Algebraic techniques for decoding Reed-Solomon codes and cryptanalyzing McEliece-like cryptosystems

Rocco Mora

SORBONNE UNIVERSITÉ
CRÉATEURS DE FUTURS DEPUIS 1257

Inria

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Rocco Mora

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Algebraic techniques for decoding Reed-Solomon codes and cryptanalyzing McEliece-like cryptosystems

soutenue publiquement le 7 avril 2023

devant le jury composé de :

Jean-Pierre Tillich	Inria de Paris	Directeur
Ayoub Otmani	Université de Rouen	Rapporteur
Daniel Smith-Tone	NIST, University of Louisville	Rapporteur
Magali Bardet	Université de Rouen	Examinatrice
Alain Couvreur	Inria de Saclay	Examinateur
Vincent Neiger	Sorbonne Université	Examinateur
Joachim Rosenthal	University of Zurich	Examinateur
Nicolas Sendrier	Inria de Paris	Examinateur

Acknowledgments

Dio t' strabenedìssa!

Marisa

First of all, I would like to express my deepest gratitude to Jean-Pierre, my thesis advisor. Thank you for responding to that email, now four years ago, in which a student wondered about the possibility of pursuing a Ph.D. under your supervision. Thank you also for proposing to me this thesis project, which has shown new and unexpected developments and potential over time and still has a lot to offer.

Your guidance has been invaluable and your advice and insights have helped me to navigate through various challenges. I appreciated your constructive directness, which has been a catalyst for my growth and improvement. In parallel, you have always encouraged me to pursue and not give up when I was stuck. You have been a virtuous example not only of scientific quality but also of ethics and integrity in research. I hope to have learned as much as possible from your teachings.

I am also profoundly indebted to my other co-author, Magali. Your expertise in Gröbner bases has been immensely helpful, and your patient help with algebraic cryptanalysis over the years has been priceless. You have instilled in me a keen attention to detail and a preference for accuracy without detriment to overall clarity. I enjoyed every time you were right in discussions with Jean-Pierre, whether it was about solving systems or predicting Covid-related government measures.

My sincere thanks then go to Alain for the many discussions you have taken the time to devote to me, despite your increasing commitments. Talking with you is always a pleasure, you are an inexhaustible source of ideas and your deep and always instantaneous understanding of the subject never ceased to amaze me.

I am very grateful to Ayoub Otmani and Daniel Smith-Tone for agreeing to be reviewers of this dissertation and for their accurate reading of my manuscript. I extend my gratitude to all the other researchers and professors who agreed to be members of the jury: Magali Bardet, Alain Couvreur, Vincent Neiger, Joachim Rosenthal and Nicolas Sendrier. I am truly honored that these names appear on the first page of this document.

Surviving the French bureaucracy is certainly consuming and frustrating: biblical times, redundant documentation, inefficiency and incompetence often threaten our sound sleeps. The task becomes even more arduous for foreigners, for whom an

additional collection of headaches is graciously reserved. Thanks SU for teaching me this from the very first day and reminding me to the very last.

Fortunately, my doctorate took entirely place within the Cosmiq team at the INRIA Paris center. This has been an extremely motivating environment: the continuing achievements of its members have been an incentive to roll up my sleeves and work harder. Thanks to all the people who work here permanently or who transited for a more or less long period during my Ph.D.: André C., André S., Andrea, Anne, Anthony, Augustin, Aurélie, Aurelien, Charles, Christelle, Christina, Clara, Clémence, Daniel, Étienne, Ferdinand, Freja, Gaëtan, Jean-Pierre, Johanna, Jules, Kévin, Léo, Loïc, Lucien, María, Mathys, Matthieu, Maxime, Nicolas D., Nicolas S., Nicholas, Pascale, Paul, Pierre, Rémi, Ritam, Simona, Thomas, Valentin, Virgile, Xavier. I hope I have forgotten as few as possible. A special thought goes to the members of the so-called "1pm squad", I had a very good time during our conversations and I hope that we will stay in touch.

Un ringraziamento va a tutti i miei familiari e in particolare ai miei genitori, sempre felici di accogliermi nelle mie ormai rare visite. Grazie mamma per il tuo costante supporto in tutti questi anni di studio. Cosa sia questo "dottorato" e come trascorra le mie giornate rimane tuttoggi avvolto dal mistero, ma spero che questo manoscritto sia la prova che non ho trascorso gli ultimi tre anni e mezzo nel perseguire attività illecite. Grazie papà per la tua perenne fiducia e per sapere che, anche qualora si fosse trattato di affari loschi, avrei saputo cavarmela egregiamente.

Infine, grazie Martina. Per aver condiviso gioie ed esperienze ma anche insicurezze e momenti di sconforto. Per la compagnia nei lunghi mesi di confinamento (nonché per aver provveduto al mio sostentamento!). Per essere sempre stata presente quando avevo bisogno o ero in difficoltà. E, ancora di più, per la forza che hai dimostrato nell'affrontare le avversità.

Contents

List of Tables

Introduction (Français)

Le contexte

Les **codes correcteurs d'erreurs linéaires** sont le leitmotiv de ce manuscrit. Un code linéaire $\mathscr{C}$ est défini comme un sous-espace vectoriel sur un corps fini $\mathbb{F}$. En particulier, la *longueur du code* est le nombre n tel que $\mathscr{C}$ est un sous-espace de $\mathbb{F}^n$. La *dimension du code* est le nombre k qui exprime la dimension de $\mathscr{C}$ en tant que sous-espace de $\mathbb{F}$. Le rapport k/n est le *rendement du code*. L'autre notion clé d'un code linéaire est la *distance minimale* $d_{\min}$, définie comme la plus petite distance possible entre deux éléments, appelés *mots de code*, dans le code. Différentes définitions peuvent être adoptées pour calculer une distance. La métrique la plus courante est celle dite de *Hamming* : la distance de deux vecteurs est le nombre de coordonnées dans lesquelles ils diffèrent.

La notion de codes correcteurs d'erreurs (linéaires ou non) trouve son origine dans la sous-branche de la théorie de l'information appelée **théorie des codes**, à la frontière entre les mathématiques discrètes, l'informatique et le génie électrique, et qui a maintenant plus de 70 ans d'histoire. Ils ont été introduits à l'origine par Richard Hamming aux Bell Telephone Laboratories pour supprimer et/ou détecter les erreurs survenant dans les calculatrices mécaniques. Le problème central de la théorie des codes est donc ce qu'on appelle le "problème du décodage", qui peut être énoncé de la manière suivante. Étant donné un code linéaire $\mathscr{C} \subseteq \mathbb{F}^n$ et un vecteur $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}^n$, où $\boldsymbol{c} \in \mathscr{C}$ et le vecteur $\boldsymbol{e} \in \mathbb{F}^n$ obéit à une certaine distribution qui dépend du canal de communication, trouver efficacement $\boldsymbol{c}$. Cependant, le problème du décodage d'un code linéaire (aléatoire) générique est bien connu pour appartenir à la classe de complexité algorithmique NP-complet. La plupart des efforts déployés dans ce contexte ont donc été orientés vers la recherche de codes ayant une structure spécifique et qui, par conséquent, admettent des algorithmes de décodage efficaces.

À cet égard, plusieurs familles de codes ont été découvertes en l'espace d'une décennie environ à partir de 1950 : les codes de Hamming, de Reed-Muller (RM), de Bose-Chauduri-Hocquenghem (BCH) et de Reed-Solomon (RS), qui portent tous le nom de leur inventeur. Ces familles, et bien d'autres, peuvent être classées dans la catégorie des *codes algébriques*. En effet, elles sont caractérisées par des codeurs et des décodeurs, c'est-à-dire par des algorithmes permettant respectivement de coder et de décoder un message, basés sur des propriétés algébriques. En d'autres termes, leur structure mathématique permet de concevoir des algorithmes spécifiques qui atteignent de très bonnes capacités de correction d'erreurs et qui sont également très efficaces. En particulier, ces décodeurs traitent le problème de la résolution d'équations linéaires ou algébriques. En outre, les codes algébriques bénéficient généralement de représentations plus compactes que les codes aléatoires.

Depuis le début de la seconde moitié du 20ème siècle, la **théorie algébrique du codage** est toujours restée un domaine extrêmement actif tant dans le monde de la recherche que dans les applications industrielles, en raison de ses nombreux avantages. Une liste incomplète des applications réelles des codes mentionnés comprend les communications par téléphone mobile, les transmissions spatiales (certains d'entre eux ont été utilisés pour des missions de la NASA), le stockage de données (lecteurs de CD et de DVD, lecteurs USB et disques), les codes à barres bidimensionnels, la cryptographie.

Décodage des codes de Reed-Solomon

Considérons par exemple un code de Reed-Solomon. De manière informelle, il est défini comme l'ensemble des évaluations par composantes sur un vecteur $\boldsymbol{x} \in \mathbb{F}^n$ de tous les polynômes sur $\mathbb{F}$ avec un degré strictement limité par un entier positif k. La valeur k coïncide avec la dimension du code.

Du point de vue du décodage, la notion de distance minimale joue un rôle essentiel. Supposons qu'un message codé soit transmis et que ce que nous recevons soit $\boldsymbol{y} \in \mathbb{F}^n$. Nous voulons le décoder par rapport à un code linéaire $\mathscr{C}$ utilisé pour coder le message. Supposons également qu'il existe un mot-code $\boldsymbol{c} \in \mathscr{C}$ dont la distance par rapport à $\boldsymbol{y}$ est inférieure à la moitié de la distance minimale du code $d_{\min}$. Par définition de la distance minimale et de l'inégalité triangulaire, on obtient que tout autre mot du code $\boldsymbol{c}'$ *dans*$\mathscr{C}$ diffère de $\boldsymbol{y}$ par plus de $d_{\min}/2$ coordonnées. Ainsi, $\boldsymbol{c}$ est le mot du code le plus proche de $\boldsymbol{y}$ et il est unique. Pour cette raison, la valeur $d_{\min}/2$ est appelée *rayon de décodage unique*. Dans un canal de communication où la probabilité qu'une position unique soit mal reçue est suffisamment faible, $\boldsymbol{c}$ est le mot du code qui a le plus de chances de coïncider avec le message codé original. Un algorithme qui produit un tel mot de code est en effet appelé *décodeur à maximum de vraisemblance*.

Cependant, nous ne disposons a priori d'aucune information sur le mot du code le plus proche de $\boldsymbol{y}$ et le calcul de la distance de $\boldsymbol{y}$ pour tout élément de $\mathscr{C}$ prend un temps exponentiel dans la dimension du code.

Malgré cela, il existe un algorithme efficace, l'algorithme de Berlekamp-Welch [WB86], qui peut corriger efficacement les erreurs dans un code de Reed-Solomon jusqu'au rayon de décodage unique. Bien entendu, il exploite la structure algébrique des codes RS et utilise de manière cruciale l'interpolation de Lagrange pour reconstruire le mot de code envoyé. L'algorithme de Berlekamp-Welch est encore plus impressionnant si l'on tient compte du fait que les codes de Reed-Solomon sont des codes *maximum distance séparable* (MDS). Cela signifie qu'ils atteignent ce qu'on appelle la *borne de Singleton* et ont donc la plus grande distance minimale possible pour un code de même dimension et de même longueur.

Mais l'histoire ne s'arrête pas là. Le rayon de décodage unique garantit l'unicité dans le pire des cas. Mais même au-delà du rayon de décodage unique, la plupart du temps, il n'y a qu'un ou quelques mots de code dans la distance assignée. Dans un article révolutionnaire datant de 1997, qui a valu à son auteur le prix Nevanlinna, Sudan a modifié l'algorithme de Berlekamp-Welch en un *algorithme de décodage en liste* [Sud97], i.e. dans un décodeur qui produit une liste de mots de code dans un rayon donné, que nous appelons le rayon de Sudan, de $\boldsymbol{y}$ qui est plus grand que $d_{\min}/2$ pour les codes à haut rendement. Peu après, Guruswami et Sudan ont

encore amélioré l'algorithme [GS99], élargissant le rayon de décodage pour tout code RS jusqu'à la limite de Johnson, qui délimite la plage où la liste des solutions est polynomialement bornée.

Plus récemment, d'autres approches de décodage ont été proposées. Parmi elles, nous mentionnons le *power decoding* [Nie14; Nie18; SSB10], dont les équations clés seront étudiées, d'un point de vue algébrique, dans cette thèse.

Cryptographie à base de codes et codes de Goppa

L'un des principaux défis de la cryptographie consiste à concevoir des techniques permettant de protéger les messages et les données des adversaires, afin que seuls l'expéditeur et le destinataire prévu puissent les lire. La cryptographie moderne fait un usage intensif des mathématiques et de l'informatique théorique. En particulier, les algorithmes cryptographiques s'appuient fortement sur des hypothèses de difficulté calculatoire, c'est-à-dire sur les hypothèses selon lesquelles des problèmes spécifiques ne peuvent être résolus efficacement (où "efficacement" signifie généralement "en temps polynomial"). En d'autres termes, les schémas cryptographiques sont conçus de manière à ce qu'il soit prouvé ou raisonnablement impossible de les casser sans résoudre un problème de calcul difficile. Des exemples de ces problèmes sont la factorisation des entiers ou le logarithme discret, tous deux largement adoptés en cryptographie.

Au début de cette introduction, nous avons mentionné la difficulté prouvée de décoder un code aléatoire. Cela n'a pas échappé à l'attention des chercheurs. Le premier schéma basé sur la difficulté d'un problème emprunté à la théorie des codes, i.e. appartenant à **cryptographie à base de codes**, est le *cryptosystème de McEliece* [McE78], qui remonte à 1978, quelques mois seulement après la publication du schéma de chiffrement RSA [RSA78]. Avec ses quelque 45 ans d'histoire, c'est aussi l'un des précurseurs de la cryptographie à clé publique. Les systèmes qui appartiennent à cette catégorie sont caractérisés par deux clés, une privée et une publique. Toute personne possédant cette dernière est capable de crypter un message, mais seuls ceux qui connaissent la clé privée peuvent décrypter le texte chiffré.

Malgré un accueil initial mitigé du cryptosystème de McEliece de la part de la communauté académique, en faveur d'autres schémas à clé publique, celui-ci a plus récemment connu un regain d'intérêt. Ceci est en partie dû à l'émergence de l'information quantique. En effet, il existe des algorithmes quantiques qui peuvent résoudre de manière exponentiellement plus rapide certains problèmes de calcul employés en cryptographie. Par exemple, l'algorithme de Shor peut factoriser de grands entiers et calculer le logarithme discret en temps polynomial. Par conséquent, dans un avenir proche, l'ordinateur quantique pourrait être capable de casser presque tous les systèmes utilisés aujourd'hui dans la pratique. Le système cryptographique de McEliece, et plus généralement l'ensemble du domaine de la cryptographie à base de codes, est unanimement considéré comme n'étant pas vulnérable aux attaques quantiques. En d'autres termes, il est considéré comme une alternative à *résistance quantique* et est un candidat de *cryptographie post-quantique*. L'Institut national des normes et de la technologie (NIST) des États-Unis a lancé en 2016 un processus d'évaluation et de normalisation des systèmes de cryptage à clé publique post-quantiques. L'inconvénient du schéma McEliece est la taille énorme de la clé, par rapport à d'autres algorithmes cryptographiques, ce qui restreint l'éventail des

applications possibles. D'un autre côté, il possède des procédures d'encodage et de décodage extrêmement rapides et a survécu à une longue histoire de cryptanalyse. Pour ces raisons, Classic McEliece [Alb+20] a été admis au quatrième tour du concours du NIST [Ala+22] et on pense qu'il est prêt pour la normalisation.

Nous avons dit que la cryptographie basée sur le code s'appuie sur des problèmes difficiles de la théorie du codage, comme le problème du décodage. Cependant, dans des constructions comme le schéma de McEliece, le récepteur légitime doit être capable de décoder le texte chiffré envoyé. On ne sait pas s'il est possible d'y parvenir, même avec la clé privée, si le code utilisé est aléatoire. En d'autres termes, le code doit être choisi dans une famille dotée d'un algorithme de décodage efficace. Celui-ci est ensuite masqué de telle sorte que, sans connaître la clé privée, il n'est pas possible d'appliquer le décodeur. La proposition originale de McEliece suggère l'utilisation de **codes binaires de Goppa**, une sous-classe de **codes alternants**. Ces derniers sont la restriction à un sous-corps des codes Reed-Solomon généralisés (ou plus brièvement GRS), une extension des codes RS. Plusieurs autres familles ont été recommandées et étudiées, au fil des ans, comme les codes GRS eux-mêmes. Cependant, la plupart de ces variantes de McEliece ont ensuite été attaquées avec succès. Au contraire, les codes de Goppa ont résisté à la cryptanalyse jusqu'à présent, car les meilleures attaques connues à ce jour sont des algorithmes génériques de décodage de codes aléatoires et ont donc une complexité exponentielle.

Quoi qu'il en soit, il convient de souligner que les preuves de sécurité connues pour le cryptosystème de McEliece reposent sur deux hypothèses. La première, comme déjà expliqué, est que le décodage d'un code linéaire générique est difficile. La seconde est que les codes de Goppa se comportent comme des codes aléatoires. Pour être plus précis, nous disons que le *Goppa distinguishing problem*, qui demande de discriminer si un ensemble de vecteurs est la base d'un code de Goppa ou d'un code aléatoire, est intraitable du point de vue informatique. Les codes de Goppa se comportent en fait comme des codes aléatoires sous de nombreux aspects et pendant de nombreuses années, la croyance en cette hypothèse était robuste. Cependant, il y a environ 10 ans, un **distingueur** en temps polynomial, c'est-à-dire un algorithme qui résout le problème de distinguer, a été présenté [Fau+13]. Le contexte où il est efficace est cependant limité : il ne fonctionne que pour un code de Goppa, et plus généralement un code alternant, dont le rendement est assez élevé. En particulier, elle ne s'applique pas au McEliece classique, mais elle soulève tout de même quelques inquiétudes quant à la plausibilité de l'hypothèse de difficulté de distinguer un code de Goppa. De plus, elle invalide la preuve de sécurité d'autres schémas, tels que la signature numérique CFS [CFS01], qui est effectivement construite sur des codes de Goppa à haut rendement. Dans tous les cas, les auteurs de [Fau+13] ont laissé les problèmes d'atténuer davantage les contraintes du distingueur et de le transformer en attaque comme problèmes ouverts. Dans ce manuscrit, nous aborderons le distingueur de plusieurs points de vue, nous en donnerons un meilleur aperçu et nous tenterons de relever les deux défis mentionnés.

Bases de Gröbner

Contrairement à un système linéaire, la résolution d'un système polynomial multivarié est généralement difficile. Les principaux outils qui s'avèrent utiles à cet effet sont les bases de Gröbner introduites par Buchberger en 1965 [Buc65], ainsi qu'un algorithme

simple pour les calculer. D'autres algorithmes plus avancés ont été découverts par la suite (par exemple [Fau99; Fau02; Fau+93]) et les bases de Gröbner représentent toujours un domaine de recherche florissant. Étant donné un système de polynômes $f_1, \ldots, f_m$ définissant un système d'équations, les algorithmes de bases de Gröbner produisent récursivement de nouveaux polynômes appartenant à l'idéal généré par les polynômes initiaux $\mathcal{I} = \langle f_1, \ldots f_n \rangle$. L'idée principale est qu'ils généralisent la division polynomiale au cas non linéaire multivarié. En particulier, elles permettent de réduire un polynôme par rapport à un ensemble de polynômes de telle sorte que le reste ne dépende pas de l'ordre dans lequel les éléments de l'ensemble sont traités. Une base de Gröbner est en effet un ensemble générateur d'un idéal polynomial où toutes les réductions complètes d'un polynôme par la base produisent le même résultat. Cela peut dépendre de l'*ordre monomial* associé à l'anneau de polynômes.

Lorsque le système est affine, certaines combinaisons polynomiales des polynômes générés jusqu'à un certain point par un algorithme de base de Gröbner conduisent à des polynômes de petit degré. Lorsque cela se produit, on dit qu'une ou plusieurs *chutes de degré* se sont produites. Les chutes de degré sont souvent critiques dans la résolution d'un système. En effet, en raison de leur degré exceptionnellement bas, elles peuvent déclencher une chaîne d'autres chutes de degré après avoir été multipliées par d'autres monômes/polynômes, qui fournissent finalement une base de Gröbner. De là, il est possible de dériver la variété $\boldsymbol{V}(\mathcal{I})$ correspondant à l'idéal $\mathcal{I}$, dont les éléments sont les solutions de notre système multivarié. Les algorithmes de base de Gröbner ont une complexité exponentielle lorsqu'ils sont appliqués à la grande majorité des idéaux. Néanmoins, la résolution de certains systèmes spécifiques peut être réalisée en temps pratique (parfois même en temps polynomial dans les paramètres), en exploitant leur structure spéciale.

Dans ce manuscrit, les bases de Gröbner servent d'appareil technique pour aborder plusieurs problèmes issus de la théorie algébrique du codage et de la cryptographie. Plus précisément, le problème du décodage des codes RS peut être modélisé comme la résolution d'un système multivarié, dont la solution fournit directement le message codé sans positions en erreur. De même, le problème de la récupération de la clé privée dans le schéma de McEliece basé sur des codes alternants/de Goppa peut être exprimé par un système polynomial, dont la solution est la clé privée ou un équivalent. De plus, pour les codes alternants à haut rendement, le système devient plus facile à résoudre. Dans un cadre cryptographique, ce type d'analyse prend le nom de **cryptanalyse algébrique** et est devenu ces dernières années l'un des principaux outils pour inspecter la sécurité des schémas à base de codes, des schémas multivariés ainsi que certains chiffrements symétriques.

Contributions

Nous allons maintenant énumérer brièvement les contributions de cette thèse, chapitre par chapitre.

Chapitre 2.

Nous étudions un système algébrique bien connu qui modélise le problème de décodage des codes RS (ou de manière équivalente GRS) à l'aide de méthodes de base de Gröbner. Le système est bilinéaire, c'est-à-dire qu'il est linéaire par rapport à deux

blocs dans lesquels les variables sont réparties. Une littérature riche a été développée pour les systèmes bilinéaires génériques, mais le calcul d'une base de Gröbner devrait toujours avoir une complexité exponentielle pour les paramètres ciblés. Nous prouvons que ce n'est pas le cas pour la modélisation du décodage de Reed-Solomon, où même un algorithme de base de Gröbner simplifié s'exécute en temps polynomial jusqu'au même rayon de décodage atteint par l'algorithme de Sudan. Nous montrons que les nouveaux polynômes obtenus à partir du calcul de la base de Gröbner sont strictement liés au power decoding [Nie14; Nie18]. Il s'agit d'une approche alternative par rapport aux décodeurs de liste cités précédemment. Sa version améliorée atteint le même rayon de décodage que l'algorithme de Guruswami-Sudan, c'est-à-dire le rayon de Johnson. Cela suggère que notre approche pourrait fonctionner même au-delà de la borne de Sudan. À cet égard, nous montrons expérimentalement que, pour certains paramètres, notre méthode peut corriger un certain nombre d'erreurs jusqu'à et même légèrement au-delà de la borne de Johnson. Ce faisant, nous dérivons également de nouvelles identités polynomiales dans un seul bloc de variables qui ne sont pas exploitées par la stratégie de power decoding.

Publication associée: Magali Bardet, Rocco Mora et Jean-Pierre Tillich, *Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach*, International Symposium in Information Theory 2021 [BMT21].

Chapitre 3.

Le distingueur à haut rendement pour les codes alternants et de Goppa a été présenté à l'origine dans [Fau+13] comme le rang exceptionnellement petit d'une matrice construite à partir d'une base du code alternants/de Goppa. Pour être précis, aucune borne supérieure ou inférieure pour ce rang n'était donnée, mais seulement des explications algébriques basées sur des heuristiques. Grâce au lien donné dans [MP12], le distingueur peut être étudié de manière équivalente en termes de dimension du *carré* du dual du code alternant/de Goppa. Le carré d'un code est une construction de la théorie des codes qui a déjà été utilisée avec succès pour distinguer et/ou attaquer d'autres schémas à base de codes, par exemple, des variantes du schéma de McEliece basé sur des codes GRS [Cou+14]. Dans notre cas, ce point de vue alternatif permet de prouver une borne supérieure pour la dimension du code carré cible, rendant ainsi le distingueur plus rigoureux. De plus, notre preuve couvre le cas des codes de Goppa non binaires, pour lesquels [Fau+13] n'a fourni que des preuves empiriques. Les bornes supérieures sont serrées pour tous les paramètres et correspondent aux résultats expérimentaux. Afin de prouver les bornes supérieures, nous trouvons également de nouveaux résultats concernant la structure du produit et du carré des sous-codes sur de sous-corps en général et des codes alternants/de Goppa en particulier. Ce chapitre est donc ambivalent : il peut être considéré comme une contribution à la théorie algébrique du codage mais, étant donné l'intérêt bien connu de cette famille de codes en cryptographie, il peut également présenter un intérêt pour cette dernière.

Publication associée: Rocco Mora et Jean-Pierre Tillich, *On the dimension and structure of the square of the dual of a Goppa code*, Designs, Codes and Cryptography [MT22].

Chapitre 4.

Parfois, le distingueur peut être transformé en attaque. En cryptographie à base de codes, c'était le cas des codes GRS [Cou+14]. Il n'était pas clair si le distingueur de [Fau+13] aurait pu être exploité pour casser des instances de codes alternants/de Goppa à haut rendement du schéma de McEliece. Plusieurs attaques ont été trouvées sur des variantes liées aux codes de Goppa. Parmi elles, on peut citer les codes de Goppa quasi-cycliques et quasi-dyadiques [Fau+10b; GL09] ou les Wild Goppa codes [COT14a; FPP14]. La cryptanalyse algébrique a joué un rôle clé dans certains de ces exemples. En effet, la structure supplémentaire caractérisant le code sous-jacent a permis de réduire considérablement le nombre de variables dans le système multivarié modélisant le problème de récupération des clés. Cependant, aucune attaque n'était connue pour les codes alternants/de Goppa non structurés, même à rendement élevé. Dans ce chapitre, nous montons une attaque en temps polynomial qui fonctionne contre les codes alternants aléatoires binaires ou ternaires. L'algorithme se compose de deux parties. Tout d'abord, une *filtration* de codes alternants d'ordre décroissant, c'est-à-dire une séquence de codes alternants dont chacun est contenu dans le suivant, est calculée de manière itérative. Une fois qu'un code alternant d'ordre 3 est produit, la modélisation algébrique connue est mise en place. Dans la deuxième partie de l'attaque, nous fournissons un algorithme de base de Gröbner efficace et adapté à ce système spécifique. À partir de la variété associée à l'idéal généré par la base de Gröbner, une clé équivalente à la clé privée est récupérée. L'explication théorique est complétée par du code implémenté dans MAGMA. De manière assez surprenante, l'attaque ne fonctionne pas sur les codes de Goppa, même s'ils forment une sous-classe de codes alternants. Nous donnons également un aperçu des problèmes qui empêchent une adaptation directe du résultat aux codes de Goppa.

Publication associée: Magali Bardet, Rocco Mora et Jean-Pierre Tillich, *Polynomial time key-recovery attack on high rate random alternant codes*, Preprint [BMT23].

Chapitre 5.

Nous présentons une méthode qui permet d'améliorer le distingueur pour les codes alternants et de Goppa à haut rendement. Notre stratégie est cohérente avec la présentation donnée au Chapitre 3, car elle exploite également la construction du code carré. Cependant, nous l'améliorons en raccourcissant d'abord le code dual et en calculant ensuite le carré. Nous illustrons empiriquement que cette modification diminue le rendement minimal de code distinguable pour certains paramètres. En particulier, d'après nos expériences, cette stratégie semble être plus efficace pour les petits degrés d'extension et les grandes tailles de corps, et les codes alternants aléatoires sont plus affectés par cette approche que les codes de Goppa. Les résultats empiriques sont complétés par une explication algébrique partielle de la dimension du code carré qui en résulte.

Introduction

The context

Linear error-correcting codes are the leitmotif of this manuscript. A linear code $\mathscr{C}$ is defined as a vector subspace over a finite field $\mathbb{F}$. In particular, the *code length* is the number n such that $\mathscr{C}$ is a subspace of $\mathbb{F}^n$. The *code dimension* is the number k that expresses the dimension of $\mathscr{C}$ as an $\mathbb{F}$-subspace. The ratio k/n is the *code rate*. The other key notion of a linear code is the *minimum distance* $d_{\min}$, defined as the smallest possible distance between two elements, called *codewords*, in the code. Different definitions can be adopted for computing a distance. The most common metric is the so-called *Hamming metric*: the distance of two vectors is the number of coordinates in which they differ.

The notion of error-correcting codes (either linear or not) finds its origins in that sub-branch of information theory called **coding theory**, on the border among discrete mathematics, computer science and electrical engineering, and now boasting a history of more than 70 years. They were originally introduced by Richard Hamming at Bell Telephone Laboratories to remove and/or detect errors occurring in mechanical calculators. The central problem of coding theory is thus the so-called **decoding problem**, which can be stated in the following way. Given a linear code $\mathscr{C} \subseteq \mathbb{F}^n$ and a vector $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbb{F}^n$, where $\boldsymbol{c} \in \mathscr{C}$ and the vector $\boldsymbol{e} \in \mathbb{F}^n$ obeys to some distribution that depends on the communication channel, find efficiently $\boldsymbol{c}$. However, the decoding problem for a generic (random) linear code is well known to belong to the NP-complete computational complexity class. Much of the effort spent in this context has therefore been directed toward finding codes with a specific structure and which, consequently, admit efficient decoding algorithms.

In this respect, several families of codes have been discovered within approximately a decade starting from 1950: Hamming, Reed-Muller (RM), Bose-Chauduri-Hocquenghem (BCH) and Reed-Solomon (RS) codes, all of them taking the names from their inventors. These families, and many others, can be categorized as *algebraic codes*. Indeed, they are characterized by encoders and decoders, i.e. by algorithms to encode and decode a message respectively, based on algebraic properties. In other words, their mathematical structure allows to design specific algorithms that achieve very good error-correcting capabilities and that are also highly efficient. In particular, these decoders deal with the problem of solving linear or algebraic equations. Furthermore, algebraic codes typically benefit from more compact representations than random codes.

Since the beginning of the second half of the 20th century, **algebraic coding theory** has always remained an extremely active field both in the research world and in industrial applications, because of its many advantages. An incomplete list of real-

life applications of the mentioned codes includes mobile phone communications, space transmissions (some of them have been used for NASA missions), data storage (CDs and DVDs players, USB and disk drives), two-dimensional bar codes, **cryptography**.

Decoding of Reed-Solomon codes

Let us consider for instance a Reed-Solomon code. Informally speaking, this is defined as the set of component-wise evaluations on a vector $\boldsymbol{x} \in \mathbb{F}^n$ of all polynomials over $\mathbb{F}$ with degree strictly upper bounded by a positive integer k. The value k coincides with the dimension of the code.

From the point of view of decoding, the notion of minimum distance plays a key role. Suppose an encoded message is transmitted and what we receive is $\boldsymbol{y} \in \mathbb{F}^n$. We want to decode it with respect to a linear code $\mathscr{C}$ used for encoding the message. Let us also assume that there exists a codeword $\boldsymbol{c} \in \mathscr{C}$ whose distance from $\boldsymbol{y}$ is smaller than half the code minimum distance $d_{\min}$. By definition of minimum distance and triangular inequality, we obtain that any other codeword $\boldsymbol{c}' \in \mathscr{C}$ differs from $\boldsymbol{y}$ by more than $d_{\min}/2$ coordinates. Thus, $\boldsymbol{c}$ is the closest codeword to $\boldsymbol{y}$ and it is unique. For this reason, the value $d_{\min}/2$ is the so-called *unique decoding radius*. In a communication channel where the probability for a single position to be wrongly received is low enough, $\boldsymbol{c}$ is the codeword which more likely coincides with the original encoded message. An algorithm that outputs such a codeword is indeed called *maximum likelihood decoder*.

However, a priori we do not have any information about what is the closest codeword to $\boldsymbol{y}$ and computing the distance from $\boldsymbol{y}$ for any element of $\mathscr{C}$ takes exponential time in the code dimension.

In spite of that, there exists an efficient algorithm, the Berlekamp-Welch algorithm [WB86], that can efficiently correct errors in a Reed-Solomon code up to the unique decoding radius. Of course, it exploits the algebraic structure of RS codes and makes use in a crucial way of Lagrange interpolation to reconstruct the sent codeword. Berlekamp-Welch algorithm is even more impressive in light of the fact that Reed-Solomon codes are *maximum distance separable* (MDS) codes. This means that they attain the so-called *Singleton bound* and have therefore the largest possible minimum distance for a code of the same dimension and length.

This is not the end of the story, though. The unique decoding radius guarantees uniqueness in the worst case. But even beyond the unique decoding radius, most of the times there is just one or few codewords within the assigned distance. In a groundbreaking paper from 1997, which earned its author a Nevanlinna prize, Sudan modified the Berlekamp-Welch algorithm in a *list-decoding algorithm* [Sud97], i.e. in a decoder that outputs a list of codewords within a given radius, that we call Sudan's radius, from $\boldsymbol{y}$ that is larger than $d_{\min}/2$ for high-rate codes. Shortly after, Guruswami and Sudan further improved the algorithm [GS99], enlarging the decoding radius for any RS code up to the Johnson bound, which delimits the range where the list of solutions is polynomially bounded.

More recently, alternative decoding approaches have been proposed. Among them, we mention *power decoding* [Nie14; Nie18; SSB10], whose key equations will be studied, from an algebraic point of view, in this thesis.

Code-based cryptography and Goppa codes

One of the main challenges cryptography deals with is to design techniques to keep messages and data secure from adversaries, enabling only the sender and the intended receiver to read them. Modern cryptography makes extensive use of mathematics and theoretical computer science. In particular, cryptographic algorithms heavily rely on computational hardness assumptions, i.e. on the hypotheses that specific problems can not be solved efficiently (where "efficiently" usually means "in polynomial time"). In other words, cryptographic schemes are designed in such a way that it is provably or reasonably impossible to break them without solving a hard computational problem. Examples of these problems are the integer factorization or the discrete logarithm, both widely adopted in cryptography.

At the beginning of this introduction, we mentioned the provable difficulty of decoding a random code. This has not escaped the attention of researchers. The first scheme based on the hardness of a problem borrowed from coding theory, i.e. belonging to **code-based cryptography**, is the *McEliece cryptosystem* [McE78], which dates back to 1978, after only few months from tehe publication of the RSA encryption scheme [RSA78]. With its approximately 45 years of history, this is also one of the forerunners of public-key cryptography. Systems that belong to this category are featured by two keys, a private and a public one. Anyone with the latter is capable of encrypting a message but only those who know the private key can decrypt the ciphertext.

Despite an initial mild reception of the McEliece cryptosystem from the academic community, in favor of other public-key schemes, this has more recently experienced renewed interest. This is partially due to the emergence of quantum information. Indeed, there exist quantum algorithms that can solve exponentially faster some computational problems employed in cryptography. For instance, Shor's algorithm can factor large integers and compute the discrete logarithm in polynomial time. Therefore, in the near future, quantum computer could be capable of breaking almost all the schemes used nowadays in practice. McEliece cryptosystem, and more in general the whole field of *code-based cryptography*, is unanimously believed not to be vulnerable to quantum attacks, though. In other words, it is considered a *quantum-resistant* alternative and is a candidate of *post-quantum cryptography*. The U.S. National Institute of Standards and Technology (NIST) launched a process to evaluate and standardize quantum-resistant public-key cryptosystems in 2016. The drawback of the McEliece scheme is the huge key size, compared to other cryptographic algorithms, which restricts the range of possible applications. On the other hand, it has extremely fast encoding and decoding procedures and survived a long history of cryptanalysis. For these reasons, Classic McEliece [Alb+20] has been admitted to the fourth round of NIST competition [Ala+22] and is believed to be ready for standardization.

We have said that code-based cryptography relies on hard problems from coding theory, like the decoding problem. However, in constructions like the McEliece scheme, the legitimate receiver must be able to decode the sent ciphertext. It is not known whether it is possible to achieve this, even with the private key, if the code used is random. In other words, the code must be chosen from a family equipped with an efficient decoding algorithm. This is then masked in such a way that, without knowing the private key, it is not possible to apply the decoder. The original proposal

from McEliece suggests the use of **binary Goppa codes**, a subclass of **alternant codes**. The latter are the restriction to a subfield of generalized Reed-Solomon (or more briefly GRS) codes, an extension of RS codes. Several other families have been recommended and studied, over the years, like GRS codes themselves. Most of these McEliece-like variants have then been successfully attacked, though. Instead, Goppa codes resisted cryptanalysis up to now, because the best attack known so far are generic algorithms for decoding random codes and thus have exponential complexity.

Anyway, we should still emphasize that the known security proof for the McEliece cryptosystem rely on two assumptions. The first one, as already explained, is that decoding a generic linear code is hard. The second is that Goppa codes behave like random codes. To be more accurate, we say that the *Goppa distinguishing problem*, which asks to discriminate whether a set of vectors is the basis of a Goppa code or a random code, is computationally intractable. Goppa codes actually behave like random codes under many aspects and for many years the belief in this assumption was robust. However, approximately 10 years ago, a polynomial-time **distinguisher**, i.e. an algorithm that solves the distinguishing problem, was presented [Fau+13]. The context where it is effective, however, it is limited: it only works for a Goppa code, and more in general an alternant code, whose rate is enough high. In particular, it does not apply to Classic McEliece, but it still raises some concerns about the plausibility of the distinguishing hardness assumption. Moreover, it invalidates the security proof of other schemes, such as the digital signature CFS [CFS01], which is indeed built upon high-rate Goppa codes. In any case, the authors of [Fau+13] left as open problems to further mitigate the constraints of the distinguisher and to turn it into an attack. In this manuscript, we will approach the distinguisher from several points of view, give a better insight into it and try to tackle both the mentioned challenges.

Gröbner bases

Differently from a linear system, solving a multivariate polynomial system is generally difficult. The main tools that come in handy for this purpose are Gröbner bases, introduced by Buchberger in 1965 [Buc65], together with a simple algorithm to compute them. Later, other more advanced algorithms were discovered (for instance [Fau99; Fau02; Fau+93]) and still Gröbner bases represent a thriving field of research. Given a system of polynomials $f_1, \dots, f_m$ defining a system of equations, Gröbner bases algorithms recursively produce new polynomials belonging to the ideal generated by the initial polynomials $\mathcal{I} = \langle f_1, \dots f_n \rangle$. The key idea is that they generalize the polynomial division to the multivariate non-linear case. In particular, they allow to reduce a polynomial with respect to a set of polynomials in such a way that the remainder does not depend on the order in which the elements of the set are processed. A Gröbner basis is indeed a generating set of a polynomial ideal where all the complete reductions of a polynomial by the basis produce the same result. This can depend on the *monomial order* associated to the polynomial ring.

When the system is affine, some polynomial combinations of the polynomials generated up to some point by a Gröbner basis algorithm lead to low-degree polynomials. Whenever this happens, we say that one or more *degree falls* occurred. Degree falls are often critical in the resolution of a system. Indeed, because of their unusually low degree, they can trigger a chain of other degree falls after being

multiplied by other monomials/polynomials, which ultimately provide a Gröbner basis. From this, it is possible to derive the variety $\boldsymbol{V}(\mathcal{I})$ corresponding to the ideal $\mathcal{I}$, whose elements are the solutions of our multivariate system. Gröbner basis algorithms have exponential complexity when applied to the great majority of ideals. Nevertheless, solving some specific systems can be achieved in practical time (sometimes even polynomial time in the parameters), by exploiting their special structure.

In this manuscript, Gröbner bases serve as a technical apparatus for addressing several problems stemming from algebraic coding theory and cryptography. More precisely, the decoding problem of RS codes can be modeled as solving a multivariate system, whose solution directly provides the encoded message without positions in error. Similarly, the problem of recovering the private key in the McEliece scheme based on alternant/Goppa codes can be expressed by a polynomial system, whose solution is the private key or an equivalent one. Moreover, for high-rate alternant codes, the system becomes easier to solve. In a cryptographic setting, this kind of analysis takes the name of **algebraic cryptanalysis** and has become in recent years one of the leading tools for inspecting the security of code-based schemes, multivariate schemes as well as some symmetric ciphers.

Contributions

We now briefly list the contributions of this thesis, chapter by chapter.

Chapter 2.

We study a well-known algebraic system that models the decoding problem referred to RS (or equivalently GRS) codes using Gröbner basis methods. The system is bilinear, i.e. it is linear with respect to two blocks in which variables are split. A rich literature has been developed for generic bilinear systems, but computing a Gröbner basis should still have exponential complexity for the targeted parameters. We prove that this is not the case for Reed-Solomon decoding modeling, where even a simplified Gröbner basis algorithm runs in polynomial time up to the same decoding radius reached by Sudan's algorithm. We show that the new polynomials obtained from the Gröbner basis computation are strictly related to power decoding [Nie14; Nie18]. This is an alternative approach with respect to the list decoders cited before. Its improved version attains the same decoding radius as the Guruswami-Sudan algorithm, i.e. Johnson's radius. This suggests that our approach could work even beyond Sudan's bound. In this regard, we experimentally show that, for some parameters, our method can correct a number of errors up to and even slightly beyond Johnson's bound. By doing so, we also derive new polynomial identities in only one block of variables that are not exploited by the power decoding strategy.

Related publication: Magali Bardet, Rocco Mora and Jean-Pierre Tillich, *Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach*, International Symposium in Information Theory 2021 [BMT21].

Chapter 3.

The high-rate distinguisher for alternant and Goppa codes was originally presented in [Fau+13] as the unusually small rank of a matrix built from a basis of the alternant/Goppa code. To be precise, neither upper nor lower bounds for this rank were given, but only algebraic explanations based on heuristics. Thanks to the link given in [MP12], the distinguisher can be equivalently studied in terms of the dimension of the *square code* of the dual of the alternant/Goppa code. The square of a code is a coding theory construction that has already been successfully used to distinguish and/or attack other code-based schemes, for instance, variants of the McEliece scheme based on GRS codes [Cou+14]. In our case, this alternative point of view permits to prove an upper bound for the target square code dimension, thus making the distinguisher more rigorous. Moreover, our proof covers the case of non-binary Goppa codes, for which [Fau+13] only provided empirical evidence. The upper bounds are tight for all parameters and match experimental results. In order to prove the upper bounds, we also find new results concerning the structure of the product and square of subfield subcodes in general and of alternant/Goppa codes in particular. Hence, this chapter is ambivalent: it can be framed as a contribution to algebraic coding theory but, given the well-known interest of this family of codes in cryptography, it can be of interest in the latter as well.

Related publication: Rocco Mora and Jean-Pierre Tillich, *On the dimension and structure of the square of the dual of a Goppa code*, Designs, Codes and Cryptography [MT22].

Chapter 4.

Sometimes distinguisher can be turned into an attack. In code-based cryptography, this was the case of GRS codes [Cou+14]. Whether the distinguisher from [Fau+13] could have been exploited to break high-rate alternant/Goppa code instances of the McEliece scheme was not clear. Several attacks have been found on variants related to Goppa codes. Among them, we mention quasi-cyclic and quasi-dyadic Goppa codes [Fau+10b; GL09] or Wild Goppa codes [COT14a; FPP14]. Algebraic cryptanalysis played a key role in some of these examples. This is because the additional structure characterizing the underlying code allowed to reduce significantly the number of variables in the multivariate system modeling the key-recovery problem. However, no attacks were known for non-structured alternant/Goppa codes, even at a high rate. In this chapter, we mount a polynomial-time attack that works against binary or ternary random alternant codes. The algorithm consists of two parts. First, a *filtration* of alternant codes of decreasing order, i.e. a sequence of alternant codes each one contained in the following, is computed iteratively. Once an alternant code of order 3 is produced, the known algebraic modeling is set up. In the second part of the attack, we provide an efficient Gröbner basis algorithm that is adapted to this specific system. From the variety associated with the ideal generated by the Gröbner basis, a key equivalent to the private one is recovered. The theoretical explanation is complemented by some code implemented in MAGMA. Quite surprisingly, the attack does not work on Goppa codes, even though they form a subclass of alternant codes. We also give insight into the issues that prevent a direct adaptation of the result to Goppa codes.

Related publication: Magali Bardet, Rocco Mora and Jean-Pierre Tillich, *Polynomial time key-recovery attack on high rate random alternant codes*, Preprint [BMT23].

Chapter 5.

We present a method that enhance the distinguisher for high-rate alternant and Goppa codes. Our strategy is consistent with the presentation given in Chapter 3, as it also exploits the square code construction. However, this is improved by first shortening the dual code and then computing the square. We empirically illustrate that this tweak decreases the minimum distinguishable code rate for some parameters. In particular, from our experiments this strategy seems to be more effective for small extension degrees and large field sizes, and random alternant codes are more impacted by this approach than Goppa codes are. The empirical results are complemented by a partial algebraic explanation of the arising square code dimension.

Publications

[BMT21] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. *Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach.* In: *2021 IEEE International Symposium on Information Theory (ISIT).* Melbourne, Australia, July 2021, pp. 872–877.

[MT22] Rocco Mora and Jean-Pierre Tillich. *On the dimension and structure of the square of the dual of a Goppa code.* In: *Designs, Codes and Cryptography* (2022), pp. 1–22.

[BMT23] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. *Polynomial time key-recovery attack on high rate random alternant codes.* In: *preprint* (2023).

Notation

We gather here as a reminder some non-standard notation regarding codes and vectors that is encountered in the manuscript.

Schur's product and square of vectors and codes. We sometime imply the symbol $\star$ in the product of vectors (see Definition 1.18), in order to ease the readability. More precisely, we can denote with $\boldsymbol{cd}$ the product $\boldsymbol{c} \star \boldsymbol{d}$. In the same spirit, we sometimes denote the component-wise division of two vectors with $\frac{\boldsymbol{c}}{\boldsymbol{d}}$, the component-wise evaluation of a polynomial $P : \mathbb{F} \to \mathbb{K}$ in $\boldsymbol{c}$ with $P(\boldsymbol{c})$ and the component-wise product $\boldsymbol{c}^{*a}$ of $\boldsymbol{c}$ with itself $a-1$ times with $\boldsymbol{c}^a$. Regarding codes, Schur's power of codes $\mathscr{C}^{*a}$ should not be confused with $\mathscr{C}^{q^l}$, which is the set (actually the code) whose elements are $\boldsymbol{c}^{q^l}$'s such that $\boldsymbol{c} \in \mathscr{C} \subseteq \mathbb{F}_{q^m}$.

Subfield subcodes and extension of a code over a field extension. The subfield subcode over $\mathbb{F}_q$ of a code $\mathscr{C} \subseteq \mathbb{F}_{q^m}^n$ is denoted with $\mathscr{C}_{|\mathbb{F}_q}$. On the other hand, we use the notation $\mathscr{C}_{\mathbb{F}_{q^m}}$ to write the $\mathbb{F}_{q^m}$-linear span of a code $\mathscr{C} \subseteq \mathbb{F}_q^n$ in $\mathbb{F}_{q^m}^n$.

Shortening of vectors. If $\boldsymbol{x} = (x_i)_{i \in [\![1,n]\!]}$ and $\mathcal{I}$ is a subset of positions, we denote by $\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}}$ the vector $\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}} \overset{\text{def}}{=} (x_i)_{i \in [\![1,n]\!] \setminus \mathcal{I}}$. In particular, we do not contract the indexes but we still associate the original index to each remaining coordinate. When there is just one position i in $\mathcal{I}$ we simply write $\boldsymbol{x}_{\overset{\vee}{i}}$.

Reduction of a vector. With $\boldsymbol{c} \overset{\mathscr{C}}{\to} \boldsymbol{d}$ we mean that $\boldsymbol{d}$ can be obtained from $\boldsymbol{c}$ by adding a suitable element of $\mathscr{C} \subseteq \mathbb{F}^n$ and multiplying by some element in $\mathbb{F}$, i.e. this is equivalent to $\boldsymbol{c} - \lambda \boldsymbol{d} \in \mathscr{C}$ for a suitable element λ in $\mathbb{F}$.

Chapter 1
Preliminaries

In this preliminary chapter, we review all the fundamental concepts that will be needed for understanding the results contained in this manuscript. The material included within this chapter is part of a well-established theory, more or less recent, from several domains and the greater emphasis put on certain topics naturally reflects the focus of this thesis. In any case, this treatment does not contain original contributions, to whom all the next following part of the manuscript is dedicated, instead. The current chapter has a tripartite structure. Indeed, in accordance with the topics presented in the introduction, we will provide basics about algebraic coding theory, code-based cryptography and Gröbner bases. In particular, we will start by recalling definitions and notions in coding theory, with particular attention on GRS codes and related families, as well as on some standard constructions of codes. Then we will clarify the general aim of public key and post-quantum cryptography, before moving to schemes based on codes, from both the viewpoints of cryptology and cryptanalysis. Finally, we will see how Gröbner basis techniques can be deployed in the context of solving polynomial systems, with an accent on some applications from coding theory and cryptography.

Contents

1.1 Algebraic Coding theory

With the Third Industrial Revolution several technologies arose in the context of digital communications, for instance the possibility of transmitting information (in the shape of bits) through a channel or store it on some digital support. Both these procedures require methods to protect the information from errors. The latter might indeed occur either during the transmission over a noisy channel or because of the natural deterioration of physical supports.

To this end, **error-correcting codes** come to the rescue. We will give the proper definition of these objects further, but informally the idea consists in adding redundant bits to each piece of information we want to send or store, so that if some bits are altered, the other ones allow to *detect* (or even better *correct*) the errors. We can let the birth of *coding theory* (and more in general *information theory*) match with the seminal work of Shannon in 1948 [Sha48].

