Thus we get $\mathrm{I}(i\,; \boldsymbol{q}_{|T}) = 0$.

To sum up, the following theorem holds:

Theorem 4.40. *Let D be a database with k entries over $\mathbb{F}_q$, and $\mathcal{T} = t$-TD(ℓ, s) be a t-transversal design, whose incidence matrix has rank $\ell s - k$ over $\mathbb{F}_q$. Then, there exists an ℓ-server $(t-1)$-private PIR protocol with:*

- *only* 1 *symbol over* $\mathbb{F}_q$ *to read for each server,*
- $\ell - 1$ *field operations for the user,*
- $\ell \log(sq)$ *bits of communication,*
- *a storage rate* $R_{\mathcal{C}} = k/\ell s$.

4.4.2 Instances

In this subsection we propose instances of t-transversal designs to be used in PIR protocols.

t-transversal designs from curves of degree $\leq t-1$ in the affine plane. Looking for instances of t-transversal designs, it is natural to try to generalise the transversal designs given in Definition 4.12. An idea is to turn affine lines into higher degree curves.

Definition 4.41. Let X be the set of points in the affine plane $\mathbb{F}_q^2$, and $\mathcal{G} = \{G_1, \ldots, G_q\}$ be a partition of $X = \mathbb{F}_q^2$ into q parallel lines. Without loss of generality we choose the following one: $G_i = \{(\alpha_i, x), x \in \mathbb{F}_q\}$ for each $\alpha_i \in \mathbb{F}_q$. Blocks in $\mathcal{B}$ are now defined as sets B_F of the form

$$B_F = \{(F(x), x), x \in \mathbb{F}_q\}, \text{ where } F \in \mathbb{F}_q[T], \deg F \leq t-1.$$

Lemma 4.42. *The design $(X, \mathcal{B}, \mathcal{G})$ given in Definition 4.41 defines a t-transversal design t-$\mathrm{TD}_1(q, q)$.*

Proof. The set of groups $\mathcal{G}$ we propose is indeed a partition of X into q groups, each of size q. It remains to check the incidence property. Let $\{G_{T_1}, \ldots, G_{T_t}\}$ be a set of t distinct groups, and let $((y_{T_1}, x_{T_1}), \ldots, (y_{T_t}, x_{T_t})) \in G_{T_1} \times \cdots \times G_{T_t}$. From Lagrange interpolation theorem, we know there exists a unique polynomial $F \in \mathbb{F}_q[T]$ of degree $\leq t-1$ such that:

$$F(x_{T_j}) = y_{T_j} \quad \forall 1 \leq j \leq t\,.$$

Said differently, there is a unique block B_F which contains the t points $\{(y_{T_j}, x_{T_j})\}_{1 \leq j \leq t}$. □

We postpone the analysis of the dimension of these codes, since it corresponds to a particular case of the generic construction we give below.

t-transversal designs from orthogonal arrays of strength t. Here we give a generic construction of t-transversal designs, which is actually the same as the construction of transversal designs with orthogonal arrays (Subsection 4.3.3).

Construction 4.43. *Let $\boldsymbol{A}$ be an orthogonal array $\mathrm{OA}_\lambda(t, \ell, s)$ define over a symbol set S. Recall that the array $\boldsymbol{A}$ is composed of rows $\boldsymbol{a}_i = (a_{i,j})_{1 \leq j \leq \ell}$ for $1 \leq i \leq \lambda s^t$. We define a design $(X, \mathcal{B}, \mathcal{G})$, where*

- *the point set is $X = S \times [1, \ell]$;*
- *the group set is $\mathcal{G} = \{S \times \{i\}, 1 \leq i \leq \ell\}$;*
- *the blocks are $B_i = \{(a_{i,j}, j), 1 \leq j \leq \ell\}$ for all $\boldsymbol{a}_i \in \mathrm{Rows}(A)$.*

Proposition 4.44. *If $\boldsymbol{A}$ is an $\mathrm{OA}_\lambda(t, \ell, s)$, then the design defined with $\boldsymbol{A}$ by Construction 4.43 is a t-$\mathrm{TD}_\lambda(\ell, s)$.*

Proof. It is clear that $\mathcal{G}$ is a partition of X and that the sets of blocks and groups both have the size we claim. Now focus on the incidence property. Let $T \subset [1, \ell]$ with $|T| = t$, and let $(x_1, \ldots, x_t) \in G_{T_1} \times \ldots \times G_{T_t}$. We need to prove that there are exactly λ blocks $B \in \mathcal{B}$ such that $\{x_1, \ldots, x_t\} \subset B$.

Consider the map from blocks in $\mathcal{B}$ to rows of $\boldsymbol{A}$ given by:

$$\begin{array}{rccc} \psi: & \mathcal{B} & \to & \mathrm{Rows}(\boldsymbol{A}) \\ & B_i = \{(a_{i,j}, j), 1 \leq j \leq \ell\} & \mapsto & (a_{i,1}, \ldots, a_{i,\ell}) \end{array}$$

Since we assumed that orthogonal arrays have no repeated row, the map ψ is one-to-one. Let $\boldsymbol{x} = (x_1, , \ldots, x_t) \in X^t$, where $x_u = (b_u, j_u) \in S \times [1, \ell]$. Denote by $\boldsymbol{b} = (b_1, \ldots, b_t) \in S^t$ the tuple formed by the first coordinates of $\boldsymbol{x}$. From the definition of an orthogonal array of strength t and index λ, we know that elements in $\boldsymbol{b}$ appear exactly λ times in the submatrix of $\boldsymbol{A}$ defined by the columns indexed by T. Hence this defines λ preimages in $\mathcal{B}$, which proves the result. □

Remark 4.45. As we noticed before, Construction 4.41 is a particular case of Construction 4.43. Indeed, a block $B_F = \{(F(x), x), x \in \mathbb{F}_q\}$, with $\deg F \leq t-1$ is in one-to-one correspondence with a codeword $\boldsymbol{c}_F$ of a Reed-Solomon code $\mathrm{RS}(t-1)$.

Corollary 4.46. *Let $\mathcal{C}_0$ be a code of length ℓ and dual distance $t+2 \leq \ell$ over a set S of size s. Then, the PIR protocol defined by $\mathrm{IC}_q(\mathcal{C}_0)$ is t-private.*

Proof. Let $\boldsymbol{A}$ be the orthogonal array defined by $\mathcal{C}_0$. We know that $\boldsymbol{A}$ has strength $t+1$ (see Proposition 4.22), hence from Proposition 4.44, the associated transversal design is a $(t+1)$-$\mathrm{TD}(\ell, s)$. Theorem 4.40 then ensures that the PIR protocol induced by this transversal design is t-private. □

As in Subsection 4.3.4, when the code $\mathcal{C}_0$ is divisible, we can lower bound the rate of its incidence code. We provide two examples in finite (and small) length.

Example 4.47. A first example would be to consider extended Golay codes. Indeed, they are known to be divisible by their characteristic [MS77, ch.II.6], they have large dual distance, and Proposition 4.34 then ensures their incidence codes have non-trivial rate. In Remark 4.37, we noticed that the binary Golay code does not give a practical PIR protocol due to a large communication complexity. Thus, let us instead consider the $[12, 6, 6]_3$ extended ternary Golay code, that we denote Golay_3. It is self-dual, hence $\mathrm{d}^\perp(\mathrm{Golay}_3) = 6$. Then, $\mathcal{C} = \mathrm{IC}_{3^e}(\mathrm{Golay}_3)$, $e \geq 1$, has length 36 and Proposition 4.34 shows that $\dim \mathcal{C} \geq 18$ (the bound can be proved tight by computation). Hence, the associated PIR protocol works on a raw file of 18 symbols over $\mathbb{F}_{3^e}$, encoded into 36, and uses 12 servers (each storing 3 symbols). The upload communication complexity is cheap (only 12 symbols over $\mathbb{F}_3$) while the download communication complexity is much more heavy (12 symbols over the extension field $\mathbb{F}_{3^e}$. The main advantage thus remains the computation cost of the protocol and its resistance to any collusion of one third (*i.e.* 4) of the servers.

Example 4.48. A second example arises from the exceptional $[q+2, 3, q]_q$ MDS codes in characteristic 2 [MS77, ch.XI.6]. For instance, for $q = 4$, we obtain a 2-private PIR protocol with 6 servers, each storing 4 symbols of $\mathbb{F}_{2^e}$ for some $e \geq 1$. Once again, the dimension of the incidence code attains the lower bound, here $k = 12$.

Example 4.49. Examples of incidence codes which do not attain the lower bound of Proposition 4.34 come from binary Reed-Muller codes of order 1, denoted $\mathrm{RM}_2(m, 1)$. These codes are 2-divisible since they are known to be equivalent to extended Hamming codes. They also have length $n = 2^m$ and dual distance $\mathrm{d}^\perp(\mathrm{RM}_2(m, 1)) = n/2$. For instance, $\mathrm{RM}_2(3, 1)$ provides an incidence code of dimension $k = 11 > 8$, that is, a 2-private 8-server PIR protocol on a database with 11 symbols over $\mathbb{F}_{2^e}$, where each server stores 2 symbols. For $m = 4$, $\mathrm{RM}_2(4, 1)$ gives a 6-private 16-server PIR protocol on a database with 20 symbols over $\mathbb{F}_{2^e}$, each server storing 2 symbols. We conjecture that $\mathrm{IC}_2(\mathrm{RM}_2(m, 1))$ leads us to a $(2^{m-1} - 2)$-private 2^m-server PIR protocol on a database with $2^m + m$ symbols, each server storing 2 symbols.

As pointed out in Subsection 4.3.3, incidence codes $\mathcal{C} = \mathrm{IC}(\mathcal{C}_0)$ have the best chance to have large rate if the dimension of $\mathcal{C}_0$ is small, since the cardinality of $\mathcal{C}_0$ is the number of rows in a (non-full-rank) parity-check matrix which defines $\mathcal{C}$. Moreover, in order to define a t-private PIR protocol, we need an orthogonal array of strength $t+2$, *i.e.* a code $\mathcal{C}_0$ with dual distance $t+2$. Conciliating both constraints, we are tempted to use MDS codes of dimension $t+1$ for $\mathcal{C}_0$.

Once again we pick the family of Reed-Solomon codes as an easily example. Through the incidence code construction, these codes lead us to t-private PIR protocols with communication complexity approximately $\sqrt{n}$, where n is the length of the encoded database. For $\mathcal{C}_0 = \mathrm{RS}_{t+1}(\mathbb{F}_q)$ and varying values of q and t, we were able to compute the rate of $\mathrm{IC}_q(\mathcal{C}_0)$. These rates are presented in Figure 4.6 and as expected, the rate of families of incidence codes decreases with t, the privacy parameter. Figure 4.6 also shows that Reed-Solomon-based instances cannot expect to reach at the same time constant information rate and resistance to a constant fraction of colluding servers.

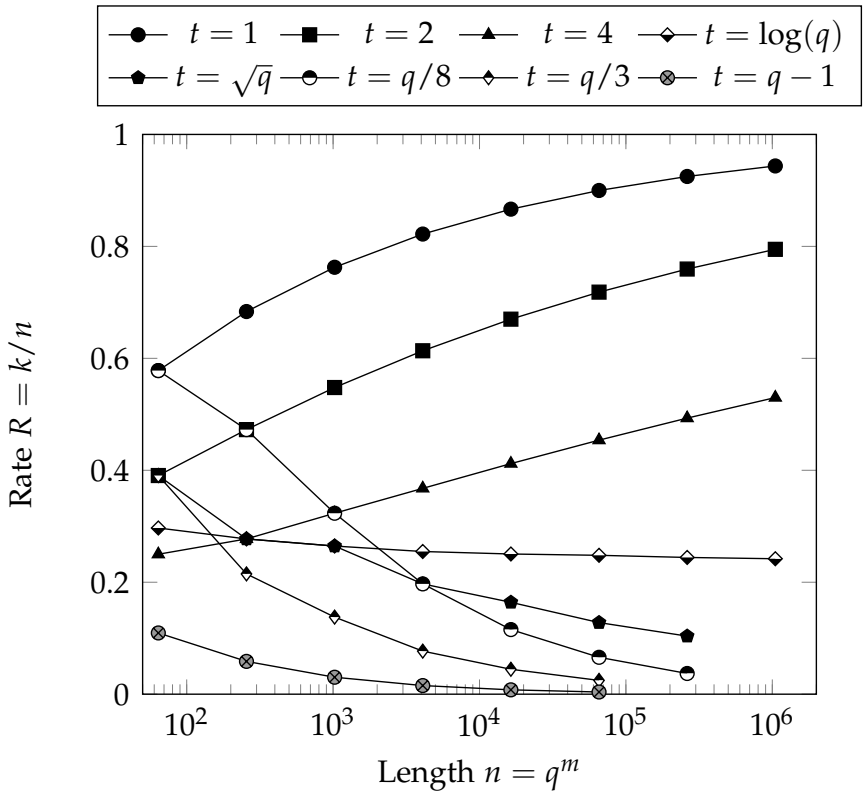


Figure 4.6 – Rate of incidence codes used for the construction of t-private PIR protocols. Base codes $\mathcal{C}_0$ are full-length Reed-Solomon codes of dimension $t+1$ (dual distance $t+2$) over $\mathbb{F}_q$. Associated PIR protocols then need q servers, each storing q symbols.

As a partial conclusion, the question of finding t-transversal designs with large strength t, moderate block size ℓ and low rank remains completely open.

4.5 Recent PIR constructions and bounds

In this section, we conclude the chapter by reporting some constructions of PIR protocols and some bounds on their parameters that appeared in the recent years.

4.5.1 PIR capacity, and capacity achieving protocols

Communication complexity is widely considered as the most crucial parameter in PIR protocols. Hence it is natural to raise the question of determining the minimum possible number of bits exchanged in a PIR protocol, all other parameters being fixed. It is commonly admitted that, theoretically, the upload communication complexity (the size of queries) is negligible compared to the download communication complexity. The usual argument is that the upload cost does not need to scale with the size of the individual message in the database [CHY15], and typical databases store large files. Therefore, given a *fixed* storage system, one would like to know what is the minimum number of bits one must download in order to retrieve privately a desired message.

For a given PIR protocol, the ratio of the number of *downloaded* bits to the message size is called the *PIR rate*. The maximum achievable PIR rate is called the *PIR capacity*.

The PIR capacity of replication-based 1-private PIR protocols were first analysed. Consider a system of ℓ non-communicating servers, each storing a copy of the database consisting of K messages of arbitrary large size L. In this context, Sun and Jafar [SJ17] proved that the PIR capacity c is:

$$c = \frac{1 - 1/\ell}{1 - (1/\ell)^K}\,. \tag{4.5}$$

One should emphasize that messages of size $L \geq \ell^K$ are necessary in their capacity-achieving construction. If one does not bound the number of files stored by the servers (*i.e.* $K \to \infty$), we get $c = 1 - \frac{1}{\ell}$. Notice that, in this setting, PIR capacity has been given and achieved in an earlier work by Shah, Rashmi and Ramchandran in [SRR14].

If the servers store an unbounded number of messages encoded by an $[\ell, R\ell]$-code (each server storing one symbol of each encoded message), Chan, Ho and Yamamoto proved that $c \leq 1 - R$, with equality for MDS codes. Tajeddine and El Rouayheb [TR16] proposed optimal constructions for a larger range of parameters, notably by the use of *file stripings*. This work was finally extended by Kumar, Rosnes and Graell i Amat [KRGiA17] for non-MDS codes. Recently, Banawan and Ulukus [BU18] showed that, when servers store $K < \infty$ messages encoded with an $[\ell, R\ell]$ code, the PIR capacity is:

$$c = \frac{1 - R}{1 - R^K}\,,$$

matching the Equation (4.5) for replication-based PIR protocols.

If servers are colluding (so-called t-private PIR, $t \geq 2$) and store K replicated messages, Sun and Jafar [SJ18b] proved that the PIR capacity is given by

$$c = \frac{1 - t/\ell}{1 - (t/\ell)^K}\,.$$

For the coded case, Freij-Hollanti, Gnilke, Hollanti and Karpuk [FHGHK17] then proposed a construction with PIR rate $\frac{1-(t+k-1)/\ell}{1-((t+k-1)/\ell)^K}$, and conjectured this quantity as being the PIR capacity. While the conjecture for finite K has been disproved in [SJ18a], the techniques involved in their construction are elegant, notably by the use of star-product of codes. This way, they manage to build PIR protocols on $[\ell, k]$-coded data, whose PIR rate equals $1 - (k + t - 1)/\ell$. Note that their *star-product scheme* generalises to arbitrary collusion patterns [TGK+17].

Finally, in the case we allow the storage system to depend on the parameters of the database (*e.g.* the number of servers depend on the size or number of messages), Shah, Rashmi and

Ramchandran [SRR14] proved that one can obtain 1-*private* retrieval with only 1 extra downloaded bit, compared to classical retrieval. However in their construction, the system requires $\ell \geq (L+1)^{K-1}$ servers, where L is the size of each message. Recently, Blackburn, Etzion and Paterson [BEP17] decreased this bound to $\ell \geq L+1$ with an upload complexity $\Theta(KL \log L)$. They also built ℓ-server PIR protocols with small download complexity $(1 + \frac{1}{\ell-1})L$, where $\ell - 1$ can be any divisor of L.

4.5.2 Is PIR rate the only criterion to consider?

Previous subsection mainly focused on optimal PIR protocols with respect to the download communication complexity (only). However, other parameters could be taken into account such as the storage overhead or the computational complexity, as we pointed out in the introduction of this chapter.

In [FVY15a], Fazeli, Vardy and Yaakobi gave a generic transformation of a replication-based PIR protocol into a coded PIR protocol, whose storage overhead is drastically reduced. Informally, the idea is to encode answers to user's queries in a way that preserves privacy, by the use of a so-called *PIR code*. Including the extended version of Fazeli *et al.*'s seminal paper [FVY15b], several works focused on studying bounds on PIR codes and optimal constructions of such codes [RV16, BE17, AY17, Ska18]. Also notice that, with very different techniques, the reduction of the storage overhead was also addressed the works of Shah *et al.* [SRR14], Augot *et al.* [ALS14] and Blackburn *et al.* [BEP17] notably.

In most capacity-achieving PIR protocols, the answer of each server consists in computing a linear combination of all the symbols it holds. Though a few existing schemes admit a low computational complexity on the server side (*e.g.* [ALS14]), this problem was not addressed for a long time. Very recently, Zhang, Yaakobi, Etzion and Schwartz [ZYES18] proposed to consider the *access complexity* — that is, the number of bits read in order to output servers' answers — as a fundamental parameter. The authors analysed trade-offs between PIR rate, storage overhead and access complexity, and gave optimal constructions for some range of parameters by the use of *covering codes*. The PIR constructions we proposed in this chapter were motivated by the reduction of the computational complexity on the server side. However, our strategy differs from [ZYES18], since we fix the access complexity to its minimum value, being the download complexity. Given this requirement, our goal was to build PIR protocols with the minimum PIR rate and maximum storage rate.

Chapter 5

Proofs of retrievability from codes with locality

Contents

This chapter is devoted to the construction of proofs of retrievability (PoRs) which feature low computation complexity on both client and server sides, as well as small client storage (typically 512 bits). We adapt the security model initiated by Juels and Kaliski [JK07] and we use the framework of Paterson *et al.* [PSU13] to analyse our construction. We thus provide a rigorous treatment of the security of the generic design we propose; more precisely, we sharply bound the extraction failure of our protocol according to this security model. Next, we instantiate our formal construction with tensor-product codes, Reed-Muller codes and affine lifted codes. This yields PoRs with moderate communication complexity and server storage overhead, in addition to the aforementioned features.

5.1 Introduction

5.1.1 Motivation

The use of *cloud* services, such as computing and storage, has evolved quite spectacularly over the past decade. In particular, data outsourcing allows users and companies to lighten their storage burden and maintenance cost. Though, it raises several issues: for example, how can a customer check efficiently that he can retrieve without any loss a massive file that he had uploaded on a remote server and erased from his personal system?

Proofs of retrievability (PoRs) address this specific issue. They are cryptographic protocols involving two parts: a client (or a verifier) and a server (or a prover). PoRs usually consist in the following phases. First, a *key generation* creates secret material related to the file, and meant to be kept by the client only. Then the file is *initialised*, that is, it is encoded and/or encrypted according to the secret data held by the client. Only this processed file is uploaded to the server. Then, the client can run a *verification procedure*, which is the core of the PoR. Finally, if the client is convinced that the server still holds his file, the client can proceed at any time to the *extraction* of the file.

In PoRs, several parameters must be taken into account. Plainly, the verification process has to feature a low communication complexity, since the main goal is to avoid to download a large part of the file while one only wants to *check* its extractability. Second, the storage overhead induced by the protocol must be low since the amount of additional data usually impacts the customer's storage fees. Similarly, the computation cost of the verification procedure must be low for the server, but also for the client which is supposed to own a small and computationally restricted device.

Notice that *proofs of data possession* (PDPs) represent protocols close to what is needed in PoRs. However, in PDPs one does not require the client to be able to extract the file from the server. Instances of PDPs are given by Ateniese *et al.* [ABC+11]. The earlier protocols of Lillibridge *et al.* [LEB+03] and Naor and Rothblum [NR09] are very often seen as precursors for PoRs. For instance, Naor and Rothblum consider a setting in which the client directly accesses the file stored by the server. However, the actual definition of PoRs is more restrictive, since according to Shacham and Waters [SW13], PoRs allow the prover to be 'an arbitrary program' answering challenges 'in an arbitrary manner (...) as opposed to a simple memory layout' for the earlier schemes.

5.1.2 Previous works

Juels and Kaliski [JK07] gave the first formal definition of PoRs, from which our own definitions in Section 5.2 are inspired. They also proposed a seminal construction based on so-called *sentinels*: these are random parts of the file the client keeps secretly on his device, and whose presence will be checked during the verification step. Additionally, an erasure code ensures the soundness of the file to be extracted. Juels and Kaliski's seminal work [JK07] also raised several interesting points. On the one hand, it demonstrated that (i) the client must store some secret data to be used in the verification step, and (ii) coding is needed in order to retrieve the file without erasures or errors. On the other hand it highlighted a possible weakness for PoRs: in the scheme they propose, the verification step can only be performed a finite number of times, since sentinels cannot be reused many times.

As a consequence, Shacham and Waters proposed to build PoRs with *unbounded-use* [SW13]. Precisely, two families of PoRs are built. The first one is based on linear combinations of authenticators produced *via* pseudo-random functions; its security is proved using cryptographic tools such as unforgeable MAC schemes, semantically secure symmetric encryption and secure PRF. The second one is a publicly verifiable scheme based on Diffie-Hellman problem in groups equipped with a bilinear map.