Probably the easiest example one may think of to perform the task mentioned above is through a *parity bit*. Imagine we want to transmit a string of bits of fixed length. Instead, we send the string to which we append an additional bit, whose value equals the sum (mod 2) of the values of the original string bits. Then the sum (mod 2) of all bits for any string crafted in this way will always be 0. Therefore if a bit was altered during the transmission, we will be able to detect it, as the sum of the string bits becomes 1. However, this stratagem does not find which bit was modified and therefore can not be used to correct the error.

More efficient techniques to correct errors require more advanced mathematical tools, especially when dealing with algebraic linear codes, which are central in this manuscript. Hence we are now going to introduce the necessary vocabulary.

1.1.1 Error-correcting codes

Let $\mathcal{A}$ an **alphabet** of size q, i.e. a set of q distinct symbols. This alphabet is used to code information. We denote with $\mathcal{A}^n$ the set of n-tuples with entries in $\mathcal{A}$. A non-empty subset $\mathscr{C}$ of $\mathcal{A}^n$ is a **code**, the cornerstone of this section. If $q = 2$ we call it a *binary code*, if $q = 3$ a *ternary code* etc. If $|\mathscr{C}| = 1$ we say that $\mathscr{C}$ is a *trivial code*.

Definition 1.1 (Hamming distance/Hamming weight)**.** Let $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{A}^n$. The **Hamming distance** $d(\boldsymbol{x}, \boldsymbol{y})$ between $\boldsymbol{x}$ and $\boldsymbol{y}$ is defined as

$$d(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} |\{i \in [\![1, n]\!] \mid x_i \neq y_i\}|.$$

The **Hamming weight** $\mathrm{wt}(\boldsymbol{x})$ of $\boldsymbol{x}$ is defined as

$$\mathrm{wt}(\boldsymbol{x}) \stackrel{\text{def}}{=} d(\boldsymbol{x}, \boldsymbol{0}).$$

It is easy to check that the function of Definition 1.1 verifies all the conditions for being a metric. This is the most common metric used in coding theory, because it represents a good measure for the error resulting from the transmission of a message through a noisy channel. However, there exist other metrics for specific kind of codes (and alphabets) such as the *Lee metric* or the *rank metric*. Afterwards, we will give more details about the latter.

We might be interested in minimizing distance and weight over a subset of $\mathcal{A}^n$.

Definition 1.2 (Minimum distance/minimum weight)**.** Given a metric d and the corresponding weight function wt, the **minimum distance** of a non-trivial code $\mathscr{C}$ is

$$d(\mathscr{C}) \stackrel{\text{def}}{=} \min\{d(\boldsymbol{c}, \boldsymbol{d}) \mid \boldsymbol{c}, \boldsymbol{d} \in \mathscr{C}, \boldsymbol{c} \neq \boldsymbol{d}\}.$$

The **minimum weight** of $\mathscr{C}$ is

$$\mathrm{wt}\,(\mathscr{C}) \stackrel{\text{def}}{=} \min\{\mathrm{wt}\,(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathscr{C}, \boldsymbol{c} \neq \boldsymbol{0}\}.$$

A code $\mathscr{C} \subseteq \mathcal{A}^n$ with M elements and minimum distance d is called an (n, M, d) code, or (n, M) code without specifying the distance. Given a nonnegative integer r and a vector $\boldsymbol{x} \in \mathcal{A}^n$, we define the **ball of radius** r *centered in* $\boldsymbol{x}$ as

$$B_r(\boldsymbol{x}) \stackrel{\text{def}}{=} \{\boldsymbol{y} \in \mathcal{A}^n \mid d(\boldsymbol{x}, \boldsymbol{y}) \leq r\}.$$

For the metrics considered here, the **volume** of a sphere does not depend on its center. Thus we can define the volume of any sphere of radius r in $\mathcal{A}^n$, where $|\mathcal{A}| = q$ as a function in r, n and q, and we denote it with $V_q(n, r)$. A straightforward computation shows that, for the Hamming metric,

$$V_q(n, r) = \begin{cases} |B_r(\boldsymbol{0})| = \sum_{i=0}^{r} \binom{n}{i}(q-1)^i, & r \leq n \\ q^n, & r > n. \end{cases}$$

The next quantity describes how much information a code is carrying.

Definition 1.3 (Code rate)**.** Let $|\mathcal{A}| = q$ and $\mathscr{C} \subset \mathcal{A}^n$ be a code. The *information rate* R of $\mathscr{C}$ is defined as

$$R \stackrel{\text{def}}{=} \frac{\log_q |\mathscr{C}|}{n}.$$

We now want to construct codes with some algebraic structure. We then proceed to define *linear codes*. The alphabet $\mathcal{A}$ considered for these codes is a finite field $\mathbb{F}_q$.

Definition 1.4 (Linear code)**.** Let q be a prime power, k a nonnegative integer and $n \geq k$ a positive integer. A **q-ary linear code** $\mathscr{C}$ is a linear subspace of $\mathbb{F}_q^n$ of **dimension** k and is called an $[n, k]$ code. if d is the code distance, then $\mathscr{C}$ is also called an $[n, k, d]$ linear code.

From now on, whenever writing $\mathscr{C} \subset \mathbb{F}_q^n$, we will imply that $\mathscr{C}$ is a q-ary code, unless otherwise stated.

We can now revisit the previous definitions in the case of a linear code. The cardinality of a q-ary $[n, k, d]$ linear code $\mathscr{C}$ is q^k. Therefore $\mathscr{C}$ is a (n, q^k, d) code and its information rate is $R = \frac{k}{n}$. Moreover

Proposition 1.1. *For a linear code $\mathscr{C}$, minimum distance and minimum weight coincide.*

Proof. The thesis follows because vector spaces are closed under addition and each element has inverse:

$$d(\mathscr{C}) = \min_{\boldsymbol{c}, \boldsymbol{d} \in \mathscr{C}} d(\boldsymbol{c}, \boldsymbol{d}) = \min_{\boldsymbol{c}, \boldsymbol{d} \in \mathscr{C}} d(\boldsymbol{c} - \boldsymbol{d}, \boldsymbol{0}) = \min_{\boldsymbol{c}, \boldsymbol{d} \in \mathscr{C}} w(\boldsymbol{c} - \boldsymbol{d}) = \min_{\boldsymbol{x} \in \mathscr{C}} w(\boldsymbol{x}) = \mathrm{wt}\,(\mathscr{C}).$$

□

Linear subspaces can be succinctly described by their basis. The *generator matrix* of a code embodies this feature.

Definition 1.5 (Generator matrix)**.** Let $\mathscr{C}$ be an $[n, k]$ linear code. A **generator matrix** $\boldsymbol{G}$ of $\mathscr{C}$ is a $k \times n$ matrix whose rows are a basis of $\mathscr{C}$.

If $\boldsymbol{G}$ is a generator matrix for $\mathscr{C} \subset \mathbb{F}_q^n$, then $\mathscr{C} = \{\boldsymbol{x}\boldsymbol{G} \mid \boldsymbol{x} \in \mathbb{F}_q^n\}$. A generator matrix is evidently not unique. However there is a natural form for it. We say that $\boldsymbol{G}$ is in **systematic form** if $\boldsymbol{G} = [\boldsymbol{I}_k \mid \boldsymbol{P}]$, where $\boldsymbol{I}_k$ is the $k \times k$ identity matrix and $\boldsymbol{P}$ a $k \times (n-k)$ matrix. If it is possible to row-reduce $\boldsymbol{G}$ in this way, then the first k positions are called **information symbols** and $[\![1, k]\!]$ is an **information set**. This is not always the case, as the column submatrix of $\boldsymbol{G}$ corresponding to the first k positions might not be full rank. However, by linear algebra, there must exist a subset S of $[\![1, n]\!]$ of cardinality k such that S is an information set. In other words, any linear code is *equivalent* to a code which admits a generator matrix in systematic form.

Definition 1.6 (Equivalent codes)**.** Let $\mathscr{C}_1, \mathscr{C}_2 \in \mathbb{F}_q^n$ be two $[n, k]$ linear codes. We say that $\mathscr{C}_1$ and $\mathscr{C}_2$ are **equivalent codes**, and write $\mathscr{C}_1 \sim \mathscr{C}_2$, if there exists a permutation σ over $[\![1, n]\!]$ such that

$$\sigma(\mathscr{C}_1) \stackrel{\text{def}}{=} \{(\boldsymbol{c}_{\sigma^{-1}(i)})_{i \in [\![1,n]\!]} \mid \boldsymbol{c} \in \mathscr{C}\}$$

is equal to $\mathscr{C}_2$.

Definition 1.7 (Dual code)**.** Let $\mathscr{C} \subset \mathbb{F}_q^n$ be a linear code. Its **dual code** $\mathscr{C}^\perp$ is defined as

$$\mathscr{C}^\perp \stackrel{\text{def}}{=} \{\boldsymbol{d} \in \mathbb{F}_q^n \mid \forall \boldsymbol{c} \in \mathscr{C},\ \langle \boldsymbol{c}, \boldsymbol{d} \rangle = 0\},$$

where $\langle \boldsymbol{c}, \boldsymbol{d} \rangle$ is the inner product of $\boldsymbol{c}$ and $\boldsymbol{d}$.

The notion of dual code of $\mathscr{C}$ must not be confused with the one of dual vector space, i.e. the space of linear forms on $\mathscr{C}$. The dual code $\mathscr{C}^\perp$ is obviously a linear code. In particular, if $\mathscr{C}$ is an $[n, k]$ code, then $\mathscr{C}^\perp$ is an $[n, n-k]$ code. We also remark that $\mathscr{C}^\perp$ is not the orthogonal complement in the sense of vector spaces over $\mathbb{R}$. Due to the fact that the field characteristic is different from 0, it is possible that $\mathscr{C} \cap \mathscr{C}^\perp \neq \{\boldsymbol{0}\}$ and it may even occur that $\mathscr{C} = \mathscr{C}^\perp$. A code $\mathscr{C}$ with the latter property is called *self-dual*.

Definition 1.8 (Parity-check matrix)**.** Let $\mathscr{C}$ be an $[n, k]$ linear code. A **parity-check matrix** $\boldsymbol{H}$ of $\mathscr{C}$ is an $(n-k) \times n$ matrix whose rows are a basis of $\mathscr{C}^\perp$.

It readily follows from the definition of parity-check matrix and linearity that

$$\boldsymbol{c} \in \mathscr{C} \iff \boldsymbol{c}\boldsymbol{H}^T = 0.$$

This also implies that

$$\boldsymbol{G}\boldsymbol{H}^T = 0. \tag{1.1}$$

We will see how Equation (1.1) allows to set up an algebraic model when the code generators are vectors of polynomial evaluations.

Moreover, if $\boldsymbol{G} = [\boldsymbol{I}_k \mid \boldsymbol{P}]$ is the generator matrix of an $[n, k]$ systematic linear code $\mathscr{C}$, then the corresponding parity-check matrix is $\boldsymbol{H} = [-\boldsymbol{P}^T \mid \boldsymbol{I}_{n-k}]$. Indeed the parity-check matrix defined in this way satisfies Equation (1.1) and the number of rows (which are clearly linearly independent) equals the dimension of $\mathscr{C}^\perp$.

Definition 1.9 (Syndrome)**.** Let $\mathscr{C} \subset \mathbb{F}_q^n$ be a linear code with parity-check matrix $\boldsymbol{H}$. For any $\boldsymbol{x} \in \mathbb{F}_q^n$, $\boldsymbol{x}\boldsymbol{H}^T$ is called the **syndrome** of $\boldsymbol{x}$.

Looking at $\mathscr{C}$ as a subgroup of $\mathbb{F}_q^n$, the latter can be decomposed into $[\mathbb{F}_q^n : \mathscr{C}] = q^{n-k}$ cosets. The notion of syndrome is involved in the definition of cosets. In particular, $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the same coset if they have the same syndrome, i.e.

$$\boldsymbol{x} - \boldsymbol{y} \in \mathscr{C} \iff \boldsymbol{x}\boldsymbol{H}^T = \boldsymbol{y}\boldsymbol{H}^T.$$

We can identify a natural representative for each coset as the vector $\boldsymbol{e}$ with minimal weight in that coset, and we call it *coset leader*. We remark that the coset leader is not necessarily unique. To explain its usefulness, suppose we want to decode a received word $\boldsymbol{y}$ with respect to a code $\mathscr{C} \in \mathbb{F}_q^n$. Let us make two assumptions. The first one is that the errors occur with the same probability p for any bit and that they are independent events. This is the case for transmissions over a *q-ary symmetric channel*, a common communication model in information theory. Then we assume that any codeword $\boldsymbol{c}$ has the same probability $1/|\mathscr{C}|$ of being sent. The goal is to perform **maximum likelihood (ML) decoding**, i.e. finding the codeword $\boldsymbol{c} \in \mathscr{C}$ which maximizes $\mathbb{P}(\boldsymbol{c} \text{ sent} \mid \boldsymbol{y} \text{ received})$. If $d(\boldsymbol{c}, \boldsymbol{y}) = d$, with our assumptions we have

$$\mathbb{P}(\boldsymbol{c} \text{ sent} \mid \boldsymbol{y} \text{ received}) = (1-p)^{n-d} \left(\frac{p}{q-1}\right)^d.$$

If we also take $p < \frac{q-1}{q}$, maximum likelihood decoding coincides with **minimum distance decoding**, which aims at finding a codeword $\boldsymbol{c} \in \mathscr{C}$ with minimum distance from $\boldsymbol{y}$. Then $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, where $\boldsymbol{e}$ is (one of) the coset leader(s) of the same coset of $\boldsymbol{y}$. Since

$$\boldsymbol{y}\boldsymbol{H}^T = (\boldsymbol{c} + \boldsymbol{e})\boldsymbol{H}^T = \boldsymbol{c}\boldsymbol{H}^T + \boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{e}\boldsymbol{H}^T,$$

one can find $\boldsymbol{c} = \boldsymbol{y} - \boldsymbol{e}$ looking at a precomputed table of size q^{n-k}, mapping $\boldsymbol{e}\boldsymbol{H}^T$ to $\boldsymbol{e}$. Minimum distance decoding using a lookup table is also known as **syndrome decoding**.

We conclude this quick introduction to linear codes with some classical procedures to contruct new codes from existing ones.

Definition 1.10. Given a code $\mathscr{C} \subseteq \mathbb{F}^n$ and a subset $\mathcal{I} \subseteq [\![1, n]\!]$, the **punctured** code $\mathbf{Pct}_{\mathcal{I}}(\mathscr{C})$ and the **shortened** code $\mathbf{Sh}_{\mathcal{I}}(\mathscr{C})$ over $\mathcal{I}$ are defined respectively as

$$\mathbf{Pct}_{\mathcal{I}}(\mathscr{C}) = \left\{(c_i)_{i \in [\![1,n]\!] \setminus \mathcal{I}} \mid \boldsymbol{c} \in \mathscr{C}\right\},$$
$$\mathbf{Sh}_{\mathcal{I}}(\mathscr{C}) = \left\{(c_i)_{i \in [\![1,n]\!] \setminus \mathcal{I}} \mid \exists\, \boldsymbol{c} = (c_i)_{i \in [\![1,n]\!]} \in \mathscr{C} \text{ s.t. } \forall\, i \in \mathcal{I}, c_i = 0\right\}.$$

For the sake of simplicity, when $\mathcal{I} = \{i\}$, we denote the punctured and the shortened codes in $\mathcal{I}$ with $\mathbf{Pct}_i(\mathscr{C})$ and $\mathbf{Sh}_i(\mathscr{C})$ respectively.

Shortening and puncturing combine with the dual operator in a reciprocal way:

Proposition 1.2 ([HP03, Theorem 1.5.7])**.** *Let $\mathscr{C}$ be a linear code of length n and $\mathcal{I} \subset [\![1, n]\!]$. Then*

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{C}^{\perp}\right) = \mathbf{Pct}_{\mathcal{I}}\left(\mathscr{C}\right)^{\perp} \quad \textit{and} \quad \mathbf{Pct}_{\mathcal{I}}\left(\mathscr{C}^{\perp}\right) = \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{C}\right)^{\perp}.$$

1.1.2 Bounds on codes

Given a q-ary (n, M, d) code, the size M gives an indication (depending on the length n) of the code efficiency, while d express the *error-correction capability*. We could list the following desirable properties for an (n, M, d)-code:

1. large M: many messages can be encoded;
2. large d: many errors can be corrected;
3. small n: the transmission is fast.

It is therefore clear that, for a fixed length n, values of both M and d as large as possible are highly desirable for error-correction purposes. Finding good codes from this point of view is an hard task even regardless of equipped encoding/decoding algorithms and this represents a wide research area in coding theory. Unfortunately there are some inviolable limits, as M and d can not be arbitrarily large at the same time. In other words, a compromise between the transmission rate and the relative error-correction can not be avoided. In this section we will review some classic upper bounds as well as some lower bounds.

Definition 1.11. Let $\mathcal{A}$ be an alphabet of size q. Given n and d, we denote with $A_q(n, d)$ the largest possible size M for which there exists an (n, M, d)-code over $\mathcal{A}$. In other words,

$$A_q(n, d) \stackrel{\text{def}}{=} \max\{M \mid \exists (n, M, d)\text{-code over } \mathcal{A}\}.$$

Any $(n, A_q(n, d), d)$-code over $\mathcal{A}$ is called an *optimal code*.

Note that the value $A_q(n, d)$ depends only on the size of $\mathcal{A}$ but not on $\mathcal{A}$ itself. Determining $A_q(n, d)$ is a difficult challenge. As a matter of fact, this problem is known as *main coding theory problem*.

We have an analogous definition when we restrict to linear codes.

Definition 1.12. Let q be a prime power. Given n and d, we denote with $B_q(n, d)$ the largest possible size q^k for which there exists an $[n, k, d]$-code over $\mathbb{F}_q$. In other words,

$$B_q(n, d) \stackrel{\text{def}}{=} \max\{q^k \mid \exists [n, k, d]\text{-code over } \mathbb{F}_q\}.$$

We quote some extremal (in)equalities for these numbers, without proof.

Proposition 1.3 (Ling Xing Theorem 5.1.7)**.** *Let $q \geq 2$ be a prime power. Then*

1. $\forall d \in [\![1, n]\!],\ B_q(n, d) \leq A_q(n, d) \leq q^n$;
2. $B_q(n, 1) = A_q(n, 1) = q^n$;

3. $B_q(n,n) = A_q(n,n) = q$*;*

A list of upper and lower bounds (and sometimes exact values) of $A_q(n,d)$ for several q, n and d is maintained in [Gra07]. Before moving to the actual bounds, let us briefly and informally recall *Shannon's theorem* (or *Shannon's limit*), which addresses the question of the limit of error correction. In particular, it wonders what is the optimal error correction for a given noisy channel. Let H_q be the *entropy function*

$$H_q(x) \stackrel{\text{def}}{=} \begin{cases} 0, & x = 0 \\ x\log_q(q-1) - x\log_q(x) - (1-x)\log_q(1-x), & 0 < x \leq (q-1)/q. \end{cases}$$

Given a channel which transmits symbols of a q-ary alphabet and alters each symbol with independent probablity p, Shannon's limit asserts that for any $R < 1 - H_q(p)$ there always exists a code $\mathscr{C}$ (not necessarily linear) whose rate is R and such that the probability of a wrong decoding is exponentially small. On the other hand, if $R > 1 - H_q(p)$ the decoding failure will always be lower bounded by some strictly positive constant.

1.1.2.1 Gilbert-Varshamov bound

We start with a lower-bound, called the **Gilbert-Varshamov bound**.

Theorem 1.1 (Gilbert-Varshamov bound)**.** *Let* $n, d \in \mathbb{N}$*,* $d \leq n$*. Then*

$$A_q(n,d) \geq \frac{q^n}{V_q(n,d-1)} = \frac{q^n}{\sum_{i=0}^{d-1}\binom{n}{i}(q-1)^i}.$$

Proof. Let $\mathscr{C}$ be an (n, M, d) maximal code. Hence for any word $\boldsymbol{x}$ in $\mathcal{A}^n$, there exists at least a codeword in $\mathscr{C}$ whose distance from $\boldsymbol{x}$ is strictly less than d. In other words $\bigcup_{\boldsymbol{c}\in\mathscr{C}} B_{d-1}(\boldsymbol{c}) = \mathcal{A}^n$. By summing all the sphere volumes, we obtain that $|\mathscr{C}| \cdot V_q(n, d-1) \geq |\mathcal{A}^n| = q^n$. □

The proofs shows that it is possible to construct a code which attains Gilbert-Varshamov bound by starting from a codeword $\boldsymbol{c}$ and iteratively adding new codewords with distance at least d from all the previously picked codewords. Of course such a code is not necessarily linear. However, it turns out that this additional requirement does not represent a crucial restriction.

Theorem 1.2 (Gilbert-Varshamov bound for linear codes)**.** *Let* $n, k, d \in \mathbb{N}$*,* $k \leq n$*. If* $V_q(n, d-1) < q^{n-k+1}$*, then there exists a* q*-ary* $[n, k, d]$ *code.*

Gilbert-Varshamov bound has also an asymptotic counterpart. Given a q-ary code $\mathscr{C} \in \mathcal{A}^n$ with relative distance δ, we first define

$$\alpha(\delta) \stackrel{\text{def}}{=} \limsup_{n\to\infty} \frac{A_q(n, \delta n)}{n}.$$

Then

Theorem 1.3 (Asymptotic Gilbert-Varshamov bound)**.** *Let* $0 \leq \delta \leq (q-1)/q$*. Then*

$$\alpha(\delta) \geq 1 - H_q(\delta).$$

1.1.2.2 Upper bounds

The simplest upper bound is instead the so called Singleton bound, for which we present both a generic description and its specialization for linear codes.

Theorem 1.4 (Singleton bound)**.** *Let $q > 1$ an integer, and d, n two intgers such that $1 \leq d \leq n$. Then*

$$A_q(n, d) \leq q^{n-d+1}.$$

In particular, any $[n, k, d]$ linear code over $\mathbb{F}_q$ must satisfy

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

Proof. Let $\mathscr{C}$ be an (n, M, d)-code over $\mathcal{A}$, with $|\mathcal{A}| = q$, such that $M = A_q(n, d)$. We consider the code $\mathscr{C}' \subseteq \mathcal{A}^{n-d+1}$ of length $n - d + 1$ constructed by removing the last $d - 1$ coordinates from all the codewords of $\mathscr{C}$. Since the distance of $\mathscr{C}$ is d, all the codewords are still different, i.e. $\mathscr{C}'$ contains M codewords too. But then

$$A_q(n, d) = M = |\mathscr{C}'| \leq |\mathcal{A}|^{n-d+1} = q^{n-d+1}.$$

It readily follows that, if $\mathscr{C}$ is an $[n, k, d]$ linear code, then $q^k \leq q^{n-d+1}$, i.e. $k \leq n - d + 1$. □

An $[n, k, d]$ linear code which attains Singleton bound, i.e. for which $k = n-d+1$, is called **maximum distance separable** (MDS for short) code. A remarkable family of MDS codes is given by (generalized) Reed-Solomon codes.

Subtler arguments may lead to tighter upper bounds with respect to the Singleton bound. An example is given by the *Hamming bound*, also known as sphere packing bound. On the other hand additional hypotheses may be needed. We just mention the *Plotkin bound*, which is only applicable to a small range of values of d, namely when d is relatively large compared to n. The *Griesmer bound* instead is applicable to linear codes only. We postpone the discussion on another upper bound, called *Johnson's bound*, to the section about RS codes decoding.

Figure 1.1 compares some of the main bounds, in their asymptotic formulation. The colored area is enclosed within the Gilbert-Varshamov bound curve from below, and the Hamming bound and Plotkin bound curves from above. Finding codes which beat Gilbert-Varshamov bound, i.e. in the colored zone, is a big open problem. The common belief was that such codes do not even exist until the beginning of 80's. In other words, it was speculated that the Gilbert-Varshamov bound was asymptotically optimal. However, a family of codes with such parameters was found for the first time in 1982 by Tsfasman, Vladut and Zink [TVZ82], by making use of techniques borrowed from algebraic geometry. This happened after the discoveries of Goppa about constructing codes from algebraic curves [Gop71]. For more information regarding algebraic geometry codes we refer the reader to [TV13] or [CR21]. In [TVZ82], the existence of codes exceeding Gilbert-Varshamov bound was proven for any field $\mathbb{F}_q$, with $q \geq 49$ a square. However, for instance families of binary codes with the same feature are still not known and it is not even clear whether they could exist or not.

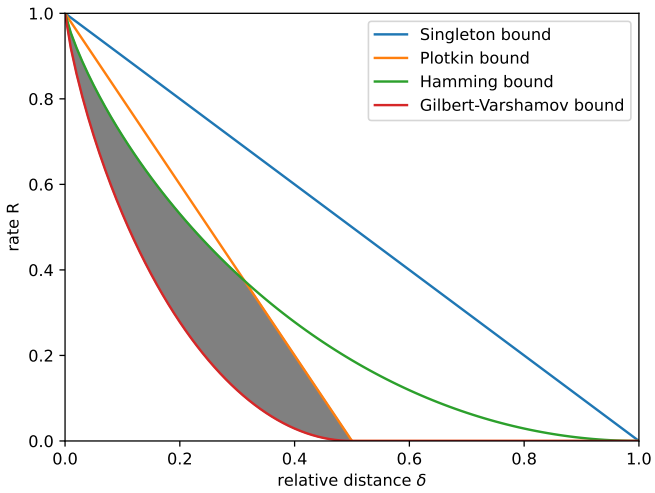


Figure 1.1: Comparison among some classical lower and upper bounds. There exist linear codes with parameters corresponding to the grey area.

1.1.3 Reed-Solomon codes

Reed-Solomon codes are a family of algebraic codes with excellent decoding capabilities and many other interesting features. It is no coincidence that these codes have been chosen for many real-life applications, sometimes in conjunction with other codes. A non exhaustive list of them includes satellite communications and space transmission (for instance in several NASA missions), bar code (QR codes and others) or data storage (such as CDs, DVDs or Blu-ray discs). They are named after Reed and Solomon, who discovered them in 1960 in their seminal work [RS60]. The attraction for these codes stems from the fact that they benefit from extremely efficient decoding algorithms.

We start by defining (generalized) Reed-Solomon codes and reviewing some classic results that shed light on the rich structure of this family and that will be useful in the next chapters. Then we will give an overview on some decoding algorithms that are specific for Reed-Solomon codes.

One of the possible ways to describe Reed-Solomon codes is to look at them as *evaluation code*. This is, incidentally, the original view from Reed and Solomon.

Definition 1.13 (Reed-Solomon code)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$ be a vector of pairwise distinct entries. The $[n, r]$ **Reed-Solomon (RS) code** with *support* $\boldsymbol{x}$ is

$$\mathbf{RS}_k(\boldsymbol{x}) \stackrel{\text{def}}{=} \{(P(x_1), \ldots, P(x_n)) \mid P \in \mathbb{F}[z], \deg P < k\}$$

Therefore, an RS code can be succinctly described by a vector, rather than a matrix. For a fixed $\boldsymbol{x}$, the RS code of prescribed dimension is indeed unique. However it is possible to obtain equivalent (but different) RS codes by permuting the vector $\boldsymbol{x}$ coordinates.

Especially for cryptograhic purposes, it will be worthwhile to consider a larger family of codes, which was instead introduced by Delsarte in [Del75]:

Definition 1.14 (Generalized Reed-Solomon code)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}^n$ be a vector of pairwise distinct entries and $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{F}^n$ a vector of nonzero entries. The $[n, k]$ **generalized Reed-Solomon (GRS) code** with *support* $\boldsymbol{x}$ and *multiplier* $\boldsymbol{y}$ is

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 P(x_1), \ldots, y_n P(x_n)) \mid P \in \mathbb{F}[z], \deg P < k\}$$

In the following, whenever two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ play the role of, respectively, support and multiplier for a (generalized) Reed-Solomon code, we will omit that they have the same length n and they respect the corresponding constraints.

Remark 1.1. An $[n, k]$ RS code is an $[n, k]$ GRS code where the multiplier is a non-zero constant vector, i.e. of the form $\bar{y} \cdot (1, \ldots, 1)$, $\bar{y} \neq 0$.

Remark 1.2. The maximal length for a (generalized) Reed-Solomon code is forced to be smaller than the field size, since the support coordinates must be different. Thus, long RS codes require a large field size. Furthermore, we say that the Reed-Solomon code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ is of *full-length* if $n = |\mathbb{F}|$, i.e. if the set of coordinates coincides with $\mathbb{F}$.

Remark 1.3. Since polynomials of bounded degree are generated by monomials up to that degree, we have

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \left\langle (y_1 x_1^i, \ldots, y_n x_n^i) \mid 0 \leq i < k \right\rangle_{\mathbb{F}}.$$

Therefore a generator matrix for $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ is given by the following rectangular Vandermonde-like matrix:

$$\boldsymbol{V}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{bmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \ddots & \vdots \\ y_1 x_1^{k-1} & \cdots & y_n x_n^{k-1} \end{bmatrix}. \tag{1.2}$$

As we have anticipated, GRS codes meet the Singleton bound:

Proposition 1.4 (p. 94 , [LX04])**.** *The code* $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ *is an* $[n, k, n - k + 1]$ *code. Hence it is an MDS code.*

Moreover, the dual of a GRS code is also a GRS code, where the support and the multiplier are related to the ones of the primal code. In order to explicit such relation we introduce the polynomial

$$\pi_{\boldsymbol{x}}(z) \stackrel{\text{def}}{=} \prod_{i=1}^{n} (z - x_i) \in \mathbb{F}[z].$$

Proposition 1.5 (Theorem 4, p. 304, [MS86])**.** *Let* $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ *be a GRS code of length* n*. Its dual is also a GRS code. In particular*

$$\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^{\perp} = \boldsymbol{GRS}_{n-k}(\boldsymbol{x}, \boldsymbol{y}^{\perp}),$$

where

$$\boldsymbol{y}^{\perp} \stackrel{\text{def}}{=} \left(\frac{1}{\pi'_{\boldsymbol{x}}(x_1) y_1}, \ldots, \frac{1}{\pi'_{\boldsymbol{x}}(x_n) y_n} \right)$$

and $\pi'_{\boldsymbol{x}}$ *is the derivative of* $\pi_{\boldsymbol{x}}$*.*

1.1.3.1 Decoding algorithms for RS codes

As already said, RS codes comes with very efficient decoding algorithms, which are able to exploit their strong algebraic structure. The Berlekamp-Welch algorithm [WB86] allows to uniquely decode whenever no more than half of the minimum distance errors occur. Such a threshold coincides with the error correction radius $\frac{1-R}{2}$, since Reed-Solomon codes are MDS codes. After this work, decoding Reed-Solomon codes beyond the error-correction radius has been a long-standing open problem in algebraic coding theory. This was eventually solved by Madhu Sudan in 1997 [Sud97] who discovered an algebraic decoder that works up to a fraction of errors $1 - \sqrt{2R}$ (which we will call the **Sudan radius** from now on). Shortly after, this result was even improved by Venkatesan Guruswami and Madhu Sudan in [GS98] with a decoder that works up to the Johnson radius $1 - \sqrt{R}$.

Remark 1.4. Any decoding algorithm for Reed-Solomon codes can be adapted to a decoder for *generalized* Reed-Solomon codes. Indeed, assume we receive a vector $\boldsymbol{v} = \boldsymbol{c} + \boldsymbol{e} = (v_1, \ldots, v_n)$, for some $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ and $\mathrm{wt}(\boldsymbol{e}) \leq t$. Then $\boldsymbol{c}' \stackrel{\text{def}}{=} (c_1 y_1^{-1}, \ldots, c_n y_n^{-1}) \in \mathbf{RS}_k(\boldsymbol{x})$. It is then enough to compute $(v_1 y_1^{-1}, \ldots, v_n y_n^{-1})$ from $\boldsymbol{v}$ and apply the decoding algorithm for $\mathbf{RS}_k(\boldsymbol{x})$. Since $\mathrm{wt}\left((e_1 y_1^{-1}, \ldots, e_n y_n^{-1})\right) = \mathrm{wt}(\boldsymbol{e}) \leq t$, the decoder recovers $\boldsymbol{c}'$, from which $\boldsymbol{c}$ can be immediately obtained. That is the reason why in the following we will simply focus on RS codes rather than GRS codes, without loss of generality. Since we got rid off the multiplier vector, we come back to usual notation where $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ represents the received word, for some $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbf{RS}_k(\boldsymbol{x})$ and $\mathrm{wt}(\boldsymbol{e}) \leq t$.

We present now a simplified version of Berlekamp-Welch algorithm.

Let $E \stackrel{\text{def}}{=} \{i \in [\![1, n]\!] \mid e_i \neq 0\}$, i.e. the set of error positions and define the *error locator polynomial* as

$$\Lambda(X) \stackrel{\text{def}}{=} \Pi_{i \in E}(X - x_i) \in \mathbb{F}[X].$$

The error locator polynomial Λ is monic and $\deg(\Lambda) = |E| = t$, so $\Lambda = X^t + \sum_{j=0}^{t-1} \lambda_i X^i$. By construction, its t roots are the $\boldsymbol{x}$ coordinates corresponding to E. Moreover, the sought codeword $\boldsymbol{c}$ is, by definition of RS codes, the evaluation vector in $\boldsymbol{x}$ coordinates of some polynomial P with $\deg(P) < k$, i.e. $\boldsymbol{c} = (P(x_1), \ldots, P(x_n))$. Therefore, two cases may occur:

- $i \in E$: then $\Lambda(x_i) = 0$.
- $i \notin E$: then $y_i = c_i + e_i = c_i = P(x_i)$.

It can be readily checked that both cases imply

$$y_i \Lambda(x_i) = P(x_i)\Lambda(x_i), \tag{1.3}$$

which hence holds for all $i \in [\![1, n]\!]$. We can look at the polynomial equation (1.3) as a system of equations obtained by equating the corresponding coefficients of left and right hand sides. Indeed, while x_i's and y_i's are known, the coefficients of both Λ and P are unknown and can be set as variables. The product $P(x_i)\Lambda(x_i)$ provides quadratic coefficients in the unknowns, and in particular they are bilinear if we split the variables in the sets of P's coefficients and Λ's coefficients. In any case, the

system is not linear. However we can perform a linearization, by defining an auxiliary polynomial:

$$g \stackrel{\text{def}}{=} P \cdot \Lambda$$

and picking g's coefficients as new variables. Hence Equation (1.3) translates into

$$\forall i \in [\![1, n]\!], \quad y_i \Lambda(x_i) = g(x_i).$$

Such a system clearly has a solution, which is the one derived from $\boldsymbol{c}$ and $\boldsymbol{e}$. Let us now count the number of unknowns and compare it with the dimension of the arising system. The error locator provides t variables, while $\deg(g) = \deg(P) + \deg(\Lambda) = k + t$, thus we have to add other $k + t + 1$ variables, for a total of $k + 2t + 1$. In this setting, once the linear system is set up, it is reasonable to find a unique solution whenever $n \geq k + 2t + 1$. In other words, the error correction capacity of the algorithm coincides with

$$t = \left\lfloor \frac{n - k - 1}{2} \right\rfloor.$$

In relative terms, the error correction capacity is $\frac{1-R}{2}$, where R is the code rate. The linear algebra part of the algorithm turns out to be the most expensive to do. In the unique decoding regime, the number of unknowns is upper bounded by n and thus the cost amounts to $\mathcal{O}(n^\omega)$ is the linear algebra constant and this term dominates the overall complexity of the Berlekamp-Welch algorithm. In fact, P can then be determined through Euclidean division of g with respect to Λ with cost $\mathcal{O}(tk)$ and $\boldsymbol{c}$ is obtained by evaluating P in all x_i's in $\mathcal{O}(kn)$.

Remark 1.5. The other standard method to makes use of the Euclidean algorithm instead of linear algebra and yields an improved complexity of $\mathcal{O}(n^2)$. However, this would require some care about the details. So we gave preference to the simplicity and immediacy of the linear algebra approach, which is too often ignored in textbooks and presentations.

The Berlekamp-Welch algorithm is already enough to make RS codes a really special family for error-correction purposes. Nevertheless, it has been non-constructively proved the existence of codes with decoding algorithm correcting errors with weight up to $1 - R - \epsilon(q)$, with $\epsilon(q) \to 0$ for $q \to \infty$, i.e. twice the value of the algorithm just described.

Although Berlekamp-Welch fails to work above half the minimum distance, it always outputs a unique codeword in the regime for which it is designed. On the other hand, an enhanced correction capability comes for a price: the resulting algorithm will output a list of candidate solutions rather than a unique solution. These decoders are called **list-decoding algorithms** and the fraction $1 - R$ is known as *list-decoding capacity*. This optimal error-correction has been asymptotically attained in a breakthrough paper from Guruswami and Rudra [GR06] for a family constructed upon RS codes, namely *folded Reed-Solomon* codes.

One might still think that these algorithms are totally useless; for instance if the output list contains all (or almost all) the codewords, the problem of correcting an error would definitely not be solved. We will see that the decoder becomes relevant if the decoding radius is such that the list size is only polynomial in the code length. In this case, we highlight two facts that make list decoders worthy of study:

- If the output list is short enough, it can be efficiently sorted with respect to the distance from the received word, then the closest word is the solution to the maximum likelihood decoding problem. Even if such a word is not unique, we still obtain some useful information in case a retransmission is possible.
- The bounds on the list size for these algorithms refer to a worst-case scenario. In practice, most of the time these decoders output a list with just one (or very few) elements.

The decoding radius for which the list size is polynomially bounded can be determined, and it is known as **Johnson's radius**.

Theorem 1.5 (Johnson's bound)**.** *Let $\mathscr{C} \in \mathbb{F}_q^n$ be an $[n, k, d]$ q-ary code, and let $\delta = d/n$ be the relative distance. Then the Johnson radius ρ is defined as*

$$\rho \stackrel{def}{=} \left(1 - \frac{1}{q}\right)\left(1 - \sqrt{1 - \frac{q\delta}{q-1}}\right)$$

and for any $\boldsymbol{y} \in \mathbb{F}_q^n$,

$$|\{\boldsymbol{c} \in \mathscr{C} \mid d(\boldsymbol{y}, \boldsymbol{c}) \leq \rho n\}| \leq qdn = \mathcal{O}(qn^2).$$

Remark 1.6. The Johnson radius depends on the field size q of $\mathscr{C}$. In the extremal binary case we get

$$\rho = \frac{1 - \sqrt{1 - 2\delta}}{2},$$

while for $q \to \infty$

$$\rho \to 1 - \sqrt{1 - \delta}.$$

In the latter case, if $\mathscr{C}$ is also an MDS code, then

$$\rho \to 1 - \sqrt{R},$$

R being the code rate. This is the case for instance of a sequence of RS codes, for which a length growing to ∞ implies a field size tending to ∞ as well.

The first step towards efficient list-decoding algorithms is due to Sudan, who proposed the first polynomial time decoder of this type in 1997 [Sho97]. Despite not reaching an error correction capacity corresponding to the Johnson bound, its work was a huge breaktrough in algebraic coding theory, which earned him a Nevanlinna prize.

The core idea consists in translating the decoding problem in terms of a root finding problem on a bivariate polynomial vanishing on several points, which can be solved through linear algebra. For a sequence of RS codes, with length growing to ∞ and constant rate R, the asymptotic list size from Sudan's algorithm is

$$\frac{1 - \rho}{R},$$

where ρ is the relative decoding radius. Prescribing that the arising linear system has more variables than equations (which guarantees the existence of solutions) leads to an asymptotic decoding radius of

$$\rho = 1 - \sqrt{2R}.$$

In order to improve on the Berlekamp-Welch algorithm, the code rate must then be low enough, so that $1 - \sqrt{2R} \geq \frac{1-R}{2}$. A straightforward computation shows that asymptotically Sudan's algorithm outperforms Berlekamp-Welch algorithm for $R \lessapprox 0.1716$. We also mention that the complexity of the algorithm is dominated by the linear algebra part and is therefore given by $\mathcal{O}(n^\omega)$.

In [GS99], Guruswami and Sudan improved the Sudan algorithm. In particular they generalized it in such a way that it now takes into account some multiplicity constraints. Finally the Guruswami-Sudan algorithm manages to achieve the asymptotic Johnson bound $\rho = 1 - \sqrt{R}$. Contrarily to the previous list decoder, this one always improves the correction capability upon half the minimum distance, since $1 - \sqrt{R} > \frac{1-R}{2}$ for any rate R. Despite being polynomial time, the decoder has a much higher complexity than *e.g.* Sudan's algorithm. Nonetheless, several improvements have been studied, eventually leading to a complexity of $\tilde{\mathcal{O}}(v^\omega sn)$ [Cho+15] (where $\tilde{\mathcal{O}}()$ indicates that we omit polylogarithmic terms, while s and v are respectively the multiplicity and powering parameters of the decoder), which makes possible practical implementations.

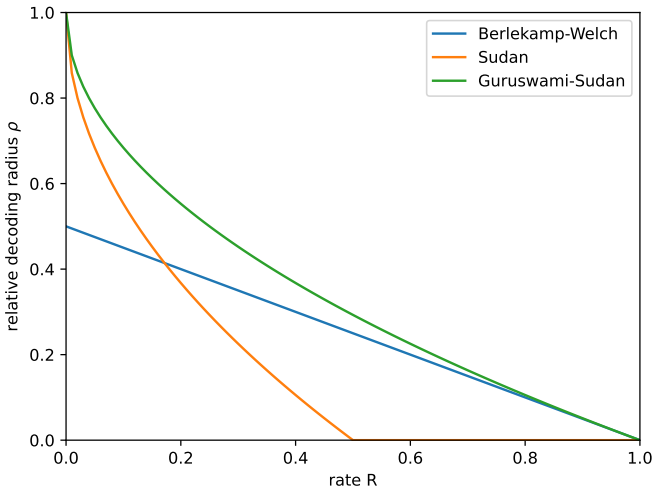


Figure 1.2: Comparison Reed-Solomon decoders radii. Note that Sudan improves upon Berlekamp-Welch only for low rates, whereas Guruswami-Sudan radius is always above Berlekamp-Welch radius.

1.1.4 Subfield subcodes of RS codes: alternant and Goppa codes

The problem with Reed-Solomon codes is that long codes need to be defined over big fields. In order to keep the field size constant, while preserving the decoding algorithms and (some of) the nice properties of RS codes, alternant codes have been introduced. Informally, these are subcodes of RS codes where only the codewords lying over a subfield are taken.

In the context of subfield subcodes we will often consider a finite field $\mathbb{F}_{q^m}$ and its subfield $\mathbb{F}_q$. It is therefore useful to fix a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$:

$$\{\alpha_0, \ldots, \alpha_{m-1}\}.$$

We will also make use of a *normal basis*

$$\{\beta, \beta^q, \ldots, \beta^{q^{m-1}}\}$$

whenever fruitful.

The field trace operator, from now on simply called **trace**, is a map that allows to map elements from $\mathbb{F}_{q^m}$ to its subfield $\mathbb{F}_q$.

Definition 1.15 (Trace)**.** Given the finite field extension $\mathbb{F}_{q^m}/\mathbb{F}_q$, we define the trace map $\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}\colon \mathbb{F}_{q^m} \to \mathbb{F}_q$ for all $x \in \mathbb{F}_{q^m}$ as

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(x) = \sum_{i=0}^{m-1} x^{q^i}.$$

The definition extends component-wise to vectors $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$:

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\boldsymbol{x}) = (\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(x_1), \ldots, \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(x_n))$$

and consequently to codes $\mathscr{C} \subseteq \mathbb{F}_{q^m}$

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}) = \{\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathscr{C}\}.$$

We remark that if $\mathscr{C} = \langle\, \boldsymbol{c}_i \mid 1 \leq i \leq k \,\rangle_{\mathbb{F}_{q^m}}$ then $\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C})$ is a linear code over $\mathbb{F}_q$ and

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}) = \left\langle\, \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\alpha_j \boldsymbol{c}_i) \mid 0 \leq j < m, 1 \leq i \leq k \,\right\rangle_{\mathbb{F}_q}.$$

Essentially, multiplying the generators of $\mathscr{C}$ inside the trace by each element α_j of the extension field basis takes into account the fact that $\mathscr{C}$ is a code over $\mathbb{F}_{q^m}$, while $\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C})$ is defined over the subfield $\mathbb{F}_q$. So, if $\dim_{\mathbb{F}_{q^m}} \mathscr{C} = k$ then we typically expect $\dim_{\mathbb{F}_q} \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}) = mk$. A counterexample is given when $\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C})$ coincides with the ambient space, since the dimension clearly can not exceed the code length. From now on, we will omit the extension field $\mathbb{F}_{q^m}/\mathbb{F}_q$ and simply write Tr, whenever the former is clear from the context.

Delsarte's theorem is a classical result linking a trace code to a **subfield subcode**, i.e. the intersection of a code with the vector space defined by a subfield.

Theorem 1.1 (Delsarte's theorem, [Del75])**.** *Let $\mathscr{C}$ be a code over $\mathbb{F}_{q^m}$. Then*

$$(\mathscr{C}_{|\mathbb{F}_q})^\perp = \mathrm{Tr}(\mathscr{C}^\perp),$$

where $\mathscr{C}_{|\mathbb{F}_q} \stackrel{def}{=} \mathscr{C} \cap \mathbb{F}_q$ denotes the subfield subcode over $\mathbb{F}_q$ of $\mathscr{C}$.

Notation 1.1. Although the code dimension is usually denoted with k, in this context it will be replaced by r. This choice is done in compliance with a significant part of the literature on Goppa codes for code-based cryptography, where r indicates the degree of a Goppa polynomial, the latter being a notion related to the dimension. By extension, the same notation will be used for GRS codes and generic alternant codes.

An **alternant code** can be defined as the subfield subcode of a GRS code:

Definition 1.16 (Alternant code)**.** Let $n \leq q^m$, for some positive integer m. Let $\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ be the GRS code over $\mathbb{F}_{q^m}$ of dimension r with support $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ and multiplier $\boldsymbol{y} \in (\mathbb{F}_{q^m}^*)^n$. The *alternant code* with support $\boldsymbol{x}$ and multiplier $\boldsymbol{y}$ and *degree* r over $\mathbb{F}_q$ is

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n.$$

The integer m is called *extension degree* of the alternant code.

Remark 1.7. By Proposition 1.5 we immediately infer that an alternant code is the subfield subcode of a GRS code:

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}^\perp) \cap \mathbb{F}_q^n.$$

Notation 1.2. We use the same notation as in [MS86] and use the dimension r and the multiplier $\boldsymbol{y}$ of the dual GRS code, which turns out to be more convenient in our setting.

It can be verified that the alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ of length n has dimension lower bounded by $n - rm$ and minimum distance strictly larger than r. In other words, it is a $[n, \geq n - rm, \geq r + 1]$ code. Since $\mathbf{GRS}_{n-r}(\boldsymbol{x}, \boldsymbol{y}^\perp)$ is a supercode with respect to $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, the same decoding algorithms used for GRS codes can be adopted for the corresponding alternant codes. This comes at the price of a scaling of the decoding radius proportional to the extension degree m. In particular, a polynomial-time decoder which works up to half the minimum distance for the former code provides a polynomial time decoding algorithm for the family of alternant codes essentially up to $\frac{n-(n-r)}{2} = \frac{r}{2}$ errors. We observe that for $m = 1$ an alternant code is simply a GRS code. Therefore from now on we will always assume $m > 1$.

From Delsarte's theorem (Theorem 1.1) and by duality,

$$\begin{aligned} \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp &= \Big(\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \cap \mathbb{F}_q^n \Big)^\perp \\ &= \mathrm{Tr}\Big((\ \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^\perp)^\perp \Big) \\ &= \mathrm{Tr}\,(\ \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y}))\,. \end{aligned} \tag{1.4}$$

The dimension of an alternant code of order r built upon an extension field of degree m has therefore dimension at least $n - rm$. There exists a subclass of alternant codes which is particularly attractive for cryptographic purposes:

Definition 1.17. Let $\boldsymbol{x} \in \mathbb{F}_{q^m}^n$ be a support vector and $\Gamma \in \mathbb{F}_{q^m}[z]$ a polynomial of degree r such that $\Gamma(x_i) \neq 0$ for all $i \in \{1, \ldots, n\}$. The *Goppa code* of degree r with support $\boldsymbol{x}$ and *Goppa polynomial* Γ is defined as

$$\mathscr{G}(\boldsymbol{x}, \Gamma) \stackrel{\text{def}}{=} \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}),$$

where $\boldsymbol{y} \stackrel{\text{def}}{=} \left(\frac{1}{\Gamma(x_1)}, \ldots, \frac{1}{\Gamma(x_n)} \right)$.

The reason why binary Goppa codes are preferable to instantiate McEliece-like schemes is that, if the Goppa polynomial has no multiple roots, there exists a polynomial time algorithm to decode up to r errors. This follows directly from

Theorem 1.2. *[Pat75] Let $\mathscr{G}(\boldsymbol{x}, \Gamma)$ be a binary Goppa code with the Goppa polynomial Γ of degree r and square-free. Then*

$$\mathscr{G}(\boldsymbol{x}, \Gamma) = \mathscr{G}(\boldsymbol{x}, \Gamma^2) = \mathscr{A}_{2r}(\boldsymbol{x}, \boldsymbol{y}),$$

where $y_i \stackrel{def}{=} \frac{1}{\Gamma(x_i)^2}$ for all $1 \leq i \leq n$.

It follows from Theorem 1.2 that a square-free binary Goppa code is a $[n, \geq n - rm, \geq 2r + 1]$ code. It is enough to apply the decoding algorithm for alternant codes on $\mathscr{A}_{2r}(\boldsymbol{x}, \boldsymbol{y})$, thus correcting $\frac{2r}{2} = r$ errors. In a cryptographic framework, this becomes relevant because it permits to choose better trade-offs between key-size and security level and design more competitive parameters. Moreover, a more efficient algorithm, which also corrects up to r errors, can be designed for binary Goppa codes. This is called Patterson's algorithm [Pat75].

1.1.5 Product and square of codes

The next operation we are going to introduce is a binary operator on codes which could be perceived as an artificial and innatural notion. However, it will find application in code-based cryptography, leading to both distinguishers and attacks on several schemes relying on structured codes.

First we define a binary operator on vectors defined over the same field and with the same length, called **component-wise product** or (**Schur's product**).

Definition 1.18 (Component-wise product of vectors)**.** The *component-wise product* of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{F}^n$ is defined as

$$\boldsymbol{a} \star \boldsymbol{b} \stackrel{\text{def}}{=} (a_1 b_1, \ldots, a_n b_n).$$

Remark 1.8. GRS codes can be conveniently generated by vectors that are component-wise products of $\boldsymbol{x}$ and $\boldsymbol{y}$.

Schur's product can then be naturally extended to codes.

Definition 1.19 (Product of codes)**.** The **component-wise product of codes** $\mathscr{C}, \mathscr{D}$ over $\mathbb{F}$ with the same length n is defined as

$$\mathscr{C} \star \mathscr{D} \stackrel{\text{def}}{=} \langle\, \boldsymbol{c} \star \boldsymbol{d} \mid \boldsymbol{c} \in \mathscr{C}, \boldsymbol{d} \in \mathscr{D} \,\rangle_{\mathbb{F}}.$$

If $\mathscr{C} = \mathscr{D}$, we call $\mathscr{C}^{\star 2} \stackrel{\text{def}}{=} \mathscr{C} \star \mathscr{C}$ the **square code** of $\mathscr{C}$.

A generating set for $\mathscr{C} \star \mathscr{D}$ over $\mathbb{F}$ can be constructed by taking the $k_1 k_2$ products between the generators of $\mathscr{C} = \langle\, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_{k_1} \,\rangle_{\mathbb{F}}$ and $\mathscr{D} = \langle\, \boldsymbol{d}_1, \ldots, \boldsymbol{d}_{k_2} \,\rangle_{\mathbb{F}}$, specifically

$$\{\boldsymbol{c}_i \star \boldsymbol{d}_j \mid 1 \leq i \leq k_1, 1 \leq j \leq k_2\}$$

is a generating set for $\mathscr{C} \star \mathscr{D}$. However, this is not always a basis, as some linear dependencies can occur among the products. For instance when $\mathscr{C} \cap \mathscr{D} \neq \{\mathbf{0}\}$, some of the above elements are obviously redundant. In the extreme case, i.e. when $\mathscr{C} = \mathscr{D}$, the square code dimension is much smaller than k^2, where $k = \dim_{\mathbb{F}} \mathscr{C}$. This is a consequence of the commutative property of the component-wise product: $c_i \star c_j = c_j \star c_i$. Thus we can give the following folklore result appearing for instance in [Cas+15].

Proposition 1.6. *Let $\mathscr{C}$ be a linear code over $\mathbb{F}$ of dimension k and length n. Then*

$$\dim_{\mathbb{F}_q} \mathscr{C}^{\star 2} \leq \min\left(n, \binom{k+1}{2}\right).$$

More in general if $\dim_{\mathbb{F}_q} \mathscr{C} = k_1, \dim_{\mathbb{F}_q} \mathscr{D} = k_2$ and $\dim_{\mathbb{F}_q}(\mathscr{C} \cap \mathscr{D}) = k_\cap$, then

$$\dim_{\mathbb{F}_q} \mathscr{C} \star \mathscr{D} \leq \min\left(n, k_1 k_2 - \binom{k_\cap}{2}\right)$$

as the generator products coming from the intersection must be counted only once.

For a random linear code $\mathscr{C}$ whose square does not fill the full space, the dimension of its square code is $\binom{k+1}{2}$ with high probability [Cas+15], where k is the dimension of $\mathscr{C}$. More precisely, we have the following result:

Theorem 1.3 ([Cas+15], Theorem 2.3)**.** *Let $n\colon \mathbb{N} \mapsto \mathbb{N}$ be such that $n(k) \geq \binom{k+1}{2}$ for all $k \in \mathbb{N}$ and let $s\colon \mathbb{N} \mapsto \mathbb{N}$ be defined as $s(k) \stackrel{def}{=} n(k) - \binom{k+1}{2}$. Then there exists a constant $c \in \mathbb{R}_{>0}$ such that for k large enough,*

$$\mathbb{P}\left(\dim \mathscr{C}^{\star 2} = \binom{k+1}{2}\right) \geq 1 - 2^{-cs(k)},$$

where $\mathscr{C}$ is chosen uniformly at random among the $[n(k), k]$ codes over $\mathbb{F}$.

Moreover, if the code length does not exceed $\binom{k+1}{2}$ then the square code fills the ambient space $\mathbb{F}^n$ with high probablity. Namely,

Theorem 1.4 ([Cas+15], Theorem 2.5)**.** *There exist two constants (depending on the field size q) $c, c' \in \mathbb{R}_{>0}$ such that, if for all $k \in \mathbb{N}$, $n : \mathbb{N} \mapsto \mathbb{N}$ satisfies*

$$k \leq n(k) \leq c\binom{k+1}{2},$$

then for k large enough

$$\mathbb{P}\left(\mathscr{C}^{\star 2} = \mathbb{F}_q^n\right) \geq 1 - 2^{-c's(k)},$$

where $\mathscr{C}$ is chosen uniformly at random among the $[n(k), k]$ codes over $\mathbb{F}_q$.

However, there exist families of codes for which the inequality in Proposition 1.6 is strict. By computing the square code one can therefore determine with good probability if the original code belongs to such families or not. Generalized Reed-Solomon codes represent an example of such behavior and turn out to display a very peculiar property with respect to the component-wise/Schur product [Wie10].

Proposition 1.7. *Let $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ be a GRS code with support $\boldsymbol{x}$, multiplier $\boldsymbol{y}$ and dimension k. We have $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^{\star 2} = \boldsymbol{GRS}_{2k-1}(\boldsymbol{x}, \boldsymbol{y}^2)$. Hence, if $k \leq \frac{n+1}{2}$,*

$$\dim_{\mathbb{F}_{q^m}}(\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y}))^{\star 2} = 2k - 1.$$

This follows on the spot from the fact that the square of a $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ can be written as

$$\begin{aligned}\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})^{\star 2} &= \left\langle (\boldsymbol{x}^a \boldsymbol{y}) \star (\boldsymbol{x}^b \boldsymbol{y}) \mid 0 \leq a, b < k \right\rangle_{\mathbb{F}_{q^m}} \\ &= \left\langle \boldsymbol{x}^{a+b} \boldsymbol{y}^2 \mid 0 \leq a, b < k \right\rangle_{\mathbb{F}_{q^m}} \\ &= \left\langle \boldsymbol{x}^c \boldsymbol{y}^2 \mid 0 \leq c < 2k-1 \right\rangle_{\mathbb{F}_{q^m}} .\end{aligned}$$

Note that the square code dimension is here $2k-1$, i.e. it is linear in k and not quadratic. This implies that (generalized) RS codes up to a constant rate can be distinguished from random. In particular it is required that $2k-1 < n$, yielding a distinguishable rate R in the interval $[0, 1/2]$. Furthermore, since the dual of a (generalized) Reed-Solomon code is again a (generalized) Reed-Solomon code, it turns out that this algebraic class is distinguishable for any rate. Indeed, if $R > 1/2$, it is possible to square the dual code and check whether its dimension lies below the code length.

For other families, it may happen that a distinguisher of this kind still exist but only occur for certain rates. For instance Wild Goppa codes (i.e. Goppa codes with additional properties) are distinguishable in the case of a quadratic extension. Secondly all Goppa codes (and more in general alternant codes) are distinguishable whenever the rate is high enough. Interestingly enough, in the latter case the distinguishing property does not hold for the primal code but for its dual code. However the maximum distinguishable rate is not constant, as the dimension of the square code of the dual code is still quadratic with respect to the dual code dimension. We will explain this more involved behaviour regarding alternant codes in the Chapter 3.

1.2 Code-based cryptography

1.2.1 Public key cryptography

Cryptography (from Ancient Greek (romanized): kryptós "hidden", and graphein, "to write") is the science of keeping information secret and ensuring secure communications. One of the greatest and most beautiful revolutions in its history is the so-called *public-key cryptography* or *asymmetric cryptography*, first introduced in the breakthrough work of Diffie and Hellman [DH76] in 1976. Before that and starting probably in Ancient Egypt, about 4000 years ago, cryptography underwent multiple transformations and inspired studying and solving problems, mathematical and otherwise, which sometimes even transcended the mere application of hiding a message. It has been a central figure in wars (from Roman times through World War II, until the present days) and has partially contributed to the development of information theory and computation.

Since the dawn of what is considered modern cryptography, following Shannon's seminal work [Sha48], it was clear that a secure system has to be based on *computationally difficult mathematical problems* and such hardness must be formalized by a model which states the *security level* one wants to reach. Nevertheless, until the second half of the 1970s, no one knew how to answer a fundamental issue: is it possible to secure secret communication between two entities that have never met and exchanged information before? Such a question seemed out of reach and even counterintuitive, hardly anyone in fact believed that it had an affirmative answer. On the other hand, the interest in it was growing year by year along with the number of companies and banks that used computers and their services and consequently needed to exchange massive amounts of information internally or to their customers, through insecure channels. This required physically distributing keys to trusted people, and while this was feasible for military purposes it soon became untenable for civil use.

This obstacle was finally overcome by the aforementioned article from Diffie and Hellman. They theorized the existence of a *trapdoor one-way function*, i.e. of a function that is easy to compute but difficult to inverse, unless one knows a secret (rightly the *trapdoor*). However, they had been unable to find a practical example of such a function. Two years later, in 1978, Rivest, Shamir and Adleman suggested to instantiate the trapdoor with the exponention modulo a product of large prime numbers, thus proposing the first public-key cryptosystem: RSA [RSA78], called in this way from the initials of their names.

Over the years, the number of trapdoor functions for instantiating asymmetric cryptosystems has multiplied, and nowadays the palette of primitives is extremely rich, embracing several difficult mathematical problems. In parallel to the development of the technology, new needs and challenges have emerged to which public key cryptography is trying to respond. We cite for example *multiparty computation*, which allows communication between multiple users with different roles and remains secure even if some of them are malicious. Or again *(fully) homomorphic encryption*, which permits to perform computations on encrypted data without having to decrypt them before, an extremely useful feature when working with sensible storaged data. Finally we talk about *post-quantum cryptography* to refer to the study of primitives that remain secure even against the threat of quantum computers.

1.2.2 Quantum computing in a nutshell

Quantum computing is an emerging technology and its study started in the 1980's as a subfield of quantum physics. Quantum computers are devices whose operations exploit the laws of *quantum mechanics*, such as superposition, interference and entanglement. For a deeper treatment on the topic, the reader can refer to the renown textbook from Nielsen and Chuang [NC02].

The basic unit for computation in classical computer are the bits, which can assume only two states, 0 or 1. In quantum computers these are replaced by *qubits*.

Definition 1.20 (Qubit)**.** A **qubit** is a two-dimensional quantum-mechanical system whose state $|\psi\rangle$ is an element of the two-dimensional Hilbert space $\mathcal{H} \equiv \mathbb{C}^2$ represented by a linear combination of the *computational basis* states $|0\rangle$ and $|1\rangle$, i.e.

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \text{s.t. } |\alpha|^2 + |\beta|^2 = 1,$$

where $\alpha, \beta \in \mathbb{C}$ are called *probability amplitudes*.

A *quantum circuit* is the quantum counterpart of classical circuits and uses a fixed number of qubits initialized to $|0\rangle$.

Differently from classical computation, where the states registered on the bits can be read at any time without affecting them, *quantum measurement* is an irreversible operation which gains information on a single qubit but collapses the state. If the state of a qubit is $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$, measuring it will result in the state $|0\rangle$ with probability $|\alpha|^2$ or $|1\rangle$ with probability $|\beta|^2$.

We want to give an insight of the intrinsic potentialities of quantum computing which are missing in classical systems. What really distinguishes qubits from bits is a phenomenon called *quantum entanglement*, that can be contemplated when at least two qubits are considered. This property means that some qubits are correlated. A *quantum register* is a system made of multiple qubits. For instance, a quantum register of two qubits is spanned by 2^2 basis states, denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. Now consider the two entangled qubits in the so called *Bell state*:

$$\frac{1}{\sqrt{2}}(|00\rangle + |11\rangle).$$

Because of what said before, a measurement of the state will end up in either the state $|00\rangle$ or the state $|11\rangle$, both with probability $\frac{1}{\sqrt{2}^2} = \frac{1}{2}$. Now suppose we start measuring the first qubit and we observe the state $|0\rangle$. Then, we have the certitude that the measurement of the second qubit will also give the state $|0\rangle$ (and viceversa). This can not be explained by classical physics.

The implications of this new model of computation are various and fall in the fields of computational biology, chemistry, cryptography, machine learning etc. In the following we will focus on a particular aspect of the impact of quantum computers in cryptography.

1.2.3 Post-quantum cryptography

Public-key cryptosystems used currently mainly rely on the integer factorization (RSA) and discrete logarithm over a finite field (DSA) or an elliptic curve (ECDSA) problems.

While efficient classical algorithms to tackle these problems are not known, a quantum computer would be able to solve them in polynomial time thanks to a *quantum algorithm* found by Peter Shor in 1994 [Sho94]. We remark that this algorithm actually solves the *Hidden Subgroup Problem* for finite abelian groups [Joz01] of which integer factorization and discrete logarithm can be seen as particular subinstances.

Nowadays, designing quantum-secure schemes has become one of the main trends in asymmetric cryptography. Shor's algorithm is indeed a huge breakthrough with potential serious consequences for digital security, even though the algorithm requires large and reliable quantum computers to be run, which do not exist yet. However, in the last decade, big companies such as IBM, Google, Microsoft, D-Wave, Rigetti etc., started heavily financed research projects for developing large scale quantum computers. Right now, the experimental quantum computers realized by these companies definitely lack of sufficient power processing to break any classic cryptosystem. Indeed there are several physical and engineering difficulties to overcome depending on the different possible technologies adopted, *e.g.* ion traps, transmons or topological quantum computer. Without going into details, since an in-depth discussion is beyond the scope of this document, we just say that various computing models have been theorized, such as quantum logic gates, adiabatic quantum computation or quantum annealing.

Note that quantum computers obey the *Church-Turing thesis*, and thus can solve exactly the same problems that can be attacked by a classical computer. On the other side, for certain problems, the former can have significantly lower time complexities than the latter, even when considering supercomputers. This feature is sometimes called *quantum supremacy*.

As already said we are still far from quantum computers becoming a real world technology, and it is not even certain that quantum computers will ever reach the point where they can break classical schemes like RSA. However the cryptographic community for some time now started working on new primitives that are believed to be quantum-safe, both in the designing and cryptanalysis sides, in order to stave off the quantum threat. The reason is that the post-quantum transition is a slow process, which demands a joint effort from multiple personalities with different expertise. The realization of a quantum computer before quantum-safe cryptographic schemes get properly studied and deployed would have terrible consequences and we can not take such a risk.

The interest in studying quantum-secure schemes is reflected by the choice from the American National Institute of Standards and Technologies (NIST) to announce in 2016 a standardization procedure for post-quantum cryptography, then launched in the late 2017[1]. NIST competition consisted so far of three rounds, after which one *public-key encryption/key-establishment algorithm* (PKE/KEM for short) and three *digital signature* algorithms were selected in 2022. Currently a fourth round is ongoing for schemes that are considered secure and/or promising but need further investigation.

We can identify multiple hard problems that are believed not to be substantially more vulnerable against quantum computers than against classical ones. We can

[1] https://csrc.nist.gov/projects/post-quantum-cryptography.

gather them in five families which we briefly describe. For a more complete overview on post-quantum cryptography, see *e.g.* [BBD].

Lattice-based cryptography. A *lattice* is defined as a discrete subgroup of $\mathbb{R}^n$. Some hard problems related to this object can be defined, for instance the *Shortest Vector Problem* (SVP for short), either in its exact or approximated form, which asks to determine the vector of smallest norm lying in the lattice. Or the *Closest Vector Probem* (CVP for short), where a point in the lattice with minimal distance from a given point is sought. A classic cryptosystem based on the hardness of lattice problems is NTRU [HPS98]. More recently, another computational problem called *Learning With Error* (LWE for short) [Reg05], for which Regev won the Gödel prize, and its variants have been used as hardness assumption for PKEs [Pei14].

At present, lattice-based schemes are the ones which achieve better overall performance, for what concern both PKEs and digital signatures. This is confirmed by NIST standardization process (see Table 1.1). Moreover, lattices represents the main solution to advanced form of encryption, *e.g. Fully Homomorphic Encryption* (FHE for short) [Gen09].

Code-based cryptography. Cryptography based on error-correcting codes is by far the oldest quantum-safe alternative. As a matter of fact, McEliece proposed a cryptosystem of this kind back in 1978 [McE78], the same year of RSA. It was based on the difficulty of decoding a random linear code (an NP-complete problem). Unfortunately, the drawback of this scheme are its very large public key (in the order of one thousand times larger than RSA), reason why it did not receive much attention from the community at the time. However, it has been revalued in more recent times thanks to the fact that it is believed to be quantum-resistant. Because of its age, it is nowadays considered one of the most trusted schemes in terms of security. New frameworks and techniques have also been discovered since then, leading to new and more efficient code-based schemes. We will discuss this family of cryptosystems in much deeper detail in the rest of this subsection.

Multivariate cryptography. This includes the primitives based on multivariate (typically quadratic) polynomial systems over a finite field. Solving systems of multivariate equations is proven to be NP-complete and the best generic techniques, i.e. *Gröbner basis* algorithms (see Section 1.3), have exponential complexity.

The first multivariate scheme, from Matsumoto and Imai, is called C* [MI88] and dates back to 1988. Although it has been broken in [Pat95], C* inspired other cryptosystems relying on similar ideas, such as *Hidden Field Equation* (HFE for short) [Pat96], *Balanced Oil & Vinegar* [Pat97] and *Unbalanced Oil & Vinegar* (UOV for short) [KPG99]. Despite not being extremely competitive on the PKE side, multivariate cryptography is an excellent approach for building digital signatures as it provides the shortest signatures among all post-quantum alternatives. On the other hand, recent catastrophic key-recovery attacks [Bae+21; Beu22; TPD21] on the third round NIST candidates Rainbow and GeMSS demonstrate how difficult it is to design long-term secure multivariate schemes. This is related to the hardness of understanding the real power of Gröbner bases computation, as new algebraic modelings and strategies appeared.

Isogeny-based cryptography. This is the youngest family of quantum-safe primitives. These are based on the hardness of finding isogenies between supersingular elliptic curves. An analogue of Diffie-Hellman key-exchange, called *Supersingular Isogeny Diffie-Hellman* (SIDH for short), has first been proposed by Jao and De Feo

	1st Round			2nd Round			3rd Round			Standards+4th Round		
	KEMs	Sig.s	Overall	KEMs	Sig.s	Overall	KEMs	Sig.s	Overall	KEMs	Sig.s	Overall
Lattice-based	21	5	26	9	3	12	3+2	2+0	5+2	1+0	2+0	3+0
Code-based	17	2	19	7	0	7	1+2	0+0	1+2	0+3	0+0	0+3
Multivariate	2	7	9	0	4	4	0+0	1+1	1+1	0+0	0+0	0+0
Hash-based	0	3	3	0	2	2	0+0	0+2	0+2	0+0	0+1	0+1
Other	5	2	7	1	0	1	0+1	0+0	0+1	0+1	0+0	0+1
Total	45	19	64	17	9	26	4+5	3+3	7+8	1+4	2+1	3+5

Table 1.1: Breakdown of NIST post-quantum candidates for each round and category. *KEMs* stands for "key encapsulation mechanisms/encryption schemes", while *Sig.s* is an abbreviation for signatures. Round 3 is split into "finalists+alternates". *Other* includes isogeny-based cryptography.

[JF11]. These primitives offer the smallest key-sizes of all post-quantum families, but they require to perform very heavy computations, leading to a number of cycles that is over one order of magnitude larger than the other alternatives. Although we can count on previous contributions from the elliptic curve cryptography (ECC) community, as there are partial intersections, this relatively fresh field still needs and deserves much more study. This is evidenced by very recent and devastating attacks on isogenies [CD22], [MM22], [Rob22].

Hash-based cryptography. We conclude with the family of cryptographic schemes relying on the security of *hash functions*. Hash-based cryptography is particularly devised to design digital signatures but also for protocols such as *zero-kwnoledge proofs* with advanced features. A precursor of hash-based schemes can be identified in Lamport's signature [Lam79]. Since hash-based schemes exploits one-time signature schemes, even combining several of them as building blocks, the former can be used to sign securely only a limited amount of messages.

We remark that, despite relying sometimes on very different hard problems, these families are not always completely unconnected. For instance multivariate and some code-based schemes are united by the fact that can be both attacked with algebraic techniques (this manuscript indeed address the field of algebraic cryptanalysis on code-based cryptosystems). On another plane, some studies have been recently dedicated to adapt and carry over techniques from lattice-based to code-based cryptography (and viceversa) [DDW22], [BCD22], [Deb+22].

1.2.4 Hard problems from coding theory

Code-based cryptography finds its roots in 1978 in the seminal work of McEliece [McE78]. The security of this PKE relies indeed on the hardness of *decoding a random linear code*. We now define the *search version* of the problem.

Problem 1.1 (Generic decoding (GD) problem)**.** *Given a matrix* $\boldsymbol{G} \in \mathbb{F}^{k\times n}$*,* $\boldsymbol{y} \in \mathbb{F}^n$ *and* $w \in \mathbb{N}$*, find a vector* $\boldsymbol{e} \in \mathbb{F}^n$*, such that* $wt(\boldsymbol{e}) \leq w$ *and* $\boldsymbol{y} - \boldsymbol{e} = \boldsymbol{m}\boldsymbol{G}$ *for some* $\boldsymbol{m} \in \mathbb{F}^k$*.*

The term *generic* refers to the fact that $\boldsymbol{G}$ is interpreted as the generator matrix of a *random* (linear) code. For binary linear codes, the *decision version* of the

problem, i.e. the one where is only asked whether such $\boldsymbol{e}$ exists, has been proven to be NP-hard in the worst case [BMT78] by reducing *Three-dimensional Matching*, another problem known to be NP-complete, to it. It is easy to see that the decoding problem is also in NP. Thus it is NP-complete. A fortiori, the search version is NP-complete, too. The proof has then been generalized to any field size in [Bar94].

Remark 1.9. The proof in [BMT78] only shows the hardness of the decoding problem in the worst case. However, it is believed to be difficult in the average case, too.

Problem 1.1 has a dual version, called *Syndrome decoding problem*, which is the one typically adopted. Again we provide the search version of the problem.

Problem 1.2 (Syndrome decoding (SD) problem)**.** *Given a matrix* $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}^{n-k}$ *and* $w \in \mathbb{N}$*, find a vector* $\boldsymbol{e} \in \mathbb{F}^n$*, such that* $wt(\boldsymbol{e}) \leq w$ *and* $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}$.

Notice that finding a codeword of bounded weight in a code, a major problem in coding theory which is NP-complete too [Var97], can be seen as an instantiation of SD problem.

Problem 1.3 (Bounded weight Codeword problem)**.** *Given a matrix* $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$ *and* $w \in \mathbb{N}$*, find a nonzero vector* $\boldsymbol{c} \in \mathbb{F}^n$*, such that* $wt(\boldsymbol{c}) \leq w$ *and* $\boldsymbol{c}\boldsymbol{H}^T = \boldsymbol{0}$.

For the sake of the presentation, we have ignored a subtlety about the McEliece framework. As it will become more clear in the next subsection, the aforementioned scheme is not exactly based on GD/SD problems. Indeed, since a specific code $\mathscr{C}$ must be chosen, the weight w here is naturally bounded by the error correction capacity of $\mathscr{C}$. This leads to the definition of another variant of SD problem:

Problem 1.4 (Bounded Syndrome decoding (BSD) problem)**.** *Given a matrix* $\boldsymbol{H} \in \mathbb{F}^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbb{F}^{n-k}$, $d \in \mathbb{N}$ *such that every set of* d *columns of* $\boldsymbol{H}$ *is linearly independent and* $w = \left\lfloor \frac{d-1}{2} \right\rfloor$*, find a vector* $\boldsymbol{e} \in \mathbb{F}^n$*, such that* $wt(\boldsymbol{e}) \leq w$ *and* $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}$.

This problem is believed to be NP-hard [Bar94] but, differently from syndrome decoding, is likely not in NP.

Remark 1.10. For all the problems introduced, one could consider the corresponding variants where the sought vector must have weight equal to w, instead of less or equal. These variants live in the same complexity class as their original counterpart. One side of the reduction is completely obvious. If we are able to find (when it exists) a codeword with weight w and satisfying the problem equation, the repeating the process up to weight $\bar{w}$, will solve the "less or equal" version of the problem.

So far, we implicitly considered problems on codes endowed with the Hamming metric. We will now adapt the GD and SD problems to the rank metric. In particular, the definitions adopted in the literature mirror the use of codes in rank-metric cryptography. Indeed, in this setting the codes employed are linear codes over an extension field $\mathbb{F}_{q^m}$ of degree m of $\mathbb{F}_q$. The codewords can be seen as elements of $\mathbb{F}_{q^m}^n$ but also as matrices in $\mathbb{F}_q^{m\times n}$. In particular, a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ corresponds to

$$\mathbf{Mat}(\boldsymbol{x}) \stackrel{\text{def}}{=} (x_{i,j})_{i,j} \in \mathbb{F}_q^{m\times n},$$

where $x_j = \sum_{i=1}^m b_i x_{i,j}$ for any $j \in [\![1, n]\!]$ and for a fixed basis $(b_1, \ldots, b_m)$ of $\mathbb{F}_{q^m}/\mathbb{F}_q$. Then we define the weight in the rank metric as $\mathrm{wt}_{\mathbf{Rank}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \mathbf{Rank}(\mathbf{Mat}(\boldsymbol{x}))$. Note that this distance does not depend on the choice of the basis.

Problem 1.5 (Rank decoding (RD) problem)**.** *Given a matrix* $\boldsymbol{G} \in \mathbb{F}_{q^m}^{k\times n}$ *of rank* k*,* $\boldsymbol{y} \in \mathbb{F}_{q^m}^n$ *and* $w \in \mathbb{N}$*, find a vector* $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$*, such that* $wt_{\mathbf{Rank}}(\boldsymbol{e}) \leq w$ *and* $\boldsymbol{y} - \boldsymbol{e} = \boldsymbol{mG}$ *for some* $\boldsymbol{m} \in \mathbb{F}_{q^m}^k$*.*

Problem 1.6 (Rank Syndrome decoding (RSD) problem)**.** *Given a matrix* $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *of rank* $n-k$*,* $\boldsymbol{s} \in \mathbb{F}_{q^m}^{n-k}$ *and* $w \in \mathbb{N}$*, find a vector* $\boldsymbol{e} \in \mathbb{F}_{q^m}^n$*, such that* $wt_{\mathbf{Rank}}(\boldsymbol{e}) \leq w$ *and* $\boldsymbol{eH}^T = \boldsymbol{s}$*.*

We conlude with a more recent problem on rank-metric, introduced in [Gab+16].

Problem 1.7 (Rank Support Learning (RSL) problem)**.** *Given a matrix* $\boldsymbol{H} \in \mathbb{F}_{q^m}^{(n-k)\times n}$ *of rank* $n-k$ *and the product* $\boldsymbol{EH}^T$*, where* $\boldsymbol{E} \in \mathbb{F}_{q^m}^{N\times n}$ *is such that all its entries belong to a subspace* $V \subseteq \mathbb{F}_{q^m}$ *of dimension* $w \in \mathbb{N}$*, find* V*.*

RSL problem essentially consists of N RSD instances $\boldsymbol{e}_i\boldsymbol{H}^T = \boldsymbol{s}$ sharing a common support V of dimension w. This problem has an analogue in the Hamming metric, called *Support Learning* problem, but it seems to have much more relevance in the rank-metric setting. Indeed, it is a versatile problem which, differently from many other problems, allows to devise code-based primitives whose security is based only on its difficulty.

1.2.5 McEliece's scheme

The structure of McEliece's PKE is fairly standard as it starts with the key-generation and then consists of encryption and decryption. We now describe it in its general form, e.g. without specifying the underlying code.

First of all, Alice chooses an $[n,k]$ code $\mathscr{C}_{\text{sec}} \in \mathbb{F}_q^n$ that can efficiently decode up to w errors with a decoding algorithm $\mathcal{D}$ which exploits the knowledge of the generator matrix $\boldsymbol{G}_{\text{sec}}$. The representation of the secret code is given by this matrix and hence it can not be safely shared, because it would reveal the algebraic shape of $\mathscr{C}_{\text{sec}}$ and how to decode it. To hide the structure, Alice first applies a scrambling to it, i.e. she chooses at random $\boldsymbol{S} \in \mathbf{GL}_k(\mathbb{F}_q)$ and a permutation matrix $\boldsymbol{P} \in \mathbf{GL}_n(\mathbb{F}_q)$ and computes $\boldsymbol{G}_{\text{pub}} = \boldsymbol{SG}_{\text{sec}}\boldsymbol{P}$. The goal is to produce a generator matrix $\boldsymbol{G}_{\text{pub}}$ of a code $\mathscr{C}_{\text{pub}}$ that seems random to someone not knowing $\boldsymbol{G}_{\text{sec}}$. Therefore $\boldsymbol{G}_{\text{pub}}$ and w are public, while $\boldsymbol{P}, \boldsymbol{G}_{\text{sec}}$ and $\boldsymbol{S}$ are kept secret.

It is now the turn for Bob to encrypt a message $\boldsymbol{m} \in \mathbb{F}_q^k$. He first encodes it by multiplying with $\boldsymbol{G}_{\text{pub}}$ and then adds a random error vector $\boldsymbol{e} \in \mathbb{F}_q^n$ of weight up to w to it. Thus, he sends the cipher $\boldsymbol{c} = \boldsymbol{mG}_{\text{pub}} + \boldsymbol{e}$ back to Alice. For a potential eavesdropper Eve, recovering the original message $\boldsymbol{m}$ would require to decode $\mathscr{C}_{\text{pub}}$, a task related to the NP-complete problem of generic decoding.

On the other hand, Alice, being in possession of the secret key, can compute

$$\boldsymbol{cP}^{-1} = \boldsymbol{mSG}_{\text{sec}} + \boldsymbol{eP}^{-1}.$$

Since $\boldsymbol{P}$ is a permutation matrix, $\text{wt}\left(\boldsymbol{eP}^{-1}\right) = \text{wt}\left(\boldsymbol{e}\right) \leq w$. Hence, Alice can use the decoding algorithm $\mathcal{D}$ to retrieve $\boldsymbol{mS}$ and thus recover $\boldsymbol{m}$ by multiplying with $\boldsymbol{S}^{-1}$.

The McEliece scheme is sketched in Table 1.2.

ALICE		BOB
Key generation • Choose a linear code $\mathscr{C}_{sec} \subseteq \mathbb{F}_q^n$ of dimension k equipped with a decoding algorithm $\mathcal{D}$ correcting up to w errors. Let $\boldsymbol{G}_{\text{sec}}$ be a $k \times n$ generator matrix of $\mathscr{C}_{\text{sec}}$. • Sample randomly $\boldsymbol{S} \in \mathbf{GL}_k(\mathbb{F}_q)$ and an $n \times n$ permutation matrix $\boldsymbol{P}$. • Compute $\boldsymbol{G}_{\text{pub}} = \boldsymbol{S}\boldsymbol{G}_{\text{sec}}\boldsymbol{P}$. **Public key:** $\mathcal{P} = (\boldsymbol{G}_{\text{pub}}, w)$ **Secret key:** $\mathcal{S} = (\boldsymbol{G}_{\text{sec}}, \boldsymbol{S}, \boldsymbol{P})$	$\xrightarrow{\mathcal{P}}$	
		Encryption • Choose a message $\boldsymbol{m} \in \mathbb{F}_q^k$ and a random error vector $\boldsymbol{e} \in \mathbb{F}_q^n$ of weight up to w. • Compute $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$.
	$\xleftarrow{\boldsymbol{c}}$	
Decryption • Compute $\boldsymbol{c}\boldsymbol{P}^{-1} = \boldsymbol{m}\boldsymbol{S}\boldsymbol{G}_{\text{sec}} + \boldsymbol{e}\boldsymbol{P}^{-1}$. • Decode $\boldsymbol{m}\boldsymbol{S}\boldsymbol{G}_{\text{sec}} + \boldsymbol{e}\boldsymbol{P}^{-1}$ using $\mathcal{D}$ and recover $\boldsymbol{m}\boldsymbol{S}$. • Compute $(\boldsymbol{m}\boldsymbol{S})\boldsymbol{S}^{-1} = \boldsymbol{m}$.		

Table 1.2: McEliece's scheme

Remark 1.11. As observed in [Cou19], this historical presentation, which dates back to McEliece's article [McE78], might lead to some misunderstanding. Primarily because for most of the code families $\mathcal{F}$, if $\mathscr{C} \in \mathcal{F}$, then also the image of $\mathscr{C}$ through a permutation of coordinates is inside the same family $\mathcal{F}$. This is the case for instance of Reed-Solomon codes and their subfield subcodes (alternant and Goppa codes). In other words, specifying $\boldsymbol{P}$ mistakenly prompts that all the orbits of $\mathcal{F}$ with respect to the symmetric group have a canonical representative. On the other side, the matrix $\boldsymbol{S}$ suggests that codes always have a canonical generator matrix and that from the latter a decoding algorithm can be reconstructed. However, this is often not the case. Our attack on high-rate alternant codes (see Chapter 4) is illustrative of these misconceptions. We do not aim to recover $\boldsymbol{S}$ and $\boldsymbol{P}$ but rather to find a secret s (not necessarily unique) which describes the code structure. Despite sticking to the classical description, we therefore deem it appropriate to outline the scheme from an alternative viewpoint.

- $\mathcal{F}$ is a family of $[n, k]$ codes.

- $\mathcal{S}$ is a set of *secrets*.
- $\mathscr{C}: \mathcal{S} \to \mathcal{F}$ is a surjective map sending secrets into codes: $s \mapsto \mathscr{C}(s)$.
- $\mathscr{C}(s)$ is equipped with a decoding algorithm $\mathcal{D}(s)$, for any $s \in \mathcal{S}$.
- The secret key is some $s \in \mathcal{S}$, while the public key is $(\boldsymbol{G}, w)$, where $\boldsymbol{G}$ is a generator matrix of $\mathscr{C}(s)$.
- The message $\boldsymbol{m} \in \mathbb{F}_q^k$ is encrypted into $\boldsymbol{mG} + \boldsymbol{e}$, where $\boldsymbol{e} \in \mathbb{F}_q^n$ is sampled randomly among vectors of weight at most w.
- The message $\boldsymbol{m}$ is recovered from $\boldsymbol{mG} + \boldsymbol{e}$ by applying $\mathcal{D}(s)$.

1.2.6 Niederreiter's scheme

The Niederreiter framework [Nie86] is the dual counterpart of the McEliece one. Indeed the parity-check matrix of a code is used instead of the generator matrix, hence it refers to Problem 1.2 rather than Problem 1.1. The original proposal built upon GRS codes, and this has been broken after 6 years by Sidelnikov and Shestakov attack [SS92] and then again in [Cou+13], [Cou+14] with different techniques. However, the weakness of the scheme could be always traced back to the family of GRS codes. In other words, it is not an inherent flaw of the framework, the latter being provably equivalent to the McEliece one [LDW94]. As a matter of fact, the Niederreiter version, instantiated with binary Goppa codes, is the one used in Classic McEliece [Alb+20]. The latter is an evolution of the Niederreiter scheme, designed to achieve indistiguishability under adaptive chosen ciphertext attack (IND-CCA2) thanks to a well-known tight conversion from one-way chosen plaintext attack (OW-CPA) security, and currently taking part at the fourth round of NIST competition. Again, and even more so in light of the above, we are going to give a description of Niederreiter's scheme that is unfettered by any specific class of codes.

As in the McEliece scheme, Alice chooses an $[n, k]$ code $\mathscr{C}_{\text{sec}} \in \mathbb{F}_q^n$ equipped with a decoding algorithm $\mathcal{D}$ that can efficiently decode up to w errors. She also applies a scramble, this time to a parity-check matrix $\boldsymbol{H}_{\text{sec}}$, i.e. she samples at random a matrix $\boldsymbol{S} \in \mathbf{GL}_{n-k}(\mathbb{F}_q)$ and an $n \times n$ permutation matrix $\boldsymbol{P}$ and computes $\boldsymbol{H}_{\text{pub}} = \boldsymbol{S}\boldsymbol{H}_{\text{sec}}\boldsymbol{P}$, which is the parity-check matrix of a code $\mathscr{C}_{\text{pub}} \in \mathbb{F}_q^n$. The secret is then represented by $\boldsymbol{H}_{\text{sec}}, \boldsymbol{S}$ and $\boldsymbol{P}$, while Alice publishes $\boldsymbol{H}_{\text{pub}}$ (and the value w).

Bob can encrypt a message $\boldsymbol{m} \in \mathbb{F}_q^n$ of weight up to w as its syndrome through the parity-check matrix $\boldsymbol{H}_{\text{pub}}$. Hence, the cipher is given by $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{H}_{\text{pub}}^T$. Recall that, differently from McEliece's scheme, no error vector is added, but the message itself has the weight constraint. Therefore a generic message needs to be mapped into the set of weight-$\leq w$ vectors through a function $\phi_{n,w}: \mathbb{F}_q^l \mapsto \mathcal{W}_{n,w}$, where $\mathcal{W}_{n,w} \stackrel{\text{def}}{=} \{\boldsymbol{e} \in \mathbb{F}_q^n \mid \text{wt}(\boldsymbol{e}) \leq w\}$ and $l \stackrel{\text{def}}{=} \left\lfloor \log_q(|\mathcal{W}_{n,w}|) \right\rfloor$. The eavesdropper Eve must solve the problem of syndrome decoding in order to retrieve the message $\boldsymbol{m}$, but this is NP-hard and only exponential time algorithms are known to tackle it.

Alice, on the other hand, has access to the private key. Consequently, she is able to compute

$$\boldsymbol{S}^{-1}\boldsymbol{c}^T = \boldsymbol{H}_{\text{sec}}\boldsymbol{P}\boldsymbol{m}^T,$$

ALICE		BOB
Key generation		
• Choose a linear code $\mathscr{C}_{\text{sec}} \subseteq \mathbb{F}_q^n$ of dimension k equipped with a decoding algorithm $\mathcal{D}$ correcting up to w errors. Let $\boldsymbol{H}_{\text{sec}}$ be a $(n-k) \times n$ generator matrix of $\mathscr{C}_{\text{sec}}$. • Sample randomly $\boldsymbol{S} \in \mathbf{GL}_{n-k}(\mathbb{F}_q)$ and an $n \times n$ permutation matrix $\boldsymbol{P}$. • Compute $\boldsymbol{H}_{\text{pub}} = \boldsymbol{S}\boldsymbol{H}_{\text{sec}}\boldsymbol{P}$. **Public key:** $\mathcal{P} = (\boldsymbol{H}_{\text{pub}}, w)$ **Secret key:** $\mathcal{S} = (\boldsymbol{H}_{\text{sec}}, \boldsymbol{S}, \boldsymbol{P})$	$\xrightarrow{\mathcal{P}}$	
		Encryption
		• Choose a message $m \in \mathbb{F}_q^n$ of weight up to w.
	$\xleftarrow{\boldsymbol{c}}$	• Compute $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{H}_{\text{pub}}^T$.
Decryption		
• Compute $\boldsymbol{S}^{-1}\boldsymbol{c}^T = \boldsymbol{H}_{\text{sec}}\boldsymbol{P}\boldsymbol{m}^T$. • Decode $\boldsymbol{S}^{-1}\boldsymbol{c}^T = \boldsymbol{H}_{\text{sec}}\boldsymbol{P}\boldsymbol{m}^T$ using $\mathcal{D}$ and recover $\boldsymbol{P}\boldsymbol{m}^T$. • Compute $\boldsymbol{m} = (\boldsymbol{P}^{-1}(\boldsymbol{P}\boldsymbol{m}^T))^T$.		

Table 1.3: Niederreiter's scheme

and, since $\text{wt}\left(\boldsymbol{P}\boldsymbol{m}^T\right) = \text{wt}\left(\boldsymbol{m}^T\right) \leq w$, recover first $\boldsymbol{P}\boldsymbol{m}^T$ with the decoding algorithm $\mathcal{D}$ and finally $\boldsymbol{m} = (\boldsymbol{P}^{-1}(\boldsymbol{P}\boldsymbol{m}^T))^T$.

The Niederreiter scheme is summarized in Table 1.3.

1.2.7 Other code-based PKE frameworks and schemes

Despite not being object ot this manuscript, it is wortwhile to mention some alternatives to McEliece's and Niederreiter's schemes. Their security relies on two hypotheses. The first one is the hardness of the general decoding problem (or the syndrome decoding problem for the dual version). As already seen, this is a well-studied mathematical problem, considered intractable. The second one depends on the specific class of codes used, *e.g.* Goppa codes, and therefore could be weaker. For instance, the contributions of this manuscript essentially disclaim the second hypothesis for the choice of high-rate alternant codes offering a polynomial time attack, and severely questioned it a much more general setting for both alternant and Goppa codes through a new distinguisher.

It is therefore natural to wonder if it is possible to eliminate upstream this second

hypothesis. Alekhnovich positively answered to this question in a seminal work from 2003 [Ale03]. *Alekhnovich's scheme* relies indeed solely on the hardness of the general decoding algorithm, i.e. it comes with a security proof. There exist two main variants of the Alekhnovich cryptosystem. In both of them the attacker is required to distinguish a random vector from an erroneous codeword of a given code $\mathscr{C}$, in order to recover the message. This comes with two main drawbacks from the efficiency point of view. First of all sending n bits is needed to encrypt just one bit. Furthermore, the public key size is quadratic in the ciphertext length. The latter defect is the most difficult to work around and hence it is the one which makes the original Alekhnovich's scheme impratical.

The *quasi-cyclic framework*, introduced in [Agu+18], is inspired by Alekhnovic's scheme, as it also makes public an initial code, endowed with an efficient decoding algorithm up to some value w. It also fixes the large key-size issue from Alekhnovich's scheme, because it allows to represent the keys in a more compact way. Several modern code-based schemes are built upon this framework, among them we mention the NIST candidates HQC [Agu+21] and RQC [Agu+20].

Essentially a message is encrypted in a codeword and then a large error is added, so that the error correcting capability is not high enough to decode the received word. However, Alice can use the private key to remove part of the error. The word obtained after this step is then expected to be decodable. The framework takes its name from the fact that an auxiliary code with a random double circulant parity-check matrix is used. We recall, however, that the decryption is probabilistic and deleting part of the error does not guarantee to make the resulting word decodable. In this case we speak about *decoding failure rate* (DFR for short), i.e. the probability that the added error does not permit to retrieve the original message. The DFR is inherent to the framework and therefore the aim is to design schemes that have a decoding failure rate as low as possible. In order to prevent some kind of attacks exploiting the DFR, it becomes desirable to experimentally or theoretically upper bound it with the value $2^{-\lambda}$, where λ is the number of security bits. However, this is often a difficult task.

The security of the quasi-cyclic framework does not rely on the indistiguishability of the quasi-cyclic code, as this is publicly available. On the other hand, the syndrome decoding problem is known to be NP-hard for random code, but not for codes with a quasi-cyclic structure. In particular it is a long standing open problem to obtain search to decision reductions for this problem in the case of structured codes. Despite some progress in this direction, *e.g.* [BCD22], borrowing and adapting techniques from lattice-based cryptography, a lot of work still deserves to be devoted to this matter.

We have seen several frameworks that can be instantiated in the world of code-based PKEs. Differently from digital signatures, code-based cryptography has been the stage for many proposals, attacks, tweaks and refinements since McEliece's seminal work. Most notably, McEliece/Niederreiter framework has been instantiated with a variety of different classes of codes. We try to gather a list of some major ideas developed in the long journey of code-based cryptology, aware of the fact that this will be inevitably incomplete.

As already said, Niederreiter's original scheme relied on GRS codes, but was broken by Sidelnikov-Shestakov attack. Nevertheless, several attempts to repair the scheme were proposed [Bal+11; Bee+18; BL05; BL11; KRW19; Wie06], in order to keep the same family of algebraic codes but avoid the aforementioned attack. Indeed,

GRS codes can correct a much bigger amount of errors than binary Goppa codes, thus providing better parameters. Most of these variants however suffer from the same weakness, namely the fact that GRS codes are distinguishable from random linear codes thanks to the square code construction, which eventually has led to as many breakings [CL22; Cou+14; LR20; Wie10].

In parallel, some efforts were made in choosing families which keep some of the properties of Reed-Solomon codes, although different. For instance, RS codes can be interpreted as algebraic geometry codes over a line. The adaptation to different curves led to a new scheme with features inherited from RS codes [JM96]. Reed-Muller codes also generalize Reed-Solomon codes and their use in the McEliece frameworks has been suggested [Sid94]. Gabidulin codes are the natural analogue of RS codes for the rank metric, and GPT [GPT91], the first rank-based cryptosystem, is built from them. For all the listed variants, the supposedly hidden structure has been leaked [CMP14; MS07; Ove08].

Additionally, alternative versions of Goppa codes, aiming at more compact keys, have been pointed out. Some examples include quasi-cyclic Goppa codes [Ber+09] or quasi-dyadic Goppa codes [BLM11; MB09]. Another stratagem was to move to alphabets of larger size, as in the case of non-binary Wild Goppa codes [BLP10]. These proposals have been cryptanalyzed mostly using algebraic methods [Fau+10b; Fau+14b; FPP14] similar to the ones presented in this work. Alternatively, the Goppa structure has been used in conjunction with convolutional codes [LJ12], but then broken in [LT13].

A promising family for building efficient and secure PKEs is given by codes whose parity-check matrix is sparse. They are called *Low-Density Parity-Check* (LDPC for short) codes (introduced by Gallager [Gal63] in a context unrelated to cryptography) or *Moderate-Density Parity-Check* (MDPC for short) codes. When instead the code admits a sparse generator matrix, it is called *Low-Density Generator-Matrix* (LDGM for short) code. Since the number of non-zero positions for a dual basis is small, it is enough (in the binary case) to store them instead of the full vector of coordinates. Thus these codes benefit from a compact representation. Moreover, they are equipped with efficient probabilistic decoders. The fact that they do not have algebraic structure makes them appreciated in cryptography. However, the matrix sparsity led to severe flaws. Indeed, the permutation matrix used to mask the secret parity-check matrix representation in Niederreiter-like schemes leads to a public key that still has low density. This is a weakness that has been exploited to cryptanalyze several early proposals [MRA00; OTD08]. The idea of [BBC08] is to replace the permutation matrix with a denser transformation matrix. In this way the density of the public parity-check matrix increases providing an MDPC code. However, the MDPC matrix is not random in this case, as it is obtained from the product of two LDPC matrices. The authors of [Apo+20] took advantage of this additional structure to practically break the second round NIST candidates LEDAcrypt [Bal+19b]. In [Mis+13], instead, it has been proposed to consider directly MDPC codes, thus avoiding the weakness of LDPC codes without adding any structure to the public key. An example of this approach is the scheme BIKE [Ara+17], based on quasi-cyclic MDPC codes, which advanced to the fourth round of NIST standardization process. Its encryption algorithm can be seen as the analog in the Hamming metric of NTRU [HPS98] and they are believed to have the same security. The family of codes with sparse parity-check matrices has also a counterpart in the rank-based cryptography,

the so called *Low-Rank Parity-Check* (LRPC for short) codes [Gab+13].

1.2.8 Digital signatures: definitions and main approaches

A *digital signature* aims at verifying the *authenticity* and *integrity* of digital messages/documents. This means that if a digital signature is valid then the receiver can expect the message to be created by a legitimate sender (authenthicity) and that no one altered the message during its transmission (integrity) with very high probability. A digital signature consists of three phases:

1. key generation;
2. signing;
3. verification.

In particular a signature scheme involves two parties: a *prover*, that has to prove his identity, and a *verifier* which verifies the prover's identity. The prover first constructs a secret key $\mathcal{S}$ and a public one $\mathcal{P}$, sharing the latter. Then, using $\mathcal{S}$, he creates a signature $\boldsymbol{s}$ for the message $\boldsymbol{m}$ that he wants to transmit and appends the former to the latter. Thus the verifier can read the message $\boldsymbol{m}$ and ensure the prover legitimacy by checking that the signature is consistent with the message and the public key $\mathcal{P}$. Regarding the notion of integrity, we introduce a third party, called *impersonator*, who tries to act as a prover and to deceive the verifier, without however knowing the secret key $\mathcal{S}$. If he alters the message he also needs to modify the signature, but the verifier will not discover the tampering with very low probability, called *cheating probability* or *soundness error*, using again $\mathcal{P}$.