Bowers, Juels and Oprea [BJO09] modelled PoRs with so-called *inner* and *outer codes*. The encoded file which is uploaded lies in the outer code, while the inner code models the space in which the server's answers theoretically evolve. The authors then adopted this coding-theoretic approach to compare variants of the constructions given in [JK07, SW13]. They focused on the practical efficiency of the schemes, and proved that, despite bounded-use, optimised variants of the construction of Juels and Kaliski are highly competitive compared to other existing schemes.

In [PSU13], Paterson, Stinson and Upadhyay provide a general framework for PoRs in the unconditional security model. They show how the question of the retrievability of the file can be expressed as an error-correction problem in a so-called *response code*. This allows them to precisely quantify the extraction success as a function of the success probability of a proving algorithm: indeed, in this setting, extraction can be naturally seen as nearest-neighbour decoding in the response code. They notably apply their framework to examine the security of a modified version of Shacham-Waters scheme. Also notice that, prior to [PSU13], Dodis, Vahan and Wichs [DVW09] proposed another coding-theoretic model for PoRs that allowed them to build efficient bounded-use and unbounded-use PoR schemes.

With practicality in mind, other features have been deployed for PoRs. For instance, Wang *et al.* [WWR$^+$11] presented a PoR construction based on Merkle hash trees, which allows efficient *file updates* on the server. Their scheme is provably secure under cryptographic assumptions (hardness of Diffie-Hellman in groups with bilinear maps, unforgeable signatures, etc.), and has been improved by Mo, Zhou and Chen [MZC12] in order to prevent unbalanced trees. More recently, other features have been proposed for PoRs, such as multi-prover PoRs (see for instance the unpublished work [PSU18] of Paterson, Stinson and Upadhyay), or PoRs with public verifiability (*e.g.* Sengupta and Ruj's construction [SR16]).

5.1.3 Our approach

As we noticed earlier, most PoR schemes rely on two techniques: (i) the client stores on his device some secret data in order to check the integrity of the file and (ii) the client encodes the file in order to repair a small number of erasures/errors that may remain unnoticed by the Verifier during the verification steps.

In this chapter, we propose to build a generic PoR scheme based on codes with local properties. When equipped with a suitable cipher, we prove that the construction fulfils both previous requirements. More precisely, our idea is the following. Given a file F, a code $\mathcal{C}$ and a suitable cipher $\mathtt{E}_\kappa$, the client sends to the server an encoded and encrypted version $\boldsymbol{w} = \mathtt{E}_\kappa(\mathcal{C}(F))$ of his file[1]. Then, the verification step consists in checking short relations between symbols of $\boldsymbol{w}$, that arise for instance from low-weight parity-check equations for $\mathcal{C}$. Finally, if the file is considered as extractable, the code $\mathcal{C}$ provides the redundancy necessary to repair erasures and potential unnoticed errors.

[1] the abusive notation $\mathcal{C}(F)$ is used to suggest that the specific encoding map of F in $\mathcal{C}$ is meaningless

We here develop ideas published in [LL16], where we proposed a construction of PoRs based on lifted codes. In this chapter we provide a more generic construction, and give a sharper analysis of the security of the PoR.

The construction of PoRs we propose does not feature updatability nor public verifiability. Though, we emphasize its generality since it is based on well-studied and broad families of algebraic structures, namely codes and their parity-check equations. From a practical perspective, we highlight two main attributes of the construction. First, the client must only store the few bits corresponding to the secret material necessary to the cipher. Second, an honest server simply needs to read pieces of $\boldsymbol{w}$ during the verification step, and therefore has no computational burden compared to many other PoR schemes.

5.1.4 Organisation

Section 5.2 is devoted to the definition of proofs of retrievability and their security model. Despite the great disparity of models in PoR literature, we try to keep close to the definitions given in [JK07, PSU13] for the sake of uniformity. Section 5.3 presents our construction of PoR. Precisely, in Subsection 5.3.1, we introduce objects called *verification structures for a code* $\mathcal{C}$ that will be used in the definition of our PoR scheme (Subsection 5.3.2). A rigorous analysis of our scheme is the purpose of the remainder of that section. The performance of our construction are given in Section 5.4. We then provide several instances in Section 5.5, proving the practicality of our PoR schemes for some classes of codes.

5.2 Proofs of retrievability

5.2.1 Definition

In the context of proofs of retrievability, a user wants to estimate if a message $\boldsymbol{m} \in \mathcal{M}$ can be retrieved from a encoded version $\boldsymbol{w} \in \mathcal{W}$ of the message stored on a server. In all what follows, the user will be known as the Verifier (wants to verify the retrievability of the message) while the server is the Prover (aims at proving the retrievability). We also denote by $\mathcal{K}$ the set of *secret values* (or *keys*) kept by the Verifier, and by $\mathcal{R}$ the space of responses to challenges.

Definition 5.1 (Proof of retrievability)**.** A keyed *proof of retrievability* (PoR) is a tuple of algorithms (KeyGen, Init, Verify, Extract) running as follows:

1. The *key generation* algorithm KeyGen generates uniformly at random a key $\kappa \leftarrow_{\mathrm{R}} \mathcal{K}$. The key κ is secretly kept by the Verifier.
2. The *initialisation algorithm* Init is a deterministic algorithm which takes as input a message $\boldsymbol{m} \in \mathcal{M}$ and a key $\kappa \in \mathcal{K}$, and outputs a file $\boldsymbol{w} \in \mathcal{W}$. Init is run by the Verifier who initially holds the message $\boldsymbol{m}$. After the process, the file $\boldsymbol{w}$ is sent to the Prover and the message $\boldsymbol{m}$ is erased on Verifier's side. Upon receipt of $\boldsymbol{w}$, the Prover sets a deterministic algorithm $\mathsf{P}^{(w)}$ that will be run during the verification procedure.
3. The *verification algorithm* Verify is a randomised algorithm initiated by the Verifier which needs a secret key $\kappa \in \mathcal{K}$, and interacts with the Prover. Verify is depicted in Figure 5.1 and works as follows:
   (i) the Verifier runs a random query generator that outputs a challenge $u \in \mathcal{Q}$ ($\mathcal{Q}$ being the so-called *query set*);
   (ii) the challenge u is sent to the Prover;
   (iii) the Prover outputs a response $\boldsymbol{r}_u \leftarrow \mathsf{P}^{(w)}(u) \in \mathcal{R}$;

(iv) the Verifier checks the validity of $\boldsymbol{r}_u$ according to u and κ by the means of a subroutine $\mathsf{Check}(u, \boldsymbol{r}_u, \kappa)$, which outputs `True` if $\boldsymbol{r}_u$ is compliant with u and κ.

4. The *extraction algorithm* Extract is run by the Verifier. It takes as input κ and $\boldsymbol{r} = (\boldsymbol{r}_u : u \in \mathcal{Q}) \in \mathcal{R}^{\mathcal{Q}}$, and outputs either a message $\boldsymbol{m}' \in \mathcal{M}$, or a failure symbol $\perp$. We say that extraction *succeeds* if $\mathsf{Extract}(\boldsymbol{r}, \kappa) = \boldsymbol{m}$.

The vector $\boldsymbol{r} = (\boldsymbol{r}_u \leftarrow \mathsf{P}^{(\boldsymbol{w})}(u))_{u \in \mathcal{Q}} \in \mathcal{R}^{\mathcal{Q}}$ is called the *response word* associated to $\mathsf{P}^{(\boldsymbol{w})}$.

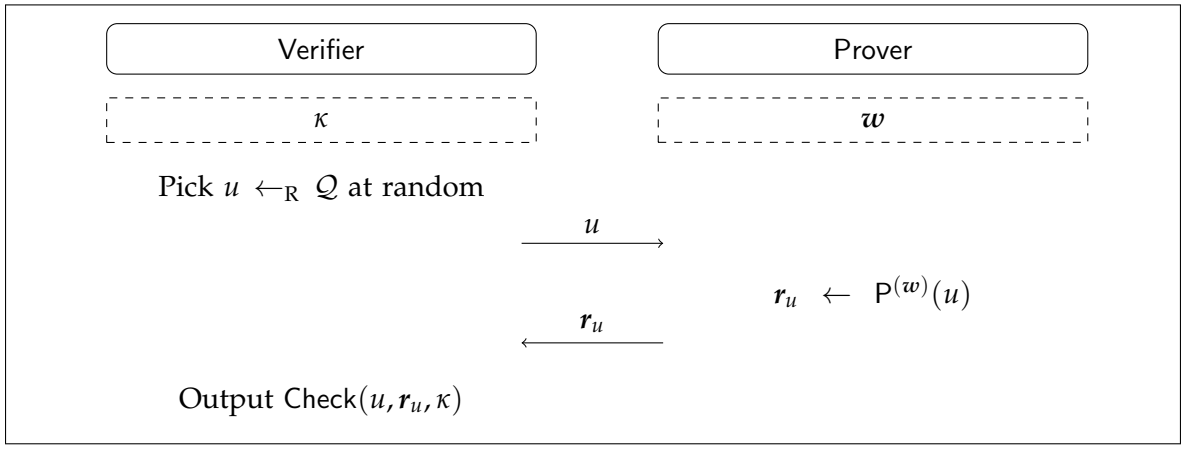


Figure 5.1 – Definition of algorithm Verify

Note that, in assuming that response algorithms $\mathsf{P}^{(\boldsymbol{w})}$ are deterministic and non-adaptive (meaning that their behaviour only depends on the *value* of the challenge u, and not on the success of past calls to the verification algorithm), we follow the work of Paterson *et al.* [PSU13]. The authors justify determinism of response algorithms by the fact that any probabilistic prover can be replaced by a deterministic prover whose success probability is at least as good as the probabilistic one.

In Definition 5.1, we can see that the deterministic algorithm $\mathsf{P}^{(\boldsymbol{w})}$ can be represented by the vector of its outputs $\boldsymbol{r} = (\mathsf{P}^{(\boldsymbol{w})}(u), u \in \mathcal{Q})$, called the response word of $\mathsf{P}^{(\boldsymbol{w})}$. Therefore, we can assume that before the verification step, the Prover produces a word $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ related to the file $\boldsymbol{w}$ he holds. In other words, we model provers as algorithms P which, given an input $\boldsymbol{w}$, return a word $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$.

Following [PSU13], we also assume in this chapter that the extraction algorithm Extract is deterministic, though in general it can be randomised. Finally, notice that proofs of retrievability aim at *proving* the extractability of a file. The extraction algorithm is therefore a theoretical tool for obtaining such a proof. Hence, its computational efficiency is not a crucial feature.

The following table summarises the information held by the two main entities after the initialisation step:

Verifier	Prover
κ	$\boldsymbol{w}$

Let us also report the inputs and outputs of the algorithms involved in a PoR:

algorithm	KeyGen	Init	Verify	Check	Extract
input	1^{λ}	$\boldsymbol{m}, \kappa$	$\boldsymbol{r}, \kappa$	$u, \boldsymbol{r}_u, \kappa$	$\boldsymbol{r}, \kappa$
output	κ	$\boldsymbol{w}$	`True` or `False`	`True` or `False`	$\boldsymbol{m}'$ or $\perp$

5.2.2 Security models

One should first notice that, despite many efforts, proofs of retrievability lack a general agreement on the definition of their security model. For the sake of uniformity, our definitions remain very close to the ones given in the original work of Juels and Kaliski [JK07].

For a response word $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ given by the Prover and a key $\kappa \in \mathcal{K}$ kept by the Verifier, we first define the *success of* $\boldsymbol{r}$ *according to* κ as:

$$\mathrm{succ}(\boldsymbol{r}, \kappa) := \Pr\left(\mathsf{Check}(u, \boldsymbol{r}_u, \kappa) = \mathtt{True}\right),$$

where the probability is taken over the internal randomness of Verify. A first security model can be defined as follows.

Definition 5.2 (security model, strong version)**.** Let $\varepsilon, \tau \in [0,1]$. A proof of retrievability $(\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Verify}, \mathsf{Extract})$ is *strongly* (ε, τ)*-sound* if, for every file $\boldsymbol{m} \in \mathcal{M}$ and every prover $\mathsf{P} : \mathcal{W} \to \mathcal{R}^{\mathcal{Q}}$ we have:

$$\Pr\left(\begin{array}{c|l} \mathsf{Extract}(\boldsymbol{r}, \kappa) \neq \boldsymbol{m} & \kappa \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(\mathbf{1}^{\lambda}) \\ \text{and} & \boldsymbol{w} \leftarrow \mathsf{Init}(\boldsymbol{m}, \kappa) \\ \mathrm{succ}(\boldsymbol{r}, \kappa) \geq 1 - \varepsilon & \boldsymbol{r} \leftarrow \mathsf{P}^{(\boldsymbol{w})} \end{array}\right) \leq \tau, \tag{5.1}$$

the probability being taken over the internal randomness of KeyGen.

Let us propose a short scenario to illustrate this definition. The Verifier is the owner of a file $\boldsymbol{m} \in \mathcal{M}$, and uploads on a server a file $\boldsymbol{w} \in \mathcal{W}$ computed according to the initialisation phase of an (ε, τ)-sound PoR. During a certain period of time, the Verifier proceeds to several verification procedures for his file. This allows him to estimate[2] the value of $\mathrm{succ}(\boldsymbol{r}, \kappa) \in (0,1)$. Notice that the Verifier may only choose the most recent verification procedures, since we need to assume that the response word $\boldsymbol{r}$ does not change over time. Finally, if the estimate convinces the Verifier that $\mathrm{succ}(\boldsymbol{r}, \kappa) \geq 1 - \varepsilon$, then he knows that he can theoretically retrieve his file from the server, with very high probability $\geq 1 - \tau$.

Regarding parameters ε and τ, in the light of the above it is highly desirable to have τ very small. For its part, the parameter ε measures the rate of unsuccessful audits which leads the Verifier to believe the extraction will fail. Therefore, one does not necessarily need to have large ε, though in practice large values of ε afford flexibility, for instance if communication errors occur between the Prover and the Verifier during the verification procedure.

Definition 5.2 provides a strong security model, in the sense that (i) it does not require any bound on the response algorithms given by the Prover (ii) the probability in (5.1) is taken over fixed messages $\boldsymbol{m}$ (informally, it means the Prover knows $\boldsymbol{m}$). However, keyed proofs of retrievability are usually insecure according to Definition 5.2. For instance, in [PSU13] Paterson *et al.* noticed that, given the knowledge of $\boldsymbol{m}$ and $\boldsymbol{w}$, in the Shacham-Waters scheme [SW13] an unbounded Prover may be able to

1. compute (or at least guess) a key κ such that $\mathsf{Init}(\boldsymbol{m}, \kappa) = \boldsymbol{w}$,
2. build $\boldsymbol{m}' \neq \boldsymbol{m}$ such that $\mathsf{Init}(\boldsymbol{m}', \kappa) = \boldsymbol{w}'$, and
3. define a malicious response word $\boldsymbol{r}' \leftarrow \mathsf{P}^{(\boldsymbol{w}')}$ which (i) successfully passes every audit and (ii) leads to the extraction of $\boldsymbol{m}' \neq \boldsymbol{m}$.

Hence, we choose to use a weaker but still realistic security model, where informally, the Prover only knows what he stores (that is, $\boldsymbol{w}$) and has no information on the initial message $\boldsymbol{m}$. Therefore, this security model is also conform with the one given by Paterson *et al.*

[2] this can be formally done with estimation theory, as we can see for instance in [PSU13]

Definition 5.3 (security model, weak version)**.** Let $\varepsilon, \tau \in [0,1]$. A proof of retrievability $(\mathsf{KeyGen}, \mathsf{Init}, \mathsf{Verify}, \mathsf{Extract})$ is *weakly* (ε,τ)*-sound* (or simply (ε,τ)*-sound*) if, for every prover $\mathsf{P} : \mathcal{W} \to \mathcal{R}^{\mathcal{Q}}$, we have:

$$\Pr\left(\begin{array}{c|l} \mathsf{Extract}(\boldsymbol{r},\kappa) \neq \boldsymbol{m} & \boldsymbol{m} \leftarrow_{\mathrm{R}} \mathcal{M}, \\ \text{and} & \kappa \leftarrow_{\mathrm{R}} \mathsf{KeyGen}(\mathbf{1}^{\lambda}) \\ \mathrm{succ}(\boldsymbol{r},\kappa) \geq 1-\varepsilon & \boldsymbol{w} \leftarrow \mathsf{Init}(\boldsymbol{m},\kappa) \\ & \boldsymbol{r} \leftarrow \mathsf{P}^{(w)} \end{array}\right) \leq \tau\,,$$

Since we deal with values of τ very close to 0, we also say that a strongly (ε,τ)-sound PoR admits $\lambda = -\log_2(\tau)$ bits of security.

Informally, stating that a PoR is *not* weakly sound amounts to finding a deterministic algorithm P which

- takes as input a file $\boldsymbol{w} \in \mathcal{W}$ and outputs a response word $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$,
- makes the extraction fail with non-negligible probability (over messages $\boldsymbol{m} \in \mathcal{M}$ and keys $\kappa \in \mathcal{K}$ such that the corresponding response words are successfully audited).

5.3 The PoR construction

Schematically, in the initialisation phase of our construction the Verifier

(i) encodes his file according to a code $\mathcal{C}$;
(ii) scrambles the resulting codeword using a tuple of permutations over the base field;
(iii) uploads the result to the Prover.

As we explained in the introduction, the verification step then consists in checking that the server is still able to give answers that, once decrypted, satisfy low-weight parity-check equations for $\mathcal{C}$. For this purpose, we introduce objects called *verification structures* for codes, that will be used in the definition of our generic PoR scheme.

5.3.1 Verification structures for codes

Let $1 \leq \ell \leq n$, and u be an ℓ-subset of $[1,n]$. For $\boldsymbol{w} \in \mathbb{F}_q^n$, the word $\boldsymbol{w}_{|u} \in \mathbb{F}_q^u$ can be seen as a word in $\mathbb{F}_q^\ell$ *via* a bijective map $[1,\ell] \to u$ which corresponds to an ordering of elements in the set u. Throughout the chapter will abuse this identification between $\mathbb{F}_q^u$ and $\mathbb{F}_q^\ell$, that we will hide behind the notation $\mathcal{R}$.

Definition 5.4 (Verification structure)**.** Let $1 \leq \ell \leq n$ and $\mathcal{C} \subseteq \mathbb{F}_q^n$ be a code. Let also $\mathcal{Q}$ be a non-empty set of ordered ℓ-subsets of $[1,n]$. We define the collection R of *restriction maps* associated to $\mathcal{Q}$ as:

$$\begin{array}{rccc} R: & \mathcal{Q} \times \mathbb{F}_q^n & \to & \mathcal{R} \\ & (u, \boldsymbol{w}) & \mapsto & \boldsymbol{w}_{|u} \end{array}$$

where $\mathcal{R} \simeq \mathbb{F}_q^\ell$ as we explained previously. Given an integer $s \geq 1$ and a linear map $V : \mathcal{Q} \times \mathcal{R} \to \mathbb{F}_q^s$, we say that $(\mathcal{Q}, V)$ is a *verification structure* for $\mathcal{C}$ if the following two constraints are fulfilled:

1. for all $i \in [1,n]$, there exists $u \in \mathcal{Q}$ such that $i \in u$;

2. for all $u \in \mathcal{Q}$, the linear map $\mathbb{F}_q^n \to \mathbb{F}_q^s$ given by $\boldsymbol{a} \mapsto V(u, R(u, \boldsymbol{a}))$ is surjective and vanishes on the code $\mathcal{C}$. Explicitly,

$$\forall \boldsymbol{c} \in \mathcal{C}, V(u, R(u, \boldsymbol{c})) = \mathbf{0}\,.$$

The map V is then called a *verification map*, and the set $\mathcal{Q}$ a *query set* for $\mathcal{C}$. By convention, for $\boldsymbol{w} \in \mathbb{F}_q^n$ and $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, we denote by

$$R(\boldsymbol{w}) \coloneqq (R(u, \boldsymbol{w}) : u \in \mathcal{Q}) \in \mathcal{R}^{\mathcal{Q}},$$
$$V(\boldsymbol{r}) \coloneqq (V(u, \boldsymbol{r}_u) : u \in \mathcal{Q}) \in (\mathbb{F}_q^s)^{\mathcal{Q}}.$$

Finally, the code $R(\mathcal{C}) \coloneqq \{R(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{C}\}$ is called the *response code* of $\mathcal{C}$, similarly to [PSU13].

We recall that for convenience, if u is an ordered ℓ-subset of $[1, n]$, the set $\mathcal{R}$ represents either $\mathbb{F}_q^\ell$ or $\mathbb{F}_q^u$ which can be easily identified.

Remark 5.5. In Definition 5.4, the first constraint on $(\mathcal{Q}, V)$ ensures that each symbol of the file will be part of at least one query, and thus can be checked during the verification phase. The second constraint formalises the needs to distinguish non-corrupted files from corrupted ones. On the one side, a non-corrupted file $\boldsymbol{c}$ satisfies $V(u, R(u, \boldsymbol{c})) = \mathbf{0}$; on the other side, from the surjectivity of $\boldsymbol{a} \mapsto V(u, R(u, \boldsymbol{a}))$, we hope that files $\boldsymbol{a}$ with several missing or noisy symbols can be discerned. Finally, we allow flexibility in the soundness of the verification map V, which is represented by the dimension $s \geq 1$ of the codomain of V. Very informally, the larger the s, the lower the probability that a malicious server produces a response which vanishes on the verification map V.

Construction 5.6. *Let $\mathcal{C}$ be a code, and $\mathcal{H}$ be a set of parity-check equations for $\mathcal{C}$ of Hamming weight ℓ whose supports are pairwise distinct. Define the query set $\mathcal{Q} = \{\mathrm{supp}(\boldsymbol{h}), \boldsymbol{h} \in \mathcal{H}\}$, and for any $u \in \mathcal{Q}$, denote by $\boldsymbol{h}^{(u)}$ the unique parity-check equation in $\mathcal{H}$ whose support is u. Finally, we define a map V by:*

$$\begin{array}{rccc} V: & \mathcal{Q} \times \mathcal{R} & \to & \mathbb{F}_q \\ & (u, \boldsymbol{x}) & \mapsto & \sum_{i \in u} h_i^{(u)} x_i\,. \end{array}$$

By construction, it is clear that $(\mathcal{Q}, V)$ is a verification structure for $\mathcal{C}$.

Construction 5.7 (using generalised design-based codes)**.** *Let $\mathcal{D} = (X, \mathcal{B})$ be a design whose blocks $B \in \mathcal{B}$ have constant size ℓ. Let also $\mathcal{L} = (\mathcal{L}_B \subseteq \mathbb{F}_q^B : B \in \mathcal{B})$ be a family of local codes, each of dimension $\ell - s$. For every $B \in \mathcal{B}$, we denote by $H_B : \mathcal{R} \simeq \mathbb{F}_q^B \to \mathbb{F}_q^s$ a linear map such that $\ker(H_B) = \mathcal{L}_B$. Finally, let $\mathcal{C} = \mathrm{Code}_q(\mathcal{D}, \mathcal{L}) \subseteq \mathbb{F}_q^X$ the generalised design-based code associated to $\mathcal{D}$ and $\mathcal{L}$ (see Subsection 2.4.3 for the definition of generalised design-based codes).*

We can define a query set $\mathcal{Q} \coloneqq \mathcal{B}$, and a verification map:

$$\begin{array}{rccc} V: & \mathcal{B} \times \mathcal{R} & \to & \mathbb{F}_q^s \\ & (B, \boldsymbol{x}) & \mapsto & H_B(\boldsymbol{x})\,. \end{array}$$

Once again we see that $(\mathcal{Q}, V)$ is a verification structure for $\mathcal{C}$. Indeed by definition, if $\boldsymbol{c} \in \mathcal{C}$, then $\boldsymbol{c}_{|B} \in \mathcal{L}_B$ for every $B \in \mathcal{B}$.

Example 5.8. Let $\mathcal{C} = \mathrm{Had}(3) \subseteq \mathbb{F}_2^7$ be the binary Hadamard code of length $n = 7$ and dimension $k = 3$. We know that $\mathcal{C}$ is the code based on the Fano plane, *i.e.* the $(7, 3, 1)$-block-design $\mathrm{PG}_1(2, 2) = (X, \mathcal{B})$. Hence one can use Construction 5.7 to obtain a verification structure. As in Example 1.8, we write the point set $X = [1, 7]$ and the block set is therefore:

$$\mathcal{B} = \{\{1,2,3\}, \{1,4,5\}, \{1,6,7\}, \{2,5,6\}, \{2,4,7\}, \{3,4,6\}, \{3,5,7\}\}\,.$$

Recall that according to Construction 5.7, we define $\mathcal{Q} = \mathcal{B}$. Then, the verification map $V : \mathcal{Q} \times \mathbb{F}_2^3 \to \mathbb{F}_2$ can be defined as follows: if $u = \{u_1, u_2, u_3\} \in \mathcal{B}$ and $\boldsymbol{b} \in \mathbb{F}_2^u$, we define

$$V(u, \boldsymbol{b}) = \sum_{i=1}^{3} b_{u_i} .$$

Now, let $\boldsymbol{m} = (m_1, m_2, m_3) \in \mathbb{F}_2^3$. According to the previous ordering of points and blocks, the message $\boldsymbol{m}$ can be encoded into

$$\boldsymbol{c} = (m_1,\ m_2,\ m_1 + m_2,\ m_3,\ m_1 + m_3,\ m_1 + m_2 + m_3,\ m_2 + m_3) \in \mathcal{C} .$$

Hence, the word $\boldsymbol{r} = R(\boldsymbol{c}) \in (\mathbb{F}_2^3)^7$ is:

$$\begin{aligned} \boldsymbol{r} &= \left(\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}, \begin{pmatrix} c_1 \\ c_4 \\ c_5 \end{pmatrix}, \begin{pmatrix} c_1 \\ c_6 \\ c_7 \end{pmatrix}, \begin{pmatrix} c_2 \\ c_5 \\ c_6 \end{pmatrix}, \begin{pmatrix} c_2 \\ c_4 \\ c_7 \end{pmatrix}, \begin{pmatrix} c_3 \\ c_4 \\ c_6 \end{pmatrix}, \begin{pmatrix} c_3 \\ c_5 \\ c_7 \end{pmatrix} \right) \\ &= \left(\begin{pmatrix} m_1 \\ m_2 \\ m_1 + m_2 \end{pmatrix}, \begin{pmatrix} m_1 \\ m_3 \\ m_1 + m_3 \end{pmatrix}, \begin{pmatrix} m_1 \\ m_1 + m_2 + m_3 \\ m_2 + m_3 \end{pmatrix}, \begin{pmatrix} m_2 \\ m_1 + m_3 \\ m_1 + m_2 + m_3 \end{pmatrix}, \right. \\ &\qquad \left. \begin{pmatrix} m_2 \\ m_3 \\ m_2 + m_3 \end{pmatrix}, \begin{pmatrix} m_1 + m_2 \\ m_3 \\ m_1 + m_2 + m_3 \end{pmatrix}, \begin{pmatrix} m_1 + m_2 \\ m_1 + m_3 \\ m_2 + m_3 \end{pmatrix} \right) \end{aligned}$$

For each coordinate $\boldsymbol{b} = \boldsymbol{r}_i \in \mathbb{F}_2^3$ of $\boldsymbol{r} = R(\boldsymbol{c})$, one can now check that $\sum_j b_j = 0$. Hence, we get $V(R(\boldsymbol{c})) = \boldsymbol{0}$, as expected.

From now on, we denote by $N = |\mathcal{Q}|$ the length of the response code $R(\mathcal{C})$ of a code $\mathcal{C}$ equipped with a verification structure $(\mathcal{Q}, V)$.

5.3.2 A PoR scheme based on codes with locality

For pedagogical purposes, let us first present in Figure 5.2 a naive construction of PoR that unfortunately fails. It is based on a linear code $\mathcal{C}$ and its verification structure $(\mathcal{Q}, V)$. We can assume that $\mathcal{C}$ is public, and we denote by $\mathcal{R} = \mathbb{F}_q^\ell$ and $\mathcal{W} = \mathbb{F}_q^n$.

In short, in the tentative PoR scheme given in Figure 5.2, the client verifies parity-check equations in the codeword $\boldsymbol{w}$ that has been uploaded on the server. Hence such a PoR would admit several advantages: first, it requires no computation for the server; second, it can achieve low communication complexity if the parity-check equations have low weight; third, the storage cost for the client is zero.

However, the protocol proposed in Figure 5.2 does not define a sound proof of retrievability. Indeed, a malicious server can very easily produce a response word $\boldsymbol{r}^{(\mathrm{mal})}$ which (i) always pass the verification procedure, and (ii) leads to the extraction of a file different from $\boldsymbol{m}$. For instance, let $\boldsymbol{w}^{(\mathrm{mal})} \neq \boldsymbol{w} \in \mathcal{C}$, and define $\boldsymbol{r}^{(\mathrm{mal})} = R(\boldsymbol{w}^{(\mathrm{mal})})$. Then it is clear that $V(u, \boldsymbol{r}_u^{(\mathrm{mal})}) = 0$ for every challenge $u \in \mathcal{Q}$. However, the extraction procedure outputs the message $\boldsymbol{m}^{(\mathrm{mal})}$ whose encoded version is $\boldsymbol{w}^{(\mathrm{mal})}$, and we have $\boldsymbol{m}^{(\mathrm{mal})} \neq \boldsymbol{m}$.

To tackle the attack on the protocol, we need to prevent the Prover to create *other* codewords than $\boldsymbol{w}$, the original one. The solution we propose makes use of permutations, aiming at breaking the structure of $\mathcal{C}$. Let us formalise this idea.

• **Key generation:** No key generation.
• **Initialisation:** The Verifier first encodes his file $\boldsymbol{m} \in \mathbb{F}_q^k$ into a codeword $\boldsymbol{w} \in \mathcal{C}$. The word $\boldsymbol{w}$ is sent to the Prover.
• **Verification:** Assume the Prover produced a response word $\boldsymbol{r} \leftarrow \mathsf{P}^{(w)}$.
1. The Verifier picks uniformly at random $u \leftarrow_{\mathrm{R}} \mathcal{Q}$, where $u = \{u_1, \ldots, u_\ell\}$. Then, the Verifier sends u to the Prover, who is asked to send back $R(u, \boldsymbol{w}) = \boldsymbol{w}_{|u} \in \mathcal{R}$.
2. The Prover sends back to the Verifier the u-th coordinate $\boldsymbol{r}_u \in \mathcal{R}$ of $\boldsymbol{r}$.
3. On input $\boldsymbol{r}_u \in \mathcal{R}$, the Verifier checks whether $V(u, \boldsymbol{r}_u) = 0$.

• **Extraction:** The Verifier first collects $\boldsymbol{r} = (\mathsf{P}^{(w)}(u) : u \in \mathcal{Q}) \in \mathcal{R}^{\mathcal{Q}}$. He then defines a word $\boldsymbol{r}' \in (\mathcal{R} \cup \{\perp\})^{\mathcal{Q}}$ as follows:

$$\boldsymbol{r}'_u = \begin{cases} \boldsymbol{r}_u & \text{if } V(u, \boldsymbol{r}_u) = 0, \\ \perp & \text{otherwise.} \end{cases}$$

Then he calls a bounded-distance error-and-erasure decoding algorithm for $R(\mathcal{C})$ with input $\boldsymbol{r}' \in (\mathcal{R} \cup \{\perp\})^{\mathcal{Q}}$. It outputs either a word $\boldsymbol{m} \in \mathbb{F}_q^k$, or the failure symbol $\perp$.

Figure 5.2 – A tentative PoR scheme that is not sound.

Let $(\mathcal{Q}, V)$ be a verification structure for $\mathcal{C} \subseteq \mathbb{F}_q^n$, and let $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n) \in \mathfrak{S}(\mathbb{F}_q)^n$ be an n-tuple of permutations which acts on $\mathbb{F}_q^n$ by $\boldsymbol{\sigma}(\boldsymbol{x}) := (\sigma_1(x_1), \ldots, \sigma_n(x_n))$. We define $\boldsymbol{\sigma}(\mathcal{C}) = \{\boldsymbol{\sigma}(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{C}\}$. Let also

$$\begin{array}{rccc} V^{\boldsymbol{\sigma}} : & \mathcal{Q} \times \mathcal{R} & \to & \mathbb{F}_q^s \\ & (u, \boldsymbol{y}) & \mapsto & V(u, \boldsymbol{\sigma}_{|u}^{-1}(\boldsymbol{y})) \end{array}$$

where $\boldsymbol{\sigma}_{|u}^{-1}(\boldsymbol{y}) = (\sigma_{u_1}^{-1}(y_1), \ldots, \sigma_{u_\ell}^{-1}(y_\ell))$. The map $V^{\boldsymbol{\sigma}}$ is designed in order to satisfy

$$V^{\boldsymbol{\sigma}}(u, R(u, \boldsymbol{\sigma}(\boldsymbol{c}))) = V(u, R(u, \boldsymbol{c}))$$

for every $(\boldsymbol{c}, u) \in \mathcal{C} \times \mathcal{Q}$. Based on these definitions, we propose a PoR construction based on codes, given in Figure 5.3. This construction is generic in the sense that any code $\mathcal{C}$ equipped with a verification structure $(\mathcal{Q}, V)$ could be used. Notice that, in the light of Constructions 5.6 and 5.7, any code admits a verification structure, although it might be too poor to lead to PoRs with good parameters. More details concerning the required properties for these structures will be given in the next section.

Algorithm 13: The extraction procedure $\mathsf{Extract}(\boldsymbol{r}, \boldsymbol{\sigma})$.

Input: $\boldsymbol{\sigma} \in \mathfrak{S}(\mathbb{F}_q)^n$ and $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$.
Output: $\boldsymbol{m} \in \mathbb{F}_q^k$, or a failure symbol $\perp$.

1 Define $\boldsymbol{r}' = \boldsymbol{\sigma}^{-1}(\boldsymbol{r})$.
2 On challenges $u \in \mathcal{Q}$ such that $V(u, \boldsymbol{r}'_u) \neq 0$, assign $\boldsymbol{r}'_u \leftarrow \perp$, where $\perp$ here denotes the erasure symbol.
3 Run a bounded-distance error-and-erasure decoding algorithm for $R(\mathcal{C})$ with input $\boldsymbol{r}' \in (\mathcal{R} \cup \{\perp\})^{\mathcal{Q}}$. It outputs either a word $\boldsymbol{m}' \in \mathbb{F}_q^k$, or the failure symbol $\perp$.
4 Return this output.

The code $\mathcal{C}$ and the verification structure $(\mathcal{Q}, V)$ for $\mathcal{C}$ are public parameters. We assume that $\mathcal{C}$ is linear, and we denote by $N = |\mathcal{Q}|$. Finally, let $\mathcal{R} = \mathbb{F}_q^\ell$ and $\mathcal{W} = \mathbb{F}_q^n$.

• **Key generation:** The Verifier generates uniformly at random an n-tuple of permutations

$$(\sigma_1, \ldots, \sigma_n) = \boldsymbol{\sigma} \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n .$$

• **Initialisation:** The Verifier first encodes his file $\boldsymbol{m} \in \mathbb{F}_q^k$ into a codeword $\boldsymbol{c} \in \mathcal{C}$ with a encoding algorithm for $\mathcal{C}$. Then, the Verifier scrambles each coordinate c_i using the permutation σ_i:

$$w_i = \sigma_i(c_i), \quad 1 \leq i \leq n .$$

Finally, $\boldsymbol{w} \in \mathcal{W}$ is sent to the Prover, and $\boldsymbol{m}$ is erased by the Verifier. To sum up, the deterministic algorithm Init is defined by

$$\boldsymbol{w} = \mathsf{Init}(\boldsymbol{m}, \boldsymbol{\sigma}) \coloneqq \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m})) \in \mathcal{W} .$$

Based on his knowledge of $\boldsymbol{w}$ and public parameters, the Prover produces a word $\boldsymbol{r} \leftarrow \mathsf{P}^{(\boldsymbol{w})}$, where $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, corresponding to the vector of outputs of the deterministic proving algorithm P on input $\boldsymbol{w}$.

• **Verification:**

1. The Verifier picks uniformly at random $u \leftarrow_{\mathrm{R}} \mathcal{Q}$, where $u = \{u_1, \ldots, u_\ell\}$. Then, the Verifier sends u to the Prover, meaning the Prover is asked to send back $R(u, \boldsymbol{w}) = \boldsymbol{w}_{|u} \in \mathbb{F}_q^\ell$ to the Verifier.
2. The Prover sends back the u-th restriction $\boldsymbol{r}_u \in \mathcal{R}$ of his response word $\boldsymbol{r}$ to the Verifier.
3. On input $\boldsymbol{r}_u \in \mathcal{R}$, the Verifier runs $V^{\boldsymbol{\sigma}}(u, \boldsymbol{r}_u)$ and outputs the result. Here we mean that:

$$\mathsf{Check}(u, \boldsymbol{r}_u, \boldsymbol{\sigma}) \coloneqq \begin{cases} \texttt{True} & \text{if } V^{\boldsymbol{\sigma}}(u, \boldsymbol{r}_u) = 0 \\ \texttt{False} & \text{otherwise.} \end{cases}$$

• **Extraction:** The Verifier first collects $\boldsymbol{r} = (\mathsf{P}^{(\boldsymbol{w})}(u) : u \in \mathcal{Q}) \in \mathcal{R}^{\mathcal{Q}}$. Then, he runs the extraction procedure given in Algorithm 13 on inputs $\boldsymbol{\sigma}$ and $\boldsymbol{r}$, and outputs his result.

Figure 5.3 – A PoR scheme based on a code equipped with a verification structure

5.3.3 Analysis

Let us now analyse the correctness and the soundness of the PoR scheme presented in Figure 5.3.

Preliminary results. We first give results concerning verification structures and response codes. The following two lemmata are straightforward to prove.

Lemma 5.9. *Let $(\mathcal{Q}, V)$ be a verification structure for a code $\mathcal{C} \subseteq \mathbb{F}_q^n$. Then $(\mathcal{Q}, V^{\boldsymbol{\sigma}})$ is a verification structure for $\boldsymbol{\sigma}(\mathcal{C})$.*

Lemma 5.10. *Let $(\mathcal{Q}, V)$ be a verification structure for a code $\mathcal{C} \subseteq \mathbb{F}_q^n$. Then its response code $R(\mathcal{C})$ is an $\mathbb{F}_q$-linear code over the alphabet $\mathcal{R} \simeq \mathbb{F}_q^\ell$.*

Remark 5.11. By considering $\boldsymbol{\sigma}(\mathcal{C})$, we lose $\mathbb{F}_q$-linearity, but one can check that verification structures still make sense and provide the result claimed in Lemma 5.9.

The next result states that the action $\mathcal{C} \mapsto \boldsymbol{\sigma}(\mathcal{C})$ does not modify the distance between codewords.

Lemma 5.12. *Let also $\boldsymbol{\sigma} \in \mathfrak{S}(\mathbb{F}_q)^n$, and R be the collection of restriction maps associated to a query set $\mathcal{Q}$. Then the following two maps*

$$\begin{array}{rccc} \boldsymbol{\sigma}: & \mathbb{F}_q^n & \to & \mathbb{F}_q^n \\ & \boldsymbol{w} & \mapsto & \boldsymbol{\sigma}(\boldsymbol{w})\,. \end{array}$$

and

$$\begin{array}{rccc} \boldsymbol{\sigma}': & R(\mathbb{F}_q^n) & \to & R(\mathbb{F}_q^n) \\ & R(\boldsymbol{w}) & \mapsto & R(\boldsymbol{\sigma}(\boldsymbol{w}))\,. \end{array}$$

are isometries of the respective metric spaces $(\mathbb{F}_q^n, \mathrm{d})$ and $(R(\mathbb{F}_q^n), \mathrm{d})$, where d is the Hamming distance.

Proof. Since every σ_i is one-to-one, for every $\boldsymbol{c}, \boldsymbol{c}' \in \mathcal{C}$ we get

$$\begin{aligned} \mathrm{d}(\boldsymbol{c}, \boldsymbol{c}') &= |\{i \in [1,n], c_i \neq c_i'\}| \\ &= |\{i \in [1,n], \sigma_i(c_i) \neq \sigma_i(c_i')\}| \\ &= \mathrm{d}(\boldsymbol{\sigma}(\boldsymbol{c}), \boldsymbol{\sigma}(\boldsymbol{c}'))\,. \end{aligned}$$

The proof for response codes relies on the same argument. □

Hence, if $\mathcal{C}$ is linear, then the minimum distance of $R(\boldsymbol{\sigma}(\mathcal{C}))$ is the minimum *weight* of $R(\mathcal{C})$, since $R(\mathcal{C})$ is linear thanks to Lemma 5.10.

Definition 5.13. Let $\varepsilon \in [0,1]$ and $(\mathcal{Q}, V)$ be a verification structure for a code $\mathcal{C} \subseteq \mathbb{F}_q^n$. Denote by $N := |\mathcal{Q}|$. We say $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ is *ε-close* to $(\mathcal{Q}, V)$ if

$$\mathrm{wt}(V(\boldsymbol{r})) = |\{u \in \mathcal{Q}, V(u, \boldsymbol{r}_u) \neq 0\}| \leq \varepsilon N\,.$$

Let now $\boldsymbol{c} \in \mathcal{C}$ and $\beta \in [0,1]$. We say that $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ is a *β-liar* for $(\mathcal{Q}, V, \boldsymbol{c})$ if

$$|\{u \in \mathcal{Q} \mid V(u, \boldsymbol{r}_u) = 0 \text{ and } \boldsymbol{r}_u \neq R(u, \boldsymbol{c})\}| \leq \beta N\,.$$

Finally, if $\boldsymbol{r}_u = R(u, \boldsymbol{c})$ we say that $\boldsymbol{r}_u$ is *authentic*.

Let $\mathcal{A} \subseteq \mathbb{F}_q^n$ be any code of minimum distance d, and let $\boldsymbol{a} \in \mathcal{A}$ be corrupted with b errors and e erasures, resulting in a word $\boldsymbol{r} \in (\mathbb{F}_q \cup \{\perp\})^n$. Then, it is well-known that, as long as $2b + e < d$, it is possible to retrieve $\boldsymbol{a}$ from $\boldsymbol{r}$ thanks to a *bounded-distance error-and-erasure decoding algorithm* (be it efficient or not). This is precisely the decoding algorithm that we employ in Algorithm 13 on the code $\mathcal{A} = R(\mathcal{C})$.

Our framework allows us to reformulate the extraction success in terms of a probability to decode corrupted codewords. More precisely:

Proposition 5.14. *Let $(\mathcal{Q}, V)$ be a verification structure for a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, and denote by $N = |\mathcal{Q}|$. Let $\boldsymbol{\sigma} \in \mathfrak{S}(\mathbb{F}_q)^n$, $\boldsymbol{m} \in \mathbb{F}_q^k$ and denote by d the minimum distance of $R(\mathcal{C})$. Let also $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ and $\boldsymbol{w} = \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m}))$. Finally, assume that $\boldsymbol{r}$ is ε-close to $(\mathcal{Q}, V^{\sigma})$ and is also a β-liar for $(\mathcal{Q}, V^{\sigma}, \boldsymbol{w})$, with $(\varepsilon + 2\beta)N < d$. Then, $\mathsf{Extract}(\boldsymbol{r}, \boldsymbol{\sigma}) = \boldsymbol{m}$, where $\mathsf{Extract}(\boldsymbol{r}, \boldsymbol{\sigma})$ is defined in Algorithm 13.*

Proof. Recall that $\boldsymbol{r}' \in (\mathcal{R} \cup \{\perp\})^{\mathcal{Q}}$ represents the word we get from $\boldsymbol{r}$ after step 2 of Algorithm 13. Let us now translate our assumptions on $\boldsymbol{r}$ in coding-theoretic terminology:

- $\boldsymbol{r}$ is ε-close to $(\mathcal{Q}, V^\sigma)$ means there are at most εN challenges $u \in \mathcal{Q}$ for which we *know* that the coordinate $\boldsymbol{r}'_u$ is not authentic. This justifies that we assign erasure symbols to these coordinates.
- $\boldsymbol{r}$ is a β-liar for $(\mathcal{Q}, V, c)$ means there are at most βN other corrupted values $\boldsymbol{r}'_u$, but we *cannot identify* them. Therefore we can assimilate these coordinates to errors.

To sum up, we see $\boldsymbol{r}'$ as a corruption of $R(\mathcal{C}(m))$ with at most εN erasures and at most βN errors, where $N = |\mathcal{Q}|$. Since we assume that $(\varepsilon + 2\beta)N < d$, we know from the previous discussion that the decoding succeeds to retrieve $\boldsymbol{m}$. □

Bounding the extraction failure. According to Definition 5.3, our PoR scheme is weakly (ε, τ)-sound if for every algorithm P outputting a response word $\boldsymbol{r}^{(w)}$ from a file $\boldsymbol{w}$, we have

$$\Pr\left(\begin{array}{c|c} \text{decoding } \boldsymbol{r}^{(w)} \text{ into } \boldsymbol{m} \text{ fails} & \boldsymbol{m} \leftarrow_{\mathrm{R}} \mathbb{F}_q^k, \\ \text{and} & \boldsymbol{\sigma} \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n, \\ \mathrm{wt}(V^\sigma(\boldsymbol{r}^{(w)})) \leq \varepsilon N & \boldsymbol{w} = \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m})) \end{array}\right) \leq \tau\,.$$

Using Proposition 5.14, the analyse of the security of our PoR scheme reduces to measuring the Prover's capability to produce a response word $\boldsymbol{r}$ which is ε-close to $(\mathcal{Q}, V^\sigma)$ and a β-liar for $(\mathcal{Q}, V^\sigma, \boldsymbol{w})$, with $(\varepsilon + \beta N) < d$.