The existence of code-based digital signatures has remained an open problem for long time. The CFS signature [CFS01] from Courtois, Finiasz and Sendrier, which is of particular interest in this manuscript, represents a forerunner in this context. However its construction requires unacceptably large public key that prevents it from being competitive against other signature schemes. Nevertheless, it has remained essentially unbroken for more than 20 years. In recent times, many code-based signatures have been proposed with better performances. In order to depict CFS and mention modern alternatives, we need to recall the main approaches in which these signatures are classified, though we remark that they do not cover the totality of schemes.

Fiat-Shamir

Fiat and Shamir introduced a method to turn an *identification scheme* into a signature in 1986 [FS87]. An identification scheme is a protocol where the prover wants to convince the verifier that he knows a secret. It generally has the following structure:

- The prover sends to the verifier a piece of information, called *commitment*;
- The verifier sends to the prover a *challenge*;
- The prover returns an answer to the challenge;

- The verifier checks whether the answer is consistent with the commitment and the challenge.

The Fiat-Shamir transform makes use of the identification scheme protocol but allows to avoid the interaction with the verifier. The challenge is indeed directly derived by applying a hash function to the commitment. Consequently, the prover does not choose the value for the challenge.

Hash and sign

The *hash and sign* approach is the one used in, *e.g.*, CFS. In general, a public-key encryption function can be transformed into a signature for a message $\boldsymbol{m}$ through the following steps:

1. Hash $\boldsymbol{m}$ using some hash function $\mathcal{H}$;
2. Decrypt $\mathcal{H}(\boldsymbol{m})$ as if it were a ciphertext of the PKE;
3. Append the decryption to the message.

The verifier just needs to apply the public encryption function to the message and check that the result coincides with the appended signature. While this can be done by everyone, forging a signature requires to invert the trapdoor function of the corresponding PKE. The main difficulty behind hash and sign protocols is to find instances for which a solution $\boldsymbol{e}$ for the syndrome decoding problem $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}$ can be found for a non-negligible proportion of all vectors $\boldsymbol{s}$ and an efficient decoding algorithm exists at the same time. We will see CFS signature achieves this result.

MPC-in-the-head

The *MPC-in-the-head* approach is a more recent paradigm [Ish+07] which borrows from multi-party computation (MPC) protocols to build zero-knowlegde proofs. Initially considered of theoretical interest only, it became a practical alternative thanks to Picnic post-quantum signature [Cha+20]. By reducing the soundness interest, it allows to significantly decrease the signatures size with respect to early Fiat-Shamir based proposals, such as Stern's protocol.

1.2.8.1 CFS signature

As for Niederreiter scheme, a digital signature can be built exploiting the hardness of syndrome decoding problem, thanks to the hash and sign framework. However, the construction requires to find an error vector with weight $w \leq \frac{d-1}{2}$, corresponding to a syndrome. While it is impossible to hash into decodable syndromes only, the prover can hash into the space of all syndromes instead. Therefore the document is not always decodable. There exist multiple workarounds to circumvent this. The solution we are going to describe consists in appending a counter to the document and repeat until the hashed value is decodable. So the counter is also part of the signature.

Still this technique is expensive and to make it practical the underlying code must fullfil some requirements. First of all the density of decodable syndromes in

the set of all syndromes must be high enough. Secondly, the covering radius must be close to the decoding capability. Hence, for almost every known families of codes, this construction is precluded. A remarkable counterexample is given by high-rate Goppa codes, for which such errors exist for a non-negligible proportion of syndromes. CFS scheme [CFS01], proposed by Courtois, Finiasz and Sendrier in 2001, builds exactly upon these codes. Because of the special emphasis this manuscript has on (high-rate) Goppa codes and McEliece-like constructions, we are going to detail the workflow of CFS.

During the key generation phase, the prover chooses a parity-check matrix $\boldsymbol{H}_{sec} \in \mathbb{F}_2^{(n-k)\times n}$ of a binary code (in practice a high-rate Goppa code), equipped with a decoding algorithm $\mathcal{D}$ that can efficiently decode up to w errors. Then he samples a random matrix $\boldsymbol{S} \in \mathbf{GL}_{n-k}(\mathbb{F}_2)$ and a random $n \times n$ permutation matrix $\boldsymbol{P}$ and computes $\boldsymbol{H}_{pub} = \boldsymbol{S}\boldsymbol{H}_{sec}\boldsymbol{P}$. The secret key is then given by $\mathcal{S} = (\boldsymbol{H}_{\text{sec}}, \boldsymbol{S}, \boldsymbol{P})$ and the shared public key by $\mathcal{P} = (\boldsymbol{H}_{pub}, w)$.

Regarding the signing, given a message $\boldsymbol{m}$, the prover first hashes $\boldsymbol{m}$ with a given hash function $\mathcal{H}$. Then he iterates over the integers until he finds $i \in \mathbb{N}$ such that $m_i \overset{\text{def}}{=} \mathcal{H}([\mathcal{H}(\boldsymbol{m}) \mid i])$ (where $[\cdot \mid \cdot]$ denotes a padding) is decodable, i.e. there exists an error vector $\boldsymbol{e}$ such that $\text{wt}\,(\boldsymbol{e}) \leq w$ and $\boldsymbol{e}\boldsymbol{H}_{pub}^T = \boldsymbol{m}_i$. This can be done easily by the prover, as he knows the secret code representation $\boldsymbol{H}_{pub}$ and therefore can retrieve $\boldsymbol{e}\boldsymbol{P}^T$ from the relation $\boldsymbol{e}\boldsymbol{P}^T\boldsymbol{H}_{sec}^T = \boldsymbol{m}_i(\boldsymbol{S}^T)^{-1}$. The signature for $\boldsymbol{m}$ thus becomes $[\boldsymbol{e} \mid i]$.

For the verifier it remains to check whether $\boldsymbol{e}\boldsymbol{H}_{pub}^T = \mathcal{H}([\mathcal{H}(\boldsymbol{m}) \mid i])$. If the answer is positive he accepts the signature as valid, otherwise he does not.

The hash procedure needs to be repeated until a proper error vector is found. The probability of success equals the ratio between the number of decodable syndromes $\sum_{i=0}^{w}\binom{n}{i}$ and the number of all syndromes 2^{n-k}, thus it can estimated, for a full length Goppa code, with

$$\frac{\sum_{i=0}^{w}\binom{n}{i}}{2^{n-k}} = \frac{\sum_{i=0}^{w}\binom{n}{i}}{2^{mw}} \approx \frac{\binom{n}{w}}{2^{mw}} = \frac{\binom{2^m}{w}}{2^{mw}} \approx \frac{\frac{(2^m)^w}{w!}}{2^{mw}} \approx \frac{1}{w!}.$$

The parameters originally suggested for binary Goppa codes to be used in CFS are: weight error $w = 9$, extension degree $m = 16$ and length $n = 2^{16}$. But these values are too low to prevent a generalized birthday's paradox attack. Larger parameters can be chosen, but since these do not scale well, they soon lead to a huge public key. In this manuscript, we will focus instead on a different weakness of algebraic nature, namely the distinguishability of high-rate Goppa codes.

1.2.8.2 Other code-based signatures: historical proposals and recent contributions

The first code-based scheme is due to Stern [Ste93], then improved in [Vér96]. It has the advantage of being based only on the decoding problem and the Fiat-Shamir construction. These signatures have small public keys, however the authenticity is guaranteed with only constant probability (the soundness error is 2/3 for Stern's scheme). It is therefore necessary to repeat the protocol many times to increase the mentioned probability, thus yielding large signatures (in the order of hundreds of kilobits) [AGS11]. In the case of lattices, these inherent weakness has been overcome by Lyubachevsky in [Lyu09] who managed to combine the independent

PROVER		VERIFIER
Key generation		
• Choose a linear code $\mathscr{C}_{\text{sec}} \subseteq \mathbb{F}_q^n$ of dimension k equipped with a decoding algorithm $\mathcal{D}$ correcting up to w errors. Let $\boldsymbol{H}_{\text{sec}}$ be a $(n-k) \times n$ generator matrix of $\mathscr{C}_{\text{sec}}$. • Sample randomly $\boldsymbol{S} \in \mathbf{GL}_{n-k}(\mathbb{F}_q)$ and an $n \times n$ permutation matrix $\boldsymbol{P}$. • Compute $\boldsymbol{H}_{\text{pub}} = \boldsymbol{S}\boldsymbol{H}_{\text{sec}}\boldsymbol{P}$. • Choose an hash function $\mathcal{H}$. **Public key:** $\mathcal{P} = (\boldsymbol{H}_{\text{pub}}, w, \mathcal{H})$ **Secret key:** $\mathcal{S} = (\boldsymbol{H}_{\text{sec}}, \boldsymbol{S}, \boldsymbol{P})$	$\xrightarrow{\mathcal{P}}$	
Signing		
• Choose a message $\boldsymbol{m}$ and hash it with $\mathcal{H}$. • For $i \in \mathbb{N}$ - Compute $\boldsymbol{m}_i = \mathcal{H}([\mathcal{H}(\boldsymbol{m}) \mid i])$. - If $\exists \boldsymbol{e} \in \mathbb{F}_q^n$ such that $\text{wt}(\boldsymbol{e}) \leq w \wedge \boldsymbol{e}\boldsymbol{H}_{pub}^T = \boldsymbol{m}_i$ then Output $[\boldsymbol{e} \mid i]$ as signature.	$\xrightarrow{[\boldsymbol{e}\mid i]}$	**Signing** • If $\boldsymbol{e}\boldsymbol{H}_{\text{pub}}^T = \mathcal{H}([\mathcal{H}(\boldsymbol{m}) \mid i])$ then accept else reject.

Table 1.4: CFS signature

binary challenges in a single challenge. As a consequence a remarkable saving in the key size is achieved. However, the same framework does not work equally well when lattices are replaced by codes. There have been several attempts in this direction, for instance the NIST proposal RaCoSS [Fuk+17] is based on a public matrix with columns formed by low-weight syndromes. After NIST call, other schemes came out: [Per18] relies on quasi-cyclic codes, in [Son+20] private key matrix rows have weight below GV bound instead of fixed weight and [LXY20] employs rejection sampling instead of trapdoors. All of them have been broken though, respectively by [Ara+21; Bal+21; SBC19; Xag18]. For now, Durandal scheme [Ara+19] seems to be the only one to be able to adapt Lyubachevsky's framework to codes. What entirely differs here is the metric: Durandal is a rank-metric scheme. The security relies indeed on a specific problem for this metric and thus needs further analysis.

Another early code-based scheme is KKS [KKS05; KKS97], with following improvements [BMS11; GS12]. However, even ignoring the cryptanalysis of [OT11], the only application they could have is as one-time signature, due to the attack contained in [COV07].

CFS is related to the conventional decoding problem, where is asked to find an error with minimum weight satisfying the syndrome equation. Other schemes used the relaxed requirement of finding an error with weight sufficiently low, but not necessarily minimal. This was done in [Bal+13] with LDGM codes, in [Gli+14] with convolutional codes and in the NIST first round candidate pqsigRM [Lee+17] with modified Reed-Muller codes. All of these have been succesfully attacked, respectively in [MP16; PT16] and in an official comment on the pqc-forum[2].

The Wave scheme [DST19] is also a recent up-and-coming signature that makes use of the hash and sign protocol and relies on the NP-complete problem of distinguishing generalized $(U, U+V)$ codes. Wave resolves a notoriously difficult issue that appears in the regime where the syndrome decoding problem $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}$ has several solutions. In this case, the decoding algorithm may output a particular solution that is not random among the set of solutions. This can leak some information and several cryptosystems have been broken exploiting it. To avoid this serious weakness in hash-and-sign signature schemes, it is necessary to use one-way trapdoor functions such that the inversion algorithm samples from all possible preimages according to an appropriate distribution that can not be statistically distinguished. The first collection of trapdoor functions that are many-to-one (i.e. every output has several preimages) was found in the context of lattice-based cryptography in [GPV08]. The strategy followed in [GPV08] has been adapted to WAVE. Indeed, a tight security reduction has been given, replacing the lattice assumption used in [GPV08] with the difficulty of the decoding problem.

With the exception of Durandal, all the systems listed so far fall in the setting of Hamming metric. However, some other rank metric schemes are worthy of a mention. Among them, there are the RankSign [Gab+14] submitted to NIST process but then broken [DT18], and MURAVE [LT20].

With regards to MPC-in-the-head approach, some zero-knowledge schemes have been proposed for the syndrome decoding problem [Bid+22; FJR21; FJR22; GPS22]. In particular a soundness error $1/N$ is achieved (instead of a constant one as in

[2] available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/official-comments/pqsigRM-official-comment.pdf

Stern's scheme). As a consequence, a dramatic drop of the signature size is obtained.

Finally, a digital signature which does not fall in the above classification is LESS-FM [Bar+21], which instead relies on the *Linear Code Equivalence problem*.

To summarize, some technical issues penalized the early development of code-based signature schemes and prevented them to be competitive with respect to, *e.g.*, lattices. It is not a coincidence that no signatures based on error correcting codes have passed the first round of NIST competition. However, in recent years, and in particular after the starting of the standardization process, many new constructions and proposals came out. Some have already been broken but other are very promising and may have a central role in the upcoming NIST call for digital signatures schemes.

1.2.9 Cryptanalysis on code-based schemes

In this subsection we review the main strategies to attack cryptosystems based on codes. We can distinguish two fundamental approaches:

1. Message recovery attacks;
2. Key recovery attacks.

Such distinction is actually more general and applies to other kind of cryptosystems, too. The first ones aim at inverting the encryption without discovering a trapdoor. The second consist in determining a private key (either Alice's or an equivalent one) and use it to decipher any message in the same way as Alice would.

Regarding code-based schemes, the first class of attacks is mainly represented by a family of combinatorial algorithms which go under the name of *Information Set Decoding* (ISD for short) algorithms. Instead, key recovery attacks tipically exploit properties that are specific to the class of codes through which a system is instantiated.

Indeed, the McEliece framework shows that Alice needs to decode a known linear code in order to retrieve the original message $\boldsymbol{m}$ sent by Bob. This necessity raises the problem of choosing a linear code that is equipped with an efficient decoding algorithm. In light of Section 1.1, such a code is therefore not random, but it should arguably have some structure. For instance, the original proposal from McEliece uses binary Goppa codes, which can be decoded with Patterson's algorithm. Thus, McEliece and Niederreiter-like cryptosystems actually rely on two assumptions:

1. Decoding a random linear code is hard;
2. Distinguishing the public code from a random code is unfeasible.

The first assumption clearly does not depend on the code chosen. *Non-structural attacks* simply assume that the public code is random. On the other side, the goal of *structural attacks* is to benefit from the second assumption being false in some cases. In this setting, we say that there exists a *distinguisher* between the chosen code and a random one. Although a distinguisher does not necessarily implies a flaw, it usually diminishes the confidence in a cryptosystems because an attacker could be able to recover the secret key thanks to the particular structure of the code. So the distinction "message recovery vs. key recovery" attacks almost coincide with "non-structural vs. structural" attacks.

Algorithm	$\max_{0\leq R\leq 1}\alpha(R, w_{GV}, 2)$
Prange [Pra62] 1962	0.1207
Stern [Ste88] 1988	0.1166
Dumer [Dum91] 1991	0.1164
MMT [MMT11] 2011	0.1114
BJMM [Bec+12] 2012	0.1019
MO [MO15] 2015	0.0966

Table 1.5: Comparison of ISD workfactor exponents for several algorithms.

Addressing the topic of structural attacks briefly is impossible, since they are usually extremely specific to the scheme and to the family of codes employed. We therefore leave to the next chapters a more in-depth explanation of some of these techniques, namely square-code based attacks on GRS codes related families and algebraic attacks. We proceed instead with a very high-level overview of ISD algorithms.

Information set decoding.

Non-structural attacks are clearly more general, as they could work regardless of the chosen code. For this reason, ISD algorithms, the primary representatives of the message recovery approach, are used to design the parameters of code-based schemes. In other words, the key size must be selected in a way to thwart these attacks, depending on the desired security level. Therefore they must be considered attacks in a broad sense only. Indeed, although the key size are currently chosen in order to counter their threat their average complexity is exponential, both in the classical and quantum settings. More precisely it can be estimated by the *workfactor*

$$q^{(\alpha(R,w,q)+o(1))n},$$

where $\alpha(R, w, q)$ is a function depending on the code rate R, the relative distance w and the field size q (and on the specific algorithm). Not only these kind of algorithms remained exponential but the exponent $\alpha(R, w, q)$ barely improved over approximately 60 years. Table 1.2.9 outlines how the asymptotic exponent decreased since the seminal work of Prange in 1962 [Pra62], for Gilbert-Varshamov relative distance ω_{GV} and in the binary case $q = 2$.

The exponent can be further improved if we have access to a quantum computer. Indeed, in the quantum setting we can exploit *Grover's algorithm*, which provides a speed up for the problem of unstructured search, a very common subroutine appering in ISD algorithms too. However, the speed up is at most quadratic, therefore even the best quantum ISD algorithm [KT17] can not bring the workfactor exponent below half the classical one.

We now provide the basic idea behind ISD algorithm, describing Prange's algorithm. For more information the reader can refer to *e.g.* [Deb19], [Bal+19a].

Let $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$ be a parity-check matrix of a code $\mathscr{C}$, $\boldsymbol{s} \in \mathbb{F}_q^{n-k}$ a syndrome and $w \in \mathbb{N}$. The goal is to find an error vector $\boldsymbol{e} \in \mathbb{F}_q^n$ such that $\mathrm{wt}(\boldsymbol{e}) = w$ and $\boldsymbol{e}\boldsymbol{H}^T = \boldsymbol{s}$. Therefore we have $n - k$ linear equations (one for each row of $\boldsymbol{H}$) and n unknowns (the entries of $\boldsymbol{e}$). One possible strategy to solve this underdetermined

linear system is to make $n-(n-k)=k$ bets. In other words, a subset of positions $I \subset [\![1,n]\!]$, $|I|=k$, is chosen and the corresponding coordinates of $\boldsymbol{e}$ are guessed. Since $\boldsymbol{e}$ has small weight, the most probable outcome is that the restriction $\boldsymbol{e}_I$ is the null vector. Thus, we guess $\boldsymbol{e}_I = \boldsymbol{0}$ and solve the linear system $\boldsymbol{e}_{\check{I}}\boldsymbol{H}_{\check{I}}^T = \boldsymbol{s}$ of $n-k$ equations in $n-k$ unknowns. Here $\boldsymbol{e}_{\check{I}}$ denotes the vector $\boldsymbol{e}$ restricted to $[\![1,n]\!] \setminus I$ and $\boldsymbol{H}_{\check{I}}$ the submatrix of $\boldsymbol{H}$ with only the columns indexed by $[\![1,n]\!] \setminus I$. The algorithm iterates until a good I is found.

The cost of the algorithm is given by the product of the cost of one iteration times the inverse of its success probability. Indeed, the algorithm is expected to iterate several times since finding an information set I for which a solution is such that $\boldsymbol{e}_I = \boldsymbol{0}$ is not a likely event. Since $|[\![1,n]\!] \setminus I| = n-k$, the number of vectors of weight w with support in $[\![1,n]\!] \setminus I$ is $\binom{n-k}{w}(q-1)^w$. On the other hand there are $\binom{n}{w}(q-1)^w$ vectors of weight w without other constraints. Therefore the success probability of one iteration is

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

The cost of one iteration is dominated by solving the linear system, which can be perfomed with $(n-k)^2 n$ operations over $\mathbb{F}_q$, even though more involved algorithms can be used for this routine. Overall, the total cost of Prange's algorithm is

$$(n-k)^2 n \frac{\binom{n}{\omega}}{\binom{n-k}{\omega}}$$

operations over $\mathbb{F}_q$. With respect to brute-force attacks, where on the opposite the information set is fixed and the algorithm iterates over all the possible weight distribution, ISD method clearly achieves better results.

Remark 1.12. For some instances we can predict that Prange's algorithm is not going to find a suitable error vector, regardless of the number of iterations. For instance suppose that the intersection of all information sets is not empty. Then, if a position in this intersection belongs to the support of the sought error vector, the latter can not be found using Prange's algorithm. The mentioned case, however, is highly unlikely, at least for random codes.

Modern ISD algorithms improve Prange's basic version by assuming a more likely weight distribution of the error vector. For instance, they search for a subset $I \subset [\![1,n]\!]$, such that $\mathrm{wt}(\boldsymbol{e}_I) = p \ll w$. In this way better trade-offs between number of iterations and cost of one iterations, eventually optimizing the involved parameters, can be obtained.

It's appropriate to remember that there exist variants of ISD algorithms designed for metrics different from Hamming, such as the Lee metric [HW20], [CDE21] or the rank metric [CS96; OJ02]. In the latter case, however, recent algebraic methods [Bar+20a; Bar+20b] have outperformed the combinatorial techniques.

There exist also alternative strategies to decode random linear codes. Among them we have generalized birthday algorithm, first proposed by Wagner [Wag02], which is especially suitable for decoding problems with many solutions, or statistical decoding [Jab01], which recently managed to outperform ISD algorithms for low rates [Car+22].

Structural attacks

The most efficient key-recovery attack for the original McEliece was given in [LS01] and essentially lies in guessing the Goppa polynomial and the support up to permutation, trying all the possibilities. The verification step consists in solving a code equivalence problem which is often easy with the help of the support splitting algorithm [Sen00]. However, the overall complexity is exponential and the exponent is even bigger than the one obtained from ISD algorithms. This is why message recovery attacks are considered as the main threat against McEliece-Goppa, and consequently the parameters are chosen in order to thwart them.

More in general, it is not easy to describe which algorithms fall into the category of structural attacks, as they must be typically designed ad hoc, depending precisely on the structure of the underlying family of codes. A historical representative of key-recovery attacks that work on a specific class of codes is the Sidelnikov-Shestakov attack [SS92] on GRS codes. Besides that, we can still find some similarities between examples of these attacks and isolate two classes that have successfully pushed the state-of-the-art of this kind of cryptanalysis:

- *Square code attacks:* we have already defined Schur's product and square of codes in Section 1.1.5. The key idea is that some families of codes have a square code of dimension smaller than what is expected with high-probability from random codes (see Theorems 1.3 and 1.4). In other words, these attacks belong to the category of *distinguisher attacks*. We already showed that this is what happens for GRS codes, but the list is much longer. Among the families of codes that suffer from the square-code distinguisher, in addition to GRS codes and some variants [Cou+14], we can mention: GRS subcodes [Wie10], wild Goppa codes over quadratic extensions [COT14b], algebraic geometry codes [CMP14; CMP17], GRS codes with random entries [CLT19], subspace subcodes of GRS codes [CL22].

- *Algebraic cryptanalysis:* the key-recovery problem is modeled by an algebraic system: finding its solutions implies breaking the scheme. This is typically achieved by Gröbner bases and linearization techniques, which will be introduced in the next subsection. This kind of analysis received a lot of attention in the last years in several domains. For instance, concerning code-based cryptography, it recently completely outperformed combinatorial attacks in rank-metric, where it now defines benchmark standards. On another topic, the McEliece scheme's variants based on structured Goppa codes witnessed some important developments in this sense. For instance, quasi-cyclic and quasi-dyadic Goppa codes [Fau+10b] or Wild Goppa codes [FPP14] have been attacked by solving multivariate polynomial systems.

We will show how a combination of these two macro-categories leads to an attack on unstructured high-rate alternant codes. We postpone to Chapter 4 a deeper treatment, when all the necessary background will have been introduced.

1.3 Gröbner Bases

Among the mathematical problems considered hard, solving a system of *multivariate* polynomial equations has always received a lot of attention from mathematicians, both in ancient and modern times. The problem can be reinterpreted in terms of finding a "good" representation of a polynomial ideal. Even the easier *ideal membership problem* put a strain on algebraists and some believed it was not even decidable.

Gröbner bases represent a powerful tool in computer algebra to study polynomials ideals. They have been introduced by Buchberger in 1965 in his Ph.D. thesis [Buc65] and take the name from his advisor. Several questions on the algebra of polynomials ideals may arise from applications. As announced above, two of the main problems that can be addressed through Gröbner basis are the following.

Problem 1.8 (Solving a polynomial system)**.** *Given a set of (multivariate) polynomials $f_i \in \mathbb{K}[x_1, \ldots, x_n]$, $1 \leq i \leq m$, find all the common solutions of the system*

$$\begin{cases} f_1(x_1, \cdots, x_n) &= 0 \\ & \vdots \\ f_m(x_1, \cdots, x_n) &= 0 \end{cases}. \tag{1.5}$$

From a geometric point of view, this is related to determining the affine variety

$$\boldsymbol{V}(f_1, \ldots, f_m) = \boldsymbol{V}_{\overline{\mathbb{K}}}(f_1, \ldots, f_m) = \{\boldsymbol{a} \in \overline{\mathbb{K}}^n \mid \forall i \in [\![1, m]\!],\ f_i(\boldsymbol{a}) = 0\},$$

where $\overline{\mathbb{K}}$ denotes the algebraic closure of $\mathbb{K}$.

Problem 1.9 (Ideal Membership problem)**.** *Given an ideal $\mathcal{I} = \langle f_1, \ldots, f_m \rangle \subseteq \mathbb{K}[x_1, \ldots, x_n]$ and a polynomial $f \in \mathbb{K}[x_1, \ldots, x_n]$, determine whether $f \in \mathcal{I}$.*

From a geometric point of view, this is related to determining whether the algebraic variety $\boldsymbol{V}(f_1, \ldots, f_m)$ is contained in $\boldsymbol{V}(f)$.

For readability reasons, in this section we will often denote the multivariate polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ with $\mathbb{K}[\boldsymbol{x}]$, where $\boldsymbol{x} = (x_1, \ldots, x_n)$, implying that there are n unknowns.

Remark 1.13. In both problems, the geometric counterpart refers to affine variety with respect to the algebraic closure of a field $\mathbb{K}$. For several applications in coding theory and cryptology, our interest is restricted to zeros lying in the finite field $\mathbb{K}$ itself (or in some subfield/finite extension). Since finite fields are not algebraically closed, we will rather refer to the algebraic variety

$$\boldsymbol{V}_{\mathbb{K}}(f_1, \ldots, f_m) = \{\boldsymbol{a} \in \mathbb{K}^n \mid \forall i \in [\![1, m]\!],\ f_i(\boldsymbol{a}) = 0\}.$$

The computation of a Gröbner basis can be interpreted as a generalization of two elementary algorithms of Algebra. The first one is the Gaussian elimination (or row-reduction) algorithm, mainly used for solving linear systems. The problem of solving polynomial systems is then a generalization of the former, where the polynomial equations can have a larger degree. The second algorithm is the univariate polynomial division in $\mathbb{K}[x]$. Thanks to it, we can for instance figure out if a univariate polynomial

belongs to an ideal. Again, the ideal membership problem generalizes this question to multivariate polynomial rings.

In both cases, the notion of *ordering of terms* plays a central role. For Gaussian elimination, we need to specify an order for the variables. According to it, the matrix echelonized by the algorithm is determined by a specific permutation of columns. For univariate polynomial division, the term ordering is implicit: the higher the degree the larger the monomial term.

1.3.1 Monomial orderings

Ordering of terms takes on even greater importance when we deal with multivariate non-linear polynomials. This is a delicate problem because we want to preserve some desirable properties. In particular, we would like to order the monomials appearing in any possible polynomial, i.e. we want a *total order*. Moreover, it has to behave naturally with respect to the multiplication. These requirements lead to the following definition.

Definition 1.21. A **monomial ordering** $>_{mon}$ on $\mathbb{K}[x_1, \ldots, x_n]$ is a relation on $\mathbb{N}^n$ such that:

1. $>_{mon}$ is a *total ordering* on $\mathbb{N}^n$, i.e. for any $\alpha, \beta \in \mathbb{N}^n$, exactly one of the following occurs:
$$\alpha >_{mon} \beta, \quad \alpha <_{mon} \beta, \quad \alpha = \beta.$$
2. For any $\alpha, \beta, \gamma \in \mathbb{N}^n$, $\alpha >_{mon} \beta \Rightarrow \alpha + \gamma >_{mon} \beta + \gamma$.
3. $>_{mon}$ is a *well-ordering* on $\mathbb{N}^n$, i.e. every non-empty subset of $\mathbb{N}^n$ has a smallest element with respect to $>_{mon}$.

We also denote with $\geq_{mon}$ the relation $\alpha \geq_{mon} \beta \overset{\text{def}}{\iff} \alpha >_{mon} \beta \vee \alpha = \beta$.

Remark 1.14. We defined an ordering as a relation $\mathbb{N}^n$, i.e. on sequences of natural numbers. So, why is it called *monomial* ordering? Given a polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$ and an element $\alpha = (\alpha_1, \ldots, \alpha_n) \in \mathbb{N}^n$, we can naturally identify

$$\mathbb{N}^n \ni \alpha \leftrightarrow \boldsymbol{x}^\alpha \overset{\text{def}}{=} x_1^{\alpha_1} \cdot \ldots x_n^{\alpha_n} \in \mathbb{K}[x_1, \ldots, x_n].$$

Hence a relation on $\mathbb{N}^n$ can be equivalently seen as a relation on the monomials of $\mathbb{K}[x_1, \ldots, x_n]$. In the following, we will likewise write $\alpha >_{mon} \beta$ or $\boldsymbol{x}^\alpha >_{mon} \boldsymbol{x}^\beta$ with the same meaning.

The usefulness of Condition 3 of Definition 1.21 can be explained by the next proposition:

Proposition 1.8. *An order relation $>$ on $\mathbb{N}^n$ is a well-ordering if and only if every strictly decreasing sequence*
$$\alpha(1) > \alpha(2) > \ldots$$
in $\mathbb{N}^n$ eventually terminates.

There exist several order relations on $\mathbb{N}^n$ that satisfy all the 3 conditions and are therefore valid monomials orderings. We now define some of the most common orderings. Some of them will be extensively used in the next chapters.

Definition 1.22 (Lexicographic order/Lex order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. We say that $\alpha >_{lex} \beta$ if there exists $i \in [\![1, n]\!]$ such that:

1. $\forall 1 \leq j < i,\ \alpha_j = \beta_j$;
2. $\alpha_i > \beta_i$.

Remark 1.15. Lex order owes its name to word ordering used in dictionaries with which the analogy is evident.

Remark 1.16. Reordering the indexes leads to different lex orders. In particular, for n variables, there exist $n!$ lex orders.

The following order is also helpful to define what is called the *grevlex order* later.

Definition 1.23 (Reverse Lexicographic order/Revlex order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. We say that $\alpha >_{revlex} \beta$ if there exists $i \in [\![1, n]\!]$ such that:

1. $\forall i < j \leq n,\ \alpha_j = \beta_j$;
2. $\alpha_i < \beta_i$.

With lex and revlex orders, a monomial of total degree 1 can be larger than another monomial of a much higher degree. In many contexts, we may want to take into account the total degree of monomials. This leads to the following two orders, but first, we denote with $|\alpha| \stackrel{\text{def}}{=} \sum_{i=1}^{n} \alpha_i$.

Definition 1.24 (Graded Lex order/Glex order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. We say that $\alpha >_{glex} \beta$ if

$$|\alpha| > |\beta| \vee (|\alpha| = |\beta| \wedge \alpha >_{lex} \beta)\,.$$

Definition 1.25 (Graded Reverse Lex order/Grevlex order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. We say that $\alpha >_{grevlex} \beta$ if

$$|\alpha| > |\beta| \vee (|\alpha| = |\beta| \wedge \alpha >_{revlex} \beta.)$$

Despite being less intuitive, there exist both theoretical and empirical evidence that grevlex order leads in many cases to the best computation complexity for a Gröbner basis. Even when a lex basis is sought, the usual practical approach is to first compute a grevlex basis and then rely on another algorithm (e.g. FGLM or the Gröbner Walk) to transit between two different bases.

It is also possible to define a class of orders where variables are split into blocks and then other orders (as the previous ones) are used on the blocks. For the sake of simplicity, we are going to give the definition in the case of two orders only.

Definition 1.26 (Block order/Elimination order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. Let $>_1$ and $>_2$ two monomial orders on $\mathbb{N}^i$ and $\mathbb{N}^{n-i}$ respectively. We say that $\boldsymbol{x}^\alpha >_{(>_1,>_2)} \boldsymbol{x}^\beta$ if

$$\boldsymbol{x}^{(\alpha_1,\ldots,\alpha_i)} >_1 \boldsymbol{x}^{(\beta_1,\ldots,\beta_i)} \vee \Big((\alpha_1, \ldots, \alpha_i) = (\beta_1, \ldots, \beta_i) \wedge \boldsymbol{x}^{(\alpha_{i+1},\ldots,\alpha_n)} >_2 \boldsymbol{x}^{(\beta_{i+1},\ldots,\beta_n)}\Big)\,.$$

The order above is of interest because it allows *elimination*, which corresponds to projection in the geometric viewpoint.

Finally, we show another class of orders, where a different *weight* is attributed to each variable.

Definition 1.27 (Weight order)**.** Let $\alpha = (\alpha_1, \ldots, \alpha_n), \beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. Let $>_{mon}$ be a monomial order and $\omega = (\omega_1, \ldots, \omega_n) \in (\mathbb{R}_{>0})^n$ such that $\omega_1, \ldots, \omega_n$ are linearly independent over $\mathbb{Q}$. We say that $\alpha >_\omega \beta$ if

$$\boldsymbol{x}^{(\omega_1\alpha_1,\ldots,\omega_n\alpha_n)} >_{mon} \boldsymbol{x}^{(\omega_1\beta_1,\ldots,\omega_n\beta_n)}.$$

It can be readily verified that all of them are well-defined monomials orderings.

We conclude the subsection by giving some terminology for multivariate polynomials.

Definition 1.28. Let $f(\boldsymbol{x}) = \sum_\alpha c_\alpha \boldsymbol{x}^\alpha \in \mathbb{K}[x_1, \ldots, x_n]$ and $>_{mon}$ a monomial order.

- The **multidegree** of f is $\text{multideg}(f) \stackrel{\text{def}}{=} \max\{\alpha \in \mathbb{N}^n \mid c_\alpha \neq 0\}$, where the maximum is taken with respect to $>_{mon}$.
- The **total degree** (or simply degree) of f is $\deg(f) \stackrel{\text{def}}{=} \max\{|\alpha| \in \mathbb{N}^n \mid c_\alpha \neq 0\}$.
- The **leading coefficient** of f is $LC(f) = c_{\text{multideg}(f)} \in \mathbb{K}$.
- The **leading monomial** of f is $LM(f) = \boldsymbol{x}^{\text{multideg}(f)}$.
- The **leading term** of f is $LT(f) = LC(f)LM(f)$.

From these definitions, it is straightforward to verify, given $f, g \in \mathbb{K}[x_1, \ldots, x_n]$, $f, g \neq 0$, that

- $LC(fg) = LC(f)LC(g)$, $LM(fg) = LM(f)LM(g)$ and $LT(fg) = LT(f)LT(g)$;
- If $f+g \neq 0$, then $LM(f+g) \leq_{mon} \max(LM(f), LM(g))$. Moreover if $LM(f) \neq LM(g)$ then $LM(f+g) = \max(LM(f), LM(g))$.

1.3.2 Polynomial reduction and Gröbner bases

In the univariate polynomial ring $\mathbb{K}[x]$ all the ideals are generated by one polynomial. In particular, $\mathcal{I}(f_1, \ldots, f_s) = \mathcal{I}(\gcd(f_1, \ldots, f_s))$ and the greatest common divisor is iteratively computed through euclidean division. How this generalizes in the case of many variables? We have already highlighted that one of the algorithms Gröbner bases techniques deal with is multivariate polynomial division. With the knowledge acquired in the previous subsection, we are now ready to introduce the *polynomial reduction* in Algorithm 1.1.

The remainder r outputted by Algorithm 1.1 is sometimes referred as a **normal form** of f modulo F and denoted by $r = \bar{f}^F$,

Remark 1.17. A normal form depends in general on how the elements of F are ordered.

Algorithm 1.1 Polynomial reduction

Input
f A polynomial.
F $= [f_1, \ldots, f_m]$ an (ordered) sequence in $\mathbb{K}[\boldsymbol{x}]$
$>_{mon}$ A monomial order.

$q_1, \ldots, q_m \leftarrow 0$
$r \leftarrow 0$
$p \leftarrow f$
while $p \neq 0$ **do**
  $i \leftarrow 1$
  $\texttt{div} \leftarrow false$
  while $i \leq m \wedge \texttt{div} = false$ **do**
    if $LT(f_i) \mid LT(p)$ **then**
      $q_i \leftarrow q_i + \frac{LT(p)}{LT(f_i)}$
      $p \leftarrow p - \frac{LT(p)}{LT(f_i)} f_i$
      $\texttt{div} \leftarrow true$
    else
      $i \leftarrow i + 1$
  if $\texttt{div} = false$ **then**
    $r \leftarrow r + LT(p)$
    $p \leftarrow p - LT(p)$
Output $q_1, \ldots, q_m, r$.

Definition 1.29. Let $\{0\} \neq \mathcal{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ be an ideal and $>_{mon}$ a monomial order. We define:

$$LT(\mathcal{I}) \stackrel{\text{def}}{=} \{t \mid \exists f \in \mathcal{I} \setminus \{0\} \text{ s.t. } LT(f) = t\}$$

We also denote by $\langle LT(\mathcal{I}) \rangle$ the ideal generated by $LT(\mathcal{I})$.

Theorem 1.6 (Hilbert basis theorem, Theorem 4 p. 77, [CLO15])**.** *Every ideal $\mathcal{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ is generated by a finite set.*

From a geometric perspective, the Hilbert basis Theorem says that the affine variety of an ideal can be defined by a finite set of polynomial equations.

Proposition 1.9 (Proposition 9 p. 81, [CLO15])**.** *Let $\mathcal{I} = \langle f_1, \ldots, f_s \rangle$. Then $V(\mathcal{I}) = V(f_1, \ldots, f_s)$.*

We can now define rigorously a Gröbner basis.

Definition 1.30 (Gröbner basis)**.** Let $>_{mon}$ be a monomial order on the polynomial ring $\mathbb{K}[x_1, \ldots, x_n]$. A finite set $G = \{g_1, \ldots, g_s\}$ of an ideal $\{0\} \neq \mathcal{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ is a **Gröbner basis** for the order $>_{mon}$ if

$$\langle LT(g_1), \ldots, LT(g_s) \rangle = \langle LT(\mathcal{I}) \rangle .$$

Equivalently, G is a Gröbner basis if for every $f \in \mathcal{I}$, there exists $1 \leq i \leq s$ such that $LT(g_i) \mid LT(f)$. As a consequence of Theorem 1.6, we have

Corollary 1.1 (Corollary 6 p. 78, [CLO15]). *Every ideal $\mathcal{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ has a Gröbner basis with respect to a fixed monomial order $>_{mon}$. Moreover, every Gröbner basis of $\mathcal{I}$ is a basis of $\mathcal{I}$.*

Remark 1.18. From now on, a monomial ordering is always implicitly fixed.

Even though not all the bases of $\mathcal{I}$ satisfy the conditions for being a Gröbner basis, the latter is not unique, even for a fixed monomial order. On the other hand and differently from the remainder with respect to generic ordered sets, the normal form can be proven to be unique. In other words, if G is a Gröbner basis, a normal form of f modulo G does not depend on the order of the basis elements. Thus, the notion of Gröbner basis permits to unambiguously extend the division remainder to the multivariate setting.

Furthermore, we have

Corollary 1.2 (Corollary 2 p. 84, [CLO15]). *Let $G = \{g_1, \ldots, g_s\}$ be a Gröbner basis for an ideal $\mathcal{I} \subseteq \mathbb{K}[x_1, \ldots, x_n]$. A polynomial $f \in \mathbb{K}[x_1, \ldots, x_n]$ lies in $\mathcal{I}$ if and only if the remainder on division by f by G is 0.*

Corollary 1.2 shows one of the applications of Gröbner bases we have anticipated. Indeed, if a Gröbner basis is known, this result allows to solve the ideal membership problem by computing a normal form and checking whether it is the null polynomial or not.

1.3.3 Buchberger's algorithm: a first method to compute Gröbner bases

The following definition is a key ingredient for the original Buchberger's algorithm.

Definition 1.31 (S-polynomial). Let $f, g \in \mathbb{K}[x_1, \ldots, x_n]$ be nonzero polynomials and $\boldsymbol{x}^\alpha = \operatorname{lcm}(LM(f), LM(g))$. The **S-polynomial** of f and g is defined as

$$S(f, g) \stackrel{\text{def}}{=} \frac{\boldsymbol{x}^\alpha}{LT(f)} f - \frac{\boldsymbol{x}^\alpha}{LT(g)} g.$$

The S-polynomial is essentially a polynomial combination of f and g which ensures the cancellation of leading terms, a step at the core of any Gröbner basis algorithm. Indeed we have

$$LT\left(\frac{\boldsymbol{x}^\alpha}{LT(f)} f\right) = \frac{\boldsymbol{x}^\alpha}{LT(f)} LT(f) = \boldsymbol{x}^\alpha = \frac{\boldsymbol{x}^\alpha}{LT(g)} LT(g) = LT\left(\frac{\boldsymbol{x}^\alpha}{LT(g)} g\right).$$

S-polynomials are also used to decide whether a basis is Gröbner.

Theorem 1.7 (Buchberger's Criterion/S-pair criterion, Theorem 6 p. 86, [CLO15]). *A basis $G = \{g_1, \ldots, g_v\}$ of the polynomial ideal $\mathcal{I}$ is a Gröbner basis of $\mathcal{I}$ if and only if, for all $1 \leq i < j \leq v$, the remainder on division of $S(g_i, g_j)$ by G is 0.*

Remark 1.19. The sufficient and necessary condition from Buchberger's Criterion is sometimes taken as the definition of a Gröbner basis.

Algorithm 1.2 Buchberger's algorithm

Input
$F \quad = \{f_1, \ldots, f_s\}$ Generating set.
$G \leftarrow F$
$P \leftarrow \{\{f, g\} \mid f, g \in G, f \neq g\}$
repeat
$\{f, g\} \xleftarrow{\$} P$
$P \leftarrow P \setminus \{\{f, g\}\}$
$r \leftarrow \overline{S(f,g)}^{G}$
if $r \neq 0$ **then**
$P \leftarrow P \cup \{\{r, f\} \mid f \in G\}$
$G \leftarrow G \cup \{r\}$
until $P = \emptyset$
Output G.

We are now ready to present the constructive counterpart of Corollary 1.1, i.e. Buchberger's algorithm for computing a Gröbner basis.

Buchberger's algorithm always terminates and outputs a Gröbner basis. However, it is hard to estimate the complexity of the algorithm. In Algorithm 1.2, pairs are selected in a random order among those available at the current step. However, it is possible to order them by degree: pairs with a smaller degree are chosen first.

Definition 1.32 (d-Gröbner basis [**BW12**])**.** Let $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ be an ideal of homogeneous polynomials. A finite set $G = \{g_1, \ldots, g_s\} \subset \mathcal{I}$ is a d-**Gröbner basis** (or **Gröbner basis truncated at degree** d) of $\mathcal{I}$ if:

- $\langle g_1, \ldots, g_s \rangle = \mathcal{I}$,
- $\forall g_i, g_j \in G$ s.t. $\deg(\mathrm{lcm}(LT(g_i), LT(g_2))) \leq d$, $\overline{S(g_i, g_j)}^{G} = 0$.

Thus, in the case of a homogeneous set F, if we stop the computation of Algorithm 1.2 at degree d, we obtain a d-Gröbner basis.

We now come back to the problem of non-uniqueness of Gröbner bases. In order to guarantee uniqueness, we need to ask for additional properties.

Definition 1.33. A **reduced Gröbner basis** of a polynomial ideal $\mathcal{I}$ is a Gröbner basis of $\mathcal{I}$ such that:

- $\forall g \in G,\ LC(g) = 1$,
- $\forall g \in G,$ no monomial of g belongs to $\langle LT(G \setminus \{g\}) \rangle$.

Theorem 1.8 (Theorem 5 p. 93, [CLO15])**.** *Let $\mathcal{I}$ be a nonzero polynomial ideal. Then $\mathcal{I}$ has a reduced Gröbner basis and this is unique.*

We deduce that the reduced Gröbner basis also solves the problem of determining whether two ideals are the same. Indeed, it suffices to compute the reduced Gröbner bases of the two generating sets and check if they are equal.

1.3.4 The Macaulay matrix

We are now going to introduce Gröbner basis from a different perspective, which will ultimately clarify the link with linear algebra.

Definition 1.34 (Macaulay Matrix [Mac94])**.** Let $F = \{f_1, \dots, f_m\} \subset \mathbb{K}[\boldsymbol{x}]$ such that $\deg(f_i) = d_i$. Let d be a positive integer and $>_{mon}$ a graded monomial order. The Macaulay matrix $\mathrm{Mac}_{>_{mon}}(F, d)$ of F in degree d with respect to the order $>_{mon}$ is a matrix whose rows are each indexed by a polynomial $m_j f_i$, for all the input polynomials f_i's and all the monomials m_j's of degree $\leq d - d_i$, and whose columns are indexed by all the monomials of degree $\leq d$, sorted in decreasing order. The entry corresponding to the row indexed by $m_j f_i$ and column indexed by m_l is the coefficient of m_l in $m_j f_i$. In particular, if $m_j f_i = \sum_\alpha a_\alpha \boldsymbol{x}^\alpha$ and $m_l = \boldsymbol{x}^\beta$, then the corresponding entry of $\mathrm{Mac}_{>_{mon}}(F, d)$ is a_β:

$$\mathrm{Mac}_{>_{mon}}(F, d) = \begin{array}{c} \\ m_j f_i \end{array} \begin{array}{c} \begin{array}{c} m_l \end{array} \\ \begin{bmatrix} \vdots \\ \cdots\ a_\beta\ \cdots \\ \vdots \end{bmatrix} \end{array}.$$

Example 1.1. Let $F = \{f_1, f_2\}$, with $f_1(x_1, x_2) = x_1^2 + x_1x_2 + x_2^2 - x_1 + 2$ and $f_2(x_1, x_2) = x_1^2 - 2x_1x_2 + x_2 + 1 \in \mathbb{K}[x_1, x_2]$. Then

$$\mathrm{Mac}_{>_{mon}}(F, 3) = \begin{array}{c} \\ x_1f_1 \\ x_2f_1 \\ f_1 \\ x_1f_2 \\ x_2f_2 \\ f_2 \end{array} \begin{array}{c} \begin{array}{cccccccccc} x_1^3 & x_1^2x_2 & x_1x_2^2 & x_2^3 & x_1^2 & x_1x_2 & x_2^2 & x_1 & x_2 & 1 \end{array} \\ \begin{bmatrix} 1 & 1 & 1 & 0 & -1 & 0 & 0 & 2 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & -1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & -1 & 0 & 2 \\ 1 & -2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & -2 & 0 & 0 & 1 & 1 \end{bmatrix} \end{array}.$$

Remark 1.20. When computing a Macaulay matrix, a practical measure to reduce the complexity consists in considering the column submatrix of nonzero columns, i.e. removing the columns such that the corresponding monomial does not appear in any polynomial indexing a row. This expedient is beneficial if one wants to do operations on it (e.g. Gaussian elimination) and becomes especially relevant when the arising Macaulay matrix is sparse.

Linear combinations of $\mathrm{Mac}_{>_{mon}}(F, d)$ rows can easily be read as linear combinations of the $m_j f_i$'s, hence as polynomial combinations of the f_i's. Therefore we can translate key operations such as multivariate polynomial reduction in terms of Gaussian elimination. Indeed, Lazard showed in [Laz83] that, when the degree d is large enough, the Macaulay matrix in degree d provides a Gröbner basis. Let us first look at the homogeneous case.

Theorem 1.9 (Lazard's Theorem, [Laz83])**.** *Let $F = \{f_1, \dots, f_m\} \subset \mathbb{K}[\boldsymbol{x}]$ be a set of homogeneous polynomials. There exists a degree D such that the polynomials corresponding to the rows of the row-echelon form of $Mac_{>_{mon}}(F, d)$ form a Gröbner basis of $\mathcal{I}(F)$ with respect to $>_{mon}$.*

Lazard's Theorem clearly provides a method for computing a Gröbner basis. Indeed, we can compute D-Gröbner bases of increasing degree D (starting from $\max_i(d_i)$) until the row-echelon form is a Gröbner basis (this can be checked efficiently). Differently from Buchberger's algorithm, this strategy is also useful to estimate the cost of a Gröbner basis computation. Indeed, given the minimum degree D for which the row-echelon form of $\mathrm{Mac}_{>_{mon}}(F, D)$ is a Gröbner basis, we can roughly infer a computational cost from row-reduction algorithms.

Even in the non-homogeneous case, this result implies that there exists an integer D_0, such that for all $D \geq D_0$, a truncated D-Gröbner basis, computed through Algorithm 1.3, is a Gröbner basis. Indeed, it is possible to reduce the affine case to the homogeneous one by applying a *homogenization*. More precisely, if d is the degree of an affine polynomial $P(x_1, \ldots, x_n)$, the polynomial becomes homogeneous by introducing a new variable x_0 and applying the map

$$\begin{aligned} \phi\colon\ & \mathbb{K}[x_1, \ldots, x_n] \mapsto \mathbb{K}[x_0, x_1, \ldots, x_n] \\ & P(x_1, \ldots, x_n) \to P'(x_0, x_1, \ldots, x_n) = x_0^d P(\tfrac{x_1}{x_0}, \ldots, \tfrac{x_n}{x_0}). \end{aligned}$$

We can imagine to compute the Macaulay matrix for a homogenized system and then to *specialize* the set of polynomials corresponding to the reduced rows through the map

$$\begin{aligned} \psi\colon\ & \mathbb{K}[x_0, x_1, \ldots, x_n] \mapsto \mathbb{K}[x_1, \ldots, x_n] \\ & P'(x_0, x_1, \ldots, x_n) \to P(x_1, \ldots, x_n) = P'(1, x_1, \ldots, x_n). \end{aligned}$$

Hence, Macaulay matrices can be used to mimic a D-Gröbner basis computation for non-homogeneous systems, too. To this extent, Algorithm 1.3 represents a very simple and illustrative demonstration.