For fixed $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, $\boldsymbol{\sigma} \in \mathfrak{S}(\mathbb{F}_q)^n$ and $\boldsymbol{w} = \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m}))$ the authentic file given to the prover, we define:

- $\mathcal{D}(\boldsymbol{r}, \boldsymbol{w}) := \{u \in \mathcal{Q}, \boldsymbol{r}_u \neq R(\boldsymbol{w})_u\}$ and $D(\boldsymbol{r}, \boldsymbol{w}) := |\mathcal{D}(\boldsymbol{r}, \boldsymbol{w})| = \mathrm{wt}(\boldsymbol{r} - R(\boldsymbol{w}))$. This represents challenges u on which the response word $\boldsymbol{r}$ differs from the authentic one $R(\boldsymbol{w})$.
- $\mathcal{E}(\boldsymbol{r}, \boldsymbol{\sigma}) := \{u \in \mathcal{Q}, V^\sigma(u, \boldsymbol{r}_u) \neq 0\}$ and $E(\boldsymbol{r}, \boldsymbol{\sigma}) := |\mathcal{E}(\boldsymbol{r}, \boldsymbol{\sigma})| = \mathrm{wt}(V^\sigma(\boldsymbol{r}))$. These are challenges u on which the associated coordinate $\boldsymbol{r}_u$ is not accepted by the verification map (it corresponds to erasures in the decoding process),
- $\mathcal{B}(\boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{w}) := \{u \in \mathcal{Q}, \boldsymbol{r}_u \neq R(\boldsymbol{w})_u \text{ and } V^\sigma(u, \boldsymbol{r}_u) = 0\}$ and $B(\boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{m}) := |\mathcal{B}(\boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{m})|$. These are the challenges u on which the associated coordinate $\boldsymbol{r}_u$ is accepted by the verification map, but differs from the authentic response (it corresponds to errors in the decoding process).

One can easily check that the sets $\mathcal{E}(\boldsymbol{r}, \boldsymbol{\sigma})$ and $\mathcal{B}(\boldsymbol{r}, \boldsymbol{\sigma}, \boldsymbol{w})$ define a partition of $\mathcal{D}(\boldsymbol{r}, \boldsymbol{w})$. The probability of extraction failure can thus be written:

$$\Pr\left(\begin{array}{c|c} 2D(\boldsymbol{r}, \boldsymbol{w}) - E(\boldsymbol{r}, \boldsymbol{\sigma}) \geq \mathrm{d}_{\min}(R(\mathcal{C})) & \boldsymbol{m} \leftarrow_{\mathrm{R}} \mathbb{F}_q^k, \\ \text{and} & \boldsymbol{\sigma} \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n, \\ E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \varepsilon N & \boldsymbol{w} \leftarrow \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m})) \end{array}\right). \tag{5.2}$$

Given $\boldsymbol{w} \in \mathbb{F}_q^n$ and $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, let us also rewrite the space of *admissible* permutations and messages:

$$\Phi_{\boldsymbol{w}} := \{(\boldsymbol{\sigma}, \boldsymbol{m}) \in \mathfrak{S}(\mathbb{F}_q)^n \times \mathbb{F}_q^k,\ \boldsymbol{w} = \boldsymbol{\sigma}(\mathcal{C}(\boldsymbol{m}))\}\,.$$

We also define

$$\alpha(\boldsymbol{r}, \boldsymbol{w}) := \max_{u \in \mathcal{D}(\boldsymbol{r}, \boldsymbol{w})} \Pr_{\Phi_{\boldsymbol{w}}}(V^\sigma(u, \boldsymbol{r}_u) = 0)\,.$$

and $\alpha := \max_{(\boldsymbol{r}, \boldsymbol{w})} \alpha(\boldsymbol{r}, \boldsymbol{w})$ where $(\boldsymbol{r}, \boldsymbol{w})$ are such that $D(\boldsymbol{r}, \boldsymbol{w}) \neq 0$. Here the notation $\Pr_{\Phi_{\boldsymbol{w}}}$ refers to the fact that $\boldsymbol{\sigma}$ is uniformly drawn from $\Phi_{\boldsymbol{w}}$. Similarly we will use notation $\mathbb{E}_{\Phi_{\boldsymbol{w}}}$ and $\mathbb{D}_{\Phi_{\boldsymbol{w}}}$.

The parameter $\alpha \in (0,1)$ is called the *bias* of the verification structure $(\mathcal{Q}, V)$ for $\mathcal{C}$. It corresponds to the maximum probability that a response is accepted but does not correspond to the original file.

Lemma 5.15. *For all $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ and $\boldsymbol{w} \in \mathbb{F}_q^n$, we have:*

$$\mathbb{E}_{\Phi_w}(E(\boldsymbol{r}, \boldsymbol{\sigma})) \geq (1-\alpha) D(\boldsymbol{r}, \boldsymbol{w}) .$$

Proof. A simple computation shows that:

$$\begin{aligned}
\mathbb{E}_{\Phi_w}(E(\boldsymbol{r}, \boldsymbol{\sigma})) &= \mathbb{E}_{\Phi_w}\Big(\sum_{u \in \mathcal{D}(\boldsymbol{r}, \boldsymbol{w})} \mathbb{1}_{V^{\sigma}(u, \boldsymbol{r}_u) \neq 0} \Big) \\
&= \sum_{u \in \mathcal{D}(\boldsymbol{r}, \boldsymbol{w})} \Pr_{\Phi_w}(V^{\sigma}(u, \boldsymbol{r}_u) \neq 0) \\
&\geq \sum_{u \in \mathcal{D}(\boldsymbol{r}, \boldsymbol{w})} (1-\alpha) = (1-\alpha) D(\boldsymbol{r}, \boldsymbol{w}) .
\end{aligned}$$

□

Lemma 5.15 essentially means that, if an adversary to our PoR scheme wants its response word to be (in average) ε-close to the verification structure, then he must modify at most $D(\boldsymbol{r}, \boldsymbol{w}) \leq \frac{\varepsilon N}{1-\alpha}$ responses. Below we take advantage of this fact and we measure the probability of an extraction failure.

First, for $\delta, \varepsilon \in (0,1)$, denote by

$$\begin{aligned}
p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta) &\coloneqq \Pr_{\Phi_w}\Big(2D(\boldsymbol{r}, \boldsymbol{w}) - E(\boldsymbol{r}, \boldsymbol{\sigma}) \geq \delta N \text{ and } E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \varepsilon N \Big) \\
&= \Pr_{\Phi_w}\Big(E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \min\{\varepsilon N, 2D(\boldsymbol{r}, \boldsymbol{w}) - \delta N\} \Big) .
\end{aligned}$$

The probability $p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta)$ represents the probability that the extraction fails for a response code of relative distance δ and an adversarial response word $\boldsymbol{r}$ associated to $\boldsymbol{w}$, which is ε-close to the verification structure. Recall that $\mathbb{D}(X)$ represents the variance of a random variable X, and let us bound $p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta)$.

Proposition 5.16. *Let $\delta, \varepsilon \in (0,1)$ such that $\delta \frac{1-\alpha}{1+\alpha} > \varepsilon$. Let also $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ and $\boldsymbol{w} \in \mathbb{F}_q^n$. Then we have:*

$$p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta) \leq \frac{\mathbb{D}_{\Phi_w}(E(\boldsymbol{r}, \boldsymbol{\sigma}))}{\left(\frac{1+\alpha}{2}\left(\delta \frac{1-\alpha}{1+\alpha} - \varepsilon\right)\right)^2 N^2} .$$

Proof. We distinguish three cases.

1. Case $2D(\boldsymbol{r}, \boldsymbol{w}) - \delta N < 0$. The event $E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \min\{\varepsilon N, 2D(\boldsymbol{r}, \boldsymbol{w}) - \delta N\}$ never occurs since $E(\boldsymbol{r}, \boldsymbol{\sigma}) \geq 0$. Hence $p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta) = 0$.
2. Case $\varepsilon N \leq 2D(\boldsymbol{r}, \boldsymbol{w}) - \delta N$. In that case we have $p(\boldsymbol{r}, \boldsymbol{w}; \varepsilon, \delta) = \Pr_{\Phi_w}(E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \varepsilon N)$. Moreover, if $E(\boldsymbol{r}, \boldsymbol{\sigma}) \leq \varepsilon N$ holds, then we also have

$$\begin{aligned}
E(\boldsymbol{r}, \boldsymbol{\sigma}) - \mathbb{E}_{\Phi_w}(E) &\leq \varepsilon N - (1-\alpha) D(\boldsymbol{r}, \boldsymbol{w}) \\
&\leq \varepsilon N - (1-\alpha) \frac{\varepsilon + \delta}{2} N \\
&\leq -\frac{1+\alpha}{2}\left(\delta \frac{1-\alpha}{1+\alpha} - \varepsilon\right) N .
\end{aligned}$$

Hence,

$$p(\boldsymbol{r},\boldsymbol{w};\varepsilon,\delta) \leq \Pr_{\Phi_w}\left(|E(\boldsymbol{r},\boldsymbol{\sigma}) - \mathbb{E}_{\Phi_w}(E)| \geq \frac{1+\alpha}{2}\left(\delta\frac{1-\alpha}{1+\alpha} - \varepsilon\right)N\right)$$

Finally, using Chebyshev's inequality we get

$$p(\boldsymbol{r},\boldsymbol{w};\varepsilon,\delta) \leq \frac{\mathbb{D}_{\Phi_w}(E(\boldsymbol{r},\boldsymbol{\sigma}))}{\left(\frac{1+\alpha}{2}\left(\delta\frac{1-\alpha}{1+\alpha} - \varepsilon\right)\right)^2 N^2}\,.$$

3. Case $0 \leq 2D(\boldsymbol{r},\boldsymbol{w}) - \delta N < \varepsilon N$. The argument is similar to the second case. Here, $E(\boldsymbol{r},\boldsymbol{\sigma}) \leq 2D(\boldsymbol{r},\boldsymbol{w}) - \delta N$ leads us to

$$\begin{aligned} E(\boldsymbol{r},\boldsymbol{\sigma}) - \mathbb{E}_{\Phi_w}(E) &\leq (1+\alpha)D(r,w) - \delta N \\ &\leq (1+\alpha)\frac{\varepsilon+\delta}{2}N - \delta N \\ &\leq -\frac{1+\alpha}{2}\left(\delta\frac{1-\alpha}{1+\alpha} - \varepsilon\right)N. \end{aligned}$$

Therefore, similarly to the previous case, we obtain the result we claim. □

Now it remains to bound $\mathbb{D}_{\Phi_w}(E(\boldsymbol{r},\boldsymbol{\sigma}))$. For any $u \in \mathcal{D}(\boldsymbol{r},\boldsymbol{w})$, denote by X_u the $\{0,1\}$-random variable $\mathbb{1}_{V^\sigma(u,\boldsymbol{r}_u)=0}$, when $\boldsymbol{\sigma}$ is uniformly drawn from $\Phi_{\boldsymbol{w}}$. It holds that $E(\boldsymbol{r},\boldsymbol{\sigma}) = \sum_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}(1-X_u)$. Also recall that two real random variables Y, Z are uncorrelated if $\mathbb{E}(YZ) = \mathbb{E}(Y)\mathbb{E}(Z)$. If so, it holds that $\mathbb{D}(YZ) = \mathbb{D}(Y) + \mathbb{D}(Z)$. For instance, two independent random variables are uncorrelated.

Lemma 5.17. *Let $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ and $\boldsymbol{w} \in \mathbb{F}_q^n$. If the random variables $\{X_u\}_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are pairwise uncorrelated, then:*

$$\mathbb{D}_{\Phi_w}(E(\boldsymbol{r},\boldsymbol{\sigma})) \leq D(\boldsymbol{r},\boldsymbol{w})\,.$$

Proof. By assumption, the random variables $\{X_u\}_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are pairwise uncorrelated, hence the variables $\{1-X_u\}_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are too. Therefore we get:

$$\mathbb{D}_{\Phi_w}(E(\boldsymbol{r},\boldsymbol{\sigma})) = \sum_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}\mathbb{D}_{\Phi_w}(X_u)\,.$$

The bound $\mathbb{D}_{\Phi_w}(X_u) \leq 1$ then gives the result. □

As a corollary of Proposition 5.16 and Lemma 5.17, under the same hypothesis and assuming $\delta\frac{1-\alpha}{1+\alpha} > \varepsilon$, we get

$$p(\boldsymbol{r},\boldsymbol{w};\varepsilon,\delta) \leq \frac{4}{N\left((1-\alpha)\delta - (1+\alpha)\varepsilon\right)^2}$$

since $D(\boldsymbol{r},\boldsymbol{w}) \leq N$. Moreover, if $\lim_{N\to\infty}\delta > 0$ and $\lim_{N\to\infty}\alpha = 0$, then $p(\boldsymbol{r},\boldsymbol{w};\varepsilon,\delta) = \mathcal{O}(1/N)$.

Therefore, we end up with the following theorem.

Theorem 5.18. *Let $(\mathcal{Q},V)$ be a verification structure for $\mathcal{C}$ with bias α. Denote by $N = |\mathcal{Q}|$ and $\delta = \mathrm{d}_{\min}(R(\mathcal{C}))/N$ the relative distance of the response code of $\mathcal{C}$. Finally, assume that, for any $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ and any $\boldsymbol{w} \in \mathbb{F}_q^n$ the variables $\{X_u\}_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are pairwise uncorrelated. Then, for any $\varepsilon < \delta\frac{1-\alpha}{1+\alpha}$, the PoR scheme associated to $\mathcal{C}$ and $(\mathcal{Q},V)$ is (ε,τ)-sound, where*

$$\tau = \frac{4}{N\left((1-\alpha)\delta - (1+\alpha)\varepsilon\right)^2}\,.$$

According to Theorem 5.18, we need to look for (sequences of) codes $\mathcal{C}$ and associated verification structures $(\mathcal{Q}, V)$ such that:

1. the response code $R(\mathcal{C})$ admits a good relative distance $\delta = \mathrm{d}_{\min}(R(\mathcal{C}))/N$,
2. the bias α is small,
3. random variables $\{X_u\}_{u \in \mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are pairwise uncorrelated.

Subsections 5.3.4 and 5.3.5 characterise conditions under which the last two points are fulfilled. Then, in Section 5.5 we study some response codes in the perspective of achieving good relative distance.

5.3.4 Estimation of the bias

In this subsection we prove that, assuming $\Phi_{\boldsymbol{w}}$ approximates the uniform distribution over $\mathfrak{S}(\mathbb{F}_q)^n$ in a sense we will make precise later, the bias α can be bounded according to the parameters of the verification structure.

Let us fix $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, $\boldsymbol{w} \in \mathbb{F}_q^n$ and $u \in \mathcal{Q}$. We recall that α is defined by:

$$\alpha = \max_{\boldsymbol{r},\boldsymbol{w}} \max_{u \in \mathcal{D}(\boldsymbol{r},\boldsymbol{w})} \Pr_{\Phi_{\boldsymbol{w}}}(V^{\sigma}(u, \boldsymbol{r}_u) = 0)$$

where the randomness comes from $\sigma \leftarrow_{\mathrm{R}} \Phi_{\boldsymbol{w}} = \{(\sigma, \boldsymbol{m}) \in \mathfrak{S}(\mathbb{F}_q)^n \times \mathbb{F}_q^k, \boldsymbol{w} = \sigma(\mathcal{C}(\boldsymbol{m}))\}$. We notice that this is equivalent to writing $\sigma \leftarrow_{\mathrm{R}} \{\sigma \in \mathfrak{S}(\mathbb{F}_q)^n, \sigma^{-1}(\boldsymbol{w}) \in \mathcal{C}\}$.

Recall that for convenience we see $\boldsymbol{r}_u \in \mathcal{R} \simeq \mathbb{F}_q^u$ as a vector indexed by the subset $u \subset [1, n]$. Hence we can easily denote by $r_u[i] \in \mathbb{F}_q$ the coordinate of $\boldsymbol{r}_u$ indexed by $i \in u$. We define the code $K_u \coloneqq \ker(V(u, \cdot)) \subseteq \mathbb{F}_q^u$, and by definition of V we have $\mathcal{C}_{|u} \subseteq K_u$. Moreover, for every $\sigma \in \mathfrak{S}(\mathbb{F}_q)^n$, we have $V^{\sigma}(u, \boldsymbol{r}_u) = 0$ if and only if $\sigma_u^{-1}(\boldsymbol{r}_u) \in K_u$. Finally, we denote by $Z_u \coloneqq \{i \in u, r_u[i] \neq R(\boldsymbol{w})_u[i]\}$ the set of coordinates of $\boldsymbol{r}_u$ that are not authentic.

Let $Y_u(\sigma)$ represent the event '$\sigma_u^{-1}(\boldsymbol{r}_u) \in K_u \mid \mathrm{supp}(\sigma_u^{-1}(\boldsymbol{r}_u)) = Z_u$'. We say that $\Phi_{\boldsymbol{w}}$ is *sufficiently uniform* if, for every $u \in \mathcal{Q}$, we have:

$$\gamma_u \coloneqq \frac{\Pr\left[Y_u(\sigma) \mid \sigma \leftarrow_{\mathrm{R}} \Phi_w\right] - \Pr\left[Y_u(\sigma) \mid \sigma \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n\right]}{\Pr\left[Y_u(\sigma) \mid \sigma \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n\right]} = o(1)\,,$$

when the file size $n \log(q) \to \infty$.

In other words, $\Phi_{\boldsymbol{w}}$ is sufficiently uniform if, when we measure the probability that $Y_u(\sigma)$ happens, the random distribution induced by $\Phi_{\boldsymbol{w}}$ is a good approximation of the uniform distribution over n-tuples of permutations.

Lemma 5.19. *Let $\boldsymbol{r}$, $\boldsymbol{w}$, u and Z_u be defined as above. Let also $A_u = |\{\boldsymbol{x} \in K_u, \mathrm{supp}(\boldsymbol{x}) = Z_u\}|$. Then*

$$\Pr_{\Phi_{\boldsymbol{w}}}(V^{\sigma}(u, \boldsymbol{r}_u) = 0) \leq \frac{(1 + \gamma_u) A_u}{(q-1)^{|Z_u|}}\,.$$

Proof. For every $\sigma \in \Phi_{\boldsymbol{w}}$, we know that $\sigma_u^{-1}(R(\boldsymbol{w})_u) \in K_u$, and we recall that $V^{\sigma}(u, \boldsymbol{r}_u) = 0$ if and only if $\sigma_u^{-1}(\boldsymbol{r}_u) \in K_u$. Since K_u is linear, and up to considering $\sigma_u^{-1}(R(u, \boldsymbol{w}) - \boldsymbol{r}_u)$ instead, let us assume without loss of generality that $\sigma_u^{-1}(R(\boldsymbol{w})_u) = \boldsymbol{0}$. Hence, by definition of Z_u, it holds that $\sigma_u^{-1}(\boldsymbol{r}_u)[i] = 0$ for every $i \in u \setminus Z_u$.

When σ is drawn uniformly in $\mathfrak{S}(\mathbb{F}_q)^n$, then we have

$$\Pr\left(\sigma_u^{-1}(\boldsymbol{r}_u) \in K_u \;\middle|\; \begin{array}{c}\sigma \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n \\ \mathrm{supp}(\sigma_u^{-1}(\boldsymbol{r}_u)) = Z_u\end{array}\right) = \Pr\left(\boldsymbol{x} \in K_u \;\middle|\; \begin{array}{c}\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_q^\ell \\ \mathrm{supp}(\boldsymbol{x}) = Z_u\end{array}\right).$$

Furthermore,

$$\Pr\left(\boldsymbol{x} \in K_u \;\middle|\; \begin{array}{c}\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_q^\ell \\ \mathrm{supp}(\boldsymbol{x}) = Z_u\end{array}\right) = \frac{A_u}{(q-1)^{|Z_u|}},$$

since A_u counts the number of codewords in K_u having Z_u as support. Therefore we get

$$\begin{aligned}
\mathrm{Pr}_{\Phi_w}(V^\sigma(u,\boldsymbol{r}_u)=0) &\geq \mathrm{Pr}_{\Phi_w}\left(V^\sigma(u,\boldsymbol{r}_u)=0 \;\middle|\; \mathrm{supp}(\sigma_u^{-1}(\boldsymbol{r}_u)) = Z_u\right) \\
&= (1+\gamma_u)\Pr\left(V^\sigma(u,\boldsymbol{r}_u)=0 \;\middle|\; \begin{array}{c}\sigma \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n \\ \mathrm{supp}(\sigma_u^{-1}(\boldsymbol{r}_u)) = Z_u\end{array}\right) \\
&= (1+\gamma_u)\Pr\left(\sigma_u^{-1}(\boldsymbol{r}_u) \in K_u \;\middle|\; \begin{array}{c}\sigma \leftarrow_{\mathrm{R}} \mathfrak{S}(\mathbb{F}_q)^n \\ \mathrm{supp}(\sigma_u^{-1}(\boldsymbol{r}_u)) = Z_u\end{array}\right) \\
&= (1+\gamma_u)\Pr\left(\boldsymbol{x} \in K_u \;\middle|\; \begin{array}{c}\boldsymbol{x} \leftarrow_{\mathrm{R}} \mathbb{F}_q^\ell \\ \mathrm{supp}(\boldsymbol{x}) = Z_u\end{array}\right) \\
&= \frac{(1+\gamma_u)A_u}{(q-1)^{|Z_u|}}.
\end{aligned}$$

□

Lemma 5.20. *Let S_u be the $\mathbb{F}_q$-vector space $\mathrm{Span}_{\mathbb{F}_q}\{\boldsymbol{x} \in K_u, \mathrm{supp}(\boldsymbol{x}) = Z_u\}$ and assume that $S_u \neq \{\mathbf{0}\}$. We then have:*

$$A_u \leq q^{|Z_u| - \mathrm{d_{min}}(S_u)+1}.$$

Proof. Let us prove that, if $A_u > q^e$ for some integer $e \geq 0$, then $\mathrm{d_{min}}(S_u) \leq |Z_u| - e$, which induces the result. If $A_u > q^e$, then $\dim(S_u) > e$ since $|S_u| \geq A_u$. The Singleton bound then provides:

$$\mathrm{d_{min}}(S_u) \leq |Z_u| - \dim(S_u) + 1 \leq |Z_u| - e. \qquad \square$$

Finally, we get the following upper bound on α.