Algorithm 1.3 D-Gröbner Basis

Input

D Maximal degree.

F $= \{f_1, \cdots, f_m\} \in \mathbb{K}[\boldsymbol{x}]$ set of polynomials.

$>_{mon}$ A graded monomial order.

repeat

$F \leftarrow \mathtt{Pol}(\mathtt{EchelonForm}(\mathrm{Mac}_{>_{mon}}(F, D)))$ ▷ $\mathtt{Pol}(M)$ returns the polynomials represented by the rows of M

until $\dim_{\mathbb{K}} \langle F \rangle_{\mathbb{K}}$ has not increased. ▷ $\langle F \rangle_{\mathbb{K}}$ is the $\mathbb{K}$-vector space spanned by the elements in $F \subseteq \mathbb{K}[\boldsymbol{x}]$ seen as vectors of $\mathbb{K}$

Output F.

Remark 1.21. The space generated by F and output by Algorithm 1.3 should not be confused with the space of polynomials in $\mathcal{I}(F)$ of degree at most D. Indeed, while trivially $\langle F \rangle_{\mathbb{K}} \subseteq \mathcal{I}(F)$, it is possible that some polynomials of degree at most D in $\mathcal{I}(F)$, are not produced by the initial basis $\{f_1, \ldots, f_m\}$ with computation restricted at degree D.

The degree D from Lazard's Theorem is called the *solving degree* of the affine system F. If F is homogeneous, then D is also called *degree of regularity*. Let $M_n^{(d)}$

be the set of monomials in n variables of degree d. By elementary combinatorics we have

$$|M_n^{(d)}| = \binom{n+d-1}{d}.$$

From this, a simple complexity bound can be established

Proposition 1.10 ([BFS15])**.** *Let $F = \{f_1, \ldots, f_m\} \subset \mathbb{K}[x_1, \ldots, x_n]$ be a homogeneous system. The number of arithmetic operations in $\mathbb{K}$ needed to reduce a Macaulay matrix at degree D and thus computing a d-Gröbner basis of $\mathcal{I}(F)$ with respect to a graded monomial ordering up to degree D is upper bounded by:*

$$\mathcal{O}\left(mD\binom{n+D-1}{D}^{\omega}\right), \quad \text{as } D \to \infty,$$

where ω is the linear algebra constant.

Remark 1.22. By adding one variable and homogeneizing, any system can be brought to meet the condition of Proposition 1.10. Therefore the complexity for an affine system becomes

$$\mathcal{O}\left(mD\binom{n+D}{D}^{\omega}\right), \quad \text{as } D \to \infty.$$

The reason why Algorithm 1.3 iterates the computation of a Macaulay matrix at the same degree can be explained thanks to the notion of *degree fall*.

Definition 1.35 (Degree fall)**.** A **degree fall** for the sequence $f_1, \ldots, f_m \in \mathbb{K}[\boldsymbol{x}]$ is a polynomial combination $\sum_{i=1}^m g_i f_i \neq 0$ that satisfies

$$s \overset{\text{def}}{=} \deg \sum_{i=1}^m g_i f_i < \max_{i \in [\![1,m]\!]} \deg g_i f_i.$$

We say that $\sum_{i=1}^m g_i f_i$ is a degree fall of *degree* s.

Remark 1.23. Note that the definition loses its meaning if the system $f_1, \ldots, f_m$ is homogeneous. Indeed $\deg \sum_{i=1}^m g_i f_i < \max_{i \in [\![1,m]\!]} \deg g_i f_i$ implies $\sum_{i=1}^m g_i f_i = 0$ in this case, thus the degree fall degenerates into what is called a *syzygy*.

Definition 1.36 (Syzygy)**.** Given a sequence $f_1, \ldots, f_m$ of polynomials in $\mathbb{K}[\boldsymbol{x}]$, a **syzygy** is a sequence $g_1, \ldots, g_m$ of polynomials in $\mathbb{K}[x_1, \ldots, x_n]$ such that

$$\sum_{i=1}^m g_i f_i = 0.$$

In other words, a syzygy is the tuple of coefficients of a polynomial combination that is identically 0.

Remark 1.24. The set of all possible syzygies for $f_1, \ldots, f_m$ forms a submodule $Syz(f_1, \ldots, f_m) \subset \mathbb{K}[\boldsymbol{x}]^m$.

Notice that syzygies always exist: for any $i \neq j$ we have

$$f_i f_j - f_j f_i = 0.$$

Such relations determine the so-called *trivial syzygies*.

A degree fall occurs if all the monomials of higher degree disappear, thus revealing an element of low degree in the ideal. Because of this, in the affine setting Gröbner basis algorithms benefit from degree falls when the basis is computed with respect to a graded order. Indeed, whenever a degree fall occurs, new low-degree polynomials can be computed by multiplying it by several monomials. This potentially triggers a chain of other degree falls. Therefore, understanding them can help in designing better algorithms for specific systems.

1.3.5 Advanced Gröbner basis algorithms and solving strategies

Several algorithms build upon the Macaulay matrix, whether they call it by this name or not. For several years since its publication, cryptographers have preferred to use XL algorithm [Cou+00], which is nothing but an adapted version of Lazard's method. This phenomenon can be explained by the fact that Gröbner bases were known very little in the past by the large majority of cryptographers. The idea of XL consists of row-reducing a Macaulay matrix in such a way that a univariate polynomial appears. XL algorithm can be simulated by a Gröbner basis algorithm and extensive studies over finite fields [Ars+04] suggest that its complexity is not better. For instance, the degree that needs to be reached during the computation is never smaller than that for a Gröbner basis. Moreover, the matrix size can be huge compared to other algorithms in matricial form.

Probably the most used algorithm nowadays is F_4 introduced by Faugère in [Fau99]. This is also the algorithm implemented in the MAGMA software that is mainly used in the experiments of this thesis. The detailed description of F_4 goes beyond the scope of this presentation, but we will try to highlight the main improvements with respect to Algorithm 1.2.

In Buchberger's algorithm there are a couple of degrees of freedom:

1. choosing the pair of polynomials for which the S-polynomial is computed;
2. choosing an element from a list of reductors when reducing a polynomial with respect to a list of polynomials.

Different choices may lead to substantial discrepancies in the time complexity, and some possible strategies have been studied [Gio+91]. The manner in which F_4 tackles the first issue is by not doing any choice. In particular, instead of selecting a pair of polynomials, a subset of pairs is chosen and handled simultaneously by constructing a matrix containing all the reductions. In other words, the issue is postponed to the second step of the algorithm, where Gaussian elimination is performed on the matrix.

The so-called *normal strategy* for the selection of the subset of pairs exploits the notion of d-Gröbner bases. More precisely, at each step, the pairs of minimal degree are selected. This stratagem permits to handle and take advantage of the *degree falls* occurring during the computation in affine systems. Indeed, whenever a polynomial combination produces a new polynomial of a degree lower than the Macaulay matrix degree, the algorithm F_4 will construct matrices of lower degree in the next step, in

order to exploit the information obtained. This is a critical add-on with respect to the basic Lazard's algorithm in an affine context.

The algorithm can be further refined and improved. For instance, it is also possible to add Buchberger's criterion to achieve better performance. Finally, some specific algorithms have been dedicated to speeding up the linear algebra part [FL10].

The main efficiency issue with this algorithm is that, when row-reducing a Macaulay matrix, many linear combinations of rows are expected to become the zero row (e.g. those coming from the *trivial syzygies*). Their computation is therefore useless, as they do not provide any information. The F_5 algorithm [Fau02] is an evolution of F_4, which avoids this unnecessary computation, at least when the sequence is *regular*. This requires associating a label, called *signature*, to each polynomial, i.e. to each matrix row. Even though the asymptotic complexity does not change, this allows in practice to save a huge amount of time, around 90% of it, thus gaining a magnitude order with respect to its predecessor. In [BFS15] a complexity bound for F_5 has been established, too. Moreover, for some classes of systems (*(semi)-regular systems*), all the reductions to zero are removed. An implementation of F_5 can be found in [**F10a**], together with F_4. F_5 inspired numerous variants based on signature and the interested reader can find in [CF17] an attempt to gather and classify several advances in this direction.

As already mentioned, the Gröbner basis algorithms based on the notion of Macaulay matrix are applied to graded monomial orderings. For polynomial-solving purposes, one then needs to move to a basis with respect to a different order. For 0-dimensional ideals, this can be done through FGLM algorithm from Faugère, Gianni, Lazard and Mora [Fau+93]. Although the description of the algorithm goes beyond the scope of this manuscript, we provide its computational complexity. We first recall that the degree of an ideal $\mathcal{I} \subseteq \mathbb{K}[\boldsymbol{x}]$ is the dimension of the $\mathbb{K}$-vector space $\mathbb{K}[\boldsymbol{x}]/\mathcal{I}$, and, if $\mathcal{I}$ is 0-dimensional, $\deg(\mathcal{I})$ coincides with the number of zeros of $\mathcal{I}$, counted with multiplicity.

Theorem 1.10 (FGLM algorithm, [Fau+93])**.** *Let $\mathcal{I} \subset \mathbb{K}[\boldsymbol{x}]$ be a 0-dimensional ideal and G_1 its Gröbner basis with respect to a monomial order $>_1$. Then FGLM algorithm computes a Gröbner basis G_2 for a monomial order $>_2$ in $\mathcal{O}(n \deg(\mathcal{I})^3)$ operations on $\mathbb{K}$, given G_1 as input.*

Indeed, the algorithm corresponds to a *linear* change of basis and can be done by linear algebra. Thus, its complexity can be further reduced to $\mathcal{O}(n \deg(\mathcal{I})^\omega)$ ($2 \leq \omega < 2.3727$ is the linear algebra exponent) by using fast linear algebra [Fau+14a]. There are also other *changing order algorithms* (such as *Gröbner walks*) that are less efficient but do not require the ideal to be 0-dimensional. For our purposes, FGLM algorithm is part of the best strategy. Indeed, we can sum up the comprehensive strategy to solve an algebraic system over a finite field in Algorithm 1.5.

The purpose of computing a Gröbner basis with respect to a lexicographic order can be explained by the following result which highlights the shape of a lex basis.

Proposition 1.11 (Shape of a lex basis)**.** *Let $\mathcal{I} \subset \mathbb{K}[\boldsymbol{x}]$ be a 0-dimensional ideal.*

Then a lexicographic Gröbner basis $G_{>_{lex}}$ of $\mathcal{I}$ can be written as

$$G_{>_{lex}} = \begin{Bmatrix} g_{1,1}(x_1, \dots, x_n), \\ \vdots \\ g_{1,s_1}(x_1, \dots, x_n), \\ g_{2,1}(x_2, \dots, x_n), \\ \vdots \\ g_{n-1,s_{n-1}}(x_{n-1}, x_n), \\ g_n(x_n) \end{Bmatrix}. \tag{1.6}$$

In particular, the polynomial g_{i,s_j} depends on the last $n - i + 1$ variables only. Moreover, the smallest polynomial g_n depends on x_n only. Such a polynomial is unique up to a constant factor because in the univariate case ideals are generated by one polynomial.

Thanks to its structure, the lex basis allows to compute the variety associated with a 0-dimensional ideal. The solution coordinates can be found iteratively starting from the last. Indeed, for any element which lies in the variety, the last coordinate must be a root of the univariate polynomial g_n. Finding roots of a univariate polynomial can be done efficiently over a finite field. Once these values have been found, they can substitute the corresponding variables in the other elements of the basis. In this way, the polynomials $g_{n-1,j}$ will only depend on x_{n-1}. The process can be carried on iteratively, finding the common roots of the univariate polynomials for all the valid partial specializations and specializing again. Algorithm 1.4 formalizes this explanation. We keep the same notation as in Proposition 1.11.

Algorithm 1.4 Finding the solution from a lexicographic basis

Input

G Lexicographic basis of an ideal $\mathcal{I} \subset \mathbb{F}_q[\boldsymbol{x}]$.

Output

$V_{\mathbb{F}_{q^m}}(G)$ Algebraic variety over the extension field $\mathbb{F}_{q^m}$

$V_n \leftarrow \{a_n \in \mathbb{F}_{q^m} \mid g_n(a_n) = 0\}$ ▷ Find the roots of $g_n(x_n)$ over $\mathbb{F}_{q^m}$
for $i \in \{n-1, \dots, 1\}$ **do**
  $V_i \leftarrow \emptyset$
  for $(a_{i+1}, \dots, a_n) \in V_{i+1}$ **do**
    $T \leftarrow \mathbb{F}_{q^m}$
    for $j \in [\![1, s_i]\!]$ **do**
      if $g_{i,j}(a_i, a_{i+1}, \dots, a_n) \not\equiv 0$ **then**
        $T \leftarrow T \cap \{a_i \in \mathbb{F}_{q^m} \mid g_{i,j}(a_i, a_{i+1}, \dots, a_n) = 0\}$
      $V_i \leftarrow \{(a_i, a_{i+1}, \dots, a_n) \in \mathbb{F}_{q^m}^{n-i+1} \mid a_i \in T\}$
Output V_1

Remark 1.25. The instruction $T \leftarrow \mathbb{F}_{q^m}$ in Algorithm 1.4 is just a simple way to initialize T in this pseudocode. Of course this should and can be avoided with a simple *if* condition when q^m is big.

Remark 1.26. The extension field $\mathbb{F}_{q^m}$ is not necessarily proper, meaning that we may be interested in solutions that lie in the same field as the polynomial equations.

Algorithm 1.5 Resolution of an algebraic system over a finite field generating a 0-dimensional ideal

Input

$\mathbb{F}_q$ Finite field.

$F = \{f_1, \dots, f_m\} \in \mathbb{F}_q[\boldsymbol{x}]$ Algebraic system.

Output

$V_{\mathbb{F}_{q^m}}(F)$ Algebraic variety over the extension field $\mathbb{F}_{q^m}$

1. Compute the grevlex basis $G_{>_{grevlex}}$ using F_4 or F_5.
2. Compute the lex basis

$$G_{>_{lex}} = \{g_{1,1}, \dots g_{1,s_1}, g_{2,1}, \dots, g_{n-1,s_{n-1}}, g_n\}$$

from $G_{>_{grevlex}}$ using FGLM. ▷ see (1.6)

3. Use Algorithm 1.4 with input $G_{>_{lex}}$ to find $V_{\mathbb{F}_{q^m}}(F)$.

This will be for instance the case of the Reed-Solomon decoding problem modeling of Chapter 2. Instead, regarding the modelings for alternant and Goppa codes, we will assume that the subfield subcode structure is not trivial, i.e. that $m \geq 2$.

If instead we assume that the system has a unique solution $\bar{\boldsymbol{x}} = (\bar{x}_1, \dots, \bar{x}_n)$, then the shape of a reduced Gröbner basis further simplifies and it does not depend on the chosen order.

Proposition 1.12. *Let $\mathcal{I} \subseteq \mathbb{K}[\boldsymbol{x}]$ be a radical ideal whose variety contains a single element $\bar{\boldsymbol{x}} = (\bar{x}_1, \dots, \bar{x}_n)$. Then, for any monomial order, the reduced Gröbner basis is*

$$G = \{x_1 - \bar{x}_1, \dots, x_n - \bar{x}_n\}.$$

A proof of this result can be found for instance in [Bar04].

1.3.6 The Hilbert series

Given $d \in \mathbb{N}$, we denote $\mathbb{K}[\boldsymbol{x}]_d = \{f \in \mathbb{K}[\boldsymbol{x}] \mid f \text{ homogeneous, } \deg(f) = d\} \cup \{0\}$. This is a $\mathbb{K}$-vector space with basis the degree-d monomials in n variables $M_n^{(d)}$, thus it has dimension $\binom{n+d-1}{d}$. Moreover, given an ideal $\mathcal{I}$,

$$\mathcal{I}_d \stackrel{\text{def}}{=} \mathcal{I} \cap \mathbb{K}[x_1, \dots, x_n]_d$$

is a $\mathbb{K}$-vector space, too.

Definition 1.37 (Hilbert function)**.** Let $\mathcal{I} \subset \mathbb{K}[\boldsymbol{x}]$ be a homogeneous polynomial ideal. The **Hilbert function** of $\mathbb{K}[\boldsymbol{x}]/\mathcal{I}$ is defined as

$$HF_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(d) \stackrel{\text{def}}{=} \dim_{\mathbb{K}}(\mathbb{K}[\boldsymbol{x}]_d/\mathcal{I}_d) = \dim_{\mathbb{K}}(\mathbb{K}[\boldsymbol{x}]_d) - \dim_{\mathbb{K}}(\mathcal{I}_d).$$

Hence the dimension of $\mathcal{I}_d$ as a $\mathbb{K}$-vector space can be readily computed knowing the Hilbert function and vice versa.

Definition 1.38 (Hilbert series)**.** Let $\mathcal{I} \subset \mathbb{K}[x_1, \ldots, x_n]$ be a homogeneous polynomial ideal. The **Hilbert series** of $\mathbb{K}[\boldsymbol{x}]/\mathcal{I}$ is the generating sequence of $HF_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}$, i.e.

$$HS_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(z) \stackrel{\text{def}}{=} \sum_{d \geq 0} HF_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(d) z^d.$$

Theorem 1.11. *Given a homogeneous polynomial ideal $\mathcal{I} \subset \mathbb{K}[x_1, \ldots, x_n]$, the Hilbert series of $\mathbb{K}[\boldsymbol{x}]/\mathcal{I}$ is a rational fraction and its irreducible form can be written as*

$$\frac{N(z)}{(1-z)^d}, \quad N(1) \neq 0,$$

where $d = \dim(\mathcal{I})$, $N \in \mathbb{Z}[z]$ and $N(1)$ is the degree of $\mathcal{I}$.

Remark 1.27. If $\mathcal{I}$ is 0-dimensional, then $HS_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(z) = N(z)$ is a polynomial and $HS_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(1)$ equals the number of zeros counted with multiplicities. In fact, if $\mathcal{I}$ is homogeneous, the only solution is 0 (with some multiplicity).

Remark 1.28. The Hilbert series of $\mathbb{K}[x_1, \ldots, x_n]$ is

$$HS_{\mathbb{K}[x_1,\ldots,x_n]}(z) = \frac{1}{(1-z)^n}.$$

Theorem 1.12. *Let $\mathcal{I} \subset \mathbb{K}[\boldsymbol{x}]$ be a homogeneous polynomial ideal. There exists a degree d_0 such that, for any $d \geq d_0$, $HF_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}(d)$ is a polynomial, called **Hilbert polynomial**, and we say that d_0 is the dimension of $\mathcal{I}$, denoted* $\dim(\mathcal{I})$*. The smallest integer d_0 verifying this property is called **index of regularity**. The **dimension** of $\mathcal{I}$, denoted with* $\dim(\mathcal{I})$*, is defined as the degree of the Hilbert polynomial.*

Remark 1.29. The dimension of $\mathcal{I}$ can be defined in equivalent ways and then proved to coincide with the degree of the Hilbert polynomial. For the sake of simplicity, we do not provide here alternative definitions.

For 0-dimensional ideals, the index of regularity can be read from the Hilbert polynomial:

Proposition 1.13 (Corollary 1.66, [Spa12])**.** *Let $\mathcal{I} \subset \mathbb{K}[\boldsymbol{x}]$ be a homogeneous 0-dimensional polynomial ideal. Then*

$$i_{reg} = \deg(HS_{\mathbb{K}[\boldsymbol{x}]/\mathcal{I}}) + 1.$$

The following proposition gives an upper bound for the index of regularity:

Proposition 1.14 (Macaulay's bound, [Laz83])**.** *Let $f_1, \ldots, f_m$ be a sequence of homogeneous polynomials. If $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ is 0-dimensional, then*

$$i_{reg} \leq 1 + \sum_{i=1}^{m} (\deg(f_i) - 1).$$

For affine polynomial systems, [Bar04] generalizes the notion of index of regularity to any polynomial ideal, defining the *degree of regularity* for an affine polynomial system under the condition that the generating ideal is 0-dimensional. We denote with f^h the homogeneous part of highest degree of the affine polynomial f.

Proposition 1.15. *Let $f_1, \ldots, f_s$ be a sequence of polynomials in $\mathbb{K}[\boldsymbol{x}]$. If $\dim\left(\left\langle f_1^h, \ldots, f_s^h \right\rangle\right) = 0$, then $\dim(\langle f_1, \ldots, f_s \rangle) = 0$.*

Definition 1.39 (Degree of regularity)**.** Let $f_1, \ldots, f_s$ be a sequence of polynomials in $\mathbb{K}[\boldsymbol{x}]$. If $\dim\left(\left\langle f_1^h, \ldots, f_s^h \right\rangle\right) = 0$, we define the **degree of regularity** d_{reg} of $f_1, \ldots, f_s$ as the index of regularity of $\left\langle f_1^h, \ldots, f_s^h \right\rangle$.

For homogeneous 0-dimensional ideals, the index of regularity coincides with the degree of regularity. Thus, Macaulay's bound becomes an upper bound on the complexity of computing a Gröbner basis.

1.3.7 Regular and semi-regular sequences

We present here a class of sequences for which the Hilbert series, and consequently a complexity estimate of the Gröbner basis computation, is known in advance. For some homogeneous sequences, called *regular sequences*, the only syzygies are the trivial ones: $f_i f_j - f_j f_i$. The formal definition, which turns out to be equivalent to the characterization just mentioned, is the following.

Definition 1.40 (Regular sequence)**.** A sequence $f_1, \ldots, f_m$ of homogeneous polynomials in $\mathbb{K}[\boldsymbol{x}]$ is said a **regular sequence** if $\forall i \in [\![2, m]\!]$, $\forall g \in \mathbb{K}[\boldsymbol{x}]$,

$$g \cdot f_i \in \langle f_1, \ldots, f_{i-1} \rangle \Rightarrow g \in \langle f_1, \ldots, f_{i-1} \rangle .$$

A sequence $f_1, \ldots, f_m$ is regular if the sequence of the homogeneous parts of highest degree $f_1^h, \ldots, f_m^h$ is.

Remark 1.30. A necessary condition for $f_1, \ldots, f_m$ to be regular is that $m \leq n$.

The Hilbert series of a regular sequence is determined:

Proposition 1.16 ([Bar04])**.** *The sequence $f_1, \ldots, f_m \in \mathbb{K}[\boldsymbol{x}]$, $m \leq n$, is regular if and only if*

$$HS_{\mathbb{K}[\boldsymbol{x}]/\langle f_1, \ldots, f_m \rangle}(z) = \frac{\prod_{i=1}^{m}(1 - z^{\deg(f_i)})}{(1-z)^n}.$$

The notion of regular sequence has been extended to the overdetermined case, where there are more equations than variables, i.e. $m > n$, in [Bar+05; Bar04; BFS04]. The relevance of such generalization can be appreciated for instance in the cryptographic context, where systems arising from algebraic modelings typically have more equations than variables.

Definition 1.41 (Semi-regular sequence)**.** A sequence $f_1, \ldots, f_m$ of homogeneous polynomials in $\mathbb{K}[\boldsymbol{x}]$ is said a **semi-regular sequence** if $\forall i \in [\![2, m]\!]$, $\forall g \in \mathbb{K}[\boldsymbol{x}]$,

$$g \cdot f_i \in \langle f_1, \ldots, f_{i-1} \rangle \wedge \deg(g f_i) < i_{\text{reg}}(\langle f_1, \ldots, f_m \rangle) \Rightarrow g \in \langle f_1, \ldots, f_{i-1} \rangle .$$

Remark 1.31. In [Bar+05; Bar04], the notion of semi-regularity has been adapted to systems containing field equations, in the specific case $q = 2$.

Analogously to regular sequences, the Hilbert series not only is known, but it also characterizes semi-regular sequences.

Proposition 1.17 (Proposition 3.2.5, [Bar04]). *A sequence $f_1, \ldots, f_m \in \mathbb{K}[x_1, \ldots, x_n]$, $m > n$, is semi-regular if and only if*

$$HS_{\mathbb{K}[\boldsymbol{x}]/\langle f_1,\ldots,f_m\rangle}(z) = \left[\frac{\prod_{i=1}^{m}(1-z^{\deg(f_i)})}{(1-z)^n}\right],$$

where $[S] \in \mathbb{N}[[z]]$ denotes the power series obtained by truncating $S \in \mathbb{Z}[[z]]$ at its first non-positive coefficient.

Remark 1.32. If $S \in \mathbb{Z}[[z]]$ has a non-positive coefficient, then $[S] \neq S$, and hence $[S]$ is truncated into a polynomial.

We have seen in Section 1.3.6 that it is possible to derive a complexity estimate for computing a Gröbner basis, in the case of a homogeneous system, through the notion of index of regularity. The existence of degree falls makes it more difficult for affine systems. The next proposition shows that, under some conditions, the complexity bound can be extended to affine systems. This explains why semi-regular sequences are so relevant. Recalling Definition 1.39 about the degree of regularity of an affine sequence, we have

Proposition 1.18 (Proposition 6, [Bar+05]). *Let $F = \{f_1, \ldots, f_m\} \subseteq \mathbb{K}[x_1, \ldots, x_n]$ be an affine sequence of polynomials such that the sequence $f_1^h, \ldots, f_m^h$ is semi-regular, where f_i^h denotes the homogeneous part of f_i of highest degree. Then, the number of arithmetic operations in $\mathbb{K}$ to compute a Gröbner basis of $\mathcal{I}(F)$ with respect to a graded monomial ordering is upper bounded by*

$$\mathcal{O}\left(m \cdot d_{reg}\binom{n+d_{reg}-1}{d_{reg}}^{\omega}\right),$$

where ω is the linear algebra constant.

It has been observed experimentally that systems are "almost always" semi-regular. This expected behavior has been formally conjectured in [Frö85], but proven only in a few special cases. For an infinite field $\mathbb{K}$, the formal meaning of "almost always" is "outside a Zariski proper closed subset of the space of coefficients", the latter considered as algebraically independent formal parameters. This is the so-called *genericity assumption* under which properties of *generic systems* can be proven. If $\mathbb{K}$ is a finite field, the probability that a system is semi-regular is given by ratio between semi-regular systems and all systems. For what concerns this manuscript, this assumption will be intended as the expected behavior for random systems (i.e. for random coefficients) with a specific shape.

1.3.8 Systems with a special shape: application to coding theory and cryptography

Algebraic coding theory and cryptanalysis are two of the many fields of application of Gröbner basis techniques. The contributions of the present dissertation fall exactly within these frameworks.

A common feature of these two applications is that the polynomial systems we study are typically defined over a finite field $\mathbb{F}_q$. While the preliminary results we have

presented in this section hold for any coefficient field $\mathbb{K}$, additional properties can be exploited when $\mathbb{K}$ has positive characteristic. Indeed, in the case where $\mathbb{K} = \mathbb{F}_q$, we might be interested in finding solutions over the field itself rather than its closure. In this case, the standard strategy consists in adding the *field equations* $x_i^q - x_i$ to it. However, for all the systems considered in this work, the sought solutions belong to finite fields of large size, as they are support and multiplier vectors of large field size GRS codes or of alternant codes. Therefore they do not benefit from adding field equations, as their degree is too big. Instead, we now briefly review some known results on Gröbner bases for systems with a special shape. More precisely, we focus on a case study that is related to the work presented in Chapter 2 of this thesis: *bilinear systems*.

1.3.8.1 Affine bilinear systems

A *multi-homogeneous polynomial* is a polynomial homogeneous with respect to each block in which the unknowns are partitioned. For instance, a *bi-homogeneous polynomial* $f \in \mathbb{K}[x_1, \dots, x_{n_x}, y_1, \dots, y_{n_y}]$ ($\mathbb{K}[\boldsymbol{x}, \boldsymbol{y}]$ from now on) of *bidegree* (d_1, d_2) is such that

$$\forall \lambda, \mu \in \mathbb{K},\ f(\lambda x_1, \dots, \lambda x_{n_x}, \mu y_1, \dots, \mu y_{n_y}) = \lambda^{d_1} \mu^{d_2} f(x_1, \dots, x_{n_x}, y_1, \dots, y_{n_y}).$$

A *bilinear polynomial* is a bi-homogeneous polynomial of bidegree $(1, 1)$, and a system $F = \{f_1, \dots, f_m\}$ is *bilinear* if $f_1, \dots, f_m$ are bilinear.

Several investigations, both on the theoretical and experimental sides, showed that bilinear systems behave better than quadratic systems with the same number of equations and unknowns. In particular, the syzygy module is well-understood. We summarize the main results proved in [FSS11; Spa12] and related to a generic affine bilinear system $\{f_1, \dots, f_m\}$, $m \leq n_x + n_y$, generating the ideal $\mathcal{I}^{(\text{bil})}$:

- They enjoy a property of regularity that extends the standard definition to bi-homogeneous systems and that is called *bi-regularity*.
- Define the *Hilbert bi-series* (an analog of the Hilbert series for bi-homogeneous ideal) as

$$HS_{\mathbb{K}[\boldsymbol{x},\boldsymbol{y}]/\mathcal{I}^{(\text{bil})}}(z_1, z_2) \stackrel{\text{def}}{=} \sum_{(d_1,d_2)\in\mathbb{N}^2} \dim_K \left(\mathbb{K}[\boldsymbol{x}, \boldsymbol{y}]_{d_1,d_2} / \mathcal{I}^{(\text{bil})}_{d_1,d_2} \right) z_1^{d_1} z_2^{d_2},$$

  where $\mathcal{I}^{(\text{bil})}_{d_1,d_2} \stackrel{\text{def}}{=} \mathbb{K}[\boldsymbol{x}, \boldsymbol{y}]_{d_1,d_2} \cap \mathcal{I}$. Then

$$HS_{\mathbb{K}[\boldsymbol{x},\boldsymbol{y}]/\mathcal{I}^{(\text{bil})}}(z_1, z_2) = \frac{(1 - z_1 z_2)^m + N_m(z_1, z_2) + N_m(z_2, z_1)}{(1 - z_1)^{n_x+1}(1 - z_2)^{n_y+1}},$$

  with

$$N_m(z_1, z_2) = \left[\sum_{l=1}^{m-n_y-1} (1 - z_1 z_2)^{m-n_y-1-l} z_1 z_2 (1 - z_2)^{n_y+1} \cdot \left(1 - (1 - z_1)^l \sum_{k=1}^{n_y+1} z_1^{n_y+1-k} \binom{l + n_y - k}{n_y + 1 - k} \right) \right].$$

- The degree of regularity d_{reg} is upper bounded by

$$d_{\text{reg}}(\mathcal{I}^{(\text{bil})}) \leq n_x + n_y + 1,$$

which is obtained by rewriting Macaulay's bound. Moreover, for a 0-dimensional affine bilinear system with $m = n_x + n_y$,

$$d_{\text{reg}}(\mathcal{I}^{(\text{bil})}) \leq \min(n_x + 1, n_y + 1).$$

Therefore, under genericity assumptions, the number of arithmetic operations in $\mathbb{K}$ to compute a grevlex basis of an affine bilinear system $f_1, \ldots, f_{n_x+n_y} \in \mathbb{K}[\boldsymbol{x}, \boldsymbol{y}]$ is upper bounded by

$$\mathcal{O}\left(min(n_x, n_y) \cdot (n_x + n_y) \binom{n_x + n_y + min(n_x + 2, n_y + 2)}{min(n_x + 2, n_y + 2)}^{\omega}\right),$$

which is polynomial as long as the size of one block is fixed.

Chapter 2
Decoding of Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach

In this chapter, we study the bilinear system already introduced in Section 1.1.3.1 that models the decoding problem for a Reed-Solomon code. We will address its resolution by using Gröbner basis techniques and exploiting the key-equations that identify the generalized power decoding algorithm. In this particular case, these computations are much more efficient than for generic bilinear systems with the same number of unknowns and equations, where these techniques have exponential complexity. We explain why the calculation of a Gröbner basis permits to solve the decoding problem in polynomial time up to the Sudan radius. Moreover, beyond this radius these techniques recover automatically polynomial identities that are at the heart of improvements of the power decoding approach for reaching the Johnson decoding radius. They also allow to derive new polynomial identities in only one block of variables that can be used to derive new algebraic decoding algorithms for Reed-Solomon codes. We experimentally compare our approach with power decoding algorithm and provide numerical evidence that our method sometimes allows to correct efficiently slightly more errors than the Johnson radius.

Contents

2.1 Introduction

Decoding a large number of errors in Reed-Solomon codes. In Section [RS decoding] we reviewed the Berlekamp-Welch algorithm for decoding Reed-Solomon codes up to the error-correction radius $\frac{1-R}{2}$. We also offered an overview of classical list-decoding algorithms, which respond to the long-standing open problem from algebraic coding theory of decoding Reed-Solomon codes beyond half the minimum distance. We mentioned, in particular, Sudan's decoder, which works up to a fraction of errors $1-\sqrt{2R}$ (the so-called Sudan radius) and the subsequent improvement from Guruswami and Sudan [GS98], which pushed the decoding up to the Johnson radius $1-\sqrt{R}$. This represents in a sense the limit for such decoders since these decoders are list decoders that output all codewords up to this radius and beyond this radius the list size is not guaranteed to be polynomial anymore. However, if we do not insist on having a decoder that outputs all codewords within a certain radius, or if we just want a decoder that is successful most of the time on the q-ary symmetric channel of crossover probability p, then we can still hope to have an efficient decoder beyond this bound. Moreover, it is even interesting to investigate if there are decoding algorithms of subexponential complexity above the radius $1-\sqrt{R}$.

As already mentioned in Remark 1.4, we recall that decoding t errors for the code $\mathbf{GRS}_k(\boldsymbol{x},\boldsymbol{y})$ from a received word $\boldsymbol{r}=(r_1,\ldots,r_n)$ is equivalent to decoding the same amount of errors for $\mathbf{RS}_k(\boldsymbol{x})$. Therefore, without loss of generality, we can restrict our attention to the decoding problem for the k- dimensional Reed-Solomon code of length n over $\mathbb{F}_q$ with support $\boldsymbol{x}=(x_i)_{1\leq i\leq n}\in\mathbb{F}_q^n$:

$$\mathbf{RS}_k(\boldsymbol{x})=\{(P(x_i))_{1\leq i\leq n}:\ P\in\mathbb{F}_q[X],\ \deg P<k\}.$$

We call $d=n-k+1$ the code's minimum distance (recalling that Reed-Solomon codes are MDS codes).

Since we do not need to reserve a symbol for the multiplier vector, we adhere to the practice of denoting the received word by $\boldsymbol{y}=(y_i)_{1\leq i\leq n}$.

A Gröbner basis approach. Our approach for decoding is to model the problem by an algebraic system and then solve it with Gröbner bases techniques. At first sight, it might seem that this approach is not new in this setting: such techniques have already been used here, mainly to solve algebraic systems involved in the Guruswami-Sudan approach [AK11; Han18; LO06; LO08; Tri10; ZS10]. They were used up to now on systems where such techniques are expected to run efficiently just because the number of variables was very small for instance: for instance [AK11; LO06; LO08; Tri10] consider only two variables X and Y corresponding to the variables of the interpolation polynomial which is sought.

The approach followed in this chapter is different. Let $\boldsymbol{e}$ be the weight-t error vector and E its support, i.e. the set of positions in error. Then the error locator is defined as usual

$$\Lambda(X)\stackrel{\text{def}}{=}\prod_{i\in E}(X-x_i).$$

From this, we can write the bilinear system with unknowns the coefficients p_i of the polynomial $P(X)=\sum_{i=0}^{k-1}p_iX^i$ corresponding to the codeword that was sent and the coefficients λ_j of the error locator polynomial $\Lambda(X)=X^t+\sum_{j=0}^{t-1}\lambda_jX^j$ if we

assume that there were t errors. We have n bilinear equations in the $k+t$ variables p_i's and λ_j's coming from the n relations $P(x_\ell)\Lambda(x_\ell) = y_\ell\Lambda(x_\ell)$, $\ell \in [\![1, n]\!]$, namely

$$\sum_{i=0}^{k-1}\sum_{j=0}^{t} x_\ell^{i+j} p_i \lambda_j = \sum_{j=0}^{t} y_\ell x_\ell^j \lambda_j, \quad \ell \in [\![1, n]\!] \text{ and } \lambda_t = 1. \tag{2.1}$$

Gröbner basis techniques: a simple and automatic way for obtaining a polynomial time algorithm in our case. Standard Gröbner bases techniques can be used to solve this system. However, if we use directly the estimates for solving generic bilinear systems of Section 1.3.8.1 we would expect an exponential computational complexity. Nevertheless, Gröbner basis techniques solve typically in polynomial time this specific decoding problem when the fraction of errors is below the Sudan radius. This is explained in Section 2.4.1. The reason why the Gröbner basis approach works in polynomial time is related to power-decoding [Nie14; SSB10] and can be explained by similar arguments. However, the nice thing about this Gröbner basis approach is that the algorithm itself is very simple and could be ideally given without any reference to power decoding (or the Sudan algorithm). The computation of the Gröbner basis reveals degree falls which are instrumental for its very low complexity. Understanding these degree falls can be explained by the polynomial equations used by power decoding. However, this simple algorithm also appears to be very powerful beyond the Sudan bound: experimentally it seems that it is efficient up to the Johnson radius and that it is even able to correct more errors in some cases than the refinement of the original power decoding algorithm [Nie18] (which reaches asymptotically the Johnson radius). This is demonstrated in Section 2.5.

Understanding the nice behavior of the Gröbner basis approach. Moreover, trying to understand theoretically why this algorithm behaves so well, is not only explained by the polynomial relations which are at the heart of the power decoding approach, but it also reveals new polynomial relations that are not exploited by the power decoding approach as shown in Section 2.4. In other words, this approach not only gives an efficient algorithm but also exploits other polynomial relations. It seems fruitful to understand and describe them, this namely paves the road towards new algebraic decoders of Reed-Solomon codes.

Notation. Throughout this chapter, we will use the following notation. For a polynomial $Q(X) = \sum_{i=0}^{m} q_i X^i$, coeff $(Q(X), X^s)$ stands for the coefficient q_s of X^s in $Q(X)$. For two polynomials $Q(X)$ and $G(X)$, $[Q(X)]_{G(X)}$ stands for the remainder of $Q(X)$ divided by $G(X)$. Finally let $Q(X)$ be a polynomial whose coefficients belong to a multivaraite polynomial ring $\mathbb{K}[\boldsymbol{u}]$ and $\mathcal{F}$ a vector space of polynomials of $\mathbb{K}[\boldsymbol{u}]$. We say that $Q \in_{\text{coef}} \mathcal{F}$ if all the coefficients of Q belong to $\mathcal{F}$.

2.2 Power decoding

In Section 1.1.3.1, we reviewed the Berlekamp-Welch algorithm for decoding Reed-Solomon codes up to half the minimum distance and we recalled that this can be adapted to obtain list decoders, namely the Sudan and the Guruswami-Sudan algorithms, which improve the error correction capability up to the Sudan bound and the Johnson bound respectively. Alternative methods to decode RS codes exist

and we are now going to recall the *Power Decoding* strategy, whose key-equations play a central role in the Gröbner basis analysis of this chapter. Power decoding has been originally proposed by Schmidt, Sidorenko and Bossert [SSB10]. In its original form, this algorithm was designed for low-rate RS codes, as it can decode the same number of errors as Sudan's algorithm. On the other hand, the approach from [SSB10] presents a one-pass algorithm, i.e. it processes the input data only once and accomplishes the decoding by solving a simultaneous shift-register problem, which is especially suitable for hardware implementations. We also remark that power decoding has been adapted to several codes related to the Reed-Solomon family, such as interleaved RS codes.

While the Guruswami-Sudan algorithm, by taking into account the multiplicity parameter, enhances Sudan's, it has been an open problem for several years whether the same upgrade was possible for power decoding. This question has been positively answered by Nielsen in 2018 [Nie18], who generalized the key equations used in this approach, eventually reaching a decoding radius that is almost the same as that of the Guruswami-Sudan algorithm. Contrarily to the latter, this new method removes the final root-finding step. Moreover, it always returns a closest codeword. Therefore, beyond half the minimum distance, it will fail for a few error patterns (for which it outputs either no result or a wrong result) but the probability of decoding failure is not easy to analyze. Indeed it has been tightly upper bounded for a specific choice of parameters, while for other ones [Nie18] only provides experimental results.

In the rest of this section, we will recall the key equations used for power decoding as presented in [Nie18] and which fraction of errors they allow to correct.

As already mentioned in Remark 1.4, decoding t errors for the code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ from a received word $\boldsymbol{r} = (r_1, \ldots, r_n)$ is equivalent to decoding the same amount of errors for $\mathbf{RS}_k(\boldsymbol{x})$. Therefore, without loss of generality, we can restrict our attention to the decoding problem for the k- dimensional Reed-Solomon code of length n over $\mathbb{F}_q$ with support $\boldsymbol{x} = (x_i)_{1 \leq i \leq n} \in \mathbb{F}_q^n$:

$$\mathbf{RS}_k(\boldsymbol{x}) = \{(P(x_i))_{1 \leq i \leq n} : \ P \in \mathbb{F}_q[X], \ \deg P < k\}.$$

We call $d = n - k + 1$ the code's minimum distance (recalling that Reed-Solomon codes are MDS codes).

Since we do not need to reserve a symbol for the multiplier vector, we adhere to the practice of denoting the received word by $\boldsymbol{y} = (y_i)_{1 \leq i \leq n}$. The vector $\boldsymbol{e}$ represents the weight-t error vector and E its support, i.e. the set of positions in error. Then the error locator is defined as usual

$$\Lambda(X) \stackrel{\text{def}}{=} \prod_{i \in E} (X - x_i). \tag{2.2}$$

We also introduce two other crucial polynomials. The first one is the interpolation polynomial with respect to the received values, i.e. the unique polynomial $R(X)$ of degree $\leq n - 1$ such that

$$\forall i \in [\![1, n]\!], \ R(x_i) = y_i.$$

The second is

$$G(X) \stackrel{\text{def}}{=} \Pi_{\ell=1}^{n} (X - x_\ell).$$

Note that these two polynomials are immediately computable by the receiver. Moreover, G can be precomputed because it does not depend on the received word but only on the support $\boldsymbol{x}$.

The first relation among all these polynomials is nothing but a rewriting of the key equation implicit in Gao's decoder [Gao03]:

$$\Lambda(X)R(X) \equiv \Lambda(X)P(X) \mod G(X). \tag{2.3}$$

This is a non-linear equation in the unknowns Λ and P. The strategy adopted by the algorithm is to linearize the equation: Λ and ΛP are replaced by λ and ψ respectively, leading to

$$\lambda R \equiv \psi \mod G. \tag{2.4}$$

Equation (2.4) is now linear, however there exist infinitely many solutions. Since ψ is a substitute for ΛP and $\deg(\Lambda P) \leq \deg(\Lambda) + (k-1)$, one can further restrict the solutions by adding the constraint

$$\deg(\lambda) + k - 1 \geq \deg(\psi).$$

Then we solve for such λ and ψ, with λ monic and of minimal degree and we hope that this weaker relation still contains $\lambda = \Lambda$ as a solution. This is indeed the case every time $t < \frac{d}{2}$. The decoding then terminates with the computation of $P = \psi/\lambda$. However, whenever the number of errors exceeds half the minimum distance, this approach never works.

Nevertheless, Gao's key equations can be powered to enable an improved correction capability for some parameters.

Lemma 2.1 (Simply powered key-equations [Nie18], Lemma 2.2)**.** *Let $u \in \mathbb{N}^*$, then*

$$\Lambda R^u = \Lambda P^u \mod G. \tag{2.5}$$

Proof. We have

$$\Lambda P^u = \Lambda(R+(P-R))^u = \Lambda R^u + \Lambda(P-R)\left(\sum_{i=1}^{u}\binom{u}{i}(P-R)^{i-1}\right) \equiv \Lambda R^u \mod G,$$

where the equivalence at the end follows from the fact that $\Lambda(P-R) \equiv 0 \mod G$. □

Again the non-linear equation can be linearized by replacing Λ and ΛP^u's with λ and ψ_u's respectively, thus obtaining for any $u \in \mathbb{N}^*$,

$$\lambda R^u \equiv \psi_u \mod G. \tag{2.6}$$

Similarly to what has been done before, the additional condition

$$\deg(\lambda) + u(k-1) \geq \deg(\psi_u) \tag{2.7}$$

is required to be fulfilled. In this case, the arising question is for which finite set of values u these powered equations should be considered. Equation (2.7) naturally answers this question. Indeed, whenever $\deg(\lambda) + u(k-1) \geq n$, Equation (2.6) is satisfied for any λ by setting $\psi_u = \lambda R \mod G$. The equations which restricts the space of solutions for λ are therefore those for $u \in [\![1, q_1]\!]$, where

$$q_1 \stackrel{\text{def}}{=} \max\{u : t + (k-1)u \leq n-1\} = \left\lfloor \frac{n-t-1}{k-1} \right\rfloor. \tag{2.8}$$

Hence, we seek $\lambda, \psi_1, \ldots, \psi_{q_1}$ satisfying (2.6) and (2.7) and with λ monic and of minimal degree. After finding the linear variables, $P = \psi_1/\lambda$ is computed. The difference between available coefficients and constraints gives the maximum number of decodable errors:

$$t \leq \frac{q_1}{q_1+1}n - \frac{q_1}{2}(k-1) - \frac{q_1}{q_1+1}, \tag{2.9}$$

which asymptotically approaches the so-called Sudan's radius $1-\sqrt{2R}$, corresponding to error-correction radius achieved by Sudan's algorithm. Indeed, let $\rho = \frac{t}{n}$ be the relative radius. When $n \to \infty$, then q_1 $to \frac{n-t}{k} = \frac{1-\rho}{R}$, and therefore Equation (2.9) gives, after dividing by n,

$$\begin{aligned}
&\rho \to \left(1 - \frac{1}{q_1}\right) - \frac{q_1}{2}R \\
\iff &\rho \to 1 - \frac{R}{1-\rho} - \frac{1-\rho}{2} \\
\iff &2\rho(1-\rho) - 2(1-\rho) + (1-\rho)^2 \to -2R \\
\iff &(1-\rho)^2 \to 2R \\
\iff &\rho \to 1 - \sqrt{2R}.
\end{aligned}$$

To reach Johnson's radius, though, it is necessary to take the multiplicity for the *error evaluator* polynomial Ω into consideration:

$$\Omega(X) \stackrel{\text{def}}{=} -\sum_{i \in E} e_i \zeta_i \prod_{j \in E \setminus \{i\}} (X - x_j),$$

where $\zeta_i = \prod_{j \neq i}(x_i - x_j)^{-1}$. Alternatively, the *error evaluator* polynomial can be defined as the unique polynomial of degree $\leq t-1$ such that

$$\Omega(x_i) = -e_i, \quad \text{for all } i \in [\![1, n]\!] \text{ for which } e_i \neq 0. \tag{2.10}$$

From (2.3) we know that G must divide $\Lambda(P-R)$. It can be readily checked that the quotient coincides with Ω:

$$\Lambda(X)(P(X) - R(X)) = \Omega(X)G(X). \tag{2.11}$$

Additionally, this relation is used in the generalization of power decoding to derive further identities:

Proposition 2.1 ([Nie18], Theorem 3.1)**.** *For any $s, u \in \mathbb{N}^*$, $u \geq s$,*

$$\Lambda(X)^s P(X)^u = \sum_{i=0}^{u} \left(\Lambda(X)^{s-i}\Omega(X)^i\right) \binom{u}{i} R(X)^{u-i} G(X)^i \qquad u \in [\![1, s-1]\!], \tag{2.12}$$

$$\Lambda(X)^s P(X)^u \equiv \sum_{i=0}^{s-1} \left(\Lambda(X)^{s-i}\Omega(X)^i\right) \binom{u}{i} R(X)^{u-i} G(X)^i \mod G(X)^s \qquad u \geq s. \tag{2.13}$$

Proof. Since $\Lambda(P-R) = \Omega G$, we obtain

$$\Lambda^s P^u = \Lambda^s (R + (P-R))^u = \sum_{i=0}^{u} \binom{u}{i} \Lambda^s (P-R)^i R^{u-i} = \sum_{i=0}^{u} \binom{u}{i} \Lambda^{s-i} \Omega^i R^{u-i} G^i,$$

which is the thesis for $u < s$. For $u \geq s$, it is enough to notice that all the summands of index $i \geq s$ equal 0 modulo G^s. □

This time the linearization is carried out by replacing $\Lambda^{s-i}\Omega^i$, $i \in [\![0, s-1]\!]$ and $\Lambda^s P^u$'s with λ_i's and ψ_u's respectively. As for simply powered key equations, the maximum power u is naturally bounded by

$$q_s \stackrel{\text{def}}{=} \max\{u : st + u(k-1) \leq sn - 1\} = \left\lfloor \frac{s(n-t)-1}{k-1} \right\rfloor.$$

Indeed it is readily seen that taking larger values of u increases the number of variables in the linear system more than it does for the number of available coefficients and thus it is not cost-effective. The resulting linear system is therefore

$$\begin{array}{rll} \psi_u = \sum_{i=0}^{u} \lambda_i \left(\binom{u}{i} R^{u-i} G^i \right), & & u \in [\![1, s-1]\!] \\ \psi_u = \sum_{i=0}^{s-1} \lambda_i \left(\binom{u}{i} R^{u-i} G^i \right) & \mod G^s, & u \in [\![s, q_s]\!], \end{array}$$

to which the "degree contraints"

$$\begin{array}{lll} \deg(\lambda_0) \geq & \deg(\lambda_i) + i, & i \in [\![1, s-1]\!] \\ \deg(\lambda_0) \geq & \deg(\psi_u) + -u(k-1), & u \in [\![1, q_s]\!]. \end{array}$$

are added. As in the previous cases, a solution such that λ_0 is monic and of minimal degree. The decoding concludes by computing $P = \psi_1/\lambda_0$.