Proposition 5.21. *Denote by $\Delta = \min\{\mathrm{d_{min}}(K_u), u \in \mathcal{Q}\}$. If every $\Phi_{\boldsymbol{w}}$ is sufficiently uniform (for $\boldsymbol{w} \in \mathbb{F}_q^n$), then*

$$\alpha \leq (1+\gamma)(1+\tfrac{1}{q-1})^\ell q^{-\Delta+1},$$

where $\gamma = o(1)$ when $n\log(q) \to \infty$.

Proof. The vector space $S_u := \mathrm{Span}_{\mathbb{F}_q}\{\boldsymbol{x} \in K_u, \mathrm{supp}(\boldsymbol{x}) = Z_u\}$, defined in the previous lemma, is a subcode of $\mathrm{Short}(K_u, u \setminus Z_u)$. Hence $\mathrm{d_{min}}(K_u) \leq \mathrm{d_{min}}(S_u)$. Due to our assumption on $\Phi_{\boldsymbol{w}}$, we can apply Lemmata 5.19 and 5.20 to obtain the desired bound:

$$\alpha \leq \max_{u,\boldsymbol{r}}\left\{(1+\gamma_u)\left(\frac{q}{q-1}\right)^{|Z_u|} q^{-\mathrm{d_{min}}(K_u)+1}\right\} \leq (1+\gamma)(1+\tfrac{1}{q-1})^\ell q^{-\Delta+1}$$

where $\gamma = \max_u \gamma_u = o(1)$ when $n\log(q) \to \infty$. □

5.3.5 Pairwise uncorrelation of variables $\{X_u\}_{u\in\mathcal{D}}$

This section is devoted to proving that variables $\{X_u\}_{u\in\mathcal{D}(\boldsymbol{r},\boldsymbol{w})}$ are pairwise uncorrelated if the supports of challenges $u \in \mathcal{D}(\boldsymbol{r},\boldsymbol{w})$ have small pairwise intersection. Let us recall that for fixed $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$, $\boldsymbol{w}$ and $u \in \mathcal{D}(\boldsymbol{r},\boldsymbol{w})$, the random variable X_u represents $\mathbb{1}_{V^\sigma(u,\boldsymbol{r}_u)=0}$, when σ is uniformly picked in $\Phi_{\boldsymbol{w}}$.

Proposition 5.22. *If* $\max\{|u\cap v|, u\neq v\in\mathcal{Q}\} < \min\{\mathrm{d}^\perp(\mathcal{C}_{|u}), u\in\mathcal{Q}\}$*, then the random variables* $\{X_u\}_{u\in\mathcal{Q}}$ *are pairwise uncorrelated.*

Proof. Recall that we denote by $K_u := \ker V(u,\cdot)$, and that by definition of a verification structure, we have $\mathcal{C}_{|u}\subseteq K_u$. For $u\neq v\in\mathcal{Q}$, let us prove that $\mathbb{E}(X_uX_v)=\mathbb{E}(X_u)\mathbb{E}(X_v)$. First,

$$\begin{aligned}\mathbb{E}(X_uX_v) &= \Pr\left(V^\sigma(u,\boldsymbol{r}_u)=0 \text{ and } V^\sigma(v,\boldsymbol{r}_v)=0\right)\\ &= \Pr\left(\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \text{ and } \sigma^{-1}(\boldsymbol{r}_v)_{|v}\in K_v\right).\end{aligned}$$

Denote by $t=|u\cap v|$ and let $(\boldsymbol{a},\boldsymbol{b})\in(\mathbb{F}_q^t)^2$. We denote by $Y(\sigma,\boldsymbol{a},\boldsymbol{b})$ the event

$$\sigma^{-1}(\boldsymbol{r}_u)_{|u\cap v}=\boldsymbol{a} \text{ and } \sigma^{-1}(\boldsymbol{r}_v)_{|u\cap v}=\boldsymbol{b}.$$

We first notice that $\{\sigma^{-1}_{|u\cap v}, \sigma\in\Phi_{\boldsymbol{w}}\}=\mathfrak{S}(\mathbb{F}_q)^t$. We can here use an argument similar to the proof of Lemma 4.21: the constraint $\sigma^{-1}(\boldsymbol{w})\in\mathcal{C}$ is ineffective on $\sigma^{-1}_{|u\cap v}$, since $|u\cap v|\leq t < \mathrm{d}^\perp(\mathcal{C}_{|z})$ for every $z\in\mathcal{Q}$. Therefore, for every $(\boldsymbol{a},\boldsymbol{b})\in(\mathbb{F}_q^t)^2$, we have

$$\Pr(Y(\sigma,\boldsymbol{a},\boldsymbol{b}))=q^{-2t},$$

and it follows that:

$$\mathbb{E}(X_uX_v)=\frac{1}{q^{2t}}\sum_{\boldsymbol{a},\boldsymbol{b}\in(\mathbb{F}_q^t)^2}\Pr\left(\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \text{ and } \sigma^{-1}(\boldsymbol{r}_v)_{|v}\in K_v \mid Y(\sigma,\boldsymbol{a},\boldsymbol{b})\right).$$

Recall that $t<\min\{\mathrm{d}^\perp(\mathcal{C}_{|u}), u\in\mathcal{Q}\}\leq\min\{\mathrm{d}^\perp(K_u), u\in\mathcal{Q}\}$. Hence, for fixed $\boldsymbol{a}$ and $\boldsymbol{b}$, the variables $\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \mid Y(\sigma,\boldsymbol{a},\boldsymbol{b})$ and $\sigma^{-1}(\boldsymbol{r}_v)_{|v}\in K_v \mid Y(\sigma,\boldsymbol{a},\boldsymbol{b})$ are independent (once again we can adapt Lemma 4.21). Therefore:

$$\begin{aligned}\mathbb{E}(X_uX_v)=\frac{1}{q^{2t}}\sum_{\boldsymbol{a},\boldsymbol{b}\in(\mathbb{F}_q^t)^2}&\Pr\left(\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \mid Y(\sigma,\boldsymbol{a},\boldsymbol{b})\right)\\ &\times\Pr\left(\sigma^{-1}(\boldsymbol{r}_v)_{|v}\in K_v \mid Y(\sigma,\boldsymbol{a},\boldsymbol{b})\right).\end{aligned}$$

Then,

$$\begin{aligned}\mathbb{E}(X_uX_v)=\frac{1}{q^{2t}}\sum_{\boldsymbol{a},\boldsymbol{b}\in(\mathbb{F}_q^t)^2}&\Pr(\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \mid \sigma^{-1}(\boldsymbol{r}_u)_{|u\cap v}=\boldsymbol{a})\\ &\times\Pr(\sigma^{-1}(\boldsymbol{r}_v)_{|v}\in K_v \mid \sigma^{-1}(\boldsymbol{r}_v)_{|u\cap v}=\boldsymbol{b}).\end{aligned}$$

and we conclude since $\mathbb{E}(X_u)=q^{-t}\sum_{\boldsymbol{a}\in\mathbb{F}_q^t}\Pr(\sigma^{-1}(\boldsymbol{r}_u)_{|u}\in K_u \mid \sigma^{-1}(\boldsymbol{r}_u)_{|u\cap v}=\boldsymbol{a})$. □

5.4 Performance

5.4.1 Efficient scrambling of the encoded file

In the PoR scheme we propose, the storage cost of an n-tuple of permutations in $\mathfrak{S}(\mathbb{F}_q)^n$ is excessive, since it is superlinear in the original file size. In this subsection, we propose a storage-efficient way to scramble the codeword $\boldsymbol{c} \in \mathcal{C}$ produced by the Verifier.

Precisely, given a codeword $\boldsymbol{c} \in \mathcal{C} \subseteq \mathbb{F}_q^n$, we want to define a map $\sigma : \boldsymbol{c} \mapsto \boldsymbol{w} \in \mathbb{F}_q^n$ with the following requirements:

- the map σ is efficiently computable and requires a low storage,
- if $\boldsymbol{w} = \sigma(\boldsymbol{c})$, $\boldsymbol{c} \in \mathcal{C}$, then for every $i \in [1, n]$ the local inverse map $w_i \mapsto c_i$ is efficiently computable,
- given the knowledge of $\boldsymbol{w} = \sigma(\boldsymbol{c})$ and $\mathcal{C}$, it is hard to produce a response word $\boldsymbol{r} \in \mathcal{R}^{\mathcal{Q}}$ such that $V^{\sigma}(u, \boldsymbol{r}_u) = 0$ and $\boldsymbol{r}_u \neq \boldsymbol{w}_{|u}$ for many $u \in \mathcal{Q}$.

We here propose to use an n-tuple of pseudo-random permutations $(\sigma_1, \ldots, \sigma_n) = \sigma$ derived from a suitable block cipher. Let us first give an explicit construction, so that we can discuss its subtleties afterwards.

The construction. Let IV denote a random initialisation vector for a cipher in CTR mode (IV could be a nonce concatenated with a random value). For example, we choose the AES block cipher. Vector IV is kept secret by the Verifier, as well as a randomly chosen key κ for the cipher. Let also f be a permutation polynomial over $\mathbb{F}_q$ of degree $d > 1$. For instance one could choose $f(x) = x^d$ with $\gcd(d, q-1) = 1$. Notice that polynomial f can be published.

Let $s = \left\lfloor \frac{256}{\lceil \log_2 q \rceil} \right\rfloor$ be the number of $\mathbb{F}_q$-symbols one can store in a 256-bit word[3]. Up to appending a few random bits to $\boldsymbol{c}$, we assume that $s \mid n$, and we denote by $t = n/s$. Let us fix a partition of $[1, n]$ into s-tuples $\boldsymbol{i} = (i_1, \ldots, i_s)$; it can be for instance $(1, \ldots, s)$, $(s+1, \ldots, 2t), \ldots, ((t-1)s+1, \ldots, n)$. Notice that this partition does not need to be chosen at random. Given $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathcal{C}$ and $\boldsymbol{i}$ an element of the above partition, we now define

$$b_{\boldsymbol{i}} = \big(f(c_{i_1}) \mid \cdots \mid f(c_{i_s})\big) \oplus \mathrm{AES}_{\kappa}(IV \oplus \boldsymbol{i}) \ \in \{0,1\}^{256}.$$

If $\log_2 q \nmid 256$, trailing zeroes can be added to evaluations of f. Finally, the pseudo-random permutation σ is defined by:

$$\sigma(\boldsymbol{c}) \coloneqq (b_1, \ldots, b_t)\,.$$

Informal analysis. One could first question the necessity to use $\boldsymbol{i}$ as a part of the input of the AES cipher. Assume that we do not. Then, the local permutation σ_j, $1 \leq j \leq n$, would not depend on j. As a consequence, for certain class of codes the local verification map $\boldsymbol{r}_u \mapsto V^{\sigma}(u, \boldsymbol{r}_u)$ would not depend on u, and a malicious Prover would then be able to produce accepted answers while storing only a small piece of the file $\boldsymbol{w}$ (*e.g.* $\boldsymbol{w}_{|u}$ for only one $u \in \mathcal{Q}$).

Second, one could wonder why a permutation polynomial f of degree $d > 1$ is used. Assume for instance that $f = \mathrm{id}$. Then, given the knowledge of $\boldsymbol{w} = \sigma(\boldsymbol{c})$, it would be very easy for a malicious Prover to produce a word $\boldsymbol{w}' \neq \boldsymbol{w}$, such that $\boldsymbol{r}' = R(\boldsymbol{w}')$ is always accepted by the Verifier. Simply, the Prover defines $\boldsymbol{w}' = \boldsymbol{w} + \boldsymbol{c}'$, where $\boldsymbol{c}'$ is any non-zero codeword of

[3] in the scheme we propose, we will always have $\log(q) < 256$

$\mathcal{C}$. Hence, one sees that the polynomial f must be nonlinear in order to prevent such kind of attacks.

5.4.2 Parameters

Let us consider a PoR built upon a code $\mathcal{C} \subseteq \mathbb{F}_q^n$, with verification structure $(\mathcal{Q}, V)$ satisfying $\mathcal{R} \simeq \mathbb{F}_q^\ell$ and $V(\mathcal{R}) = \mathbb{F}_q^s$. We also assume we use an n-tuple of pseudo-random permutations as described in the previous subsection.

Communication complexity. At each verification step, the client sends an ℓ-subset of coordinates $u \subset [1, n]$. The server then answers with the corresponding symbols $w_i \in \mathbb{F}_q$, for $i \in u$. Therefore the upload communication cost is $\ell \log_2 n$ bits while the download communication cost is $\ell \log_2 q$, thus a total of $\ell(\log_2 n + \log_2 q)$ bits.

Computation complexity. During the initialisation phase, following the encryption process described above, the client essentially has:

- to compute the codeword $\boldsymbol{c} \in \mathcal{C}$ associated to its message,
- to make n evaluations of the permutation polynomial f over $\mathbb{F}_q$, and
- to compute $t = \frac{n \log_2 q}{256}$ AES ciphertexts to produce the word $\boldsymbol{w}$ to be sent to the server.

Given a generator matrix of $\mathcal{C}$, the codeword $\boldsymbol{c} \in \mathcal{C}$ can be computed in $\mathcal{O}(kn)$ operations over $\mathbb{F}_q$ with a matrix-vector product. Notice that quasi-linear-time encoding algorithms exist for some classes of codes. Moreover, if a monomial or a sparse permutation polynomial f is used, then the cost of each evaluation of f is $\mathcal{O}((\log_2 q)^3)$. If we denote by c the bitcost of an AES encryption, we get a total bitcost of $\mathcal{O}(n^2(\log_2 q)^2 + n(\log_2 q)^3 + cn \log_2 q)$ for the initialisation phase. Recall this is a worst-case scenario in which the encoding process is inefficient.

At each verification step, an honest server only needs to read ℓ symbols from the file it stores. Hence it has no computation complexity. The client has to compute a matrix-vector product over $\mathbb{F}_q$, where the matrix has size $s \times \ell$ and the vector has size ℓ, thus a computation cost of $\mathcal{O}(\ell s)$ operations over $\mathbb{F}_q$.

Storage needs. The client only stores 2×256 bits for the secret material: the key κ and the initialisation vector IV used in the AES cipher. The server storage overhead exactly corresponds to the redundancy of the linear code $\mathcal{C}$, that is $(n - \dim(\mathcal{C})) \log_2 q$ bits.

Other features. The PoR scheme we propose in Section 5.2 is *unbounded-use*, since each challenge reveals nothing about the secret data held by the client. Unfortunately it does not feature dynamic updates of files. However, we must emphasize that the file $\boldsymbol{w}$ the client produces can be split among several servers, and the verification step remains possible even if the servers do not communicate with each other. Indeed, computing a response to a challenge does not require to mix distinct symbols w_i of the file. Therefore, the scheme we propose is well-suited for the storage of *large static distributed* databases.

Client storage:	512 bits
Server total storage:	$n \log_2 q$ bits
Communication complexity (verif.):	$\ell \log_2(nq)$ bits
Client computational complexity (verif.):	ℓ decryptions, ℓs operations over $\mathbb{F}_q$
Server computational complexity (verif.):	ℓ reads, no computation

Table 5.1 – Summary of parameters of the proposed PoR construction. We assume the file has size $k \log(q)$ bits, and the underlying code $\mathcal{C} \subseteq \mathbb{F}_q^n$ has dimension k and is equipped with a verification structure $(\mathcal{Q}, V)$ such that $|u| = \ell$, and $\operatorname{rank} V(u, \cdot) \leq s$ for all $u \in \mathcal{Q}$.

5.5 Instantiations

In this section we present several instantiations for the PoR protocol presented in Section 5.2. According to Theorem 5.18 and the study of parameters in the last section, we especially look for codes $\mathcal{C}$ with positive rate (when its length n tends to infinity), such that the response code $R(\mathcal{C})$ has positive relative distance (when its length N tends to infinity).

5.5.1 Tensor-product codes

Let us first recall a specific construction of codes called the *tensor product*. For $\boldsymbol{x} \in \mathbb{F}_q^A$ and $\boldsymbol{y} \in \mathbb{F}_q^B$, we denote by $\boldsymbol{x} \otimes \boldsymbol{y}$ the tuple $\boldsymbol{c} \in \mathbb{F}_q^{A \times B}$ and given by $c_{a,b} = x_a y_b$. Then, if $\mathcal{C} \subseteq \mathbb{F}_q^A$ and $\mathcal{C}' \subseteq \mathbb{F}_q^B$ are two linear codes, their tensor product $\mathcal{C} \otimes \mathcal{C}' \subseteq \mathbb{F}_q^{A \times B}$ is the $\mathbb{F}_q$-linear code generated by words $\boldsymbol{c} \otimes \boldsymbol{c}'$, for $\boldsymbol{c} \in \mathcal{C}$ and $\boldsymbol{c}' \in \mathcal{C}'$. If $\mathcal{C}$, $\mathcal{C}'$ have respective parameters $[n, k, d]$ and $[n', k', d']$, then $\mathcal{C} \otimes \mathcal{C}'$ is $[nn', kk', dd']$. We also denote by

$$\mathcal{C}^{\otimes s} := \underbrace{\mathcal{C} \otimes \cdots \otimes \mathcal{C}}_{s \text{ times}} \subseteq \mathbb{F}_q^{A^s}$$

the s-fold tensor product of $\mathcal{C}$ with itself.

Next paragraph illustrates the PoR construction we propose with a simple code, though it cannot be used in practice. More practical PoR instances appear in the following paragraphs.

A simple but non-practical instance. Let $n = N\ell$, and for $0 \leq i \leq N-1$, denote by $u_i = \{i\ell + 1, i\ell + 2, \ldots, (i+1)\ell\}\}$. We define $\mathcal{Q} = \{u_i, i \in [0, N-1]\}$. The set $\mathcal{Q}$ defines a partition of $[1, n]$. We define the code

$$\mathcal{C} = \{c \in \mathbb{F}_q^n \mid \sum_{j \in u} c_j = 0, \, \forall u \in \mathcal{Q}\} \subseteq \mathbb{F}_q^n .$$

In other words, up to re-indexing the coordinates we have $\mathcal{C} = \mathrm{Par}(\ell) \otimes \mathbb{F}_q^N$, and a parity-check matrix $\boldsymbol{H}$ for $\mathcal{C}$ is given by:

$$\boldsymbol{H} = \begin{pmatrix} 1 & \cdots & 1 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 1 & \ddots & \vdots & \vdots & \vdots \\ \vdots & & & \ddots & & & \ddots & & & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 & 1 & \cdots & 1 \end{pmatrix} .$$

The verification map $V : \mathcal{Q} \times \mathbb{F}_q^\ell \to \mathbb{F}_q$ is defined by $V(u, \boldsymbol{b}) := \sum_{j=1}^{\ell} b_{u_j}$, for all $(u, \boldsymbol{b}) \in \mathcal{Q} \times \mathbb{F}_q^\ell$. By construction (see Construction 5.6), the pair $(\mathcal{Q}, V)$ defines a verification structure for $\mathcal{C}$.

Lemma 5.23. *Let $\mathcal{C} = \mathrm{Par}(\ell) \otimes \mathbb{F}_q^N$ as above. Then the response code $R(\mathcal{C})$ has minimum distance* 1.

Proof. We see that the restriction map R sends the codeword $(1, -1, 0, 0, \ldots, 0) \in \mathcal{C}$ to a word of weight 1. Moreover, the map R is injective so $\mathrm{d}_{\min}(R(\mathcal{C})) > 0$. □

Since $\delta = \mathrm{d}_{\min}(R(\mathcal{C}))/N = 1/N \to 0$ when $N \to \infty$, according to Theorem 5.18 a tentative PoR scheme built upon $\mathcal{C}$ cannot be sound.

Remark 5.24. One can also see $\mathcal{C}$ as a design-based code. The underlying design is a 1-$(n, \ell, 1)$ design $(X, \mathcal{B})$, also simply known as a partition. Indeed, the point set X is $[1, n]$, and each point $x_{i,j} = i\ell + j \in X$, for $0 \leq i \leq n/\ell - 1$ and $1 \leq j \leq \ell$, belongs to exactly one block $u_i \in \mathcal{B}$ of size ℓ.

Higher order tensor-product codes. We first state a link between tensor-product codes and generalised design-based codes. For $1 \leq i \leq s$ and $\boldsymbol{x} \in X := \prod_{j=1}^{s} X_j$, we need to define $L_{i,\boldsymbol{x}} \subseteq X$, the '$i$-th axis-parallel line with basis $\boldsymbol{x}$', as

$$L_{i,\boldsymbol{x}} := \{\boldsymbol{y} \in X \mid y_j = x_j, \forall j \neq i\}\,.$$

Notice that the line $L_{i,\boldsymbol{x}}$ can be seen as a *translation* of the axis X_i, whence its designation. One should also pay attention to the fact that two distinct tuples $\boldsymbol{x} \neq \boldsymbol{x}'$ can define the same line $L_{i,\boldsymbol{x}} = L_{i,\boldsymbol{x}}$.

Proposition 5.25. *Let $\mathcal{A}_i \subseteq \mathbb{F}_q^{X_i}, 1 \leq i \leq s$, be s linear codes of length ℓ. Then, the tensor-product code $\mathcal{C} = \mathcal{A}_1 \otimes \cdots \otimes \mathcal{A}_s$ is a generalised design-based code* $\mathrm{Code}(\mathcal{D}, \mathcal{L})$*, with the following parameters.*

- *The underlying design $\mathcal{D} = (X, \mathcal{B})$ is a 1-(ℓ^s, ℓ, s) design. Its point set is $X = \prod_i X_i$, and its block set is $\mathcal{B} = \{L_{i,\boldsymbol{x}}, 1 \leq i \leq s, \boldsymbol{x} \in X\}$.*
- *The set of local codes $\mathcal{L} = (\mathcal{L}_B, B \in \mathcal{B})$ is defined by $\mathcal{L}_B = \mathcal{A}_i$ if $B = L_{i,\boldsymbol{x}}$ for some $\boldsymbol{x} \in X$ (using a one-to-one map $X_i \to L_{i,\boldsymbol{x}}$).*

Proof. Let us prove that $\mathcal{C} \subseteq \mathrm{Code}(\mathcal{D}, \mathcal{L})$. The code $\mathcal{C}$ is generated by elementary tensors of the form

$$\boldsymbol{a}^{(1,j_1)} \otimes \cdots \otimes \boldsymbol{a}^{(s,j_s)}, \quad 1 \leq j_i \leq \dim \mathcal{A}_i\,,$$

where for every $1 \leq i \leq s$, the set $\{\boldsymbol{a}^{(i,j)}, 1 \leq j \leq \dim(\mathcal{A}_i)\}$ defines a basis of $\mathcal{A}_i$. So, by linearity, let us prove the result for an elementary tensor $\boldsymbol{e}^{(\boldsymbol{j})} := \boldsymbol{a}^{(1,j_1)} \otimes \cdots \otimes \boldsymbol{a}^{(s,j_s)}$, where $\boldsymbol{j} = (j_1, \ldots, j_s)$.

We denote $a^{(i,j_i)}[x_i]$ the symbol of $\boldsymbol{a}^{(i,j_i)}$ supported by $x_i \in X_i$. We can see that

$$\boldsymbol{e}^{(\boldsymbol{j})}_{|L_{i,\boldsymbol{x}}} = (\boldsymbol{a}^{(1,j_1)} \otimes \cdots \otimes \boldsymbol{a}^{(s,j_s)})_{|L_{i,\boldsymbol{x}}} = \left(\prod_{t \neq i} a^{(t,j_t)}[x_t]\right) \boldsymbol{a}^{(i,j_i)} \in \mathcal{A}_i\,, \tag{5.3}$$

since $\boldsymbol{a}^{(i,j_i)}$ is an element of the basis we have given for $\mathcal{A}_i$. Notice that we made here the identification between X_i and $L_{i,\boldsymbol{x}}$. Finally, we get $\boldsymbol{e}^{(\boldsymbol{j})}_{|L_{i,\boldsymbol{x}}} \in \mathcal{A}_i$ for every $L_{i,\boldsymbol{x}} \in \mathcal{B}$, which proves by linearity that:

$$\mathcal{C} \subseteq \{\boldsymbol{c} \in \mathbb{F}_q^X \mid \forall L_{i,\boldsymbol{x}} \in \mathcal{B}, \boldsymbol{c}_{|L_{i,\boldsymbol{x}}} \in \mathcal{A}_i\} = \mathrm{Code}(\mathcal{D}, \mathcal{L})\,.$$

The reverse inclusion can be proved very similarly. Finally, it is easy to check that $\mathcal{D}$ is a 1-(ℓ^s, ℓ, s) design: every point $\boldsymbol{x} \in X$ belongs to exactly s distinct lines $L_{i,\boldsymbol{x}}$ of cardinality ℓ. □

For convenience we now consider the case where all $X_i = X_1$ and $\mathcal{A}_i = \mathcal{A}_1$, for every $1 \leq i \leq s$. So let $\mathcal{A} \subseteq \mathbb{F}_q^A$ be a non-degenerate $[\ell, k_{\mathcal{A}}, d_{\mathcal{A}}]_q$-linear code, and define $\mathcal{C} = \mathcal{A}^{\otimes s} \subseteq \mathbb{F}_q^X$ where $X = A^s$, and $n = |X| = \ell^s$. Thanks to Proposition 5.25, it is easy to define a verification structure $(\mathcal{Q}, V)$ for $\mathcal{C}$, see Construction 5.7. So let us analyse its response code.

Lemma 5.26. *Let $\mathcal{C} = \mathcal{A}^{\otimes s}$ as above. Then, $R(\mathcal{C})$ has minimum distance $s \cdot \mathrm{d}_{\min}(\mathcal{A})^{s-1}$.*

Proof. Let us first prove that the minimum distance of the response code is greater than $s \cdot \mathrm{d}_{\min}(\mathcal{A})^{s-1}$. Let $\boldsymbol{r} = R(\boldsymbol{c}) \in R(\mathcal{C})$, and assume $\boldsymbol{r} \neq \boldsymbol{0}$. Then, there exists $L \in \mathcal{Q}$ such that $\boldsymbol{0} \neq \boldsymbol{r}_L = \boldsymbol{c}_{|L} \in \mathcal{A}$. Therefore $\boldsymbol{c}_{\boldsymbol{x}} \neq \boldsymbol{0}$ for some $\boldsymbol{x} \in L \subset X$. Now, for some $1 \leq i \leq s$, consider the set

$$S_{i,\boldsymbol{x}} = \{\boldsymbol{y} \in X, y_i = x_i\} \subseteq X\,.$$

Very informally, the set $S_{i,\boldsymbol{x}}$ corresponds to the hyperplane passing through $\boldsymbol{x}$ and 'perpendicular' to the i-th axis. By definition of $\mathcal{C} = \mathcal{A}^{\otimes s}$, we know that $\boldsymbol{c}_{|S_{i,\boldsymbol{x}}} \in \mathcal{A}^{\otimes(s-1)}$ for every $1 \leq i \leq s$. Denote by $U_i = \mathrm{supp}(\boldsymbol{c}_{|S_{i,\boldsymbol{x}}})$, and $t_i := |U_i| \geq \mathrm{d}_{\min}(\mathcal{A}^{\otimes(s-1)}) = \mathrm{d}_{\min}(\mathcal{A})^{s-1}$. One can write $U_i = \{\boldsymbol{u}^{(i,1)}, \ldots, \boldsymbol{u}^{(i,t_i)}\}$. For every $\boldsymbol{u}^{(i,j)} \in U_i$, the line $L_{i,\boldsymbol{u}^{(i,j)}}$ is such that $\boldsymbol{c}_{|L_{i,\boldsymbol{u}^{(i,j)}}}$ is a non-zero codeword of $\mathcal{A}$. Therefore,

$$\mathrm{wt}(\boldsymbol{r}) = |\{L \in \mathcal{Q}, \boldsymbol{r}_L \neq \boldsymbol{0}\}| \geq \left|\bigcup_{i=1}^{s} \left\{L_{i,\boldsymbol{u}^{(i,j)}}, 1 \leq j \leq t_i\right\}\right| \geq \sum_{i=1}^{s} t_i \geq s \cdot \mathrm{d}_{\min}(\mathcal{A})^{s-1}\,.$$

Let us now build a word $\boldsymbol{r} \in R(\mathcal{C})$ of weight $s \cdot \mathrm{d}_{\min}(\mathcal{A})^{s-1}$. Let $\boldsymbol{w} \in \mathcal{A} \setminus \{\boldsymbol{0}\}$ be a minimum-weight codeword of $\mathcal{A}$, and denote by $W := \mathrm{supp}(\boldsymbol{w}) \subseteq A$. Define $\boldsymbol{c} = \boldsymbol{w}^{\otimes s} \in \mathcal{C}$; then $\mathrm{supp}(\boldsymbol{c}) = W^s$. Let finally $\boldsymbol{r} = R(\boldsymbol{c})$. We see that $\boldsymbol{r}_{L_{i,\boldsymbol{x}}} \neq \boldsymbol{0}$ if and only if $\boldsymbol{x} \in W^s$. Hence we get

$$\mathrm{wt}(\boldsymbol{r}) = |\{L \in \mathcal{Q}, \boldsymbol{r}_L \neq \boldsymbol{0}\}| = \left|\bigcup_{i=1}^{s} \{L_{i,\boldsymbol{x}}, \boldsymbol{x} \in W^s\}\right| = s \cdot \mathrm{d}_{\min}(\mathcal{A})^{s-1}\,.$$

since each line $L_{i,\boldsymbol{x}}$ is counted $\mathrm{d}_{\min}(\mathcal{A})$ times when $\boldsymbol{x}$ runs over W^s. □

Now denote by $N = |\mathcal{Q}| = s\ell^{s-1}$ the length of $R(\mathcal{C})$. If $\mathcal{A}$ is an MDS code, we summarise the parameters of a PoR scheme based on $\mathcal{A}^s$ in the next proposition.

Proposition 5.27. *Let $s \geq 0$, $0 < \delta < 1$ and $\mathcal{A}$ be an $[\ell, \ell(1-\delta)+1, \ell\delta]_q$ MDS code. Define $\mathcal{C} = \mathcal{A}^{\otimes s}$ and $(\mathcal{Q}, V)$ as above. If every Φ_w is sufficiently uniform, then the PoR scheme associated to $\mathcal{C}$ and $(\mathcal{Q}, V)$ is (ε, τ)-sound for $\tau = \mathcal{O}\left(\frac{1}{(\delta\ell)^s s}\right)$ and every $\varepsilon < \varepsilon_0$, where $\varepsilon_0 = (1 + \mathcal{O}(q^{-\delta\ell+1}))\delta^s$. Asymptotics are given when $n = \ell^s \to \infty$.*

Proof. First, the relative distance of $R(\mathcal{C})$ is δ^s according to Lemma 5.26. Then, the random variables $\{X_u\}_{u \in \mathcal{D}}$ are pairwise uncorrelated because the inequality

$$\max_{u \neq v \in \mathcal{Q}^2} |u \cap v| = 1 < \ell(1-\delta) + 2 = \min_{u \in \mathcal{Q}} \mathrm{d}^{\perp}(\mathcal{C}_{|u})$$

allows us to apply Proposition 5.22. Furthermore, if every Φ_w is sufficiently uniform, the bias α satisfies $\alpha \in \mathcal{O}(q^{-\delta\ell+1})$. Hence $\frac{1-\alpha}{1+\alpha} = 1 + \mathcal{O}(q^{-\delta\ell+1})$. Therefore we can use Theorem 5.18 and we get the desired result. □

Parameters. We mainly focus on the download communication complexity in the verification step and on the server storage overhead, since they are the most crucial parameters which depend on the family of codes $\mathcal{C}$ we use. Moreover, we consider as more relevant to analyse the ratio of these quantities to the file size, rather than their absolute values.

For the PoR scheme based on tensor product of MDS codes analysed in Proposition 5.27, and an initial file $\boldsymbol{m}$ of size $|\boldsymbol{m}| = ((1-\delta)q+1)^s \log_2 q$ bits, we get

- a relative redundancy $\frac{n \log_2 q}{|\boldsymbol{m}|} = \left(\frac{q}{(1-\delta)q+1}\right)^s \leq \frac{1}{(1-\delta)^s}$;
- a relative communication complexity $\frac{\ell \log_2 q}{|\boldsymbol{m}|} = \frac{q}{((1-\delta)q+1)^s} \leq \frac{1}{(1-\delta)^s} q^{1-s}$.

Example 5.28. We present various parameters of PoR instances based on tensor-product codes, for files of size approaching 10^4, 10^6 and 10^9 bits. We only report instances with comparable success bound ε_0, here $0.10 \leq \varepsilon_0 \leq 0.16$. Formally, recall that $\mathcal{A}$ is a $[q, (1-\delta)q+1, \delta q]_q$ MDS code (*e.g.* a Reed-Solomon code), and $\mathcal{C} = \mathcal{A}^{\otimes s}$.

q	δq	s	file size (bits)	comm. rate	redundancy rate	ε_0
16	10	4	9604	$6.664 \cdot 10^{-3}$	27.3	0.153
25	13	3	10 985	$1.138 \cdot 10^{-2}$	7.112	0.141
64	24	2	10 086	$3.807 \cdot 10^{-2}$	2.437	0.141
32	21	5	1 244 160	$1.286 \cdot 10^{-4}$	134.8	0.122
47	28	4	960 000	$2.938 \cdot 10^{-4}$	30.5	0.126
101	47	3	1 164 625	$6.071 \cdot 10^{-4}$	6.193	0.101
512	180	2	998 001	$4.617 \cdot 10^{-3}$	2.364	0.124
128	85	5	1 154 413 568	$7.762 \cdot 10^{-7}$	208.3	0.129
256	150	4	1 048 636 808	$1.953 \cdot 10^{-6}$	32.77	0.118
1024	550	3	1 071 718 750	$9.555 \cdot 10^{-6}$	10.02	0.155
12167	3900	2	957 037 536	$1.78 \cdot 10^{-4}$	2.166	0.103
16384	5500	2	1 658 765 150	$1.383 \cdot 10^{-4}$	2.266	0.113

Example 5.28 shows that, if the communication rate is reasonable for the PoR instances based on tensor-product codes, their storage overhead is large. In fact, the problem lies in the fact that δq must be large in order to reach non-vanishing values of ε. In the next subsection, we will see that Reed-Muller codes, and more generally codes equipped admitting a verification structure with a richer query set, allow us to reach better PoR parameters.

5.5.2 Reed-Muller and related codes

Low-degree Reed-Muller codes are known to admit many distinct low-weight parity-check equations, whose supports corresponds to affine subspaces of the ambient space. Therefore they seem naturally adapted to our construction. Let us first consider the plane (or bivariate) Reed-Muller code case.

The plane Reed-Muller code $\mathrm{RM}_q(2, q-2)$**.** Let $\mathcal{C}$ be the Reed-Muller code $\mathrm{RM}_q(2, q-2)$. Recall that $\mathcal{C}$ has length q^2 and dimension $(q-1)(q-2)/2$. Furthermore, we have seen in Chapter 2 that for every affine line $L = \{\boldsymbol{x} = (at+b, ct+d), t \in \mathbb{F}_q\} \subset \mathbb{F}_q^2$ and every $\boldsymbol{c} \in \mathcal{C}$, it holds that $\sum_{\boldsymbol{x} \in L} c_{\boldsymbol{x}} = 0$.

Therefore, we can define $\mathcal{Q}$ as the set of affine lines L of $\mathbb{F}_q^2$, and the verification map V by $V(L, \boldsymbol{b}) = \sum_{j=1}^{\ell} b_j \in \mathbb{F}_q$ for every $\boldsymbol{b} \in \mathcal{R}$. From the previous discussion we know that $(\mathcal{Q}, V)$

is a verification structure for $\mathcal{C}$. Also, there are $q(q+1)$ distinct affine lines in $\mathbb{F}_q^2$, hence $N = q(q+1)$.

Lemma 5.29. *Let $\mathcal{C} = \mathrm{RM}_q(2, q-2)$ equipped with its verification structure defined as above. Then, its response code $R(\mathcal{C})$ has minimum distance q^2+2.*

Proof. Any non-zero codeword $\boldsymbol{c} \in \mathcal{C}$ consists in the evaluation of a non-zero polynomial $f(X,Y) \in \mathbb{F}_q[X,Y]$ of degree at most $q-2$. Denote by $L_1, \dots, L_a \subset \mathbb{F}_q^2$, the affine lines on which f vanishes; *i.e.* $f(\boldsymbol{x}) = 0$ for every $\boldsymbol{x} \in L_i$, $1 \leq i \leq a$. We claim that $a \leq q-2$. Indeed, since f has total degree $< q-1$, f also vanishes on *closed* lines $\overline{L_1}, \dots, \overline{L_a}$, *i.e.* affine lines seen in the closure $\overline{\mathbb{F}_q}^2$, where $\overline{\mathbb{F}_q}$ denotes the algebraic closure of $\mathbb{F}_q$. Denote by $g_i \in \mathbb{F}_q[X,Y]$ the monic polynomial of degree 1 which defines $\overline{L_i}$. From Hilbert's *Nullstellensatz*, there exists $r > 0$ such that $(\prod_{i=1}^{q} g_i) | f^r$. Since the g_i's have degree 1 and are distinct, we get $a \leq \deg f \leq q-2$. Hence, the affine lines different from $L_1, \dots, L_a$ correspond to non-zero coordinates of $R(\boldsymbol{c})$. There are $q(q+1) - a \geq q^2+2$ such lines so $\mathrm{d}_{\min}(R(\mathcal{C})) \geq q^2+2$.

Now we claim there exists a word $\boldsymbol{r} \in R(\mathcal{C})$ of weight $N - q + 2 = q^2+2$. Let $L^{(0)}$ and $L^{(1)}$ be two distinct parallel affine lines respectively defined by $X = 0$ and $X = 1$. Now consider the word $\boldsymbol{c} \in \mathbb{F}_q^{\mathbb{F}_q^2}$ such that $c_{\boldsymbol{x}} = -1$ for $\boldsymbol{x} \in L^{(0)}$, $c_{\boldsymbol{y}} = 1$ for $\boldsymbol{y} \in L^{(1)}$ and $c_{\boldsymbol{z}} = 0$ elsewhere. One can check that $\boldsymbol{c} \in \mathcal{C}$; indeed $\boldsymbol{c}$ is the evaluation vector of $\prod_{\alpha \in \mathbb{F}_q \setminus \{0,1\}} (\alpha - X)$, of degree less than $q-2$. Now, if we want to compute $\mathrm{wt}(R(\boldsymbol{c}))$, we only need to count the number of lines which do not intersect $L^{(0)}$ or $L^{(1)}$. There are only $q-2$ such lines, namely the lines of equation $X = \beta$, where $\beta \in \mathbb{F}_q \setminus \{0,1\}$. Hence $\mathrm{wt}(R(\boldsymbol{c})) = q(q+1) - (q-2)$ and this concludes the proof. □

Proposition 5.30. *Let $\mathcal{C} = \mathrm{RM}_q(2, q-2)$, and $(\mathcal{Q}, V)$ its associated verification structure. If every Φ_w is sufficiently uniform, then the PoR scheme associated to $\mathcal{C}$ and $(\mathcal{Q}, V)$ is (ε, τ)-sound for $\varepsilon = 1 - o(1)$ and $\tau = \mathcal{O}\left(\frac{1}{(1-\varepsilon)q^2}\right)$, when $n = q^2 \to \infty$.*

Proof. One can check that the random variables $\{X_u\}_{u \in \mathcal{D}}$ are pairwise uncorrelated since

$$\max_{u \neq v \in \mathcal{Q}^2} |u \cap v| = 1 < \ell(1-\delta) + 2 = \min_{u \in \mathcal{Q}} \mathrm{d}^{\perp}(\mathcal{C}_{|u}) \,.$$

Moreover, the relative distance of $R(\mathcal{C})$ is $\frac{q^2+2}{q(q+1)} \to 1$ according to Lemma 5.29. If every Φ_w is sufficiently uniform, the bias α satisfies $\alpha \in \mathcal{O}(1/q)$; hence $\frac{1-\alpha}{1+\alpha} = 1 + \mathcal{O}(1/q)$. Therefore we can use Theorem 5.18 and we get the desired result. □

Parameters. For the plane Reed-Muller code setting and an initial file of size $|\boldsymbol{m}| = \frac{1}{2}(q-1)(q-2)\log_2 q$ bits, we get

- a relative redundancy $\frac{q^2 \log_2 q}{|\boldsymbol{m}|} = \frac{2}{(1-1/q)(1-2/q)} \to 2$;
- a relative communication complexity $\frac{q \log_2 q}{|\boldsymbol{m}|} = \frac{2}{q} \frac{1}{(1-1/q)(1-2/q)} = \mathcal{O}(1/q)$.

Storage improvements *via* lifted codes. The redundancy rate of Reed-Muller codes presented above stays stuck above 2. Affine lifted codes, introduced by Guo, Kopparty and Sudan [GKS13], allow us to break this barrier while keeping the same verification structure. We refer to Chapters 2 and 3 for more details about these codes. Here we focus on $\mathrm{Lift}(\mathrm{RS}_q(q-2), 2)$, since it can be compared to $\mathrm{RM}_q(2, q-2)$. Furthermore, by definition of lifted codes,

$\mathrm{Lift}(\mathrm{RS}_q(q-2),2)$ admits the same verification structure as the one presented previously for $\mathrm{RM}_q(2,q-2)$.

Lemma 5.31. *The response code of* $\mathrm{Lift}(\mathrm{RS}_q(q-2),2)$ *has minimum distance at least* q^2-q+2.

Proof. The *rationale* is similar to the proof of Lemma 5.29. Let $\mathbf{0} \neq \boldsymbol{c} \in \mathcal{C}$, where $\boldsymbol{c} = (f(\boldsymbol{x}))_{\boldsymbol{x}\in\mathbb{F}_q^2}$, $f \in \mathbb{F}_q[X,Y]$. recall that we can assume $\deg(f) \leq 2q-2$. Denote by $L_1,\ldots,L_a \subset \mathbb{F}_q^2$ the lines on which f vanishes. The restriction of f along L_i can be interpolated as a univariate polynomial $f_{|L_i}(T)$ of degree at most $q-2$, since $(f(\boldsymbol{Q}))_{\boldsymbol{Q}\in L_i}$ lies in the Reed-Solomon code $\mathrm{RS}_q(q-2)$, by definition of lifted codes. Therefore $f_{|L_i}(T)=0$, and f vanishes on $\overline{L_i}$. Using arguments in the proof of Lemma 5.29, we get $a \leq \deg(f) \leq 2q-2$, and $\mathrm{d}_{\min}(\mathcal{R}) \geq q^2+q-2q+2 = q^2-q+2$, where $\mathcal{R}$ is the response code of $\mathrm{Lift}(\mathrm{RS}_q(q-2),2))$. □

We believe the bound given in Lemma 5.31 is not tight, but it is sufficient to have a response code $\mathcal{R}$ of relative minimum distance $\mathrm{d}_{\min}(\mathcal{R})/N \to 1$ when $N \to \infty$. Similarly to Proposition 5.32, we can then prove that practical PoRs can be constructed with the family of lifted codes $\mathrm{Lift}(\mathrm{RS}_q(q-2),2)$.

Proposition 5.32. *Let* $\mathcal{C} = \mathrm{Lift}(\mathrm{RS}_q(q-2),2)$, *and* $(\mathcal{Q},V)$ *its associated verification structure. If every* Φ_w *is sufficiently uniform, then the PoR scheme associated to* $\mathcal{C}$ *and* $(\mathcal{Q},V)$ *is* (ε,τ)*-sound for every* $\varepsilon<1$ *and* $\tau = \mathcal{O}\left(\frac{1}{(1-\varepsilon)q^2}\right)$.

The crucial improvement is that lifted codes potentially have much higher dimension than Reed-Muller codes. For instance, if $q=2^e$, we have seen in Chapter 3 that the dimension of $\mathrm{Lift}(\mathrm{RS}_q(q-2),2)$ is 4^e-3^e, which is much larger that $2^{e-1}(2^e-1)$.

Example 5.33. We present parameters of PoRs based on Reed-Muller codes and lifted codes, using files of size approaching 10^4, 10^6 and 10^9 bits.

code	q	file size	comm. rate	redundancy rate
Lift	32	3905	$4.10\cdot10^{-2}$	1.311
RM	64	11 718	$3.28\cdot10^{-2}$	2.097
Lift	64	20 202	$1.90\cdot10^{-2}$	1.217
Lift	256	471 800	$4,34\cdot10^{-3}$	1.111
RM	512	1 172 745	$3,93\cdot10^{-3}$	2.012
Lift	512	2 182 149	$2.11\cdot10^{-3}$	1.081
Lift	8192	851 689 033	$1.25\cdot10^{-4}$	1.024
RM	16384	1 878 704 142	$1.22\cdot10^{-4}$	2.000
Lift	16384	3 691 134 818	$6.21\cdot10^{-5}$	1.018

Note that the family of lifted codes has been used in the PoR proposal [LL16].