Given the multiplicity degrees s and v, the decoding radius of this generalized power decoding approach turns out to be

$$t_{pow}(s, v) = \frac{2v - s + 1}{2(v+1)} n - \frac{v}{2s}(k-1) - \frac{v}{s(v+1)}. \tag{2.14}$$

Whenever the number of errors is below the Johnson radius, calculations analogous to those done before for the Sudan radius show that there are infinitely many choices of s and v such that the power decoding algorithm can correct them (and the best choice of v for a fixed s is clearly q_s). Advanced subroutine algorithms allow to achieve an overall complexity of power decoding of $\tilde{\mathcal{O}}(v^\omega sn)$ [RS16], matching the best realization of Guruswami-Sudan algorithm. In [Nie18] the decoding failure is also studied, but this is not our main concern here.

2.3 The Algorithm

We will assume in the following that the polynomial ideal $\mathcal{I}$ generated by the affine bilinear equations, generically called f_i's, is radical, meaning that whenever there is a polynomial f and a positive integer s such that f^s is in $\mathcal{I}$, then f is in $\mathcal{I}$. For finite fields, bringing in the system the field equations ensures that the ideal is radical. Although field equations are not effective in this case, as the field size for RS codes must be very large, namely not smaller than the block length, the ideal radicality

is still usually verified. Moreover, in a large part of this work, we also expect the algebraic system (2.1) to have a unique solution. Indeed, while this is guaranteed only up for error correction up to half the minimum distance, this seems to be the typical case when the number of errors is below the Gilbert-Varshamov bound. In such a case, Proposition 1.12 says that the reduced Gröbner basis of the ideal $\mathcal{I}$ is given by linear polynomials and in each of them only one variable appears.

The algorithm we use, in its simplest form, consists of computing a truncated Gröbner basis in degree D of the affine bilinear system (2.1), for some integer D. We have seen that this can be done by iteratively computing the row echelon forms of Macaulay matrices w.r.t to a graded monomial order. Despite not being the best option in terms of efficiency, we will still refer to Algorithm 1.3. It will indeed come in handy several times during this chapter to prove results about the membership of polynomials to some linear spaces. In particular, the monomial order chosen is $>_{grevlex}$ as this usually achieves the best performance and no evidence suggests the opposite for this specific system. By Lazard's Theorem, we have the guarantee that if D is large enough, then the D-Gröbner basis is in fact a Gröbner basis.

However, we noticed that the aforementioned approach is not the most efficient (unless $t \leq \frac{n-k}{2}$ where direct row echelonization (2.1) is enough) because during the Gaussian elimination process we have a sequence of degree falls which are instrumental for computing a Gröbner basis by staying at a very small degree. This is evident if we use for instance Faugère's F4 algorithm on (2.1)).

We recall that a *degree fall* of degree s is a polynomial combination $\sum_{i=1}^{m} g_i f_i \neq 0$ of the f_i's which satisfies

$$s \stackrel{\text{def}}{=} \deg \sum_{i=1}^{m} g_i f_i < \max_{i=1}^{m} \deg g_i f_i.$$

The simplest example of such a degree fall occurs in (2.1) when $t < n - k$. Here there are linear combinations of the bilinear equations of (2.1) giving linear equations. This can be verified by linearization, i.e. by performing the change of variables $z_s \stackrel{\text{def}}{=} \sum_{i,j:i+j=s} p_i \lambda_j$ in (2.1) and getting the system

$$\sum_{s=0}^{t+k-1} x_\ell^s z_s = \sum_{j=0}^{t} y_\ell x_\ell^j \lambda_j, \quad \ell \in [\![1, n]\!]. \tag{2.15}$$

In other words, by eliminating the z_s's in these equations we obtain linear equations involving only the λ_i's. When $t \leq \frac{n-k}{2}$ there are enough such equations to recover from them the λ_i's and by substituting for them in (2.1) the p_i's by solving again a linear system. We will see in the next section that there is a parallel between this linearization and the one used in power decoding, which follows from the equivalence of the corresponding bilinear systems. Despite the described procedure is in its essence already known and much more efficient algorithms for solving this system exist, it is still interesting to notice that the Gröbner basis approach already yields a polynomial time algorithm for the particular bilinear system (2.1), while for a large range of parameters this would be exponential for generic bilinear systems with the same number of unknowns and equations as (2.1), see Section 1.3.8.1.

A less trivial degree fall behavior is obtained in the case the fraction of errors is at most Sudan's radius. Here, after substituting for the λ_i's which can be expressed

as linear functions of the other λ_i's by using the aforementioned linear equations involving the λ_i's we obtain new bilinear equations $f'_1, \cdots, f'_m$. It turns out that we can perform linear combinations on these f'_i's to eliminate the monomials of degree 2 in them and derive new linear equations involving only the λ_i's. This is proved in Subsection 2.4.1. This process can be iterated and there are typically enough such linear equations to recover the λ_i's in this way as long as t is below or equal to the Sudan decoding radius. As explained above, this allows us to recover the right codeword by plugging the values for λ_i in (2.1) and solving the corresponding linear system in the p_i's. This will be described more thoroughly again in Subsection 2.4.1.

This behavior shows that we can decode up to the Sudan decoding radius with constant degree $D = 2$, thus implying a polynomial-time algorithm. However, when the number of errors becomes bigger, $D = 2$ is not enough to exhibit more degree falls. The latter appear already starting from $D = 3$ and we will explain why they are instrumental to the generalization of the power decoding approach of [Nie18] decoding up to the Johnson radius. In light of this and for some range of parameters, we will also propose an alternative algorithm that directly computes the degree falls and exploits them from the very beginning.

2.4 A partial explanation of the algebraic behavior

2.4.1 Correcting up to the Sudan bound in polynomial time

The efficiency of Algorithm 1.3 is already demonstrated by the fact that choosing $D = 2$ in it corrects in polynomial time as many errors as Sudan's algorithm. Choosing $D = 2$ means that we just keep the equations of degree 2 and try to produce new linear equations by linear combinations of the equations of degree 2 aiming at eliminating the degree 2 monomials. The efficiency of this algorithm is related to power decoding [SSB10]: the algorithm finds automatically the linear equations exploited by the power decoding approach. We can show that the system originated by the key equation implicit in Gao's decoder is equivalent to the one which interpolates the received values. We first need to recall the following preliminary lemma.

Lemma 2.2. *For any polynomial $Q(X) \in \mathbb{F}_q[X]$ of degree $< n$, the coefficients of Q can be expressed as linear combinations of $Q(x_1), \cdots, Q(x_n)$.*

Proof. This fact is just a consequence that Q coincides with its interpolation polynomial on the points $(a_\ell, Q(a_\ell))$ and that this interpolation polynomial is given by

$$Q(X) = \sum_{\ell=1}^{n} Q(x_\ell) \frac{\Pi_{j \neq \ell}(X - a_j)}{\Pi_{j \neq \ell}(x_\ell - x_j)}.$$

□

From the previous lemma, it follows that

Proposition 2.2. *The bilinear systems (2.1) and (2.3) are equivalent: (2.3) can be obtained from linear combinations of (2.1) and vice versa.*

Proof. We start by proving that (2.3) can be derived from (2.1). If we bring in

$$\begin{aligned} Q(X) &\stackrel{\text{def}}{=} P(X)\Lambda(X) - R(X)\Lambda(X) \\ S(X) &\stackrel{\text{def}}{=} Q(X) \mod G(X), \end{aligned}$$

then

- (2.1) amounts to write $Q(a_\ell) = 0$ for ℓ in $[\![1, n]\!]$ and to express the $Q(x_\ell)$'s as quadratic forms in the λ_i's and the p_j's.
- Since $Q(x_\ell) = S(x_\ell)$ for all ℓ in $[\![1, n]\!]$ and since S is of degree $< n$ we can use the previous fact and express its coefficients linearly in terms of the $S(x_\ell) = Q(x_\ell)$'s.
- Since (2.3) is nothing but expressing that the coefficients of $S(X)$ are all equal to 0, we obtain that the equations of (2.3) can be obtained from linear combinations of the equations of (2.1).

Conversely, since $S(x_\ell)$ can be written as a linear combination of the coefficients of $S(X)$, the quadratic equations in the λ_i's and the p_i's obtained by writing $S(x_\ell) = 0$ are linear combinations of the quadratic equations given by (2.3). These equations $S(x_\ell) = 0$ coincide with the equations in (2.1), since $Q(x_\ell) = S(x_\ell)$ for all ℓ in $[\![1, n]\!]$. □

The point of using (2.3) is that

- These equations are more convenient to work with to understand what is going on algebraically during the Gröbner basis calculations.
- They give directly $n - k - t + 1$ linear equations, since (i) the coefficient of $S(X)$ of degree $d \in [\![t + k, n - 1]\!]$ coincides with the coefficient of the same degree in $-R(X)\Lambda(X) \mod G(X)$ since
  1. $\Lambda(X)P(X)$ is of degree $\leq t + k - 1$;
  2. the coefficient of $S(X)$ of degree $t + k - 1$ is equal to p_{k-1} − coeff $\left([\Lambda(X)R(X)]_{G(X)}, X^{t+k-1}\right)$ because $\Lambda(X)$ is monic and of degree t.

This motivates the use of the following modeling.

Modeling 2.1 (Modeling for decoding Reed-Solomon codes)**.**
System:

$$\{\text{coeff}\left([\Lambda(X)P(X) - \Lambda(X)R(X)]_{G(X)}, X^u\right) = 0 \mid u \in [\![0, n - 1]\!]\}$$

Unknowns: *k unknowns p_i's, $i \in [\![0, k - 1]\!]$,+ t unknowns λ_j, $j \in [\![0, t - 1]\!]$ (since Λ is monic, $\lambda_t = 1$).*
Equations: *n equations of which:*

- *$t + k - 1$ affine bilinear equations in the blocks of coefficients p_i's and λ_j's;*
- *$n - t - k - 2$ affine linear equations in the coefficients λ_i's;*

- *1 affine linear equation in the coefficients λ_i's and p_{k-1}.*

We can now prove that

Proposition 2.3. *Let q_1 be defined as in Equation (2.8). All affine functions in the λ_i's of the form* coeff $\left(\left[\Lambda(X)R^j(X)\right]_{G(X)}, X^u\right)$ *for $j \in [\![1, q_1]\!]$ and $u \in [\![t + (k-1)j + 1, n-1]\!]$ are in the linear span of the 2-Gröbner basis for the bilinear system (2.1).*

Remark 2.1. The fact that these are indeed affine functions follows on the spot from generalizing the degree considerations above: $\Lambda(X)P(X)^j$ is of degree $\leq t + (k-1)j$.

Proof. Let us refer to Algorithm 1.3 for the computation of a truncated Gröbner basis in the affine case. In particular we are interested in the set F and the subspace spanned by it. The space $\langle F \rangle_{\mathbb{F}_q}$ contains initially (and therefore all the time) the space of affine functions in the λ_i's generated by

$$\text{coeff}\left(\left[-\Lambda(X)R(X)\right]_{G(X)}, X^u\right) = \text{coeff}\left(\left[\Lambda(X)P(X) - \Lambda(X)R(X)\right]_{G(X)}, X^u\right),$$

for all $u \in [\![t + k, n-1]\!]$. Now proceed by induction on j, and assume that at some point the space generated by F contains the linear span of the affine functions

$$\text{coeff}\left(\left[-\Lambda(X)R^j(X)\right]_{G(X)}, X^u\right) = \text{coeff}\left(\left[\Lambda(X)P(X)^j - \Lambda(X)R(X)^j\right]_{G(X)}, X^u\right),$$

for all $u \in [\![t + (k-1)j + 1, n-1]\!]$ where j is some integer in the interval $[\![1, q_1 - 1]\!]$. Note that

$$\begin{aligned} &\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right) \mod G && (2.16)\\ &= \left(P(\Lambda P^j - \Lambda R^j) + R^j(\Lambda P - \Lambda R)\right) \mod G \\ &= \left(P(\Lambda P^j - \Lambda R^j \mod G) + R^j(\Lambda P - \Lambda R \mod G)\right) \mod G. && (2.17)\end{aligned}$$

We use the equality between the polynomials (2.16) and (2.17) to claim that their coefficients should coincide for all the degrees $[\![t + (j-1)(k-1), n-1]\!]$. Note now that after the elimination of variables performed so far, this makes that all coefficients of degree in $[\![t + (k-1)j + 1, n-1]\!]$ in $\Lambda P^j - \Lambda R^j \mod G$ vanish, since they were affine functions by the induction hypothesis and become 0 after the variable elimination step. This implies that $\Lambda P^j - \Lambda R^j \mod G$ becomes a polynomial of degree $\leq t + (k-1)j$ after elimination of variables. Therefore $P(\Lambda P^j - \Lambda R^j \mod G)$ is a polynomial of degree $\leq t + (k-1)(j+1)$. From the equality of the polynomials (2.16) and (2.17), this implies that the coefficient of degree u in $\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right)$ mod G coincides with the coefficient of the same degree in $(R^j(\Lambda P - \Lambda R \mod G))$ mod G for u in $[\![t + (k-1)(j+1) + 1, n-1]\!]$. We observe now that the last coefficient is nothing but a linear combination of the coefficients of $\Lambda P - \Lambda R \mod G$, which are precisely the initial polynomial equations. Since the polynomial $\left(\Lambda P^{j+1} - \Lambda R^{j+1}\right)$ mod G has all its coefficients that are affine functions in the λ_i's by Remark 2.1 for all the degrees $u \in [\![t + (k-1)(j+1) + 1, n-1]\!]$ we obtain that after the Gaussian elimination step, $\langle F \rangle_{\mathbb{F}_q}$ contains the space generated by these aforementioned affine functions. This proves the proposition by induction on j. □

These linear equations that we produce coincide exactly with the linear equations produced by the power decoding approach [SSB10] and this allows us to correct as many errors as the power decoding approach based on the same assumption, namely that they are all independent, which is actually the typical scenario. However, contrarily to power decoding that is bound to make such an assumption to work, the Gröbner basis is more versatile, as it allows to decode even without this assumption as explained in Section 2.5.

2.4.2 Decoding up to the Johnson radius

Equations (2.12) and (2.13) generalize power decoding to decode up to the Johnson radius by bringing in the "error evaluator" polynomial $\Omega(X)$ defined in (2.10). Interestingly enough, our Gröbner basis approach also exhibits degree falls of degree s that are related to (2.12) and (2.13). This can be understood by using an equivalent definition of the error evaluator polynomial Ω as

$$\Omega \stackrel{\text{def}}{=} -\Lambda R \div G. \tag{2.18}$$

We can prove that the definitions of Ω from (2.10) and (2.18) are actually equivalent. Indeed, once Ω's coefficients are written in functions of Λ's coefficients, (2.11) holds with definition (2.18).

Proposition 2.4. *The coefficients of $\Omega \stackrel{def}{=} -\Lambda R \div G$. are affine functions of the λ_i's. Moreover, if $t \leq n-k$, $\Lambda(P-R) = \Omega G$.*

Proof. The polynomial $\Omega(X) = \sum_{i=0}^{t-1} \omega_i X^i$ is determined by the linear constraints $\text{coeff}(\Lambda R + \Omega G, d) = 0$, for $d \in [\![n, n+t-1]\!]$, which translate into the system

$$\left\{ \sum_{i=d-n}^{t-1} \omega_i g_{d-i} = -\sum_{j=d-n+1}^{t} \lambda_j r_{d-j} \mid \lambda_t = g_n = 1, d \in [\![n, n+t-1]\!] \right\},$$

where we adopt the notation $\Lambda(X) = \sum_{i=0}^{n-1} \lambda_i X^i$, $R(X) = \sum_{i=0}^{n-1} r_i X^i$, $G(X) = \sum_{i=0}^{n} g_i X^i$ (with $\lambda_t = g_n = 1$ because Λ and G are monic). We observe that in the equation $\sum_{i=d-n}^{t-1} \omega_i g_{d-i} = -\sum_{j=d-n+1}^{t} \lambda_j r_{d-j}$, only the coefficients of Ω corresponding to degree at least $d-n$ appear. Therefore, the ω_i coefficients can be recursively retrieved in reverse order (i.e. from the largest to the smallest index) from the previous system as

$$\begin{aligned} \omega_{t-1} &= -r_{n-1}, \\ \omega_l &= -\textstyle\sum_{j=l+1}^{t} \lambda_j r_{d-j} - \sum_{i=l+1}^{t-1} \omega_i g_{d-i}. \end{aligned}$$

As long as $t \leq n-k$, (2.11) follows from (2.18) and (2.3). Indeed $[\Lambda R]_G = \Lambda P$. This follows from (2.3) and $t+k-1 \leq n-1$ implying that $\Lambda P = \Lambda R \mod G$. This and (2.18) then imply that $\Lambda R = -\Omega G + \Lambda P$ which is obviously equivalent to (2.11), i.e. $\Lambda(P-R) = \Omega G$. □

From these considerations, note that if we equate the coefficients of the polynomials in (2.12) for all the degrees in $[\![st + u(k-1) + 1, st + u(n-1)]\!]$ and in (2.13) for all the degrees in $[\![st + u(k-1) + 1, s(n-1)]\!]$, the coefficient of the left-hand term vanishes and the coefficient in the righthand term is a polynomial of

degree s in the λ_i's (this follows from the fact that the coefficients of Ω are affine functions in those λ_i's). This gives polynomial equations in the λ_i's of degree s. In a sense, they can be viewed as generalizations at degree s of the linear equations that are produced by a truncated Gröbner basis at degree $D = 2$. These equations are actually produced as degree falls that are in the linear span of intermediate sets F produced in Algorithm 1.3 when $D = s + 1$. At the same degree, other degree falls of degree s also occur. To explain this point it makes sense to bring in the notation for the right-hand term in (2.12) and (2.13). Let us define

$$\chi(s,u) \stackrel{\text{def}}{=} \sum_{i=0}^{u} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i = \Lambda^{s-u} \left(\Lambda R + \Omega G\right)^u \quad \text{if } u < s,$$

$$\chi(s,u) \stackrel{\text{def}}{=} \left[\sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i \right]_{G^s} \quad \text{if } u \geq s$$

We also let $\chi(s,u)_H$ be the polynomial where we dropped all the terms of degree $\leq ts+u(k-1)$ in $\chi(s,u)$, i.e. if $\chi(s,u) = \sum_i a_i X^i$, then $\chi(s,u)_H = \sum_{i>ts+u(k-1)} a_i X^i$. The degrees corresponding to the non-identically null coefficients of $\chi(s,u)_H$ are exactly the same above the degree of the left-hand sides in (2.12) and (2.13).

Theorem 2.1. *Let $\mathcal{F}_D = \langle F \rangle_{\mathbb{F}_q}$ where F is the set output by Algorithm 1.3 with the bilinear system (2.3) and degree D as inputs. We have for all nonnegative integers s, s', $u \leq q_s$, $u' \leq q_{s'}$*

$$\chi(s,u)_H \in_{coef} \mathcal{F}_{s+1} \tag{2.19}$$

$$\chi(s,u)\chi(s',u') - \chi(s+s',u+u') \in_{coef} \mathcal{F}_{s+s'+1}. \tag{2.20}$$

where $P \in_{coef} \mathcal{F}_v$ (where P is a polynomial with coefficients that are polynomials in the λ_i's and the p_i's) means that all the coefficients of P belong to $\mathcal{F}_v$.

We point out a couple of observations:

1. The algebraic manipulations used in the proof of Proposition 2.1 involved only polynomial additions, multiplications and division remainders. This implies that the coefficients of (2.12) and (2.13) belong to the ideal generated by the bilinear system. In addition, $\Lambda(P-R)$'s factors in the right-hand side summands were replaced with ΩG's factors whenever possible. All these operations can be reinterpreted as polynomial combinations of the equations of the original bilinear system (2.3), thus answering the ideal membership problem.

2. It is of course clear that $\chi(s,u)\chi(s',u') - \chi(s+s',u+u')$ belongs to the ideal generated by the polynomial equations (2.12) and (2.13) since they basically come from the identity $(\Lambda^s P^u)(\Lambda^{s'} P^{u'}) = (\Lambda^{s+s'} P^{u+u'})$.

The two points above imply that $\chi(s,u)\chi(s',u') - \chi(s+s',u+u')$ belongs to the ideal generated by the bilinear system (2.3) as well. In principle we should expect to find out such equations at degree $s+s'+u+u'$, because they are originated by the identity $(\Lambda^s P^u)(\Lambda^{s'} P^{u'}) = (\Lambda^{s+s'} P^{u+u'})$, whose left and right-hand sides have this degree.

What is somehow surprising is that these equations are discovered at a rather small degree Gröbner basis computation. In particular, we can rigorously upper bound the degree with the value $s+s'+1$, and experimentally they already appear

from computations at degree $s+s'$ in some cases, i.e. at their same degree. Moreover, these equations only involve the λ_i's. By inspection of the behavior of the Gröbner basis computation, it seems that the linear equations that we produce later on are first produced by degree falls only involving these equations of degree s. It is therefore tempting to change the Gröbner basis decoding procedure strategy: instead of feeding Algorithm 1.3 with the initial system (2.1) or (2.3) we run it with the equations of degree s given by Theorem 2.1. Once we have recovered the λ_i's in this way we recover the p_i's by solving a linear system as explained earlier. In the next section, the behavior of this strategy on non-trivial examples will be explained. Theorem 2.1 is proved in the following subsection.

2.4.3 Proof of Theorem 2.1

It will be convenient here to notice that $\chi(s,s)$ has a slightly simpler expression which avoids the reduction modulo G^s.

Lemma 2.3.

$$\chi(s,s) = (\Lambda R + \Omega G)^s.$$

Proof. $\chi(s,s)$ is defined as

$$\begin{aligned}
\chi(s,s) &\stackrel{\text{def}}{=} \left[\sum_{i=0}^{s-1} \binom{s}{i} \Lambda^{s-i} R^{s-i} \Omega^i G^i\right]_{G^s} \\
&= \left[\sum_{i=0}^{s} \binom{s}{i} \Lambda^{s-i} R^{s-i} \Omega^i G^i\right]_{G^s} \\
&= \left[(\Lambda R + \Omega G)^s\right]_{G^s} \\
&= (\Lambda R + \Omega G)^s
\end{aligned}$$

□

It will also be helpful to observe that $\chi(s,u)$ and $\chi(s,u+1)$ are related by the following identity

Lemma 2.4.

$$\chi(s,u)P - \chi(s,u+1) = \Lambda^{s-u-1}(\Lambda R + \Omega G)^u (\Lambda P - \Lambda R - \Omega G) \quad \textit{for } u \in [\![0, s-1]\!]$$

$$[\chi(s,u)P - \chi(s,u+1)]_{G^s} = \left[(\Lambda P - \Lambda R - \Omega G) \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s} \quad \textit{for } u \in [\![s, q_s - 1]\!].$$

Proof. For $u \in [\![0, s-1]\!]$ we have (for the case $u = s-1$ we use Lemma 2.3 for the term $\chi(s,u+1)$):

$$\begin{aligned}
\chi(s,u)P - \chi(s,u+1) &= \Lambda^{s-u} P (\Lambda R + \Omega G)^u - \Lambda^{s-u-1} P (\Lambda R + \Omega G)^{u+1} \\
&= \Lambda^{s-u-1} (\Lambda R + \Omega G)^u (\Lambda P - \Lambda R - \Omega G).
\end{aligned}$$

For $u \in [\![s, q_s - 1]\!]$ we observe that

$$\begin{aligned}
[\chi(s,u)P]_{G^s} &= \left[P \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-i} R^{u-i} \Omega^i G^i\right]_{G^s} \\
&= \left[\Lambda P \sum_{i=0}^{s-1} \binom{u}{i} \Lambda^{s-1-i} R^{u-i} \Omega^i G^i\right]_{G^s} \qquad (2.21)
\end{aligned}$$

and

$$\begin{aligned}(\Lambda R+\Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i &= \sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-i}R^{u+1-i}\Omega^i G^i+\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^{i+1}G^{i+1}\\ &= \Lambda^s R^{u+1}+\binom{u}{s-1}R^{u-s+1}\Omega^s G^s\\ &\quad+\sum_{i=1}^{s-1}\left(\binom{u}{i}+\binom{u}{i-1}\right)\Lambda^{s-i}R^{u+1-i}\Omega^i G^i\\ &= \binom{u}{s-1}R^{u-s+1}\Omega^s G^s+\sum_{i=0}^{s-1}\binom{u+1}{i}\Lambda^{s-i}R^{u+1-i}\Omega^i G^i\end{aligned}$$

This implies

$$\chi(s,u+1)=\left[(\Lambda R+\Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i\right]_{G^s}. \tag{2.22}$$

The second equation of the lemma follows directly from (2.21) and (2.22). □

A last lemma will be helpful now

Lemma 2.5. *For all nonnegative integers s and $u<q_s$*

$$\chi(s,u)P-\chi(s,u+1)\in_{coef}\mathcal{F}_{s+1} \tag{2.23}$$

$$\chi(s,u+1)_H\in_{coef}\mathcal{F}_{s+1}. \tag{2.24}$$

Proof. We will prove this lemma by induction on u. For $u\le s-1$ we observe from Lemma 2.4 that

$$\begin{aligned}\chi(s,u)P-\chi(s,u+1) &= \Lambda^{s-u-1}(\Lambda R+\Omega G)^u(\Lambda P-\Lambda R-\Omega G)\\ &\in_{\text{coef}}\mathcal{F}_{s+1}\end{aligned} \tag{2.25}$$

The last point follows from the fact that (2.25) implies that the coefficients of $\chi(s,u)P-\chi(s,u+1)$ are clearly in the space spanned by $\mathcal{S}$ once we multiply the original f_i's (i.e. the coefficients of $\Lambda P-\Lambda R-\Omega G$) by all monomials of degree $\le s-1$) because the coefficients of $\Lambda^{s-u-1}(\Lambda R+\Omega G)^u$ are polynomials of degree $\le s-1$ in the λ_i's.

This also implies that $\chi(s,u+1)_H\in_{\text{coef}}\mathcal{F}_{s+1}$, since $\deg\chi(s,u)P=ts+u(k-1)$. Now let us assume that $\chi(s,u-1)P-\chi(s,u)\in_{\text{coef}}\mathcal{F}_{s+1}$ and $\chi(s,u)_H\in_{\text{coef}}\mathcal{F}_{s+1}$, for some $s\le u<q_s$. From Lemma 2.4, we know that

$$[\chi(s,u)P-\chi(s,u+1)]_{G^s}=\left[(\Lambda P-\Lambda R-\Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i\right]_{G^s}.$$

Therefore

$$[\chi(s,u)P-\chi(s,u+1)]_{G^s}\in_{\text{coef}}\mathcal{F}_{s+1}$$

since clearly

$$(\Lambda P-\Lambda R-\Omega G)\sum_{i=0}^{s-1}\binom{u}{i}\Lambda^{s-1-i}R^{u-i}\Omega^i G^i\in_{\text{coef}}\mathcal{F}_{s+1}.$$

By the induction hypothesis $\chi(s,u)_H \in_{\text{coef}} \mathcal{F}_{s+1}$ and such coefficients have degree s, then the coefficients corresponding to degrees $> ts + (u+1)(k-1)$ of $\chi(s,u)P$ belong to $\mathcal{F}_{s+1}$ too. Since $[\chi(s,u)P - \chi(s,u+1)]_{G^s} = [\chi(s,u)P]_{G^s} - \chi(s,u+1)$, it follows that

$$\chi(s,u)P - \chi(s,u+1) \in \mathcal{F}_{s+1}.$$

Thus, we also have $\chi(s,u+1)_H \in \mathcal{F}_{s+1}$. □

We are ready now to prove Theorem 2.1.

Proof of Theorem 2.1. We proceed by induction on u_1 and u_2. We first observe that we trivially have $\chi(s_1,0)\chi(s_2,0) - \chi(s_1+s_2,0) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1}$ since

$$\chi(s_1,0)\chi(s_2,0) - \chi(s_1+s_2,0) = \Lambda^{s_1}\Lambda^{s_2} - \Lambda^{s_1+s_2} = 0.$$

Now assume that we have

$$\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1},$$

for some positive integers s_1 and s_2 and non-negative integers $u_1 < q_{s_1}$ and $u_2 \leq q_{s_2}$. Since $\chi(s_1,u_1)\chi(s_2,u_2)$ and $\chi(s_1+s_2,u_1+u_2)$ are polynomials where all coefficients are polynomials in the λ_i's of degree $\leq s_1+s_2$, we also have

$$P\left(\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2)\right) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1}. \tag{2.26}$$

By Lemma 2.5 we know that

$$P\chi(s_1,u_1) - \chi(s_1,u_1+1) \in_{\text{coef}} \mathcal{F}_{s_1+1}.$$

This implies

$$P\chi(s_1,u_1)\chi(s_2,u_2) - \chi(s_1,u_1+1)\chi(s_2,u_2) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1}. \tag{2.27}$$

On the other hand, still by Lemma 2.5, we have

$$P\chi(s_1+s_2,u_1+u_2) - \chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1}. \tag{2.28}$$

From (2.27) and (2.28) we derive that

$$-P\chi(s_1,u_1)\chi(s_2,u_2)+\chi(s_1,u_1+1)\chi(s_2,u_2)+P\chi(s_1+s_2,u_1+u_2)-\chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1} \tag{2.29}$$

(2.29) and (2.26) imply that

$$\chi(s_1,u_1+1)\chi(s_2,u_2) - \chi(s_1+s_2,u_1+u_2+1) \in_{\text{coef}} \mathcal{F}_{s_1+s_2+1}.$$

This proves the theorem by induction (the induction on u_2 follows directly from the fact we can exchange the role of u_1 and u_2). □

2.5 Experimental Results

In this section, we compare the behavior of a D-Gröbner basis computation on the bilinear system (2.3), with a system involving equations in the λ_j's only. We give examples where Johnson's bound is attained and passed.

The systems in λ_j's we use contains equations $\chi(s,u)_H$ and some relations $\chi(s,u)\chi(s',u') - \chi(s+s',u+u')$. Experimentally, they are linearly dependent from $\chi(s+s'-1,u+u')\chi(1,0) - \chi(s+s',u+u')$ and $\chi(s,q_s)_H$ (we recall that $\chi(1,0)=\Lambda$). Moreover, $\chi(s-1,u)\chi(1,0) \mod G^{s-1} = \chi(s,u) \mod G^{s-1}$, so we will consider equations $\mathcal{M}_{s,u}$ defined by

$$(\chi(s-1,u)\chi(1,0) - \chi(s,u)) \div G^{s-1}. \qquad (\mathcal{M}_{s,u})$$

We do not add equations that are polynomially dependent from $\chi(s,q_s)_H$ or $\mathcal{M}_{s+1,q_s}$ at degree at most D, and thus unnecessary for the computation.

Tables 2.1, 2.2 and 2.3 show results for $[n,k]_q$ taking values $[64,27]_{64}$, $[256,63]_{256}$ and $[37,5]_{61}$. The column $\#\lambda_j$ indicates the number of remaining λ_j's after elimination of the linear ones from the $\chi(1,*)_H$ relations. The column "Eq" indicates the equations used. The column "#Eq" contains the degrees of the equations[1].

We do our experiments using the `GroebnerBasis(S,D)` function in the computer algebra system `magma v2.25-6`. The practical complexity $\mathbb{C}$ is given by the `magma` function `ClockCycles`. For instance, on our machine with an Intel® Xeon® 2.00GHz processor, $2^{30.9}$ clock cycles are done in 1 second, $2^{36.8}$ in 1 minute and $2^{42.7}$ in 1 hour. "Max Matrix" indicates the size of the largest matrix during the process. The complexities include the computation of the equations $\chi(i,j)_H$ and $\mathcal{M}_{i,j}$ that could be improved.

For systems where the number of remaining λ_j's is small compared to the number of p_i's, e.g. Table 2.1 or Table 2.2, it is clearly interesting to compute a Gröbner basis for a system containing only polynomials in λ_j's: even if the maximal degree D is larger than for the bilinear system, the number of variables is much smaller and the computation is faster. For instance for $[n,k]_q = [64,27]_{64}$ in Table 2.1, on Johnson bound $t=23$ the Gröbner basis for the bilinear system requires more than 6 hours of computation and 47 GB of memory, whereas the computation in λ_j's only takes less than a second. For $t=24$ we couldn't solve the bilinear system directly, whereas the system in λ_j's only solves in less than a minute.

Table 2.2 gives an example where the number of λ_j's variables is quite large, but still smaller than the number of p_i's. The benefit of using equations in λ_j's only is clear. We can appreciate even more the experimental results from Table 2.2 if compared with the error-correction in for a multiplicity parameter equal to the maximal degree reached during the Gröbner basis computation. For instance, a $[n,k]_q = [256,63]_{256}$ GRS code can correct with our approach up to 120 errors by staying at degree 2, and thus by using only key-equations corresponding to multiplicity not higher than 2. On the other hand

$$\left\lfloor \max_v t_{pow}(2,v) \right\rfloor = \lfloor t_{pow}(2,4) \rfloor = 116,$$

i.e. using key-equations of degree at most 2, power decoding corrects in this case 4 errors less than 2-Gröbner basis.

[1] 2:45 means that the system contains 45 equations of degree 2.

Table 2.1: Experimental results for a $[n,k]_q = [64,27]_{64}$ RS-code. System (2.3) contains 26 variables p_i. Johnson's bound is $t = 23$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
19	1	(2.3)	2:45	2	65×57	$2^{22.2}$
		$\chi(2,3)_H$	2:11	2	45×28	$2^{23.7}$
20	3	(2.3)	2:46	3	1522×1800	$2^{26.5}$
		$\chi(2,3)_H$	2:9	2	47×28	$2^{24.4}$
21	5	(2.3)	2:47	3	1711×2889	$2^{27.1}$
		$\chi(2,3)_H + \chi(3,4)_H$	2:7, 3:24	3	66×56	$2^{26.8}$
22	7	(2.3)	2:48	4	31348×35972	$2^{36.1}$
		$\chi(2,3)_H + \chi(3,4)_H$	2:5, 3:21	4	271×283	$2^{27.6}$
23	9	(2.3)	2:49	5	428533×406773	$2^{45.4}$
		$\chi(2,3)_H + \mathcal{M}_{3,3}$	2:4, 3:22	5	1466×1641	$2^{30.1}$
24	11	(2.3)	2:50	≥ 6	–	–
		$\mathcal{M}_{3,3}$	2:1, 3:23	7	28199×23536	$2^{35.8}$

Table 2.2: Experimental results for a $[n,k]_q = [256,63]_{256}$ RS-code. System (2.3) contains 62 variables p_i. Johnson's bound is $t = 130$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
120	36	(2.3)	2:182	3	20023×128018	$2^{38.0}$
		$\chi(2,3)_H$	2:85	2	119×703	$2^{34.5}$
121	39	(2.3)	2:183	3	21009×143741	$2^{38.9}$
		$\mathcal{M}_{2,2}$	2:111	3	9780×8517	$2^{35.0}$
122	42	(2.3)	2:184	3	22050×160434	$2^{39.7}$
		$\mathcal{M}_{2,2}$	2:113	3	4858×14189	$2^{35.3}$
123	45	(2.3)	2:185	3	23112×178090	$2^{40.1}$
		$\mathcal{M}_{2,2}$	2:115	3	5289×17295	$2^{35.8}$
124	48	(2.3)	2:186	≥ 4	–	–
		$\mathcal{M}_{2,2}$ + $\mathcal{M}_{4,6}$	2:117, 3:1, 4:189	4	164600×270725	$2^{45.2}$

On the contrary, Table 2.3 shows that for a small value of k compared to the number of λ_j's, the maximal degree for the bilinear system is smaller than the one for a system involving only λ_j's, but the total number of variables is almost the same, hence it is more interesting to solve directly the bilinear system. Moreover, here computing the $\mathcal{M}_{i,j}$ equations (that are equations in λ_j's of degree i) takes time. Note that, for $t \geq 26$ we may have several solutions: the Gröbner basis computation performs a list decoding and returns all the solutions.

2.6 Conclusions

This chapter shows why a Gröbner basis computation on the bilinear system (2.3) for decoding a Reed-Solomon code is of polynomial complexity below Sudan's radius. The Gröbner basis computation reveals polynomial equations of small degree involving only the coefficients λ_i of the error locator polynomial, i.e. they do not depend on

Table 2.3: Experimental results for a $[n,k]_q = [37,5]_{61}$ RS-code. System (2.3) contains 4 variables p_i. Johnson's bound is $t = 24$, Gilbert-Varshamov's bound is $t = 28$.

t	$\#\lambda_j$	Eq.	#Eq.	D	Max Matrix	$\mathbb{C}$
24	12	(2.3)	2:28	3	1065×1034	$2^{26.0}$
		$\mathcal{M}_{2,3}$	2:37	3	454×454	$2^{28.0}$
25	15	(2.3)	2:29	3	2520×1573	$2^{28.0}$
		$\chi(2,5)_H+$	2:25,	4	3193×3311	$2^{34.3}$
		$\chi(3,8)_H+$	3:40			
		$\mathcal{M}_{2,2} + \mathcal{M}_{3,5}$				
26	18	(2.3)	2:30	4	20446×15171	$2^{33.1}$
		$\chi(2,5)_H+$	2:25,	5	38796×22263	$2^{38.1}$
		$\mathcal{M}_{2,2} + \mathcal{M}_{3,5}$	3:37,			
		$+ \mathcal{M}_{4,8}$	4:37			
27	21	(2.3)	2:31	4	27366×24894	$2^{36.0}$

the block of p_i's variables. These polynomial equations are derived by manipulations of power decoding key-equations [Nie18]. We give a theorem explaining why these polynomial relations are obtained at a surprisingly small degree. This is a first step for understanding why the Gröbner basis approach still works well beyond the Sudan radius and is successful by staying at a small degree. We have also explored an alternative strategy, namely feeding directly the initial system with some of the aforementioned polynomial relations before running a Gröbner basis algorithm. This results in some cases in a considerable complexity gain. We have considered some of the examples given in [Nie18] and shown that the latter can be outperformed by our Gröbner basis approach when comparing the Gröbner basis degree with the multiplicity parameter used in the power decoding algorithm. Especially for small parameters, we experimentally demonstrated that it can still be effective slightly beyond Johnson's bound, which is a no-go for power decoding, for any possible multiplicity. This means that the power decoding approach does not fully take advantage of the polynomial equations and some of the information provided by key-equations (2.13) and (2.12) is lost after linearization. At the same time, breaking the barrier of Johnson's bound encourages further research on decoding techniques based on polynomial equation solving. Moreover, even above the unique decoding radius and contrarily to the power decoding approach, the Gröbner basis computation is also able to compute all solutions. This approach opens new roads for decoding algebraically a Reed-Solomon code.

Chapter 3
On the dimension the square of the dual of alternant and Goppa codes

In this chapter, we revisit a distinguisher for high-rate alternant and Goppa codes through a new approach, namely by studying the dimension of square codes. This partially solves the Goppa Code Distinguishing (GD) problem, which asks to distinguish efficiently a generator matrix of a Goppa code from a randomly drawn one. We provide here a rigorous upper bound for the dimension of the square of the dual of an alternant or Goppa code, while the previous approach only provided algebraic explanations based on heuristics. Moreover, for Goppa codes, our proof extends to the non-binary case as well, thus providing an algebraic explanation for the distinguisher which was missing up to now. All the upper bounds are tight and match experimental evidence. Our work also introduces new algebraic results about products of trace codes in general and of dual of alternant and Goppa codes in particular, clarifying their square code structure. For instance, we will show that the square of the dual of a Goppa code is contained into the dual of an alternant code of large degree, and for some parameters they even coincide. In Chapter 4 we will see how some of these structural results can serve for cryptanalysis purposes.

Contents

3.1 Introduction

3.1.1 A distinguisher for high-rate alternant and Goppa codes

In Section 1.2.5 we have described the McEliece encryption scheme [McE78], the oldest code-based cryptosystem, dating back to 1978. It benefits from very fast encryption and decryption algorithms and has very small ciphertexts. Despite the very general framework of this quantum-safe cryptosystem, we are here interested in its original version, the one built upon the family of binary Goppa codes, or the generalized variant based on alternant codes over any field. We also recall the existence of the CFS scheme [CFS01], a code-based digital signature, which relies on high-rate Goppa codes. The reasons why Goppa codes should belong to such a rate regime have already been explained in Section 1.2.8.1.

We have also had a high-level overview of *message-recovery attacks*, which consist in inverting the McEliece encryption without finding a trapdoor but making use of general decoding algorithms. Despite being the primary threat to consider when designing parameters, they have exponential complexity, even with a quantum speedup, and it is unlikely that future improvements would lead to a complete break of the McEliece scheme.

Therefore the original McEliece cryptosystem [McE78] based on binary Goppa codes remains, after more than forty years, the oldest unbroken quantum-secure public-key cryptosystem.

If one hopes for a polynomial (or even a subexponential) time attack, it will be presumably necessary to exploit the non-random structure of the secret code. As already explained in the preliminaries, the other macro-family of attacks is made by *key-recovery attacks*, where the intruder seeks to recover the private key. For this purpose, the first step is being able to detect the presence of this special structure from the public key. For a long time it was widely believed that even this simpler task of distinguishing efficiently a generator matrix of a Goppa code from a randomly drawn generator matrix with non-negligible probability was unfeasible. This is the so-called *Goppa Code Distinguishing (GD) problem* as introduced by the authors of [CFS01]. The nice feature of this problem is that it is possible to devise a security proof for the McEliece scheme based solely on the intractability of this problem and decoding a generic linear code [Sen10]. The belief about GD problem hardness was justified by the fact that Goppa codes behave like random codes in many aspects. For instance, they asymptotically meet the Gilbert-Varshamov bound, their weight distribution is roughly the same as those of random codes and they generally have a trivial permutation group. The absence of significant breakthroughs in key-recovery attacks also strengthened the idea that the Goppa Code distinguishing problem is difficult. This problem was used for a long time as a problem that basically captures the hardness of recovering the private key of a Goppa code.

However, this belief was severely questioned in [Fau+11; Fau+13] which gave a polynomial time algorithm that distinguishes between Goppa codes (or more generally alternant codes) and random ones from their generator matrices at least for very high rate codes. It is based on the kernel of a linear system related to an algebraic system that encodes the key-recovery problem for the McEliece cryptosystem instantiated with alternant or Goppa codes. Indeed, it was shown to have an unexpectedly high dimension. This distinguisher was later on given another interpretation in [MP12],

where it was proved that this dimension is related to the dimension of the square of the dual of the public code. The algebraic explanations given in [Fau+13] do not represent however a rigorous proof of the dimension of the kernel sought, but they rely on heuristic considerations. Indeed, while a set of vectors is proposed as a candidate for the kernel basis, its elements are neither proved to be independent nor a set of generators. Although the experiments run in [Fau+13] show a regular behavior when alternant codes are defined by picking at random support and multiplier vectors, it is possible to artificially choose alternant codes whose kernel dimensions are even larger than for random ones. Moreover, in the case of Goppa codes, even if a general formula for the dimension of the kernel was provided that matches the experimental evidence, an algebraic explanation was only provided in the case of binary Goppa codes with square-free Goppa polynomials. This explanation crucially relies on Theorem 1.2. Clearly, this approach does not generalize to non-binary Goppa codes.

There exist already many examples where a distinguisher has then been turned into an attack. In the code-based cryptography setting, this is for instance the case for GRS codes. The uncommon dimension of the square of a GRS code leads to a successful key recovery for several proposed variants of the McEliece cryptosystem built upon this family of codes for any rate [Cou+14]. Despite the strong relationship between generalized Reed-Solomon codes and alternant codes, the same attacks cannot be carried over from the former to the latter, because of the additional subfield subcode structure. A similar idea has been successfully exploited for Wild Goppa codes though [COT17]. But in this case, the distinguisher is based on considerations of the square of Goppa codes themselves, which only apply to a very restricted class of parameters. Indeed the attack can only work for extensions of degree $m = 2$ and there is no way to go beyond it, because for $m > 2$ the square code fills the whole space. In our case, our distinguisher is based on squaring the *dual* of a Goppa code (or an alternant code) and works for *any* field extension degree.

3.1.2 Our contribution

In the present article, we revisit the distinguisher for random alternant codes and Goppa codes. We do so by exploiting the link given by [MP12]. Indeed we provide a rigorous upper bound on the dimension of the square code of the dual of an alternant or a Goppa code that coincides with the experiments. By using [MP12], this also gives a lower bound on the dimension of the kernel of the matrix considered in [Fau+13]. Together with results about the typical dimension of the square of random codes [Cas+15], this provides the first rigorous analysis of the effectiveness of the approach pioneered in [Fau+11], because the typical dimension of the square of a random code is way larger than this upper-bound on the dimension of the square of the dual of a Goppa or alternant code.

Our approach relies on several new ingredients

- a new result about the square of trace codes showing that if essentially the square of a code is abnormally small then the square of its trace code is also abnormally small in a certain region of parameters. Interpreting the dual of an alternant code or a Goppa code as the trace of a generalized Reed-Solomon code (whose dimension of the square is known to be abnormally small [Wie10]) shows

Classic McEliece parameter sets [Ber+19]	n	m	r	R	Largest distinguishable r	Corresponding R
kem/mceliece348864	3488	12	64	0.77982	12	0.95872
kem/mceliece460896	4608	13	96	0.72917	12	0.96615
kem/mceliece6688128	6688	13	128	0.75120	15	0.97084
kem/mceliece6960119	6960	13	119	0.77773	16	0.97011
kem/mceliece8192128	8192	13	128	0.79688	19	0.96985

Table 3.1: Comparison between Classic McEliece and smallest distinguishable code rates. This table provides a comparison between the parameters proposed for Classic McEliece and the largest order r of a binary Goppa code that is distinguishable, with the corresponding relative rate R. Such an order can be computed as the maximum value of r for which the upper bound of the dimension of the square of a binary Goppa code given in Corollary 3.6 (Classic McEliece uses binary Goppa codes) is strictly smaller than $\min\left(n, \binom{mr+1}{2}\right)$. The latter quantity is the typical dimension of the square of a random code of the same length and dimension as the corresponding Goppa code (see [Cas+15]). We see in this table that the code rates suggested for Classic McEliece oscillate between 0.7 and 0.8 [Ber+19, Section 3], while for the same length n and degree of the field extension m, the distinguisher works for rates closer to 1, meaning that the Goppa order r must be smaller.

that the square of a dual of an alternant code or a Goppa code is abnormally small.

- While this approach explains rigorously why alternant codes or Goppa codes can be distinguished for extremely large rates, lower rates require a much more delicate analysis, in particular in the Goppa case. We do so, by noticing that the square of a trace of a code $\mathscr{C}$ can be interpreted as a sum of traces of products of $\mathscr{C}$ with $\mathscr{C}^{q^i}$ (which denotes i applications of the Frobenius map to $\mathscr{C}$). In the case of Goppa codes, we show that the traces of these products turn out to be duals of alternant codes of a remarkably low degree at least for small values of i (see Theorem 3.3). This accounts for the remarkably low dimension of the square of the dual of Goppa codes in all cases considered in [Fau+13].

Interestingly enough, the latter argument applies to *all* kinds of Goppa codes, be they binary or not and provides now not only a rigorous explanation of the distinguisher found in [Fau+11], but covers the non-binary Goppa code case as well. Note that even if this approach is not able to distinguish the Goppa codes proposed in the NIST competition as shown in Table 3.1, because it only works for very high rate Goppa codes, this still raises the issue of whether this distinguishing approach can be improved to lower the dimension of the Goppa codes that can be distinguished by this approach. A better understanding of the distinguisher obtained here might help to address this issue.

3.2 The relationship between the distinguisher of [Fau+11; Fau+13] and the square code construction

Analogously to the square code distinguisher for GRS codes seen in Section 1.1.5, the dual of an alternant (or Goppa) code can also be distinguished from random codes when the primal code has a high enough rate. This phenomenon was already observed in [Fau+11]. Here, however, the distinguisher was presented in terms of the kernel dimension of a linear system obtained by properly linearizing the algebraic system that encodes the key-recovery problem for the McEliece cryptosystem endowed with alternant or Goppa codes. More precisely, let $\boldsymbol{P} = (p_{i,j})_{\substack{1\leq i\leq k\\ 1\leq j\leq n}}$ be a generator matrix of an $[n,k]$ alternant (or Goppa) code $\mathscr{C}$ in systematic form, i.e. with its first k columns that form an identity block. In other words, $p_{i,i} = 1$ for any $i \in [\![1,k]\!]$ and $p_{i,j} = 0$ for any $i,j \in [\![1,k]\!], i \neq j$. Therefore the generator matrix can be written as $\boldsymbol{P} = [\boldsymbol{I}_k \mid \boldsymbol{P}']$ with $\boldsymbol{P}' = (p_{i,j})_{\substack{1\leq i\leq k\\ k+1\leq j\leq n}} \in \mathbb{F}_{q^m}^{k\times(n-k)}$. The following linear system turns out from the linearization of an algebraic system that models the support and multiplier recovery problem for alternant/Goppa codes.