Remark 5.34. Once again, we emphasize that the lifted codes we consider are actually design-based codes. Here, the underlying design is the classical affine geometric design $\mathrm{AG}_1(2,q)$. It is a 2-$(q^2,q,1)$ design.

On more generic families of codes. We have presented several families of codes producing practical instances of PoR. We also revealed their link with (generalised) design-based codes, and noticed that the PoR instances based on 2-designs give better parameters than the one relying on 1-designs. Let us quickly mention other families of designs and codes that could be interesting to consider.

First, lifted codes over evaluation support $\mathbb{A}^m$ or $\mathbb{P}^m$, where $m > 2$, (or equivalently, codes based on $AG_1(m,q)$ or $\mathrm{PG}_1(m,q)$) would decrease the communication complexity of the PoR schemes we propose. Indeed, lines are smaller compared to the size of the ambient space. We must however pay attention to the storage rate which could vanish if m gets large. Indeed, we have seen previously that lifted codes and codes based on geometric designs of fixed field size q have vanishing rate when $m \to \infty$.

In order to decrease the bound on the failure probability of the PoR schemes we propose, one could also consider lifted codes $\mathrm{Lift}(\mathrm{RS}_q(d),2)$ with a lower degree $d < q-2$. A good point is that the communication complexity remains unchanged; however if d is too small, we could once again observe an overwhelming storage overhead.

5.5.3 Experimental estimate of the bias α

In this last section, we confirm our heuristic on the fact that Φ_w is sufficiently uniform, by providing experimental estimates of α for finite length codes.

Setup. We consider PoR schemes using Reed-Muller codes $\mathcal{C} = \mathrm{RM}_q(2, q-2)$, as presented in Section 5.5.2. We also fix the word $\boldsymbol{w} \in \mathbb{F}_q^n$ uploaded on the server during the initialisation step, and without loss of generality we assume that $\boldsymbol{w} = \mathbf{0}$. Proposition 5.21 claims that in this context, α should be $\mathcal{O}(1/q)$ since $\Delta = 2$ and $\ell \leq q$. For convenience, we denote by $p_\Phi := \mathbb{P}_{\Phi_w}(V^\sigma(u, \boldsymbol{r}_u) = 0)$, and we recall that α is an upper bound on p_Φ (for varying u and $\boldsymbol{r}$).

We proceed to three kinds of tests in order to estimate α:

- **Test 1.** We sample N challenges u, and for each sample, we fix $t \in [2, \ell]$ and $\boldsymbol{r}_u$ in $\{\boldsymbol{x} \in \mathbb{F}_q^\ell, |Z_u| = t\}$. Then, we estimate p_Φ by running M trials and computing the average number of times $V^\sigma(u, \boldsymbol{r}_u) = 0$ occurs. We denote by $\xi_M(p_\Phi)$ this *estimator*. We then collect the maximum value of $\xi_M(p_\Phi)$ among the N samples of u.
- **Test 2.** A challenge u is fixed, and for several values of t, we pick N responses $\boldsymbol{r}_u$ randomly in $\{\boldsymbol{x} \in \mathbb{F}_q^\ell, |Z_u| = t\}$. For every $\boldsymbol{r}_u$, we estimate p_Φ with M samples. We collect the maximum value of $\xi_M(p_\Phi)$ among the N values of $\boldsymbol{r}_u$ that have been picked.
- **Test 3.** A challenge u is fixed, and for several values of $t \in [2, \ell]$, we also set a response $\boldsymbol{r}_u$ to this challenge which satisfies $|Z_u| = t$. We then run M trials and collect $\xi_M(p_\Phi)$.

Influence of M and the chosen test on the estimator. At the end of the document, Figures 5.4, 5.5 and 5.6 confirm that, for fixed N and q, and for any Test i we use, $i \in \{1,2,3\}$, our estimator $\xi_M(p_\Phi)$ converges to a value close to $1/(q-1)$.

Influence of N on the estimator. Table 5.2 shows experimentally that, for M large enough and fixed q, the number N has few influence on the estimator (N being respectively the number of responses $\boldsymbol{r}_u$ sampled in Test 2, and the number of challenges u sampled in Test 1). The minor increase we can observe on $\xi_M(p_\Phi)$ can be thought as a standard deviation due to the fact that the number of samples $M = 100,000$ is finite.

Influence of q on the estimator. In Table 5.3, we show that estimator $\xi_M(p_\Phi)$ converges to an expected value $1/(q-1)$, for any value of q.

N	$q=8$	$q=64$	N	$q=8$	$q=64$
1	0.1418	0.0152	1	0.1414	0.0158
5	0.1433	0.0163	5	0.1431	0.0162
10	0.1443	0.0165	10	0.1452	0.0166
50	0.1455	0.0169	50	0.1450	0.0168
100	0.1452	0.0167	100	0.1458	0.0168
500	0.1464	0.0169	500	0.1470	0.0168
$1/(q-1)$	0.1429	0.01587	$1/(q-1)$	0.1429	0.01587

Table 5.2 – Estimators $\xi_M(p_\Phi)$ using Test 1 (on the left) and Test 2 (on the right) with $M = 100,000$ and $t = 2$, for $q \in \{8, 64\}$ and various values of N. The quantity $1/(q-1)$ represents an estimated upper bound on α that $\xi_M(p_\Phi)$ should approximate.

q	$\xi_M(p_\Phi)$	$1/(q-1)$
4	0.333	0.3333
8	0.143	0.1429
16	0.0665	0.06667
32	0.032	0.03226
64	0.0161	0.01587
128	0.00791	0.007874
256	0.00382	0.003922

q	$\xi_M(p_\Phi)$	$1/(q-1)$
7	0.166	0.1667
17	0.0627	0.0625
31	0.0335	0.03333
257	0.00398	0.004000

Table 5.3 – Estimators $\xi_M(p_\Phi)$ using Test 3 with $M = 1,000,000$ and $t = 2$, for various values of prime powers q. The quantity $1/(q-1)$ represents an estimated upper bound on α that $\xi_M(p_\Phi)$ should approximate.

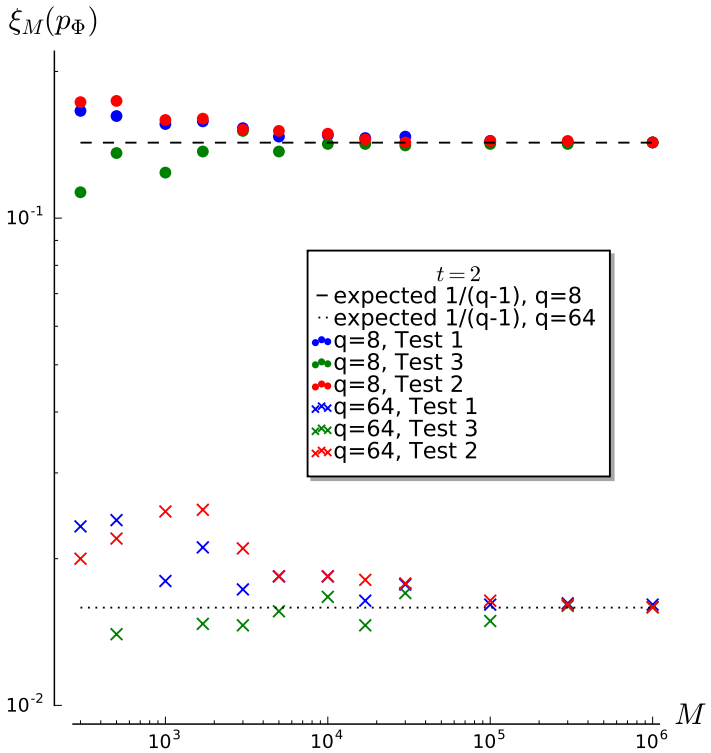


Figure 5.4 – Estimators $\xi_M(p_\Phi)$ for various values of $M \in [10^3, 10^6]$, of $q \in \{8, 64\}$, and of Test i, $i \in \{1, 2, 3\}$. Support size $t = 2$ is fixed. For Tests 1 and 2, the parameter N is set to 10. Black horizontal lines represent the expected value of α.

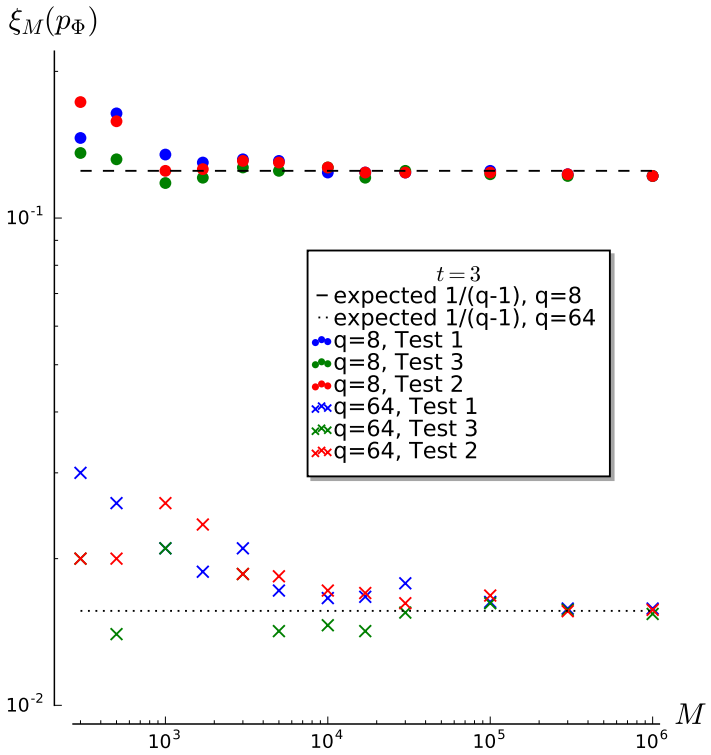


Figure 5.5 – Estimators for various values of $M \in [10^3, 10^6]$, of $q \in \{8, 64\}$, and of Test i, $i \in \{1, 2, 3\}$. Support size $t = 3$ is fixed. For Tests 1 and 2, the parameter N is set to 10. Black horizontal lines represent the expected value of α.

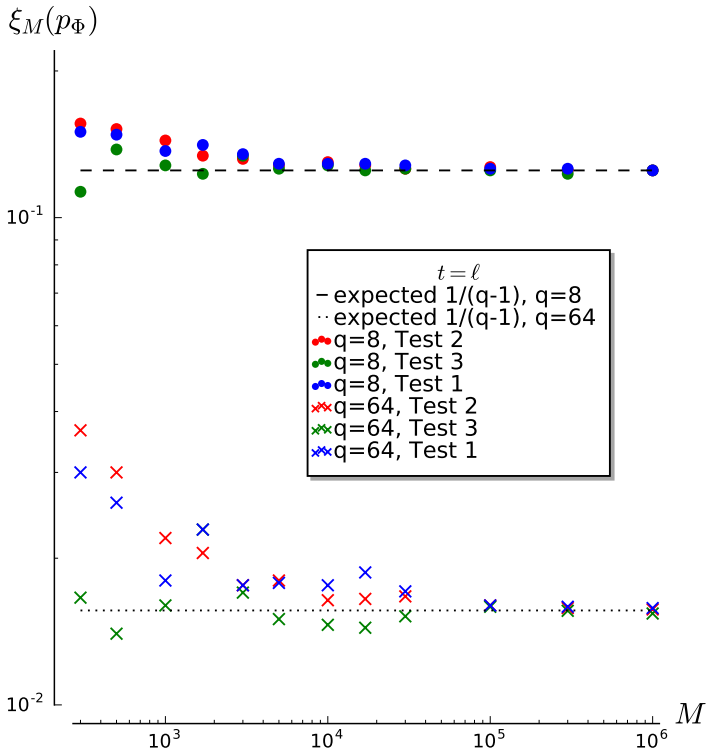


Figure 5.6 – Estimators for various values of $M \in [10^3, 10^6]$, of $q \in \{8, 64\}$, and of Test i, $i \in \{1, 2, 3\}$. Support size $t = \ell$ is fixed. For Tests 1 and 2, the parameter N is set to 10. Black horizontal lines represent the expected value of α.

Conclusion

As a conclusion, we propose several tracks of research on topics we addressed in this thesis.

Codes with locality and designs. In Chapter 2 we proved that generalising the classical construction of codes based on designs yields good LCCs. It could be of interest to pursue this study, *e.g.* by providing bounds on their parameters (for instance their dimension, length or locality). New good instances of design-based LCCs which do not come from classical geometries or algebraic constructions would also be of interest, since they would highly differ from the good LCCs we currently know.

More theoretically, one could also ask whether the *combinatorial* version of LCCs we give (based on designs) can achieve parameters as good as general LCCs. Indeed, design-based LCCs intrinsically give rise to local reconstruction algorithms (*i.e.* they never fail when the codeword is not corrupted), which is not required in the definition of LCCs.

Finally, we know that, in the constant locality regime, there is a strict distinction between LDCs and LCCs in terms of parameters. Thus, the construction of locally *decodable* codes from designs might be worthwhile to address.

Lifted codes. At the end of Chapter 3, we proposed a thorough study of the degree set (and *a fortiori* the dimension) of affine lifted codes $\mathrm{Lift}(\mathrm{RS}_q(k), m = 2)$. An immediate and interesting project would be to extend the investigation to larger orders $m \geq 3$.

A more ambitious and theoretical avenue of research would consider the generalisation of the lifting process to other kind of varieties and embeddings. The work of Guo [Guo16], where Hermitian codes are lifted, can be seen as a precursor to this study. But to the author's opinion (and ours), the construction admits minor weaknesses. For instance, Guo's so-called Φ-*lifting* of a code $\mathcal{C}_0 \subseteq \mathbb{F}_q^X$ leads to a longer code $\mathcal{C}$ which, by construction, is supported by direct products X^m. The local properties of $\mathcal{C}$ come from embeddings $X \to X^m$, but they highly depend on the richness of the automorphism group of X, and intrinsically give *non-perfectly smooth* local correcting algorithms. Ideally speaking, one would like to result in codes $\mathcal{C}$ supported by a variety $\mathcal{V}$ such that a nicely structured set of embeddings $X \to \mathcal{V}$ yields interesting local properties to $\mathcal{C}$.

Private information retrieval. First natural questions related to our construction of PIR protocols concern the existence of transversal designs yielding codes with better parameters. We would also be very curious to know if divisible codes over large alphabets can exist, given that they lead to PIR protocols with storage overhead less than two.

In the vein of recent works in PIR (see Section 4.5), one could also look for the *capacity* of PIR protocols with *no* computation on the server side. Indeed, the PIR rate we currently obtain is very far from the capacity for usual coded PIR, due to the computational constraints on servers. In that sense, it would also be of interest to relax a bit these constraints in order to obtain better PIR rates — for instance we could require constant, or sublinear computational complexity in the size of the database.

Other cryptographic applications? We proved in Chapters 4 and 5 that codes with locality allow the construction of cryptographic protocols with low communication complexity. We have also seen that a crucial point for proving the security of the underlying protocols is the richness of the structure of low-weight parity-check equations for the code. We strongly believe that the codes we presented above can be applied for the construction of *other* cryptographic protocols that require low communication. Hence, we complete this thesis by encouraging research in this direction.

List of Algorithms

List of Figures

List of Tables

Bibliography

[ABC+11] Giuseppe Ateniese, Randal C. Burns, Reza Curtmola, Joseph Herring, Osama Khan, Lea Kissner, Zachary N. J. Peterson, and Dawn Song. Remote Data Checking using Provable Data Possession. *ACM Trans. Inf. Syst. Secur.*, 14(1):12:1–12:34, 2011. (Cited on page 114).

[AK92] Edward F. Assmus and Jennifer D. Key. *Designs and Their Codes*. Cambridge Tracts in Mathematics. Cambridge University Press, 1992. (Cited on pages 8, 9, and 63).

[AL96] Noga Alon and Michael Luby. A Linear Time Erasure-Resilient Code with Nearly Optimal Recovery. *IEEE Trans. Information Theory*, 42(6):1732–1736, 1996. (Cited on page 35).

[ALM+98] Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof Verification and the Hardness of Approximation Problems. *J. ACM*, 45(3):501–555, 1998. (Cited on page 16).

[Alo99] Noga Alon. Combinatorial Nullstellensatz. *Combinatorics, Probability and Computing*, 8(1-2):7–29, January 1999. (Cited on page 63).

[ALS14] Daniel Augot, Françoise Levy-dit-Vehel, and Abdullatif Shikfa. A Storage-Efficient and Robust Private Information Retrieval Scheme Allowing Few Servers. In Dimitris Gritzalis, Aggelos Kiayias, and Ioannis G. Askoxylakis, editors, *Cryptology and Network Security - 13th International Conference, CANS 2014, Heraklion, Crete, Greece, October 22-24, 2014. Proceedings*, volume 8813 of *Lecture Notes in Computer Science*, pages 222–239. Springer, 2014. (Cited on pages xi, xvii, 88, 89, and 111).

[AS98] Sanjeev Arora and Shmuel Safra. Probabilistic Checking of Proofs: A New Characterization of NP. *J. ACM*, 45(1):70–122, 1998. (Cited on page 16).

[AY17] Hilal Asi and Eitan Yaakobi. Nearly Optimal Constructions of PIR and Batch Codes. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 151–155. IEEE, 2017. (Cited on page 111).

[BDL14] Abhishek Bhowmick, Zeev Dvir, and Shachar Lovett. New Bounds for Matching Vector Families. *SIAM J. Comput.*, 43(5):1654–1683, 2014. (Cited on page 31).

[BdM01] Thierry P. Berger and Louis de Maximy. Cyclic Projective Reed-Muller Codes. In Serdar Boztas and Igor E. Shparlinski, editors, *Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 14th International Symposium, AAECC-14, Melbourne, Australia November 26-30, 2001, Proceedings*, volume 2227 of *Lecture Notes in Computer Science*, pages 77–81. Springer, 2001. (Cited on page 66).

[BDSS16] Arnab Bhattacharyya, Zeev Dvir, Shubhangi Saraf, and Amir Shpilka. Tight Lower Bounds for Linear 2-query LCCs over Finite Fields. *Combinatorica*, 36(1):1–36, 2016. (Cited on page 36).

[BDYW11] Boaz Barak, Zeev Dvir, Amir Yehudayoff, and Avi Wigderson. Rank Bounds for Design Matrices with Applications to Combinatorial Geometry and Locally Correctable Codes. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 519–528. ACM, 2011. (Cited on page 36).

[BE17] Simon R. Blackburn and Tuvi Etzion. PIR Array Codes with Optimal PIR Rates. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 2658–2662. IEEE, 2017. (Cited on page 111).

[BEP17] Simon R. Blackburn, Tuvi Etzion, and Maura B. Paterson. PIR Schemes with Small Download Complexity and Low Storage Requirements. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 146–150. IEEE, 2017. (Cited on page 111).

[BFLS91] László Babai, Lance Fortnow, Leonid A. Levin, and Mario Szegedy. Checking Computations in Polylogarithmic Time. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 21–31. ACM, 1991. (Cited on page 16).

[BGK+13] Eli Ben-Sasson, Ariel Gabizon, Yohay Kaplan, Swastik Kopparty, and Shubhangi Saraf. A New Family of Locally Correctable Codes Based on Degree-Lifted Algebraic Geometry Codes. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 833–842. ACM, 2013. (Cited on pages 34 and 35).

[BIKR02] Amos Beimel, Yuval Ishai, Eyal Kushilevitz, and Jean-François Raymond. Breaking the $O(n^{1/(2k-1)})$ Barrier for Information-Theoretic Private Information Retrieval. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 261–270. IEEE Computer Society, 2002. (Cited on page 87).

[BIW10] Omer Barkol, Yuval Ishai, and Enav Weinreb. On Locally Decodable Codes, Self-Correctable Codes, and t-Private PIR. *Algorithmica*, 58(4):831–859, 2010. (Cited on pages 36 and 90).

[BJO09] Kevin D. Bowers, Ari Juels, and Alina Oprea. Proofs of Retrievability: Theory and Implementation. In Radu Sion and Dawn Song, editors, *Proceedings of the first ACM Cloud Computing Security Workshop, CCSW 2009, Chicago, IL, USA, November 13, 2009*, pages 43–54. ACM, 2009. (Cited on page 115).

[BLR93] Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-Testing/Correcting with Applications to Numerical Problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. (Cited on page 16).

[BMSS11] Eli Ben-Sasson, Ghid Maatouk, Amir Shpilka, and Madhu Sudan. Symmetric LDPC Codes are not Necessarily Locally Testable. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, June 8-10, 2011*, pages 55–65. IEEE Computer Society, 2011. (Cited on page 26).

[BS11] Eli Ben-Sasson and Madhu Sudan. Limits on the Rate of Locally Testable Affine-Invariant Codes. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 14th International Workshop, APPROX 2011, and 15th International Workshop, RANDOM 2011, Princeton, NJ, USA, August 17-19, 2011. Proceedings*, volume 6845 of *Lecture Notes in Computer Science*, pages 412–423. Springer, 2011. (Cited on page 26).

[BU18] Karim A. Banawan and Sennur Ulukus. The Capacity of Private Information Retrieval From Coded Databases. *IEEE Trans. Information Theory*, 64(3):1945–1956, 2018. (Cited on page 110).

[Bus52] K. A. Bush. Orthogonal arrays of index unity. *The Annals of Mathematical Statistics*, 23(3):426–434, 1952. (Cited on page 99).

[BZ06] Alexander Barg and Gilles Zémor. Distance Properties of Expander Codes. *IEEE Trans. Information Theory*, 52(1):78–90, 2006. (Cited on page 33).

[CD06] Charles J. Colbourn and Jeffrey H. Dinitz. *Handbook of Combinatorial Designs, Second Edition*. Chapman & Hall/CRC, 2006. (Cited on pages xiii, xxi, 97, 98, and 99).

[CG97] Benny Chor and Niv Gilboa. Computationally Private Information Retrieval. In Frank Thomson Leighton and Peter W. Shor, editors, *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 304–313. ACM, 1997. (Cited on page 86).

[CGKS95] Benny Chor, Oded Goldreich, Eyal Kushilevitz, and Madhu Sudan. Private Information Retrieval. In *36th Annual Symposium on Foundations of Computer Science, Milwaukee, Wisconsin, 23-25 October 1995*, pages 41–50. IEEE Computer Society, 1995. (Cited on pages xi, xvi, xvii, and 86).

[Cha90] Pascale Charpin. Codes cycliques étendus affines-invariants et antichaînes d'un ensemble partiellement ordonné. *Discrete Mathematics*, 80(3):229–247, 1990. (Cited on page 73).

[CHY15] Terence H. Chan, Siu-Wai Ho, and Hirosuke Yamamoto. Private Information Retrieval for Coded Storage. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 2842–2846. IEEE, 2015. (Cited on page 110).

[CKdR99] Neil J. Calkin, Jennifer D. Key, and Marialuisa J. de Resmini. Minimum Weight and Dimension Formulas for Some Geometric Codes. *Des. Codes Cryptography*, 17(1-3):105–120, 1999. (Cited on page 13).

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private Information Retrieval. *J. ACM*, 45(6):965–981, 1998. (Cited on pages xi, 86, 87, and 88).

[com17] Filecoin community. Filecoin: A Cryptocurrency Operated File Storage Network. `https://filecoin.io/filecoin.pdf`, 2017. [Online; accessed 13rd July 2018]. (Cited on page xvii).

[CWX14] Yeow Meng Chee, Liyasi Wu, and Chaoping Xing. Correcting on Curves and Highly Sound Locally Correctable Codes of High Rate. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 - July 4, 2014*, pages 2964–2966. IEEE, 2014. (Cited on page 42).

[DG16] Zeev Dvir and Sivakanth Gopi. 2-Server PIR with Subpolynomial Communication. *J. ACM*, 63(4):39:1–39:15, 2016. (Cited on page 87).

[DGY11] Zeev Dvir, Parikshit Gopalan, and Sergey Yekhanin. Matching Vector Codes. *SIAM J. Comput.*, 40(4):1154–1178, 2011. (Cited on pages 31 and 32).

[DS07] Zeev Dvir and Amir Shpilka. Locally Decodable Codes with Two Queries and Polynomial Identity Testing for Depth 3 Circuits. *SIAM J. Comput.*, 36(5):1404–1434, 2007. (Cited on page 36).

[DSW17] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Superquadratic Lower Bound for 3-Query Locally Correctable Codes over the Reals. *Theory of Computing*, 13(1):1–36, 2017. (Cited on page 36).

[Dür91] Arne Dür. The Decoding of Extended Reed-Solomon Codes. *Discrete Mathematics*, 90(1):21–40, 1991. (Cited on page 60).

[DVW09] Yevgeniy Dodis, Salil P. Vadhan, and Daniel Wichs. Proofs of Retrievability via Hardness Amplification. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 109–127. Springer, 2009. (Cited on page 115).

[Efr12] Klim Efremenko. 3-Query Locally Decodable Codes of Subexponential Length. *SIAM J. Comput.*, 41(6):1694–1703, 2012. (Cited on pages xvi, 31, 32, 36, and 87).

[FHGHK17] Ragnar Freij-Hollanti, Oliver W. Gnilke, Camilla Hollanti, and David A. Karpuk. Private Information Retrieval from Coded Databases with Colluding Servers. *SIAM J. Appl. Algebra Geometry*, 1(1):647–664, 2017. (Cited on page 110).

[FVY15a] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. Codes for Distributed PIR with Low Storage Overhead. In *IEEE International Symposium on Information Theory, ISIT 2015, Hong Kong, China, June 14-19, 2015*, pages 2852–2856. IEEE, 2015. (Cited on page 111).

[FVY15b] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. PIR with Low Storage Overhead: Coding instead of Replication. *CoRR*, abs/1505.06241, 2015. (Cited on page 111).

[Gao03] Shuhong Gao. A New Algorithm for Decoding Reed-Solomon Codes. In Vijay K. Bhargava, H. Vincent Poor, Vahid Tarokh, and Seokho Yoon, editors, *Communications, Information and Network Security*, pages 55–68. Springer US, 2003. (Cited on page 8).

[GD68] Jean-Marie Goethals and Philippe Delsarte. On a Class of Majority-Logic Decodable Cyclic Codes. *IEEE Trans. Information Theory*, 14(2):182–188, 1968. (Cited on page 12).

[GK16] Alan Guo and Swastik Kopparty. List-Decoding Algorithms for Lifted Codes. *IEEE Trans. Information Theory*, 62(5):2719–2725, 2016. (Cited on pages 67, 68, and 69).

[GKS13] Alan Guo, Swastik Kopparty, and Madhu Sudan. New Affine-Invariant Codes from Lifting. In Robert D. Kleinberg, editor, *Innovations in Theoretical Computer Science, ITCS '13, Berkeley, CA, USA, January 9-12, 2013*, pages 529–540. ACM, 2013. (Cited on pages viii, ix, xi, xvi, xviii, xix, xxii, 23, 26, 27, 29, 33, 35, 45, 46, 52, 59, 76, 78, and 137).

[GKST06] Oded Goldreich, Howard J. Karloff, Leonard J. Schulman, and Luca Trevisan. Lower Bounds for Linear Locally Decodable Dodes and Private Information Retrieval. *Computational Complexity*, 15(3):263–296, 2006. (Cited on page 36).

[GLR+91] Peter Gemmell, Richard J. Lipton, Ronitt Rubinfeld, Madhu Sudan, and Avi Wigderson. Self-Testing/Correcting for Polynomials and for Approximate Functions. In Cris Koutsougeras and Jeffrey Scott Vitter, editors, *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 32–42. ACM, 1991. (Cited on page 16).

[Gro00] Vince Grolmusz. Superpolynomial Size Set-Systems with Restricted Intersections mod 6 and Explicit Ramsey Graphs. *Combinatorica*, 20(1):71–86, 2000. (Cited on page 32).

[Gro02] Vince Grolmusz. Constructing Set Systems with Prescribed Intersection Sizes. *J. Algorithms*, 44(2):321–337, 2002. (Cited on page 32).

[GS92] Peter Gemmell and Madhu Sudan. Highly Resilient Correctors for Polynomials. *Inf. Process. Lett.*, 43(4):169–174, 1992. (Cited on page 16).

[Guo16] Alan Guo. High-Rate Locally Correctable Codes via Lifting. *IEEE Trans. Information Theory*, 62(12):6672–6682, 2016. (Cited on pages 29, 35, and 143).

[GW13] Venkatesan Guruswami and Carol Wang. Linear-Algebraic List Decoding for Variants of Reed-Solomon Codes. *IEEE Trans. Information Theory*, 59(6):3257–3268, 2013. (Cited on page 24).

[Ham68] Noboru Hamada. The rank of the incidence matrix of points and d-flats in finite geometries. *Journal of Science of the Hiroshima University, Series A-I (Mathematics)*, 32(2):381–396, 1968. (Cited on page 12).

[HOW15] Brett Hemenway, Rafail Ostrovsky, and Mary Wootters. Local Correctability of Expander Codes. *Inf. Comput.*, 243:178–190, 2015. (Cited on pages viii, xvi, and 33).

[HP10] William Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. Cambridge University Press, 2010. (Cited on page 4).

[Jen95] J. O. Jensen. On Decoding Doubly Extended Reed-Solomon Codes. In *Proceedings of 1995 IEEE International Symposium on Information Theory*, pages 280–. IEEE, 1995. (Cited on page 60).

[JK07] Ari Juels and Burton S. Kaliski, Jr. PORs: Proofs of Retrievability for Large Files. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 584–597. ACM, 2007. (Cited on pages xiv, xx, 113, 114, 115, 116, and 118).

[KdW04] Iordanis Kerenidis and Ronald de Wolf. Exponential Lower Bound for 2-query Locally Decodable Codes via a Quantum Argument. *J. Comput. Syst. Sci.*, 69(3):395–420, 2004. (Cited on pages xvi and 36).

[KK17] John Y. Kim and Swastik Kopparty. Decoding Reed-Muller Codes over Product Sets. *Theory of Computing*, 13(1):1–38, 2017. (Cited on page 25).

[KLP67] Tadao Kasami, Shu Lin, and W. Wesley Peterson. Some Results on Cyclic Codes which Are Invariant under the Affine Group and Their Applications. *Information and Control*, 11(5/6):475–496, 1967. (Cited on page 73).

[KLP68] Tadao Kasami, Shu Lin, and W. Wesley Peterson. New Generalizations of the Reed-Muller Codes – I: Primitive Codes. *IEEE Trans. Information Theory*, 14(2):189–199, 1968. (Cited on page 9).

[KMRS17] Swastik Kopparty, Or Meir, Noga Ron-Zewi, and Shubhangi Saraf. High-Rate Locally Correctable and Locally Testable Codes with Sub-Polynomial Query Complexity. *J. ACM*, 64(2):11:1–11:42, 2017. (Cited on page 35).

[KR06] Tali Kaufman and Dana Ron. Testing Polynomials over General Fields. *SIAM J. Comput.*, 36(3):779–802, 2006. (Cited on pages xix, 27, 53, and 74).

[KRGiA17] Siddhartha Kumar, Eirik Rosnes Rosnes, and Alexander Graell i Amat. Private Information Retrieval in Distributed Storage Systems using an Arbitrary Linear Code. In *2017 IEEE International Symposium on Information Theory, ISIT 2017, Aachen, Germany, June 25-30, 2017*, pages 1421–1425. IEEE, 2017. (Cited on page 110).

[KS08] Tali Kaufman and Madhu Sudan. Algebraic Property Testing: the Role of Invariance. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 403–412. ACM, 2008. (Cited on pages 68 and 72).

[KSY14] Swastik Kopparty, Shubhangi Saraf, and Sergey Yekhanin. High-Rate Codes with Sublinear-Time Decoding. *J. ACM*, 61(5):28:1–28:20, 2014. (Cited on pages viii, xi, xvi, xvii, 23, 24, 25, 26, 33, and 88).

[KT00] Jonathan Katz and Luca Trevisan. On the Efficiency of Local Decoding Procedures for Error-Correcting Codes. In F. Frances Yao and Eugene M. Luks, editors, *Proceedings of the Thirty-Second Annual ACM Symposium on Theory of Computing, May 21-23, 2000, Portland, OR, USA*, pages 80–86. ACM, 2000. (Cited on pages viii, xi, xvi, xvii, 16, 17, 35, and 87).

[Lac86] Gilles Lachaud. Projective Reed-Muller Codes. In *Coding Theory and Applications*, volume 311 of *Lecture Notes in Computer Science*, pages 125–129. Springer, 1986. (Cited on page 48).

[Lac90] Gilles Lachaud. The Parameters of Projective Reed-Muller Codes. *Discrete Mathematics*, 81(2):217–221, 1990. (Cited on page 48).

[LEB+03] Mark Lillibridge, Sameh Elnikety, Andrew Birrell, Michael Burrows, and Michael Isard. A Cooperative Internet Backup Scheme. In *USENIX Annual Technical Conference, General Track*, pages 29–41. USENIX, 2003. (Cited on page 114).

[LL16] Julien Lavauzelle and Françoise Levy-dit-Vehel. New Proofs of Retrievability using Locally Decodable Codes. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1809–1813. IEEE, 2016. (Cited on pages 116 and 138).

[LPS88] A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan Graphs. *Combinatorica*, 8(3):261–277, Sep 1988. (Cited on page 33).

[MM68] F. J. MacWilliams and H. B. Mann. On the p-Rank of the Design Matrix of a Difference Set. *Information and Control*, 12(5/6):474–488, 1968. (Cited on page 12).

[MS77] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North Holland, 1977. (Cited on pages 98, 105, and 108).

[MZC12] Zhen Mo, Yian Zhou, and Shigang Chen. A Dynamic Proof of Retrievability (PoR) Scheme with $O(\log n)$ Complexity. In *Proceedings of IEEE International Conference on Communications, ICC 2012, Ottawa, ON, Canada, June 10-15, 2012*, pages 912–916. IEEE, 2012. (Cited on page 115).

[NR09] Moni Naor and Guy N. Rothblum. The Complexity of Online Memory Checking. *J. ACM*, 56(1):2:1–2:46, 2009. (Cited on page 114).

[PSU13] Maura B. Paterson, Douglas R. Stinson, and Jalaj Upadhyay. A Coding Theory Foundation for the Analysis of General Unconditionally Secure Proof-of-Retrievability Schemes for Cloud Storage. *J. Mathematical Cryptology*, 7(3):183–216, 2013. (Cited on pages xiv, xxi, xxii, 113, 115, 116, 117, 118, and 120).

[PSU18] Maura B. Paterson, Douglas R. Stinson, and Jalaj Upadhyay. *J. Mathematical Cryptology*, pages 1–1, 2018. (Cited on page 115).

[PW04] Ruud Pellikaan and Xin-Wen Wu. List Decoding of q-ary Reed-Muller Codes. *IEEE Trans. Information Theory*, 50(4):679–682, 2004. (Cited on page 9).

[RS60] Irving S. Reed and Golomb Solomon. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics*, 8(2):300–304, 1960. (Cited on pages 7 and 99).

[RTM79] Irving S. Reed, Trieu-Kien Truong, and Robert L. Miller. Simplified Algorithm for Correcting both Errors and Erasures of Reed-Solomon Codes. *Institution of Electrical Engineers*, 126:961–963, 1979. (Cited on page 8).

[RTR97] Carlos Renterìa and Horacio Tapia-Recillas. Reed-Muller Codes: an Ideal Theory Approach. *Communications in Algebra*, 25(2):401–413, 1997. (Cited on page 47).

[RV16] Sankeerth Rao and Alexander Vardy. Lower Bound on the Redundancy of PIR Codes. *CoRR*, abs/1605.01869, 2016. (Cited on page 111).

[SJ17] Hua Sun and Syed Ali Jafar. The Capacity of Private Information Retrieval. *IEEE Trans. Information Theory*, 63(7):4075–4088, 2017. (Cited on page 110).

[SJ18a] Hua Sun and Syed Ali Jafar. Private Information Retrieval from MDS Coded Data With Colluding Servers: Settling a Conjecture by Freij-Hollanti et al. *IEEE Trans. Information Theory*, 64(2):1000–1022, 2018. (Cited on page 110).

[SJ18b] Hua Sun and Syed Ali Jafar. The Capacity of Robust Private Information Retrieval With Colluding Databases. *IEEE Trans. Information Theory*, 64(4):2361–2370, 2018. (Cited on page 110).

[Ska18] Vitaly Skachek. Batch and PIR Codes and Their Connections to Locally Repairable Codes. In Marcus Greferath, Mario Osvin Pavčević, Natalia Silberstein, and María Ángeles Vázquez-Castro, editors, *Network Coding and Subspace Designs*, pages 427–442. Springer International Publishing, 2018. (Cited on page 111).

[Smi69] K.J.C. Smith. On the p-rank of the incidence matrix of points and hyperplanes in a finite projective geometry. *Journal of Combinatorial Theory*, 7(2):122–129, 1969. (Cited on page 12).

[Sør91] Anders Bjært Sørensen. Projective Reed-Muller Codes. *IEEE Trans. Information Theory*, 37(6):1567–1576, 1991. (Cited on page 48).

[SR16] Binanda Sengupta and Sushmita Ruj. Efficient Proofs of Retrievability with Public Verifiability for Dynamic Cloud Storage. *CoRR*, abs/1611.03982, 2016. (Cited on page 115).

[SRR14] Nihar B. Shah, K. V. Rashmi, and Kannan Ramchandran. One Extra Bit of Download Ensures Perfectly Private Information Retrieval. In *2014 IEEE International Symposium on Information Theory, Honolulu, HI, USA, June 29 - July 4, 2014*, pages 856–860. IEEE, 2014. (Cited on pages 110 and 111).

[SS96] Michael Sipser and Daniel A. Spielman. Expander Codes. *IEEE Trans. Information Theory*, 42(6):1710–1722, 1996. (Cited on page 33).

[Sti04] Douglas R. Stinson. *Combinatorial Designs – Constructions and Analysis*. Springer, 2004. (Cited on pages 9 and 11).

[SW13] Hovav Shacham and Brent Waters. Compact Proofs of Retrievability. *J. Cryptology*, 26(3):442–483, 2013. (Cited on pages xiv, xxi, 114, 115, and 118).

[SY11] Shubhangi Saraf and Sergey Yekhanin. Noisy Interpolation of Sparse Polynomials, and Applications. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, June 8-10, 2011*, pages 86–92. IEEE Computer Society, 2011. (Cited on page 31).

[TGK+17] Razan Tajeddine, Oliver W. Gnilke, David A. Karpuk, Ragnar Freij-Hollanti, Camilla Hollanti, and Salim El Rouayheb. Private Information Retrieval Schemes for Coded Data with Arbitrary Collusion Patterns. In *ISIT*, pages 1908–1912. IEEE, 2017. (Cited on page 110).

[TR16] Razan Tajeddine and Salim El Rouayheb. Private Information Retrieval from MDS Coded Data in Distributed Storage Systems. In *IEEE International Symposium on Information Theory, ISIT 2016, Barcelona, Spain, July 10-15, 2016*, pages 1411–1415. IEEE, 2016. (Cited on page 110).

[WBB+16] Shawn Wilkinson, Tome Boshevski, Josh Brandoff, James Prestwich, Gordon Hall, Patrick Gerbes, Philip Hutchins, and Chris Pollard. Storj: A Peer-to-Peer Cloud Storage Network. `https://storj.io/storj.pdf`, 2016. [Online; accessed 13rd July 2018]. (Cited on page xvii).

[Wol89] Jacques Wolfmann. New Bounds on Cyclic Codes from Algebraic Curves. In Gérard Cohen and Jacques Wolfmann, editors, *Coding Theory and Applications*, pages 47–62. Springer Berlin Heidelberg, 1989. (Cited on page 73).

[Woo07] David P. Woodruff. New Lower Bounds for General Locally Decodable Codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(006), 2007. (Cited on page 36).

[Woo12] David P. Woodruff. A Quadratic Lower Bound for Three-Query Linear Locally Decodable Codes over Any Field. *J. Comput. Sci. Technol.*, 27(4):678–686, 2012. (Cited on pages xvi and 36).

[Woo14] Mary K. Wootters. *Any Errors in this Dissertation Are Probably Fixable: Topics in Probability and Error Correcting Codes*. PhD thesis, University of Michigan, 2014. (Cited on page 33).

[Wu15] Liyasi Wu. Revisiting the Multiplicity Codes: A New Class of High-Rate Locally Correctable Codes. In *53rd Annual Allerton Conference on Communication, Control, and Computing, Allerton 2015, Allerton Park & Retreat Center, Monticello, IL, USA, September 29 - October 2, 2015*, pages 509–513. IEEE, 2015. (Cited on page 29).

[WWR+11] Qian Wang, Cong Wang, Kui Ren, Wenjing Lou, and Jin Li. Enabling Public Auditability and Data Dynamics for Storage Security in Cloud Computing. *IEEE Trans. Parallel Distrib. Syst.*, 22(5):847–859, 2011. (Cited on pages xiv, xxi, and 115).

[WY05] David P. Woodruff and Sergey Yekhanin. A Geometric Approach to Information-Theoretic Private Information Retrieval. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 275–284. IEEE Computer Society, 2005. (Cited on page 41).

[Yek08] Sergey Yekhanin. Towards 3-query Locally Decodable Codes of Subexponential Length. *J. ACM*, 55(1):1:1–1:16, 2008. (Cited on pages xvi, 31, 36, and 87).

[Yek12] Sergey Yekhanin. Locally Decodable Codes. *Foundations and Trends in Theoretical Computer Science*, 6(3):139–255, 2012. (Cited on pages 16, 25, 32, 41, 67, and 88).

[Zém01] Gilles Zémor. On Expander Codes. *IEEE Trans. Information Theory*, 47(2):835–837, 2001. (Cited on page 33).

[ZYES18] Yiwei Zhang, Eitan Yaakobi, Tuvi Etzion, and Moshe Schwartz. On the Access Complexity of PIR Schemes. *CoRR*, abs/1804.02692, 2018. (Cited on page 111).

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Codes à propriétés locales : constructions et applications à des protocoles cryptographiques

Mots clés : codes correcteurs, codes localement corrigibles, stockage distant, protocoles cryptographiques, *block designs*

Résumé : Les codes localement corrigibles ont été introduits dans le but d'extraire une partie de l'information contenue dans un mot de code bruité, en effectuant un nombre limité de requêtes à ses symboles, ce nombre étant appelé la localité du code. Ces dernières années ont vu la construction de trois familles de tels codes, dont la localité est sous-linéaire en la taille du message, et le rendement est arbitrairement grand. Ce régime de paramètres est particulièrement intéressant pour des considérations pratiques.

Dans cette thèse, nous donnons une rapide revue de littérature des codes localement corrigibles, avant d'en proposer un modèle combinatoire générique, à base de *block designs*. Nous définissons et étudions ensuite un analogue, dans le cas projectif, des relèvements affines de codes introduits par Guo, Kopparty et Sudan. Nous établissons par ailleurs plusieurs liens entre ces deux familles, pour finir par une analyse précise de la structure monomiale de ces codes dans le cas du relèvement plan.

Une deuxième partie de la thèse se focalise sur l'application de ces codes à deux protocoles cryptographiques. D'abord, nous proposons un protocole de récupération confidentielle d'information (*private information retrieval, PIR*) à partir de codes basés sur des designs transversaux, dont la taille des blocs s'apparente à la localité d'un code localement corrigible. Les protocoles ainsi construits ont l'avantage de n'exiger aucun calcul pour les serveurs, et de présenter une faible redondance de stockage ainsi qu'une complexité de communication modérée. Ensuite, nous donnons une construction générique de preuve de récupérabilité (*proof of retrievability, PoR*) à base de codes admettant une riche structure d'équations de parité à petit poids. Nous en donnons finalement une analyse de sécurité fine ainsi que plusieurs instanciations fondées sur des codes à propriétés locales.

Title: Codes with locality: constructions and applications to cryptographic protocols

Keywords: error-correcting codes, locally correctable codes, remote storage, cryptographic protocols, block designs

Abstract: Locally correctable codes (LCCs) were introduced in order to retrieve pieces of information from a noisy codeword, by using a limited number of queries to its symbols, this number being called the locality. Three main families of LCCs reaching sublinear locality and arbitrarily high rate have been built so far. This specific range of parameters is of particular interest concerning practical applications of LCCs.

After giving a state of the art for LCCs, we study how they can be built using block designs. We then give an analogue over projective spaces of the family of affine lifted codes introduced by Guo, Kopparty and Sudan. We exhibit several links between both families, and we give a precise analysis of the monomial structure of the code in the case of the lifting of order 2.

The second part of the thesis focuses on the application of these codes to two cryptographic protocols. We first build a new private information retrieval (PIR) protocol from codes based on transversal designs, whose block size defines the locality of the code. Our construction features no computation on the server side, low storage overhead and moderate communication complexity. Then, we propose a new generic construction of proof-of-retrievability (PoR) that uses codes equipped with an elaborate structure of low-weight parity-check equations. We give a rigorous analysis of the security of our scheme, and we finally propose practical instantiations based on codes with locality.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France