Modeling 3.1 (Alternant/Goppa codes modeling [Fau+13], linearized)**.**
System:

$$\mathscr{L}_{\boldsymbol{P}} \stackrel{def}{=} \left\{ \sum_{(j,j')\in J} p_{i,j}p_{i,j'}Z_{j,j'} = 0 \mid i \in [\![1,k]\!] \right\}, \tag{3.1}$$

where $J = \{(j,j') \in \mathbb{N}^2 \mid k+1 \leq j < j' \leq n\}$.
Unknowns: $\binom{n-k}{2}$ *unknowns* $Z_{j,j'}$, $(j,j') \in J$.
Equations: k *homogeneous linear equations over the subfield* $\mathbb{F}_q$.

Recall from (1.2) that a generator matrix of $\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y}) \subseteq \mathbb{F}_{q^m}^n$ is given by

$$\boldsymbol{V}_r(\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \begin{bmatrix} y_1 & \cdots & y_n \\ y_1x_1 & \cdots & y_nx_n \\ \vdots & \ddots & \vdots \\ y_1x_1^{r-1} & \cdots & y_nx_n^{r-1} \end{bmatrix}.$$

Therefore, by definition of alternant codes (see 1.16), it readily follows that

$$\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y}) \stackrel{\text{def}}{=} \left\{ \boldsymbol{c} \in \mathbb{F}_q^n \mid \boldsymbol{V}_r(\boldsymbol{x},\boldsymbol{y})\boldsymbol{c}^T = \boldsymbol{0} \right\} \tag{3.2}$$

Again, if $\boldsymbol{y} \stackrel{\text{def}}{=} \left(\frac{1}{\Gamma(x_1)}, \ldots, \frac{1}{\Gamma(x_n)}\right)$, for some polynomial Γ of degree r, then $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ is a Goppa code.

In [Fau+13], the expression of $Z_{j,j'}$ in terms of the variables $\{X_i : i \in [\![1,n]\!]\}$ and $\{Y_i : i \in [\![1,n]\!]\}$, representing respectively the support and multiplier coordinates, is given for field sizes in even characteristic. The computation is not given explicitly. In the proof of the next proposition, we write down the computation that leads to the algebraic system from [Fau+13] in the proof of the following proposition, generalizing the formula to fields of odd characteristic.

Proposition 3.1. *Let* $\boldsymbol{X} \stackrel{def}{=} \{X_i, \ldots, X_n\}$ *and* $\boldsymbol{Y} \stackrel{def}{=} \{Y_1, \ldots, Y_n\}$ *be two blocks of* n *unknowns each corresponding to the support and multiplier coordinates respectively of* $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$*. Assume* $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ *admits a systematic form and let* $\boldsymbol{P} = [\boldsymbol{I}_k \mid \boldsymbol{P}'] = (p_{i,j})_{i,j}$ *be its generator matrix. Then* $(\boldsymbol{x}, \boldsymbol{y})$ *is a solution of the following algebraic system:*

$$\left\{ \sum_{(j,j')\in J} p_{i,j}p_{i,j'}(X_j^\delta - X_{j'}^\delta)^{q^\ell}(Y_jY_{j'}^{q^\ell}X_j^cX_{j'}^{bq^\ell} - Y_{j'}Y_j^{q^\ell}X_{j'}^cX_j^{bq^\ell}) = 0 \right.$$
$$\left. \mid i \in [\![1,k]\!], b \in [\![0,r-2]\!], \delta \in [\![1,r-1-b]\!], c+\delta q^\ell \in [\![0,r-1]\!] \right\}, \tag{3.3}$$

where $J = \{(j,j') \in \mathbb{N}^2 \mid k+1 \leq j < j' \leq n\}$*. The linear system* $\mathscr{L}_{\boldsymbol{P}}$ *from Modeling 3.1 stems from the linearization*

$$Z_{j,j'} \stackrel{def}{=} (X_j^\delta - X_{j'}^\delta)^{q^\ell}(Y_jY_{j'}^{q^\ell}X_j^cX_{j'}^{bq^\ell} - Y_{j'}Y_j^{q^\ell}X_{j'}^cX_j^{bq^\ell}),$$

for an arbitrary admissible choice of b, δ, b, c *and* ℓ*.*

Proof. The alternative definition of alternant codes from (3.2) implies that

$$\boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y})\boldsymbol{P}^T = \boldsymbol{0}_{r\times k}.$$

This matrix equation translates into the polynomial system

$$\left\{ \sum_{j=1}^{n} g_{i,j}Y_jX_j^e = 0 \mid i \in [\![1,k]\!], e \in [\![0,r-1]\!] \right\}, \tag{3.4}$$

where $\boldsymbol{X} \stackrel{\text{def}}{=} \{X_i, \ldots, X_n\}$ and $\boldsymbol{Y} \stackrel{\text{def}}{=} \{Y_1, \ldots, Y_n\}$ are two blocks of n unknowns each corresponding to the support and multiplier coordinates respectively. Observe that the sought vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy indeed the polynomial system.

Under the assumption that $\boldsymbol{P}$ is in systematic form, the polynomial system (3.4) can be rewritten as

$$\left\{ Y_iX_i^e = -\sum_{j=k+1}^{n} p_{i,j}Y_jX_j^e \mid i \in [\![1,k]\!], e \in [\![0,r-1]\!] \right\}. \tag{3.5}$$

It is possible to derive quadratic relations among the monomials on the left-hand sides of the various polynomial equations from the system in (4.20):

$$(Y_iX_i^a)(Y_iX_i^b)^{q^\ell} = (Y_iX_i^c)(Y_iX_i^d)^{q^\ell}, \tag{3.6}$$

for any 5-tuple (a,b,c,d,ℓ) of integers such that $a + b^{q^\ell} = c + d^{q^\ell}$, with $a, b, c, d \in [\![0, r-1]\!]$ and $\ell \in [\![0, m-1]\!]$. Without loss of generality, we can assume that $d > b$ (which implies $a > c$). We define $\delta \stackrel{\text{def}}{=} d - b$, hence $a = c + \delta q^l$. Hence $\delta \in [\![1, r-1-b]\!], b \in [\![0, r-2]\!]$ and $c + \delta q^l \in [\![0, r-1]\!]$. By replacing the monomials $Y_iX_i^e$ appearing in these identities with the corresponding right-hand side expressions of (4.20), we obtain

$$\left(-\sum_{j=k+1}^{n} p_{i,j}Y_jX_j^a\right)\left(-\sum_{j=k+1}^{n} p_{i,j}Y_jX_j^b\right)^{q^\ell} = \left(-\sum_{j=k+1}^{n} p_{i,j}Y_jX_j^c\right)\left(-\sum_{j=k+1}^{n} p_{i,j}Y_jX_j^d\right)^{q^\ell}.$$

By generalizing the computation done in [Fau+13] for any field size q, we get

$$
\begin{aligned}
0 &= \left(-\sum_{j=k+1}^{n} p_{i,j} Y_j X_j^a\right)\left(-\sum_{j'=k+1}^{n} p_{i,j'} Y_{j'} X_{j'}^b\right)^{q^\ell} - \left(-\sum_{j=k+1}^{n} p_{i,j} Y_j X_j^c\right)\left(-\sum_{j'=k+1}^{n} p_{i,j'} Y_{j'} X_{j'}^d\right)^{q^\ell} \\
&= \left(\sum_{\substack{j\in[\![k+1,n]\!]\\ j'\in[\![k+1,n]\!]}} p_{i,j}p_{i,j'} Y_j Y_{j'}^{q^\ell} X_j^a X_{j'}^{bq^\ell}\right) - \left(\sum_{\substack{j\in[\![k+1,n]\!]\\ j'\in[\![k+1,n]\!]}} p_{i,j}p_{i,j'} Y_j Y_{j'}^{q^\ell} X_j^c X_{j'}^{dq^\ell}\right) \\
&= \left(\sum_{(j,j')\in J} p_{i,j}p_{i,j'}(Y_j Y_{j'}^{q^\ell} X_j^a X_{j'}^{bq^\ell} + Y_{j'} Y_j^{q^\ell} X_{j'}^a X_j^{bq^\ell})\right) + \left(\sum_{j=k+1}^{n} p_{i,j}^2 Y_j^{1+q^\ell} X_j^a X_{j'}^{bq^\ell}\right) \\
&\quad - \left(\sum_{(j,j')\in J} p_{i,j}p_{i,j'}(Y_j Y_{j'}^{q^\ell} X_j^c X_{j'}^{dq^\ell} + Y_{j'} Y_j^{q^\ell} X_{j'}^a X_j^{bq^\ell})\right) - \left(\sum_{j=k+1}^{n} p_{i,j}^2 Y_j^{1+q^\ell} X_j^c X_{j'}^{dq^\ell}\right) \\
&= \sum_{(j,j')\in J} p_{i,j}p_{i,j'}(Y_j Y_{j'}^{q^\ell} X_j^{c+\delta q^\ell} X_{j'}^{bq^\ell} + Y_{j'} Y_j^{q^\ell} X_{j'}^{c+\delta q^\ell} X_j^{bq^\ell} - Y_j Y_{j'}^{q^\ell} X_j^c X_{j'}^{bq^\ell+\delta q^\ell} - Y_{j'} Y_j^{q^\ell} X_{j'}^c X_j^{bq^\ell+\delta q^\ell}) \\
&= \sum_{(j,j')\in J} p_{i,j}p_{i,j'}(X_j^\delta - X_{j'}^\delta)^{q^\ell}(Y_j Y_{j'}^{q^\ell} X_j^c X_{j'}^{bq^\ell} - Y_{j'} Y_j^{q^\ell} X_{j'}^c X_j^{bq^\ell}).
\end{aligned}
$$

Fix a 5-tuple (a, b, c, d, l). Then (3.3) coincides with (3.1) through the change of variables

$$Z_{j,j'} \stackrel{\text{def}}{=} (X_j^\delta - X_{j'}^\delta)^{q^\ell}(Y_j Y_{j'}^{q^\ell} X_j^c X_{j'}^{bq^\ell} - Y_{j'} Y_j^{q^\ell} X_{j'}^c X_j^{bq^\ell}).$$

□

Assuming here and in the following that $k = n - rm$, the system $\mathscr{L}_{\boldsymbol{P}}$ has $\binom{n-k}{2} = \binom{rm}{2}$ linear variables $Z_{j,j'}$'s and k equations (one for each row of the generator matrix $\boldsymbol{P}$). A generic linear system (i.e. a linear system whose coefficients are independent variables) with the same number of equations and unknowns has a rank equal to $\min(k, \binom{rm}{2})$. In other words, the dimension of the solution space, i.e. the kernel, of a generic linear system is

$$k - \min\left(k, \binom{rm}{2}\right) = \max\left(0, k - \binom{rm}{2}\right).$$

This means that if we replace the $p_{i,j}$'s with random entries $r_{i,j}$'s, we would expect the dimension of the solution space of the arising system $\mathscr{L}_{\boldsymbol{R}}$ not to deviate much from the generic setting. Therefore it is reasonable to expect an analogous behavior for $\mathscr{L}_{\boldsymbol{P}}$. However, the dimension of $\mathrm{Ker}(\mathscr{L}_{\boldsymbol{P}})$ as an $\mathbb{F}_q$-vector space turns out to be much smaller in the case of high-rate Goppa or alternant codes than it is for random codes. A conjecture for the value of $\dim_{\mathbb{F}_q} \mathrm{Ker}(\mathscr{L}_{\boldsymbol{P}})$ coinciding with experimental evidence was given in [Fau+13] together with a convincing algebraic explanation for alternant and binary Goppa codes.

It has been proved in [MP12] that $\dim_{\mathbb{F}_q} \mathrm{Ker}(\mathscr{L}_{\boldsymbol{P}})$ is related to the dimension of the square of the dual code $\mathscr{C}^\perp$. The rationale behind this result is quite intuitive once a set of generators for $(\mathscr{C}^\perp)^{\star 2}$ is taken. More specifically, it follows from (the proof of) [MP12, Proposition 1] that

Proposition 3.2. *Let $\boldsymbol{P} = [\boldsymbol{I}_k \mid \boldsymbol{P}'] = (p_{i,j})_{i,j}$ be the generator matrix of the code $\mathscr{C}$. Then*

$$\dim_{\mathbb{F}} \left(\mathscr{C}^{\perp}\right)^{\star 2} = \binom{\dim_{\mathbb{F}}(\mathscr{C}^{\perp}) + 1}{2} - \dim_{\mathbb{F}_q} \operatorname{Ker}(\mathscr{L}_{\boldsymbol{P}}). \tag{3.7}$$

Proof. Since $\boldsymbol{P}$ is in systematic form, we can immediately derive the parity-check matrix of $\mathscr{C}$, i.e. the generator matrix of $\mathscr{C}^{\perp}$, as $\boldsymbol{H} = [-\boldsymbol{P}^T \mid \boldsymbol{I}_{n-k}]$. Let $\boldsymbol{e}_j \in \mathbb{F}_{q^m}^{n-k}, j \in [\![k+1, n]\!]$, be the canonical vector with respect to the index $j-k$. The component-wise product of all rows of $\boldsymbol{H}$ produces the rows

$$(p_{j,1}^2 \dots p_{j,k}^2 \mid \boldsymbol{e}_j) \quad \text{for any } j \in [\![k+1, n]\!]$$

and

$$(p_{j,1}^2 \dots p_{j',k}^2 \mid \boldsymbol{0}_{n-k}) \quad \text{for any } k+1 \le j < j' \le n.$$

Let $\boldsymbol{R}_1 \stackrel{\text{def}}{=} (p_{j,i}^2)_{\substack{k+1 \le j \le n \\ 1 \le i \le k}}$ and $\boldsymbol{R}_2 \stackrel{\text{def}}{=} (p_{j,i}p_{j',i})_{\substack{k+1 \le j < j' \le n \\ 1 \le i \le k}}$ and note that $\mathbf{Rank}(\boldsymbol{R}_2) = \mathbf{Rank}(\boldsymbol{R}_2^T) = \mathbf{Rank}(\mathscr{L}_{\boldsymbol{P}})$. Then

$$\begin{aligned}
\dim_{\mathbb{F}} \left(\mathscr{C}^{\perp}\right)^{\star 2} &= \mathbf{Rank}\left(\left[\begin{array}{c|c} \boldsymbol{R}_1 & \boldsymbol{I}_{n-k} \\ \hline \boldsymbol{R}_2 & \boldsymbol{0}_{\binom{k}{2} \times k} \end{array}\right]\right) \\
&= (n-k) + \mathbf{Rank}\left(\boldsymbol{R}_2\right) \\
&= (n-k) + \mathbf{Rank}(\mathscr{L}_{\boldsymbol{P}}) \\
&= (n-k) + \binom{n-k}{2} - \dim_{\mathbb{F}_q} \operatorname{Ker}(\mathscr{L}_{\boldsymbol{P}}) \\
&= \binom{n-k+1}{2} - \dim_{\mathbb{F}_q} \operatorname{Ker}(\mathscr{L}_{\boldsymbol{P}}) \\
&= \binom{\dim_{\mathbb{F}} \mathscr{C}^{\perp} + 1}{2} - \dim_{\mathbb{F}_q} \operatorname{Ker}(\mathscr{L}_{\boldsymbol{P}}).
\end{aligned}$$

□

In terms of dimensions of the square codes, the formula for $\dim_{\mathbb{F}_q} \operatorname{Ker}(\mathscr{L}_{\boldsymbol{P}})$ given in [Fau+13] together with (3.7) predicts that

Conjecture 3.1. *For a generic alternant code $\mathbb{F}_q$ of length n and extension degree m we have*

$$\dim_{\mathbb{F}_q}(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp})^{\star 2} = \min\left\{n, \binom{rm+1}{2} - \frac{m}{2}(r-1)\left((2e_{\mathscr{A}}+1)r - 2\frac{q^{e_{\mathscr{A}}+1}-1}{q-1}\right)\right\}, \tag{3.8}$$

whereas for a generic Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma)$ of length n over $\mathbb{F}_q$ with Goppa polynomial $\Gamma(X) \in \mathbb{F}_{q^m}[X]$ of degree r:

$$\dim(\mathscr{G}(\boldsymbol{x}, \Gamma)^{\perp})^{\star 2} = \min\left\{n, \binom{rm+1}{2} - \frac{m}{2}(r-1)(r-2)\right\}, \quad \text{if } r < q-1 \tag{3.9}$$

$$\dim(\mathscr{G}(\boldsymbol{x}, \Gamma)^{\perp})^{\star 2} = \min\left\{n, \binom{rm+1}{2} - \frac{m}{2}r\left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right)\right\}, \quad \text{else,} \tag{3.10}$$

where $e_{\mathscr{A}}$ and $e_{\mathscr{G}}$ are respectively defined by

$$e_{\mathscr{A}} \stackrel{def}{=} \max\{i \in \mathbb{N} \mid r \geq q^i + 1\} = \left\lfloor \log_q(r-1) \right\rfloor \tag{3.11}$$

$$e_{\mathscr{G}} \stackrel{def}{=} \min\{i \in \mathbb{N} \mid r \leq (q-1)^2 q^i\} + 1 = \left\lceil \log_q\left(\frac{r}{(q-1)^2}\right)\right\rceil + 1. \tag{3.12}$$

As shown in [Fau+13], these formulas agree with extensive experimental evidence. Notice that even if [Fau+13] did not prove the validity of these formulas, it gave algebraic explanations making them very likely. One of the aims of this chapter is to rigorously prove that at least the $\leq$ inequality is ensured in (3.8), (3.9) and (3.10).

3.3 A general result about the square of a trace code

The dual of alternant codes and Goppa codes are trace codes of GRS codes. From Proposition 1.7 we know that square codes of GRS codes have an abnormally small dimension. A natural question is whether or not this implies that the square of the trace of a GRS code has itself a small dimension. More generally, this raises the following fundamental issue of whether or not when the product of two codes $\mathscr{C}$ and $\mathscr{D}$ over $\mathbb{F}_{q^m}$ of length n is smaller than $\min(n, \dim_{\mathbb{F}_{q^m}} \mathscr{C} \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{D})$ (which is the dimension we expect for random codes $\mathscr{C}$ and $\mathscr{D}$) then this property survives for trace codes, namely do we have in this case

$$\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D}) < \min(n, \dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) \cdot \dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{D}))?$$

This is related to open questions raised in [Ran15, p. C.4]. This is indeed the case up to some extent, due to the following proposition:

Proposition 3.3. *Let $\mathscr{C}$ and $\mathscr{D}$ be two linear codes over $\mathbb{F}_{q^m}$ with the same length n. Then*

$$\mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D}) \subseteq \sum_{i=0}^{m-1} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{D}^{q^i}\right), \quad \textit{where } \mathscr{D}^{q^i} \stackrel{def}{=} \{d^{q^i} \mid d \in \mathscr{D}\}.$$

Proof. It is readily verified that $\mathscr{D}^{q^i}$ is a linear code over $\mathbb{F}_{q^m}$. Let $c, d \in \mathbb{F}_{q^m}$. We have

$$\begin{aligned}
\mathrm{Tr}(c) \cdot \mathrm{Tr}(d) &= \left(\sum_{0 \leq i \leq m-1} c^{q^i}\right) \cdot \left(\sum_{0 \leq i \leq m-1} d^{q^i}\right) \\
&= \sum_{\substack{0 \leq i \leq m-1 \\ 0 \leq j \leq m-1}} c^{q^i} \cdot d^{q^j} \\
&= \sum_{0 \leq j \leq m-1} \sum_{0 \leq i \leq m-1} c^{q^i} \cdot d^{q^{(i+j \mod m)}} \\
&= \sum_{0 \leq j \leq m-1} \sum_{0 \leq i \leq m-1} c^{q^i} \cdot d^{q^{i+j}} \\
&= \sum_{0 \leq i \leq m-1} \mathrm{Tr}(c \cdot d^{q^i}).
\end{aligned}$$

Because the trace acts component-wise on vectors, we also have for $\boldsymbol{c}, \boldsymbol{d} \in \mathbb{F}_q^n$,

$$\mathrm{Tr}(\boldsymbol{c}) \star \mathrm{Tr}(\boldsymbol{d}) = \sum_{0 \le i \le m-1} \mathrm{Tr}(\boldsymbol{c} \star \boldsymbol{d}^{q^i}).$$

Hence

$$\begin{aligned}
\mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D}) &= \langle\, \mathrm{Tr}(\boldsymbol{c}) \star \mathrm{Tr}(\boldsymbol{d}) \mid \boldsymbol{c} \in \mathscr{C}, \boldsymbol{d} \in \mathscr{D} \,\rangle_{\mathbb{F}_q} \\
&= \left\langle \sum_{i=0}^{m-1} \mathrm{Tr}\left(\boldsymbol{c} \star \boldsymbol{d}^{q^i}\right) \mid \boldsymbol{c} \in \mathscr{C}, \boldsymbol{d} \in \mathscr{D} \right\rangle_{\mathbb{F}_q} \\
&\subseteq \sum_{i=0}^{m-1} \left\langle \mathrm{Tr}\left(\boldsymbol{c} \star \boldsymbol{d}^{q^i}\right) \mid \boldsymbol{c} \in \mathscr{C}, \boldsymbol{d} \in \mathscr{D} \right\rangle_{\mathbb{F}_q} \\
&= \sum_{i=0}^{m-1} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{D}^{q^i}\right).
\end{aligned}$$

□

Note that for an $\mathbb{F}_{q^m}$-linear code $\mathscr{C}$, $\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) \le \min(m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}, n)$, where n is the code length of $\mathscr{C}$ and $\mathscr{D}$, and equality generally holds. An easy corollary of this proposition is that

Corollary 3.1. *Let $\mathscr{C}$ and $\mathscr{D}$ be two $\mathbb{F}_{q^m}$-linear codes of a same length and which are such that $\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) = m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}$ and $\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{D}) = m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{D}$. We have*

$$\begin{aligned}
\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) \cdot \dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{D}) - \dim_{\mathbb{F}_q} (\mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D})) \\
\ge m \cdot \left(\dim_{\mathbb{F}_{q^m}} \mathscr{C} \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{D} - \dim_{\mathbb{F}_{q^m}} (\mathscr{C} \star \mathscr{D})\right)
\end{aligned}$$

Proof. We will drop in what follows the subscript indicating in the dimension if it is taken by considering the corresponding code as an $\mathbb{F}_{q^m}$ subspace or as an $\mathbb{F}_q$ subspace– it will be clear from the context.

We have

$$\begin{aligned}
\dim\left(\mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D})\right) &\le \sum_{i=0}^{m-1} \dim \mathrm{Tr}\left(\mathscr{C} \star \mathscr{D}^{q^i}\right) \quad \text{(by Prop. 3.3)} \\
&\le m \cdot \dim\left(\mathscr{C} \star \mathscr{D}\right) + \sum_{i=1}^{m-1} m \cdot \dim\left(\mathscr{C} \star \mathscr{D}^{q^i}\right) \\
&\le m\left(\dim(\mathscr{C} \star \mathscr{D}) - \dim \mathscr{C} \cdot \dim \mathscr{D}\right) + m \cdot \dim \mathscr{C} \cdot \dim \mathscr{D} \\
&\quad + m \sum_{i=1}^{m-1} \dim \mathscr{C} \cdot \dim\left(\mathscr{D}^{q^i}\right) \\
&= m\left(\dim(\mathscr{C} \star \mathscr{D}) - \dim \mathscr{C} \cdot \dim \mathscr{D}\right) + m^2 \dim \mathscr{C} \cdot \dim \mathscr{D} \\
&= m\left(\dim(\mathscr{C} \star \mathscr{D}) - \dim \mathscr{C} \cdot \dim \mathscr{D}\right) + \dim \mathrm{Tr}(\mathscr{C}) \cdot \dim \mathrm{Tr}(\mathscr{D}).
\end{aligned}$$

□

Remark 3.1. In particular, this result implies that if we have two codes $\mathscr{C}$ and $\mathscr{D}$ over $\mathbb{F}_{q^m}$ for which $\dim_{\mathbb{F}_{q^m}}(\mathscr{C} \star \mathscr{D}) < \dim_{\mathbb{F}_{q^m}} \mathscr{C} \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{D}$, then the same property survives for the corresponding trace codes:

$$\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathscr{C}) \star \mathrm{Tr}(\mathscr{D})) < \dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) \cdot \dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{D}).$$

In the case $\mathscr{C} = \mathscr{D}$, namely if we consider square codes, Proposition 3.3 can be refined to give

Proposition 3.4. *Let $\mathscr{C}$ be a linear code over $\mathbb{F}_{q^m}$. We have*

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) = \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{m-u}}\right) \tag{3.13}$$

$$(\mathrm{Tr}(\mathscr{C}))^{\star 2} \subseteq \sum_{u=0}^{\lfloor m/2 \rfloor} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \tag{3.14}$$

$$\dim_{\mathbb{F}_q}\left(\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{m/2}}\right)\right) \leq m\frac{(\dim_{\mathbb{F}_{q^m}}(\mathscr{C}))^2}{2} \quad \textit{if } m \textit{ is even} \tag{3.15}$$

Proof. Proof of (3.13). Let $\boldsymbol{c}, \boldsymbol{d} \in \mathscr{C}$. Since the trace acts component-wise on vectors and $\mathrm{Tr}(x) = \mathrm{Tr}\left(x^{q^u}\right)$ for any $x \in \mathbb{F}_{q^m}$ and natural number u,

$$\mathrm{Tr}\left(\boldsymbol{c} \star \boldsymbol{d}^{q^u}\right) = \mathrm{Tr}\left((\boldsymbol{c} \star \boldsymbol{d}^{q^u})^{q^{m-u}}\right) = \mathrm{Tr}\left(\boldsymbol{c}^{q^{m-u}} \star \boldsymbol{d}^{q^m}\right) = \mathrm{Tr}\left(\boldsymbol{d} \star \boldsymbol{c}^{q^{m-u}}\right) \in \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{(q^{m-u})}\right).$$

This shows that $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \subseteq \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{m-u}}\right)$. By replacing u by $m-u$ in the equality above, we obtain the reverse inclusion, which finishes the proof of (3.13).
Proof of (3.14).

$$\begin{aligned}(\mathrm{Tr}(\mathscr{C}))^{\star 2} &\subseteq \sum_{u=0}^{m-1} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \quad \text{(by Prop. 3.3)} \\ &\subseteq \sum_{u=0}^{\lfloor m/2 \rfloor} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \quad \text{(by (3.13)).}\end{aligned}$$

Proof of (3.15). Let $\{\boldsymbol{c}_1, \cdots, \boldsymbol{c}_r\}$ be a basis of $\mathscr{C}$ where $r \stackrel{\text{def}}{=} \dim(\mathscr{C})$. The trace code $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{m/2}}\right)$ is generated by the $\mathrm{Tr}\left(\beta^{q^\ell} \boldsymbol{c}_i \star \boldsymbol{c}_j^{q^{m/2}}\right)$'s where $\{\beta, \beta^q, \cdots, \beta^{q^{m-1}}\}$ is a normal basis of $\mathbb{F}_{q^m}$ and ℓ ranges over $\{0, \cdots, m-1\}$ and i, j over $\{1, \cdots, r\}$. Since for any $0 \leq \ell \leq \frac{m}{2} - 1$, $1 \leq i, j \leq r$,

$$\mathrm{Tr}\left(\beta^{q^\ell} \boldsymbol{c}_i \star \boldsymbol{c}_j^{q^{m/2}}\right) = \mathrm{Tr}\left(\beta^{q^{\ell+\frac{m}{2}}} \boldsymbol{c}_i^{q^{m/2}} \star \boldsymbol{c}_j\right) = \mathrm{Tr}\left(\beta^{q^{\ell+\frac{m}{2}}} \boldsymbol{c}_j \star \boldsymbol{c}_i^{q^{m/2}}\right),$$

this implies that $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{\frac{m}{2}}}\right)$ is generated by the (smaller) set

$$\{\mathrm{Tr}\left(\beta^{q^\ell} \boldsymbol{c}_i \star \boldsymbol{c}_j^{q^{m/2}}\right) \mid \ell \in [\![0, m/2-1]\!], i, j \in [\![1, r]\!].$$

This is a set of cardinality $\frac{mr^2}{2}$. □

Note that [Ras13] which solved the highly non-trivial open problem of constructing a family of asymptotically good binary linear codes whose square is also asymptotically good proves some intermediate results in it which in some sense are equivalent to some of the results presented in this section. For instance, Proposition 4 in [Ras13] gives a basis for the space of symmetric bilinear forms of $\mathbb{F}_{q^m}$ seen as vector space of dimension m over $\mathbb{F}_q$, in terms of trace operators. In essence, our Proposition 3.4 is nothing but a corollary of Proposition 4.

Proposition 3.4 has a corollary which is similar to Corollary 3.1, namely that

Corollary 3.2. *Let $\mathscr{C}$ be an $\mathbb{F}_{q^m}$-linear code. We have*

$$\dim_{\mathbb{F}_q} (\mathrm{Tr}(\mathscr{C}))^{\star 2} \leq m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}^{\star 2} + \binom{m}{2} \left(\dim_{\mathbb{F}_{q^m}} \mathscr{C}\right)^2 . \tag{3.16}$$

Furthermore if $\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) = m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}$ then

$$\dim_{\mathbb{F}_q} (\mathrm{Tr}(\mathscr{C}))^{\star 2} - \binom{\dim_{\mathbb{F}_q} \mathrm{Tr}(\mathscr{C}) + 1}{2} \leq m \left[\dim_{\mathbb{F}_{q^m}} \mathscr{C}^{\star 2} - \binom{\dim_{\mathbb{F}_{q^m}} \mathscr{C} + 1}{2} \right]. \tag{3.17}$$

Proof.
Proof of (3.16): by using (3.14) of Proposition 3.4, we obtain

$$\dim (\mathrm{Tr}(\mathscr{C}))^{\star 2} \leq \sum_{i=0}^{\lfloor \frac{m}{2} \rfloor} \dim \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^i}\right). \tag{3.18}$$

In the case of odd m, we deduce that

$$\begin{aligned}
\dim (\mathrm{Tr}(\mathscr{C}))^{\star 2} &\leq m \cdot \dim \mathscr{C}^{\star 2} + \sum_{i=1}^{\lfloor \frac{m}{2} \rfloor} m \cdot \dim \mathscr{C} \star \mathscr{C}^{q^i} \\
&= m \cdot \dim \mathscr{C}^{\star 2} + m \sum_{i=1}^{\lfloor \frac{m}{2} \rfloor} \dim \mathscr{C} \cdot \dim \mathscr{C}^{q^i} \\
&= m \cdot \dim \mathscr{C}^{\star 2} + \frac{m(m-1)}{2} (\dim \mathscr{C})^2 \\
&= m \cdot \dim \mathscr{C}^{\star 2} + \binom{m}{2} (\dim \mathscr{C})^2 .
\end{aligned}$$

On the other hand, if m is even, we have

$$\begin{aligned}
\dim (\mathrm{Tr}(\mathscr{C}))^{\star 2} &\leq m \cdot \dim \mathscr{C}^{\star 2} + \sum_{i=1}^{\frac{m}{2}-1} m \cdot \dim \mathscr{C} \cdot \dim \mathscr{C}^{q^i} + \dim \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^{m/2}}\right) \\
&\leq m \cdot \dim \mathscr{C}^{\star 2} + \frac{m(m-2)}{2} (\dim \mathscr{C})^2 + \frac{m(\dim \mathscr{C})^2}{2} \\
&\quad \text{(by using (3.15) of Proposition 3.4)} \\
&= m \cdot \dim \mathscr{C}^{\star 2} + \binom{m}{2} (\dim \mathscr{C})^2 .
\end{aligned}$$

Proof of (3.17)*:*

$$\begin{aligned}
\dim\left(\mathrm{Tr}(\mathscr{C})\right)^{\star 2} - \binom{\dim \mathrm{Tr}(\mathscr{C})+1}{2} &\leq m \cdot \dim \mathscr{C}^{\star 2} + \binom{m}{2}(\dim \mathscr{C})^2 - \binom{\dim \mathrm{Tr}(\mathscr{C})+1}{2} \\
&\quad \text{(by using (3.16))} \\
&= m \cdot \dim \mathscr{C}^{\star 2} + \frac{m(m-1)}{2}(\dim \mathscr{C})^2 - \binom{m\dim \mathscr{C}+1}{2} \\
&= m \cdot \dim \mathscr{C}^{\star 2} + m \cdot \dim \mathscr{C} \cdot \left[\frac{m-1}{2}\dim \mathscr{C} - \frac{m\dim \mathscr{C}+1}{2}\right] \\
&= m \cdot \dim \mathscr{C}^{\star 2} - m \cdot \dim \mathscr{C} \cdot \frac{\dim \mathscr{C}+1}{2} \\
&= m\left[\dim \mathscr{C}^{\star 2} - \binom{\dim \mathscr{C}+1}{2}\right].
\end{aligned}$$

□

Similarly to Corollary 3.1, Corollary 3.2 implies that if the dimension of a square code $\mathscr{C}^{\star 2}$ over $\mathbb{F}_{q^m}$ is smaller than what we expect from a random code, namely that $\dim\left(\mathscr{C}^{\star 2}\right) < \binom{\dim \mathscr{C}+1}{2}$ (if $\binom{\dim \mathscr{C}+1}{2}$ is smaller than the code length) then this property survives for the trace code:

$$\dim\left(\mathrm{Tr}(\mathscr{C})\right)^{\star 2} < \binom{\dim \mathrm{Tr}(\mathscr{C})+1}{2}.$$

3.4 Alternant case with $e_{\mathscr{A}} = 0$ and Goppa case with $e_{\mathscr{G}} = 0$

In this section, we are going to give a first upper bound on the dimension of the square of the dual of an alternant or Goppa code which is valid for all parameters and is tight when $e_{\mathscr{A}} = 0$ for random alternant codes and when $r < q-1$ for Goppa codes. This will be a direct application of the general results of Section 3.3 from which we derive:

Theorem 3.1. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_q$. Then*

$$\dim_{\mathbb{F}_q}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)^{\star 2} \leq \binom{rm+1}{2} - \frac{m}{2}(r-1)(r-2). \tag{3.19}$$

Proof. We let $\mathscr{C} \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$. Note that $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} = \mathrm{Tr}(\mathscr{C})$. We apply Corollary 3.2 with such a $\mathscr{C}$ and get that

$$\begin{aligned}
\dim_{\mathbb{F}_q}\left(\mathrm{Tr}(\mathscr{C})\right)^{\star 2} &\leq m \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}^{\star 2} + \binom{m}{2}\left(\dim_{\mathbb{F}_{q^m}} \mathscr{C}\right)^2 \\
&= m(2r-1) + \frac{m(m-1)r^2}{2} \quad \text{(by Proposition 1.7)} \\
&= \left(2(2r-1) + (m-1)r^2\right)\frac{m}{2} \\
&= \left(r(mr+1) - (r-1)(r-2)\right)\frac{m}{2} \\
&= \binom{rm+1}{2} - \frac{m}{2}(r-1)(r-2).
\end{aligned}$$

□

3.5 Alternant case with $e_{\mathscr{A}} > 0$

In this section, we will show new linear relationships arising for alternant codes (hence also for Goppa codes) of high enough order r. More precisely, the threshold value for which new relations are guaranteed is $r \geq q+1$, i.e. $e_{\mathscr{A}} > 0$. Our main result in this section is that

Theorem 3.2. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_q$. Then*

$$\dim_{\mathbb{F}_q}(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp})^{\star 2} \leq \binom{rm+1}{2} - \frac{m}{2}(r-1)\left((2e_{\mathscr{A}}+1)r - 2\frac{q^{e_{\mathscr{A}}+1}-1}{q-1}\right). \quad (3.20)$$

Remark 3.2. Note that the upper bound on the dimension coincides with the prediction (3.8) given for generic alternant codes. In other words, this theorem shows that this prediction is an upper bound on the dimension and the experimental evidence gathered in [Fau+13] actually shows that the dimensions of random alternant codes agree with this upper bound in such a case.

The reason why we have a refinement of the upper bound of Theorem 3.1 for values of r for which $e_{\mathscr{A}} > 0$ comes from the fact that when we apply Proposition 3.4 with $\mathscr{C} \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ (which is the relevant quantity here since $\mathrm{Tr}(\mathscr{C}) = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}$) we get terms of the form $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right)$ which will have a smaller dimension than the generic upper bound mr^2. This is due to the fact that these $tr\left(\mathscr{C} \star \mathscr{C}^{q^u}\right)$'s will be duals of alternant codes for small values of u as shown by the following lemma

Lemma 3.1. *Let $\mathscr{C} \stackrel{def}{=} \boldsymbol{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ and $f \stackrel{def}{=} \lfloor \log_q(r) \rfloor$. We have*

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \subseteq \mathscr{A}_{(r-1)(1+q^u)+1}(\boldsymbol{x}, \boldsymbol{y}^{1+q^u})^{\perp} \text{ for all non-negative integers } u, \quad (3.21)$$

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) = \mathscr{A}_{(r-1)(1+q^u)+1}(\boldsymbol{x}, \boldsymbol{y}^{1+q^u})^{\perp} \text{ for all integers } u \text{ in } \{0, \cdots, f\}. \quad (3.22)$$

Proof. We first notice that

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) = \left\langle \mathrm{Tr}\left(\alpha_l \boldsymbol{x}^{a+bq^u} \boldsymbol{y}^{1+q^u}\right) \mid 0 \leq l < m, 0 \leq a, b \leq r-1 \right\rangle_{\mathbb{F}_q}.$$

Clearly, the powers $a + bq^u$ are all smaller than or equal to $(r-1)(1+q^u)$. This directly implies (3.21). For the equality case, we observe that we get all the powers in $\{0, \cdots, (r-1)(1+q^u)\}$ as long as $r-1 \geq q^u - 1$, that is $r \geq q^u$ which is equivalent to $u \leq \lfloor \log_q(r) \rfloor$. □

The previous lemma implies that

Corollary 3.3. *Let $\mathscr{C} \stackrel{def}{=} \boldsymbol{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$. For all non-negative integers u, we have*

$$\dim_{\mathbb{F}_q} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \leq m((r-1)(q^u+1)+1).$$

When r is large enough with respect to the field size q, this corollary provides a tighter upper bound with respect to the trivial $\dim_{\mathbb{F}_q} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \leq mr^2$. More precisely, since

$$r \geq q^u + 1 \Rightarrow r^2 \geq r(q^u+1) \geq (r-1)(q^u+1) + 1$$

and

$$r \leq q^u \Rightarrow r^2 \leq rq^u \leq (r-1)(q^u+1) + 1,$$

for any $0 \leq u < m/2$, the inequality

$$\dim_{\mathbb{F}_q} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \leq m \cdot \min\left((r-1)(q^u+1)+1, r^2\right)$$

becomes

$$\dim_{\mathbb{F}_q} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \leq \begin{cases} m((r-1)(q^u+1)+1) & \text{if } r > q^u \\ mr^2 & \text{else.} \end{cases} \tag{3.23}$$

From (3.23) we obtain directly Theorem 3.2. The proof goes as follows.

Proof of Theorem 3.2. Let $\mathscr{C} \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$. Recalling that

$$e_{\mathscr{A}} \stackrel{\text{def}}{=} \max\{i \in \mathbb{N} \mid r \geq q^i + 1\} = \left\lfloor \log_q(r-1) \right\rfloor,$$

we obtain

$$\begin{aligned} \dim_{\mathbb{F}_q}(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp)^{\star 2} &\leq \dim_{\mathbb{F}_q} \sum_{u=0}^{\lfloor \frac{m}{2} \rfloor} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \quad \text{(by Prop. (3.4))} \\ &\leq \sum_{u=0}^{\lfloor \frac{m}{2} \rfloor} \dim_{\mathbb{F}_q} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \\ &\leq \sum_{u=0}^{e_{\mathscr{A}}} m((r-1)(q^u+1)+1) + \left(\frac{m-1}{2} - e_{\mathscr{A}}\right) mr^2 \\ &= \binom{rm+1}{2} - \frac{m}{2}(r-1)\left((2e_{\mathscr{A}}+1)r - 2\frac{q^{e_{\mathscr{A}}+1}-1}{q-1}\right). \end{aligned}$$

□

3.6 Goppa case with $r \geq q-1$

In the previous two sections, we found linear relationships within the individual $\operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right)$ subspaces, showing that they are spanned by less than $r^2 m$ vectors if r is large enough. We will see that the dimension of some $\operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right)$ is even smaller in the Goppa case with $r \geq q-1$ (see (3.25) below). Moreover, they are no more disjoint, i.e. $\dim_{\mathbb{F}_q}\left(\operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \cap \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right)\right) > 0$ for some $0 \leq u < v \leq e_{\mathscr{G}}$. We will namely prove that

Theorem 3.3. *Let $\mathscr{C} \stackrel{def}{=} \boldsymbol{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$, where $y_i = \frac{1}{\Gamma(x_i)}$ and Γ is a polynomial of degree r. Let us define for any positive integer v*

$$\mathscr{B}_v \stackrel{def}{=} \mathscr{A}_{r(q^v - q^{v-1}+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp, \quad \text{and} \quad \mathscr{B}_0 \stackrel{def}{=} \mathscr{A}_{2r-1}(\boldsymbol{x}, \boldsymbol{y}^2)^\perp. \tag{3.24}$$

Let $f \stackrel{def}{=} \lfloor \log_q(r) \rfloor$. *Then*

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) \subseteq \mathscr{B}_v \quad \textit{for all positive integers } v, \tag{3.25}$$

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) = \mathscr{B}_u \quad \textit{for } 0 \leq u \leq f, \tag{3.26}$$

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}\right) \subseteq \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q}\right) \subseteq \ldots \subseteq \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \quad \textit{for } 0 \leq u \leq f. \tag{3.27}$$

Before proving this theorem it will be useful to state and prove two lemmas. The first one deals with the inclusion $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) \subseteq \mathscr{B}_v$. For this, we first recall Lemma 3.1 which when applied to Goppa codes yields immediately that for all integers v we have:

$$\begin{aligned} \mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) &\subseteq \left\{ \mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \mid \deg P < (r-1)(q^v+1)+1 \right\} \\ &= \mathscr{A}_{(r-1)(q^v+1)+1}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^{\perp}. \end{aligned} \tag{3.28}$$

On the other hand, by definition of $\mathscr{B}_v$ we have

$$\mathscr{B}_v = \left\{ \mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \mid \deg P < r(q^v - q^{v-1} + 1) \right\}. \tag{3.29}$$

Depending on how q compares to r, $r(q^v - q^{v-1} + 1)$ might be either greater or smaller than $(r-1)(q^v+1)+1$. The key to show the inclusion $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) \subseteq \mathscr{B}_v$ will be to show that under certain conditions a codeword of the form $\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right)$ can be written as $\mathrm{Tr}\left(\frac{Q(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) + \boldsymbol{c}$ for some codeword $\boldsymbol{c} \in \mathscr{B}_v$ with a polynomial Q such that $\deg Q < \deg P$. For performing this task we will use the following lemma.

Lemma 3.2. *Let* $P(X)$ *be a polynomial in* $\mathbb{F}_{q^m}[X]$ *and consider the Euclidean division of* P *by* $\Gamma^{q^v - q^{v-1}+1}$*:* $P = A\Gamma^{q^v - q^{v-1}+1} + B$ *where* $\deg B < \deg \Gamma^{q^v - q^{v-1}+1}$. *We have*

$$\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) = \mathrm{Tr}\left(\frac{A(\boldsymbol{x})^q \Gamma(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) + \boldsymbol{c}, \textit{ for some } \boldsymbol{c} \in \mathscr{B}_v \tag{3.30}$$

$$\deg A^q\Gamma < \deg P \quad \textit{if } \deg P < r(q^v+1). \tag{3.31}$$

Proof. For the first point, we just have to observe that

$$\begin{aligned} \mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) &= \mathrm{Tr}\left(\frac{A(\boldsymbol{x})\Gamma(\boldsymbol{x})^{q^v - q^{v-1}+1} + B(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \\ &= \mathrm{Tr}\left(\frac{A(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^{v-1}}}\right) + \mathrm{Tr}\left(\frac{B(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \\ &= \mathrm{Tr}\left(\frac{A(\boldsymbol{x})^q}{\Gamma(\boldsymbol{x})^{q^v}}\right) + \mathrm{Tr}\left(\frac{B(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \quad (\text{since } \mathrm{Tr}(\alpha^q) = \mathrm{Tr}(\alpha) \text{ for } \alpha \in \mathbb{F}_{q^m}) \\ &= \mathrm{Tr}\left(\frac{A(\boldsymbol{x})^q \Gamma(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) + \boldsymbol{c} \qquad \text{with } \boldsymbol{c} \in \mathscr{B}_v. \end{aligned}$$

The last equality follows immediately from the definition of $\mathscr{B}_v$ (see (3.29)) and the fact that $\deg B < r(q^v - q^{v-1} + 1)$. For the second point, let us just observe that either $A = 0$ in which case (3.31) clearly holds or

$$\deg A = \deg P - r(q^v - q^{v-1} + 1)$$

in which case

$$\deg A^q\Gamma = q\deg P - qr(q^v - q^{v-1}+1) + r = q\deg P - r(q-1)(q^v+1).$$

Clearly

$$\begin{aligned}\deg A^q\Gamma < \deg P &\Leftrightarrow q\deg P - r(q-1)(q^v+1) < \deg P\\ &\Leftrightarrow \deg P < r(q^v+1).\end{aligned}$$

□

A consequence of Lemma 3.2 is that

Corollary 3.4. *Let $\mathscr{B}_v$ be the dual code defined in Equation (3.24). Then, for any $r(q^v - q^{v-1}+1) \leq d \leq r(q^v+1)$,*

$$\mathscr{B}_v = \mathscr{A}_d(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp = \mathscr{G}(\boldsymbol{x}, \Gamma^{q^v+1})^\perp.$$

Proof. It follows from Lemma 3.2 that, if $P(X) \in \mathbb{F}_{q^m}[X]$ has degree strictly smaller than $r(q^v+1)$, then

$$\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \in \mathscr{B}_v.$$

Since

$$\mathscr{A}_{r(q^v+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp = \left\{\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \mid \deg P < r(q^v+1)\right\},$$

we get

$$\mathscr{B}_v \supseteq \mathscr{A}_{r(q^v+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp.$$

Since $\mathscr{B}_v \stackrel{\text{def}}{=} \mathscr{A}_{r(q^v-q^{v-1}+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp$ we have also trivially the reverse inclusion and therefore

$$\mathscr{B}_v = \mathscr{A}_{r(q^v+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp. \tag{3.32}$$

Since the coordinates of $\boldsymbol{y}^{q^v+1}$ are the inverse of the evaluation over the x_i's of the polynomial Γ^{q^v+1} of degree $r(q^v+1)$, Equality (3.32) shows that $\mathscr{B}_v$ is the dual of a Goppa code of degree $r(q^v+1)$. Moreover, (3.32) implies that all the codes

$$\mathscr{A}_d(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp,$$

with $r(q^v - q^{v-1}+1) \leq d \leq r(q^v+1)$, collapse into the same code $\mathscr{B}_v$. □

The second lemma shows that the $\mathscr{B}_v$'s form a nested family of codes:

Lemma 3.3. *We have for all $v \geq 0$:*

$$\mathscr{B}_0 \subseteq \mathscr{B}_1 \subseteq \cdots \subseteq \mathscr{B}_v \subseteq \mathscr{B}_{v+1} \subseteq \cdots$$

Proof. We will first prove that $\mathscr{B}_v \subseteq \mathscr{B}_{v+1}$ for $v \geq 1$. Let us notice that

$$\begin{aligned}\mathscr{B}_v &= \left\langle \mathrm{Tr}\left(\alpha_l \boldsymbol{x}^c \boldsymbol{y}^{q^v+1}\right) \mid 0 \leq l < m, 0 \leq c < r(q^v - q^{v-1}+1)\right\rangle_{\mathbb{F}_q}\\ &= \left\{\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) : P \in \mathbb{F}_{q^m}[X],\ \deg(P) < r(q^v - q^{v-1}+1)\right\}\end{aligned} \tag{3.33}$$

Consider now a codeword $\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right)$ in $\mathscr{B}_v$ and let us prove that it is in $\mathscr{B}_{v+1}$. By (3.33) we know that

$$\deg P < r(q^v - q^{v-1} + 1). \tag{3.34}$$

Then, let us notice that

$$\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) = \mathrm{Tr}\left(\frac{P(\boldsymbol{x})\Gamma(\boldsymbol{x})^{q^{v+1}-q^v}}{\Gamma(\boldsymbol{x})^{q^{v+1}+1}}\right)$$

and that

$$\deg P(X)\Gamma(X)^{q^{v+1}-q^v} < r(q^v-q^{v-1}+1)+r(q^{v+1}-q^v) = rq^{v+1}-rq^{v-1}+r < r(q^{v+1}+1).$$

By Corollary 3.4, $\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) \in \mathscr{B}_{v+1}$ and the thesis follows. □

We are now ready to prove Theorem 3.3.

Proof of Theorem 3.3. Proof of (3.25). Lemma 3.1 implies that

$$\begin{aligned}
\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) &\subseteq \left\{\mathrm{Tr}\left(\frac{P(\boldsymbol{x})}{\Gamma(\boldsymbol{x})^{q^v+1}}\right) : \deg P < (r-1)(q^v+1)+1\right\} \\
&= \mathscr{A}_{(r-1)(q^v+1)+1}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp \\
&\subseteq \mathscr{A}_{r(q^v+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp \\
&= \mathscr{B}_v.
\end{aligned} \tag{3.35}$$

Proof of (3.26). When $v \leq f$, Lemma 3.1 shows that the inclusion (3.35) is an equality. We therefore have

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) = \mathscr{A}_{(r-1)(q^v+1)+1}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp.$$

On the other hand, from (3.25), we know that

$$\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right) \subseteq \mathscr{B}_v = \mathscr{A}_{r(q^v-q^{v-1}+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp. \tag{3.36}$$

Observe now that $r \geq q^v$ implies that $(r-1)(q^v+1) \geq r(q^v - q^{v-1}+1) - 1$, since

$$(r-1)(q^v+1) - r(q^v - q^{v-1}+1) + 1 = (r-q)q^{v-1}.$$

This shows that

$$\mathscr{B}_v = \mathscr{A}_{r(q^v-q^{v-1}+1)}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp \subseteq \mathscr{A}_{(r-1)(q^v+1)+1}(\boldsymbol{x}, \boldsymbol{y}^{q^v+1})^\perp$$

which combined with the reverse inclusion (3.36) proves Equality (3.26).

Proof of (3.27). This is a direct consequence of Lemma 3.3 and that the $\mathscr{B}_v$'s coincide with the $\mathrm{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^v}\right)$'s in this range from (3.26). □

A direct consequence of Theorem 3.3 is that

Corollary 3.5. *Let $\mathscr{G}(\boldsymbol{x},\Gamma)$ be a Goppa code of order $r \geq q-1$ over $\mathbb{F}_q$. Then*

$$(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} \subseteq \mathscr{B}_e + \sum_{u=e+1}^{\left\lfloor \frac{m}{2} \right\rfloor} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right),$$

for any non-negative integer e.

We can therefore conclude that

Corollary 3.6. *Let $\mathscr{G}(\boldsymbol{x},\Gamma)$ be a Goppa code of order $r \geq q-1$ over $\mathbb{F}_q$. Then*

$$\dim_{\mathbb{F}_q}(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} \leq \binom{rm+1}{2} - \frac{m}{2} r\left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right).$$

Proof. From Corollary 3.3

$$\begin{aligned}
\dim_{\mathbb{F}_q}(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} &\leq \dim_{\mathbb{F}_q}(\mathscr{B}_e) + \sum_{u=e+1}^{\left\lfloor \frac{m}{2} \right\rfloor} \dim_{\mathbb{F}_q} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right) \\
&\qquad \text{(for arbitrary } e \in \{0,\cdots,\left\lfloor \frac{m}{2} \right\rfloor\}) \\
&\leq rm(q^e - q^{e-1} + 1) + \left(\frac{m-1}{2} - e\right) mr^2 \\
&= \binom{rm+1}{2} - \frac{m}{2} r\left((2e+1)r - 2(q-1)q^{e-1} - 1\right).
\end{aligned}$$

We want now to minimize the function $\binom{rm+1}{2} - \frac{m}{2} r\left((2e+1)r - 2(q-1)q^{e-1} - 1\right)$ with respect to e. By removing the constant part in e, this becomes equivalent to maximizing

$$T(e) \stackrel{\text{def}}{=} er - (q-1)q^{e-1}$$

over $\{0,\cdots,\lfloor \frac{m}{2} \rfloor\}$. We compute the discrete derivative $\Delta T : e \to T(e+1) - T(e)$,

$\Delta T(e) = T(e+1) - T(e) = ((e+1)r - (q-1)q^{e+1-1}) - (er - (q-1)q^{e-1}) = r - (q-1)^2 q^{e-1}$.

The maximum is attained at the least integer e such that $\Delta T(e) \leq 0$. This corresponds to the least integer e such that $e \geq \log_q\left(\frac{r}{(q-1)^2}\right) + 1$, i.e. $\left\lceil \log_q\left(\frac{r}{(q-1)^2}\right) + 1 \right\rceil = e_{\mathscr{G}}$ minimizes the function T and consequently

$$\dim_{\mathbb{F}_q}(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} \leq \binom{rm+1}{2} - \frac{m}{2} r\left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right).$$

□

Remark 3.3. From the computation given in the proof of Corollary 3.6 and knowing that the upper bound is usually an equality, we also infer that with a high probability

$$\mathscr{B}_{e_{\mathscr{G}}} = \sum_{u=0}^{e_{\mathscr{G}}} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right).$$

This has been verified experimentally. In other words, we expect with a very high probability to have equality in Corollary for $e = e_{\mathscr{G}}$:

$$(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} = \mathscr{B}_{e_{\mathscr{G}}} + \sum_{u=e_{\mathscr{G}}+1}^{\left\lfloor \frac{m}{2} \right\rfloor} \operatorname{Tr}\left(\mathscr{C} \star \mathscr{C}^{q^u}\right). \tag{3.37}$$

Remark 3.4. One may also wonder if it is possible to have

$$(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp})^{\star 2} = \mathscr{B}_{e_{\mathscr{G}}}, \tag{3.38}$$

i.e. $e_{\mathscr{G}} = \lfloor \frac{m}{2} \rfloor$, in some range of non-degenerate parameters and while staying in the distinguishable setting. This would reveal an even stronger structure that might be exploited by an attacker. However, the existence of such parameters is not obvious, as the number $\binom{rm+1}{2} - \frac{m}{2} r \left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right)$ decreases when $e_{\mathscr{G}}$ increases, and if the former drops below $n \leq q^m$, the code is no more distinguishable. In other terms, two opposing constraints limit the regime where the equality can be satisfied. Just to give an idea of these restrictions, note that r is bounded in the following way

$$\begin{aligned}
& \left\lceil \log_q\left(\frac{r}{(q-1)^2}\right) + 1 \right\rceil = \left\lfloor \frac{m}{2} \right\rfloor \\
\iff \quad & \left\lfloor \frac{m-2}{2} \right\rfloor \geq \log_q\left(\frac{r}{(q-1)^2}\right) > \left\lfloor \frac{m-4}{2} \right\rfloor \\
\iff \quad & (q-1)^2 q^{\frac{m-3}{2}} \geq r > (q-1)^2 q^{\frac{m-5}{2}}.
\end{aligned}$$

Now assume $e_{\mathscr{G}} = \frac{m-1}{2}$, i.e. m odd. The distinguishability constraint gives

$$\begin{aligned}
q^m \geq n \quad & > \binom{rm+1}{2} - \frac{m}{2} r \left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right) \\
& = \binom{rm+1}{2} - \frac{m}{2} r \left(\left(2\frac{m-1}{2}+1\right) r - 2(q-1)q^{\frac{m-1}{2}-1} - 1\right) \\
& = \frac{r^2m^2}{2} + \frac{rm}{2} - \frac{r^2m^2}{2} + \frac{rm}{2} + rm(q-1)q^{\frac{m-3}{2}} \\
& > m(q-1)^2 q^{\frac{m-5}{2}} (q-1) q^{\frac{m-3}{2}} \\
& = m(q-1)^3 q^{m-4} \\
\Rightarrow \quad & m < \frac{q^4}{(q-1)^3}.
\end{aligned}$$

This very rough approximation shows that for a fixed subfield size q, the extension degree m is upper bounded, hence it guarantees that, for any fixed q, a finite search provides all the good parameters. Indeed we know that the Goppa polynomial degree must belong to a closed integer interval, namely $r \in [\![\left\lfloor \frac{m-4}{2} \right\rfloor + 1, \left\lfloor \frac{m-2}{2} \right\rfloor]\!]$. For instance, $q = 2$ implies $m \leq 15$. The "m even" case leads to an even more strict constraint, because the rounding in the condition $e_{\mathscr{G}} = \lfloor \frac{m}{2} \rfloor$ makes $e_{\mathscr{G}}$ larger. For a fixed q, replacing the admissible values of r in $q^m > \binom{rm+1}{2} - \frac{m}{2} r \left((2e_{\mathscr{G}}+1)r - 2(q-1)q^{e_{\mathscr{G}}-1} - 1\right)$ provides the values of m for which the square code equals $\mathscr{B}_{e_{\mathscr{G}}}$, while remaining distinguishable (for a large enough n). Table 3.2 shows all the possible values for $q = 2$.

3.7 Conclusions

In this chapter we revisited the distinguisher for random alternant and Goppa codes presented for the first time in [Fau+13] through a different approach, namely using

		r									
		2	3	5	6	9	10	17	18	19	33
m	5	30									
	6		63								
	7		105								
	9			405	486						
	11					1683	1870				
	13							7293	7722	8151	
	15										32175

Table 3.2: Square code dimensions when the square coincides with the dual of the small alternant code. The non-empty cells correspond to the pairs (r, m) for which Equation (3.38) holds in the binary case ($q = 2$) and the code is distinguishable. The values in the cell represent the square code dimension bound given by the distinguisher.

squares of codes. With this simple but powerful tool, we were able to provide explicitly the linear relationships determining the distinguisher in a more straightforward way. We managed therefore to rigorously prove a tight upper bound for the dimension of the square of the dual of an alternant or Goppa code, while [Fau+13] only provides an algebraic explanation which does not however represent neither an upper nor a lower bound. Our proof is also valid in the case of the non-binary Goppa case, for which the conjectured distinguisher is only demonstrated experimentally in [Fau+13]. By doing this we got a unifying explanation for the behavior of all Goppa codes, which does not make use of specific features of the binary case. In essence, the bounds we derived were obtained by finding linear combinations within a set of generators of the square code of the dual code. However, such linear combinations are independent of the evaluation points, i.e. from the specific support and multiplier, because they are a consequence of polynomial combinations of a bivariate polynomial ring whose unknowns correspond to $\boldsymbol{x}$ and $\boldsymbol{y}$.

Finally, we illustrated an interesting property of the structure of the square of the dual of any Goppa code, showing that it is contained into (and sometimes coincides with) the dual of another alternant code.

The fact that the dual of a Goppa code is the trace of a generalized Reed-Solomon code rather than the subfield subcode of a generalized Reed-Solomon code seems to complicate significantly the attempts to turn this distinguisher into an attack. However, with this better understanding of the distinguisher and the square code structure at hand, we are now ready to present a key-recovery attack that works on random alternant codes. This is the main topic of Chapter 4. Despite some technicalities prevent it from succeeding on Goppa codes, it reveals some serious weaknesses at least on the 20 year old CFS signature, as the rate regime where the attacks can be performed is exactly the same as this distinguisher.

Chapter 4
A polynomial-time key-recovery attack on high-rate random alternant codes

In this chapter, we exploit the distinguisher from Chapter 3 to devise a key-recovery attack against high-rate alternant codes. Differently from GRS codes, their proper subfield subcodes, i.e. alternant and Goppa codes, have never been broken, unless they are equipped with some additional structure. The algorithm comprises two parts. First of all, the key-recovery problem of an alternant code of big order is simplified into that of an alternant code of smaller order. This is done through the computation of a filtration, i.e. a sequence of alternant codes of decreasing order. Such codes are obtained recursively with a strategy that heavily relies on the notion and properties of square codes. Once the final alternant code of the filtration is produced, we solve an algebraic system that models the key-recovery problem for that code. By designing a specific algorithm based on Gröbner bases techniques, the polynomial system can be efficiently solved if the alternant code order is equal to 3. This requires the code to be either binary or ternary, because of the filtration step. In addition to the high-rate condition and the constraint on the field size, this attack is based on heuristics that require the alternant code to be sampled at random, in order to work with high probability. In particular, the cryptanalysis fails when applied to Goppa codes. In this case, several technical issues occur, but the main problem is that the algorithm for computing the filtration does not succeed. Our algorithm runs in polynomial time and is supported by an implementation in MAGMA.

Contents

4.1 Introduction

The McEliece scheme. We have already presented the McEliece encryption scheme [McE78] and its main features in terms of security and performance in Section 1.2. We have also revised the state-of-the-art cryptanalysis on it. The main threat to its security is represented by generic decoding algorithms that go under the name of Information Set Decoding algorithms. These techniques aim at inverting the encryption without finding a trapdoor and, despite considerable improvements, they have exponential complexity. From now on, we will denote the original McEliece scheme, which is built upon the class of binary Goppa codes of rate relatively close to 1/2, by "McEliece-binary Goppa", since we will be interested in variations of the McEliece cryptosystem obtained by changing the underlying code family.

We are here interested in key-recovery attacks. Efficient strategies that fall into this category are not known against McEliece-binary Goppa. This picture changes when considering variations on the McEliece-binary Goppa by either considering very high rate binary Goppa codes or by moving from binary Goppa codes to nonbinary Goppa codes over large alphabets [BLP10; BLP11]. The first modification can be helpful to achieve particular construction, for instance, to devise signature schemes [CFS01]. The second variation allows to decrease significantly the extension degree m over which the (secret) support of the Goppa code is defined. Indeed, we recall that Goppa codes are subfield subcodes of GRS codes, thus they are defined over some finite field $\mathbb{F}_q$ whereas their support is defined over an extension field $\mathbb{F}_{q^m}$. Small field extension degrees increase the decoding radius and therefore provide better parameters for the scheme. A last class of variations includes versions of the McEliece scheme with more structured Goppa codes, for instance quasi-cyclic codes such as [Bar+17; Ber+09], quasi-dyadic codes such as [Ban+17; BLM11; MB09] or Wild Goppa codes [BLP10].

The quasi-cyclic or quasi-dyadic Goppa codes could be attacked by an algebraic modeling [Fau+10b; GL09] for the secret key which could be efficiently solved with Gröbner bases techniques because the added structure allowed to reduce drastically the number of unknowns of the algebraic system. By trying to solve the same algebraic system in the case of high rate Goppa codes it was also found that Gröbner bases techniques behaved very differently when the system corresponds to a Goppa code instead of a random linear code of the same length and dimension. This approach led to [Fau+11] that gave the distinguisher deeply analyzed in Chapter 3. In some sense, the distinguisher emerges from a failed attempt at attacking high-rate alternant/Goppa codes with linear algebra techniques.

Square code and cryptanalysis. We recall here the idea behind the structural attack against McEliece of Niederreiter schemes based on GRS codes and proposed in [Cou+14]. Recall that this scheme was proposed in [Nie86] and was subsequently broken in [SS92]. Note that when the extension degree of the Goppa code is 1 (i.e. the support of the Goppa code is defined over the same field as the Goppa code itself), a Goppa code is indeed a GRS code, so a McEliece scheme based on a Goppa code of extension degree 1 can be attacked with the [SS92] attack. However, this does not seem to generalize to higher extension degrees; i.e. on McEliece schemes based on Goppa codes in general. The point of the new attack [Cou+14], is that it uses arguments on square codes for which there is hope that they could be applied to a much broader class of Goppa codes. In this chapter, we will explain how an

adaptation of this construction can be used in a crucial way to mount an attack on McEliece or Niederreiter schemes based on high-rate alternant codes.

We recall that, for a random code $\mathscr{C}$, the upper-bound $\dim \mathscr{C}^{\star 2} \leq \min\left(n, \binom{k+1}{2}\right)$, where k and n are respectively the dimension and length of $\mathscr{C}$, is almost always an equality [Cas+15]. Instead, the situation for GRS codes is completely different: from Proposition 1.7 we namely have

$$\dim \mathscr{C}^{\star 2} = \min\left(n, 2k-1\right). \tag{4.1}$$

This follows from the fact that GRS codes are evaluation codes of polynomials of degree bounded by the GRS codes dimension k. In a sense, the square code construction "sees" the polynomial structure of the GRS code. A key recovery attack could be mounted as follows. Recall that it amounts here to recover from an arbitrary generator matrix of a GRS code $\mathscr{C} = \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ a pair $(\boldsymbol{x}', \boldsymbol{y}')$ satisfying $\mathscr{C} = \mathbf{GRS}_k(\boldsymbol{x}', \boldsymbol{y}')$. Let us define $\mathscr{C}(i)$ as the subcode of the GRS code $\mathscr{C}$ given by

$$\mathscr{C}(i) = \{(y_i P(x_i))_{1 \leq i \leq n} : \deg P < k,\ x_1 \text{ is a zero of order} \geq i \text{ of } P\},$$

then

(i) $\mathscr{C}(1)$ can be readily computed from $\mathscr{C}$ since it is the shortened code of $\mathscr{C}$ in the first position.

(ii) We have in general the equality

$$\mathscr{C}(i-1) \star \mathscr{C}(i+1) = \mathscr{C}(i)^{\star 2} \tag{4.2}$$

coming from the fact that the product of two polynomials which have a zero of order i at x_1 gives a polynomial with a zero of order $2i$ in x_1 and so does the product of a polynomial with a zero of order $i-1$ in x_1 with a polynomial which has a zero of order $i+1$ at the same place.

(iii) Solving the equation $\mathscr{X} \star \mathscr{A} = \mathscr{B}$ for two known linear codes $\mathscr{A}$ and $\mathscr{B}$ amounts to solve a linear system in the case where $\mathscr{X}$ is the maximal code satisfying $\mathscr{X} \star \mathscr{A} \subseteq \mathscr{B}$. This is indeed the case here for $\mathscr{A} = \mathscr{C}(i-1)$ and $\mathscr{B} = \mathscr{C}(i)^{\star 2}$. $\mathscr{X}$ corresponds in such a case to the *conductor* of $\mathscr{A}$ into $\mathscr{B}$ which is defined as

Definition 4.1. Let $\mathscr{C}, \mathscr{D} \subseteq \mathbb{F}^n$ be two codes. The *conductor* of $\mathscr{C}$ into $\mathscr{D}$ is

$$\mathbf{Cond}\left(\mathscr{C}, \mathscr{D}\right) \stackrel{\text{def}}{=} \{\boldsymbol{u} \in \mathbb{F}_q^n \mid \boldsymbol{u} \star \mathscr{C} \subseteq \mathscr{D}\},$$

where $\boldsymbol{u} \star \mathscr{C} \stackrel{\text{def}}{=} \{\boldsymbol{u} \star \boldsymbol{c} \mid \boldsymbol{c} \in \mathscr{C}\}$.

It is clear that getting the conductor amounts to solve a linear system. The two previous points show that we can therefore compute $\mathscr{C}(2)$ in polynomial time, because $\mathscr{C}(1)$ and $\mathscr{C}(0)$ are known (the first is the shortened code and the second is the code $\mathscr{C}$ itself). We can iterate this process and compute recursively the decreasing set of codes $\mathscr{C}(3), \mathscr{C}(4), \cdots$ and stop when we get a code of dimension 1 (i.e. $C(k-1)$) which reveals a great deal of information about the multiplier $\boldsymbol{y}$.

It is then straightforward with this approach to finish the attack to recover the whole algebraic structure of $\mathscr{C}$. Note that we have computed a decreasing set of codes

$$\mathscr{C} = \mathscr{C}(0) \supset \mathscr{C}(1) \supset \mathscr{C}(2) \cdots \supset \mathscr{C}(k-1)$$

that we will call a **filtration** in what follows.

This approach works basically like this to attack a McEliece scheme based on GRS codes [Cou+14], but interestingly enough it also applies to Wild Goppa codes of extension degree 2 as shown in [COT14a]. Such schemes were indeed proposed in [BLP10]. This extension degree corresponds to the largest extension degree where we can expect the Goppa code to behave differently from a random linear code with respect to the square code dimension. Roughly speaking in this case, even if Goppa codes are subfield subcodes of GRS codes, Equality (4.2) "almost" holds and this is sufficient to mount a similar attack. As explained in [COT14a], this approach is bound to fail when the extension degree m is bigger than 2. However, as observed in [MP12], even when $m > 2$, the square code $\mathscr{C}^{\star 2}$ can also be of unusually small dimension when the rate of the Goppa code is close to 1, but this time not by taking $\mathscr{C}$ to be the Goppa code itself, but by choosing $\mathscr{C}$ to be the *dual* of the Goppa code. This strongly suggests that an approach similar to [COT14a; Cou+14] could be followed to attack McEliece schemes based on very high rate Goppa codes. Even if the parameters of the McEliece schemes proposed in the literature are never in the regime where the dimension of the square of the dual of the Goppa code behaves differently from a random linear code, there is the notable exception of the code-based signature scheme [CFS01], which is based in a crucial way on high rate Goppa codes, and which similarly to the McEliece scheme would be broken, if we can recover the unknown support of the Goppa code from an arbitrary generator matrix for it. However, the fact that the dual code is actually the trace code of a GRS code but not a subfield subcode of a GRS code loses a lot of the original polynomial structure and seems to complicate very significantly this approach. This is still an open problem since the problem was explicitly raised in [Fau+11].

Our contribution. In the present chapter, we make what we consider to be a significant step in this direction. We will namely show that somewhat unexpectedly, an equality related to (4.2) holds, when taking duals of *(generic) high rate alternant codes*, but not when we take Goppa codes. This is extremely surprising because Goppa codes are just alternant codes with a peculiar structure.

The very unusual behavior we observe in the case of a generic alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ is that in a certain high rate regime, if we shorten its *dual* in one position i and take its square to get $\mathscr{B} \stackrel{\text{def}}{=} \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}$, where $\mathbf{Sh}_i(\mathscr{C})$ denotes the code $\mathscr{C}$ shortened in position i, then the conductor of $\mathscr{A}$ into $\mathscr{B}$ is the dual of a certain alternant code of degree $r-1$:

$$\mathbf{Cond}(\mathscr{A}, \mathscr{B}) = \mathscr{A}_{r-1}(\boldsymbol{x}_{\tilde{i}}, \boldsymbol{y}_{\tilde{i}}(\boldsymbol{x}_{\tilde{i}} - x_i))^{\perp}.$$

where $\boldsymbol{x}_{\tilde{i}}$ denotes the vector $\boldsymbol{x}$ where we have dropped the index i and $\mathscr{A}$ is the dual of the shortened alternant code in position i, i.e. $(\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})))^{\perp}$. Note that this code is actually the dual of an alternant code since $(\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})))^{\perp} = \mathscr{A}_r(\boldsymbol{x}_{\tilde{i}}, \boldsymbol{y}_{\tilde{i}})^{\perp}$ (see Proposition 4.1). In other words

$$\mathbf{Cond}\left(\mathscr{A}_r(\boldsymbol{x}_{\tilde{i}}, \boldsymbol{y}_{\tilde{i}}), \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}\right) = \mathscr{A}_{r-1}(\boldsymbol{x}_{\tilde{i}}, \boldsymbol{y}_{\tilde{i}}(\boldsymbol{x}_{\tilde{i}} - x_i))^{\perp}. \tag{4.3}$$

This means that starting from a generic alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ of degree r, we can derive in polynomial time, by first computing two auxiliary codes by taking the dual, shortening and/or computing the square $\mathscr{A} = (\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})))^{\perp}$ and $\mathscr{B} = \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}$, and then computing the conductor $\mathbf{Cond}(\mathscr{A}, \mathscr{B})$ of $\mathscr{A}$ into $\mathscr{B}$, an alternant code of degree $r-1$. It will appear, that there are only two conditions to be met for performing this task:
(i) $r \geq q+1$ where q is the alphabet size of the alternant code,
(ii) $\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}$ is not the full code $\mathbb{F}_q^{n-1}$ where n is the codelength of the alternant code.

By iterating this process, we can compute in polynomial time some kind of "filtration" of duals of alternant codes of decreasing degree

$$\mathscr{A}_r^{\perp} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} \overset{\mathbf{Sh}_{i_1}}{\supseteq} \mathscr{A}_{r-1}^{\perp} \overset{\mathbf{Sh}_{i_2}}{\supseteq} \cdots \overset{\mathbf{Sh}_{i_{r-q}}}{\supseteq} \mathscr{A}_q^{\perp} \tag{4.4}$$

with multipliers and support which are related to the original support and multiplier (and from which the original support and multiplier can be easily recovered). Here the notation $\mathscr{A} \overset{\mathbf{Sh}_i}{\supseteq} \mathscr{B}$ means that

$$\mathbf{Sh}_i(\mathscr{A}) \supseteq \mathscr{B}.$$

What can we do with this sequence? The point is that if the degree of the alternant code is small enough, we can compute its support and multiplier by solving a low degree algebraic system related to the algebraic systems considered in [Fau+10b; Fau+13]. We will detail this in the particular case where $r = 3$ and show that in this case, solving the system can be performed in polynomial time with Gröbner basis techniques. Roughly speaking the reason for this is that we have a conjunction of factors: a very overdetermined and highly structured system which gives during the Gröbner basis computation many new very low degree equations. We will also show that it is possible to speed up significantly the system-solving process by introducing in the algebraic modeling new low degree polynomial equations which are not in the ideal of the original algebraic equations from [Fau+13] and which express the fact that the multiplier vector has only non zero entries and the support vector has only distinct entries. This will result in the end in a very efficient system-solving procedure. Note that the aforementioned procedure reaches an alternant code $\mathscr{A}_3$ of degree 3 when the field size q is either equal to 2 or 3. In other words, we have at the end a way to break a McEliece scheme based on *binary or ternary alternant codes* as soon as $\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}$ is not the full code $\mathbb{F}_q^{n-1}$. By using the formula given in [MT22], this is the case when

$$n - 1 > \binom{rm+1}{2} - \frac{m}{2}(r-1)\left((2e+1)r - 2\frac{q^e-1}{q-1}\right), \tag{4.5}$$

where $e \overset{\text{def}}{=} \max\{i \in \mathbb{N} \mid r \geq q^i + 1\} = \left\lfloor \log_q(r-1) \right\rfloor$.

We give in Table 4.1 the known cases where it is possible to attack a McEliece scheme based on alternant codes together with the new attack proposed here:

In a nutshell, our contribution can be summarized as follows

- It has been a long standing open problem after the [SS92] attack on McEliece-GRS whether it is also possible to attack subfield subcodes of GRS codes, i.e.

Table 4.1: Summary of polynomial time attacks on McEliece schemes based on alternant codes with the conditions to apply them.

paper	restriction
[Cou+14; SS92]	$m = 1$
[COT14a]	$m = 2$ + Wild Goppa code
this work	$q = 2$ or $q = 3$, m arbitrary + high rate condition (4.5) (does not apply in the particular case of Goppa codes)

attack McEliece-alternant or McEliece-Goppa. A first step in this direction was made in [Cou+14] where a new attack on McEliece-GRS was derived with a hope to generalize it to McEliece-alternant or McEliece-Goppa because it is in essence only based on the fact that certain alternant or Goppa codes behave differently from random codes with respect to the dimension of the square code. This was confirmed in [COT14a] by attacking McEliece-wild Goppa in the particular case where the extension degree m is 2, but the method used there which uses squares of the (shortened) Goppa code is bound to fail for higher extension degrees. Here we break for the first time the $m = 2$ barrier, which was even conjectured at some point to be the ultimate limit for such algebraic attacks to work in polynomial time and show that we can actually attack McEliece-alternant for *any* extension degree m provided that the rate of the alternant code is sufficiently large (4.5) and the field size sufficiently low $q = 2$ or $q = 3$. Our attack is also based on square code considerations, but this time on the *dual* of the alternant code. The point is that in this case the square of the dual can also be distinguished from a random code in this regime [Fau+11; MP12]. The attack is however more involved in this case, because the dual loses the simple polynomial evaluation formulation of the Goppa code, since it is in this case the trace of a GRS code and not a subfield subcode of a GRS code. Understanding the structure of the square is more complicated as was already apparent in Chapter 3 where we tackled such a task.

- Interestingly our attack does not work at all when the alternant code has the additional structure of being a Goppa code. However, this work could open the road for also attacking this subcase, in which case we could hope to break the CFS scheme [CFS01] which operates precisely in the high rate regime where the square of the dual of the Goppa code behaves abnormally.

- Our attack consists of two phases, the first phase computes a filtration of the dual of the alternant code by computing iteratively conductors and the second phase solves with Gröbner bases techniques a variation of the algebraic system considered in [Fau+13] and recovers the support and the multiplier from the dual of the alternant code of degree 3 we have at the end of the filtration when $q = 2$ or $q = 3$. We improve rather significantly upon the complexity of solving this system by adding new equations expressing the constraints on the support (all elements are distinct) and the multipliers (all elements are nonzero). By using certain heuristics that we confirmed experimentally we are able to prove that the Gröbner basis computation takes polynomial time and give a complete algebraic explanation of each step of the computation. It is likely that this

analysis could be carried over for larger constant degree alternant codes. This would allow to break McEliece-alternant for larger field size than 3.

A proof-of-concept implementation in MAGMA of the whole attack can be found at https://github.com/roccomora/HighRateAlternant.

4.2 Notation and prerequisites

We keep using the same notation adopted until now to denote integer intervals, coordinates, vectors and their Schur's products, matrices, finite fields and function acting component-wise on vectors. Moreover, we introduce the notation to indicate the drop of a set of positions in a vector. This will come in handy in relation to shortened and/or punctured codes. If $\boldsymbol{x} = (x_i)_{i\in[\![1,n]\!]}$ and $\mathcal{I}$ is a subset of positions, we denote by $\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}}$ the vector $\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}} \overset{\text{def}}{=} (x_i)_{i\in[\![1,n]\!]\setminus\mathcal{I}}$. In particular, we do not contract the indexes but we still associate the original index to each remaining coordinate. When there is just one position i in $\mathcal{I}$ we simply write $\boldsymbol{x}_{\overset{\vee}{i}}$ in this case.

In this regard, we recall the well-known fact that a shortened alternant code is itself an alternant code.

4.2.1 Shortening and alternant codes

Proposition 4.1 (Proposition 9, [COT14a])**.** *Let $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be an alternant code of length n and $\mathcal{I}\subseteq\{1,\dots,n\}$. Then*

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})\right) = \mathscr{A}_r(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\boldsymbol{y}_{\overset{\vee}{\mathcal{I}}}).$$

and its dual counterpart

Proposition 4.2. *Let $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be an alternant code of length n and $\mathcal{I}\subseteq\{1,\dots,n\}$. Then*

$$\mathbf{Pct}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right) = \mathscr{A}_r(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\boldsymbol{y}_{\overset{\vee}{\mathcal{I}}})^{\perp}.$$

Proof. From Proposition 4.1 and Proposition 1.2,

$$\mathscr{A}_r(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\boldsymbol{y}_{\overset{\vee}{\mathcal{I}}})^{\perp} = \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})\right)^{\perp} = \mathbf{Pct}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right).$$

□

The same result can be articulated in the special case of Goppa codes.

Proposition 4.3 (Proposition 10, [COT14a])**.** *Let $\mathscr{G}(\boldsymbol{x},\Gamma)$ be a Goppa code of length n and $\mathcal{I}\subseteq\{1,\dots,n\}$. Then*

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)\right) = \mathscr{G}(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\Gamma) \quad \textit{and} \quad \mathbf{Pct}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right) = \mathscr{G}(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\Gamma)^{\perp}.$$

4.2.2 Conductors and filtrations

We have already given the definition of conductor code in the introductory section (Definition 4.1) and said that its computation has computational complexity. We also remark that there exists a simple closed-form expression for the conductor in terms of the two codes involved.

Proposition 4.4 ([COT14a])**.** *Let $\mathscr{C}, \mathscr{D} \subseteq \mathbb{F}^n$ be two codes. Then*

$$\boldsymbol{Cond}(\mathscr{C}, \mathscr{D}) = \left(\mathscr{C} \star \mathscr{D}^{\perp}\right)^{\perp}.$$

Proof. Let $\boldsymbol{a} \in \mathbf{Cond}(\mathscr{C}, \mathscr{D})$, $\boldsymbol{c} \in \mathscr{C}$ and $\boldsymbol{d}^{\perp} \in \mathscr{D}^{\perp}$. Then

$$\langle \boldsymbol{a}, \boldsymbol{c} \star \boldsymbol{d}^{\perp} \rangle = \sum_{i=1}^{n} a_i c_i d_i^{\perp} = \langle \boldsymbol{a} \star \boldsymbol{c}, \boldsymbol{d}^{\perp} \rangle = 0.$$

Hence $\mathbf{Cond}(\mathscr{C}, \mathscr{D}) \subseteq \left(\mathscr{C} \star \mathscr{D}^{\perp}\right)^{\perp}$. The other inclusion is analogous. □

Remark 4.1. Since the conductor is a linear code, if we restrict the search of vectors $\boldsymbol{a}$ such that $\boldsymbol{a} \star \mathscr{C} \subseteq \mathscr{D}$ to a subspace $\mathscr{S} \subseteq \mathbb{F}^n$, the solution space is simply given by the intersection with $\mathscr{S}$:

$$\mathbf{Cond}(\mathscr{C}, \mathscr{D}) \cap \mathscr{S}.$$

In code-based cryptanalysis, the aim of the conductor is to compute a family of nested codes starting from the knowledge of the public code. According to the terminology used in commutative algebra, such a family is called *filtration* of codes. We already showed the filtration of GRS codes presented in [Cou+14]. Other families of codes broken by this approach include Wild Goppa codes over quadratic extensions [COT14a], algebraic geometry codes [CMP17] and quasi-dyadic alternant codes over quadratic extensions from the NIST candidate DAGS [BC18].

4.2.3 Base field extension and alternant codes

It will be convenient to consider for a code defined over $\mathbb{F}_q$ its extension by scalars over $\mathbb{F}_{q^m}$, meaning the following.

Definition 4.2 (extension of a code over a field extension)**.** Let $\mathscr{C}$ be a linear code over $\mathbb{F}_q$. We denote by $\mathscr{C}_{\mathbb{F}_{q^m}}$ the $\mathbb{F}_{q^m}$-linear span of $\mathscr{C}$ in $\mathbb{F}_{q^m}^n$.

This operation goes somewhat in the opposite direction with respect to certain constructions from $\mathbb{F}_{q^m}$ to $\mathbb{F}_q$, such as subfield subcodes, trace codes or even concatenated codes. While the base field extension construction typically does not provide any benefit in practical applications, it can still be helpful to simplify proofs, as observed in [Ran15]. We will indeed exploit the base field extension to significantly boil down the proof of a key theorem for the filtration attack. Indeed, this notion comes very handy in our case, since the extension to $\mathbb{F}_{q^m}$ of the dual of an alternant code defined over $\mathbb{F}_q$ is a sum of m GRS codes as we are now going to prove. We first need a technical result about the extension of scalars for a trace code.

Proposition 4.5. *Let $\mathscr{C} \subseteq \mathbb{F}_{q^m}^n$ be an $\mathbb{F}_{q^m}$- linear code. Then*

$$\mathrm{Tr}(\mathscr{C})_{\mathbb{F}_{q^m}} = \sum_{i=0}^{m-1} \mathscr{C}^{q^i},$$

where $\mathscr{C}^{q^j} \stackrel{def}{=} \{\boldsymbol{c}^{q^j} = (c_i^{q^j})_i : \boldsymbol{c} \in \mathscr{C}\}$ is readily seen to be an $\mathbb{F}_{q^m}$-linear code of the same dimension as $\mathscr{C}$ when $\mathscr{C}$ is itself an $\mathbb{F}_{q^m}$-linear code.

Proof. Take any $\boldsymbol{c} \in \mathscr{C}$. Then $\mathrm{Tr}(\boldsymbol{c}) = \boldsymbol{c}+\boldsymbol{c}^q+\cdots+\boldsymbol{c}^{q^{m-1}}$ also belongs to $\mathscr{C}+\mathscr{C}^q+\cdots+\mathscr{C}^{q^{m-1}}$. This proves that $\mathrm{Tr}(\mathscr{C}) \subseteq \sum_{i=0}^{m-1} \mathscr{C}^{q^i}$ and therefore $\mathrm{Tr}(\mathscr{C})_{\mathbb{F}_{q^m}} \subseteq \sum_{i=0}^{m-1} \mathscr{C}^{q^i}$. On the other hand, let us prove that any $\mathscr{C}^{q^i}$ is a subspace of $\mathrm{Tr}(CC)_{\mathbb{F}_{q^m}}$ for any i. Consider an arbitrary $\mathbb{F}_q$-basis $\alpha \stackrel{\text{def}}{=} \{\alpha_1, \cdots, \alpha_m\}$ of $\mathbb{F}_{q^m}$. Let $\boldsymbol{x}_i \stackrel{\text{def}}{=} \mathrm{Tr}(\alpha_i \boldsymbol{c})$. Since

$$\boldsymbol{x}_i = \alpha_i \boldsymbol{c} + \alpha_i^q \boldsymbol{c}^q + \cdots + \alpha_i^{q^{m-1}} \boldsymbol{c}^{q^{m-1}}$$

we have that

$$\begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_m \end{pmatrix} = \begin{pmatrix} \boldsymbol{c} & \boldsymbol{c}^q & \cdots & \boldsymbol{c}^{q^{m-1}} \end{pmatrix} \underbrace{\begin{pmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \alpha_1^q & \alpha_2^q & \cdots & \alpha_m^q \\ \vdots & \vdots & \vdots & \vdots \\ \alpha_1^{q^{m-1}} & \alpha_2^{q^{m-1}} & \cdots & \alpha_m^{q^{m-1}} \end{pmatrix}}_{\stackrel{\text{def}}{=} \boldsymbol{M}(\alpha)}$$

$\boldsymbol{M}(\alpha)$ is the Moore matrix associated to $\{\alpha_1, \cdots, \alpha_m\}$ and is invertible because the α_i's are linearly independent over $\mathbb{F}_q$. Therefore

$$\begin{pmatrix} \boldsymbol{c} & \boldsymbol{c}^q & \cdots & \boldsymbol{c}^{q^{m-1}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_m \end{pmatrix} \boldsymbol{M}(\alpha)^{-1}$$

and therefore all the $\boldsymbol{c}^{q^i}$ are $\mathbb{F}_{q^m}$-linear combinations of the $\boldsymbol{x}_j$'s and belong therefore to $\mathrm{Tr}(\mathscr{C})_{\mathbb{F}_{q^m}}$. This shows that $\mathscr{C}^{q^i} \subseteq \mathrm{Tr}(\mathscr{C})_{\mathbb{F}_{q^m}}$ for any i and shows therefore the reverse inclusion

$$\mathscr{C} + \mathscr{C}^q + \cdots + \mathscr{C}^{q^{m-1}} \subseteq \mathrm{Tr}(\mathscr{C})_{\mathbb{F}_{q^m}}.$$

□

Proposition 4.6. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code over $\mathbb{F}_q$. Then*

$$\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = \sum_{j=0}^{m-1} \boldsymbol{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{q^j}.$$

where $\mathscr{C}^{q^j} \stackrel{def}{=} \{\boldsymbol{c}^{q^j} = (c_i^{q^j})_i : \boldsymbol{c} \in \mathscr{C}\}$.

Proof. It follows directly from Theorem 1.1 by taking $\mathscr{C} \stackrel{\text{def}}{=} \mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$ and Proposition 4.5. □

Moreover, while the behavior of the component-wise product with respect to subfield subcodes and trace codes is quite difficult to analyze (some questions in this regard have been partially addressed in Chapter 3, the former operates in a natural way for base field extension. We now state some basic results about how the extension of scalars operation acts with respect to coding theory notions such as the generator and the parity-check matrices.

Lemma 4.1 (Lemma 2.22, [Ran15]). *Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a code. Then*

1. *The inclusion $\mathscr{C} \otimes_{\mathbb{F}_q} \mathbb{F}_{q^m} \subseteq \mathbb{F}_q^n \otimes_{\mathbb{F}_q} \mathbb{F}_{q^m} = \mathbb{F}_{q^m}^n$ induces the identification $\mathscr{C} \otimes_{\mathbb{F}_q} \mathbb{F}_{q^m} = \mathscr{C}_{\mathbb{F}_{q^m}}$.*
2. *if $\boldsymbol{G}$ is a generator matrix of $\mathscr{C}$ over $\mathbb{F}_q$, then $\boldsymbol{G}$ is a generator matrix of $\mathscr{C}_{\mathbb{F}_{q^m}}$ over $\mathbb{F}_{q^m}$.*
3. *if $\boldsymbol{H}$ is a parity-check matrix of $\mathscr{C}$ over $\mathbb{F}_q$, then $\boldsymbol{H}$ is a parity-check matrix of $\mathscr{C}_{\mathbb{F}_{q^m}}$ over $\mathbb{F}_{q^m}$.*

Furthermore, the base field extension commutes with several other standard unary and binary operators on codes.

Lemma 4.2 (Lemma 2.23, [Ran15]). *Let $\mathscr{C}, \mathscr{D} \subseteq \mathbb{F}_q^n$ be two codes. Then*

1. $(\mathscr{C}^\perp)_{\mathbb{F}_{q^m}} = (\mathscr{C}_{\mathbb{F}_{q^m}})^\perp \subseteq \mathbb{F}_{q^m}^n$.
2. $\mathscr{C} \subseteq \mathscr{D} \iff \mathscr{C}_{\mathbb{F}_{q^m}} \subseteq \mathscr{D}_{\mathbb{F}_{q^m}}$.
3. $(\mathscr{C} + \mathscr{D})_{\mathbb{F}_{q^m}} = \mathscr{C}_{\mathbb{F}_{q^m}} + \mathscr{D}_{\mathbb{F}_{q^m}}$ *and* $(\mathscr{C} \oplus \mathscr{D})_{\mathbb{F}_{q^m}} = \mathscr{C}_{\mathbb{F}_{q^m}} \oplus \mathscr{D}_{\mathbb{F}_{q^m}}$.
4. $(\mathscr{C} \cap \mathscr{D})_{\mathbb{F}_{q^m}} = \mathscr{C}_{\mathbb{F}_{q^m}} \cap \mathscr{D}_{\mathbb{F}_{q^m}}$.
5. $(\mathscr{C} \star \mathscr{D})_{\mathbb{F}_{q^m}} = \mathscr{C}_{\mathbb{F}_{q^m}} \star \mathscr{D}_{\mathbb{F}_{q^m}}$.

4.3 The filtration

The main result of this section is to explain how from the code $\mathscr{A}_r = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, when $r \geq q+1$ we are (generally) able to compute a sequence of alternant codes such that

$$\mathscr{A}_r^\perp = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp \overset{\mathbf{Sh}_{i_1}}{\supseteq} \mathscr{A}_{r-1}^\perp \overset{\mathbf{Sh}_{i_2}}{\supseteq} \cdots \overset{\mathbf{Sh}_{i_{r-q}}}{\supseteq} \mathscr{A}_q^\perp, \tag{4.6}$$

where all the alternant codes have a support that is easily derived from the support of $\mathscr{A}_r$, since it just amounts to drop some positions of the support. This is instrumental for recovering efficiently the algebraic structure of the alternant code (i.e. the support $\boldsymbol{x}$ and the multiplier $\boldsymbol{y}$) in what follows. The core of this attack is the following theorem

Theorem 4.1. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be an alternant code such that $r \geq q+1$. Let $\mathscr{C} \overset{def}{=} (\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})))^\perp$, $\mathscr{D} \overset{def}{=} \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)\right)^{\star 2}$, for an arbitrary position i. Then*

$$\mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp \star \mathscr{C} \subseteq \mathscr{D},$$

or, equivalently,

$$\boldsymbol{Cond}(\mathscr{C}, \mathscr{D}) \supseteq \mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp.$$

Informally, given a basis of an alternant code of some degree, we expect to get the basis of an alternant code of degree decreased by 1, under some conditions. The latter has support and multiplier vectors related to the initial ones and is obtained by computing a conductor code.

It turns out experimentally that we actually have equality here $\mathbf{Cond}(\mathscr{C}, \mathscr{D}) = \mathscr{A}_{r-1}(\boldsymbol{x}_{\widetilde{i}}, \boldsymbol{y}_{\widetilde{i}}(\boldsymbol{x}_{\widetilde{i}} - x_i))^{\perp}$ when choosing a random alternant code. It is still possible to build artificial examples where equality does not hold. Notably, we also found that the subfamily of Goppa codes does not meet this property either. However, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are sampled at random, we never met a case in our experiments where equality does not hold. This leads us to state the following conjecture.

Conjecture 4.1. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be a random alternant code over $\mathbb{F}_q$, such that $r \geq q+1$ and $\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)^{\star 2}$ is not the full code. Let $\mathscr{C} \stackrel{def}{=} (\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})))^{\perp}$ and $\mathscr{D} \stackrel{def}{=} \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2}$, for an arbitrary position i. Then, for at least one among r, q or m that tends to ∞,*

$$\boldsymbol{Cond}(\mathscr{C}, \mathscr{D}) = \mathscr{A}_{r-1}(\boldsymbol{x}_{\widetilde{i}}, \boldsymbol{y}_{\widetilde{i}}(\boldsymbol{x}_{\widetilde{i}} - x_i))^{\perp}.$$

with probability $1 - o(1)$.

Remark 4.2. Note that, if $\boldsymbol{x}$ and $\boldsymbol{y}$ have been sampled at random (with the only restrictions that $x_i \neq x_j$ and $y_i \neq 0$), then the same random feature (as well as the support and multiplier constraints) is preserved for $\boldsymbol{x}_{\widetilde{i}}$ and $\boldsymbol{y}_{\widetilde{i}}(\boldsymbol{x}_{\widetilde{i}} - x_i)$. Hence, the plausibility of Conjecture and therefore nothing prevents from carrying on the filtration by replacing the original random alternant code with $\mathscr{A}_{r-1}(\boldsymbol{x}_{\widetilde{i}}, \boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\widetilde{i}} - x_i))$.

It is clear that this conjecture, if true, allows to compute in polynomial time the filtration (4.6), since computing conductors just amounts to solve a linear system. The fact that when taking a random alternant code, the whole filtration can be computed has indeed been verified experimentally. The first conductor $\mathscr{A}_{r-1} = \mathscr{A}_{r-1}(\boldsymbol{x}_{\widetilde{i_1}}, \boldsymbol{y}_{\widetilde{i_1}}(\boldsymbol{x}_{\widetilde{i_1}} - x_{i_1}))^{\perp}$ is computed by using directly Conjecture 4.1 and we iterate the process by choosing a sequence of positions $i_1, i_2, \cdots, i_{r_q}$ by which we shorten. We let $\mathcal{I}_s = \{i_1, \cdots, i_s\}$. It is readily seen that we compute iteratively from $\mathscr{A}_{r-s+1}^{\perp} \stackrel{\text{def}}{=} \mathscr{A}_{r-s+1}(\boldsymbol{x}_{\widetilde{\mathcal{I}_{s-1}}}, \boldsymbol{y}_{\widetilde{\mathcal{I}_{s-1}}} \prod_{j=1}^{s-1}(\boldsymbol{x}_{\widetilde{\mathcal{I}_{s-1}}} - x_{i_j}))$ the code

$$\mathscr{A}_{r-s}^{\perp} \stackrel{\text{def}}{=} \mathscr{A}_{r-s}(\boldsymbol{x}_{\widetilde{\mathcal{I}_s}}, \boldsymbol{y}_{\widetilde{\mathcal{I}_s}} \prod_{j=1}^{s}(\boldsymbol{x}_{\widetilde{\mathcal{I}_s}} - x_{i_j})).$$

This allows to decrease the degree of the alternant one by one. The last step ends by using the conjecture with $r = q+1$ and ends with the conductor $\mathscr{A}_q^{\perp}$. Let us now prove Theorem 4.1. Thus we have access to an alternant code of degree 3 when $q = 3$ and, interrupting the filtration one step before, when $q = 2$, the latter being the most interesting case for cryptographic applications.

Remark 4.3. Differently from the GRS codes case, where the filtration is obtained by increasing the zero multiplicity in one single position (see the introductory section), here we increase the number of positions where the code vanishes with multiplicity 1.

4.3.1 Proof of Theorem 4.1

It will be convenient to prove a slightly stronger result which implies Theorem 4.1. It is based on the observation that $\mathscr{A}_{r-1}(\boldsymbol{x}_{\widetilde{i}}, \boldsymbol{y}_{\widetilde{i}}(\boldsymbol{x}_{\widetilde{i}} - x_i))^{\perp} \subseteq \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)$. Theorem 4.1 is indeed implied by the following slightly stronger result:

Theorem 4.2. *Let $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be an alternant code such that $r \geq q+1$. Let $\mathscr{C} \stackrel{def}{=} (\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})))^\perp$ and $\mathscr{D}' \stackrel{def}{=} \mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp \star \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)$, for an arbitrary position i. Then*

$$\mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp \star \mathscr{C} \subseteq \mathscr{D}',$$

or, equivalently,

$$\boldsymbol{Cond}\left(\mathscr{C}, \mathscr{D}'\right) \supseteq \mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp.$$

To prove this theorem, it will be convenient to consider the extensions of all these codes to $\mathbb{F}_{q^m}$, in other words, we are going to prove that

$$\mathscr{B}_{\mathbb{F}_{q^m}} \star \mathscr{C}_{\mathbb{F}_{q^m}} \subseteq \mathscr{D}'_{\mathbb{F}_{q^m}}, \tag{4.7}$$

where $\mathscr{B} \stackrel{\text{def}}{=} \mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^\perp$. The point of doing this, is that (i) it is equivalent to prove (4.7) because of the points 2 and 5 of Lemma 4.2, (ii) the extended codes can be expressed as a sum of GRS codes due to Proposition 4.6.

The proof of Theorem 4.2 will proceed by following the steps below

Step 1: We first observe that the code $\mathscr{C}_{\mathbb{F}_{q^m}} = \left((\mathbf{Sh}_i(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})))^\perp\right)_{\mathbb{F}_{q^m}} = \left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}}$ (where the last equality follows from Proposition 4.1) decomposes as

$$\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}} = \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \oplus \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}.$$

This is Lemma 4.3 below. This implies that

$$\mathscr{B}_{\mathbb{F}_{q^m}} \star \mathscr{C}_{\mathbb{F}_{q^m}} = \underbrace{\mathscr{B}_{\mathbb{F}_{q^m}} \star \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}}}_{\mathscr{D}'_{\mathbb{F}_{q^m}}} + \mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}.$$

Therefore in order to prove (4.7) it will be enough to prove the inclusion

$$\mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}} \subseteq \mathscr{D}'_{\mathbb{F}_{q^m}}. \tag{4.8}$$

Step 2: To achieve this purpose, we then prove that the extended shortened code $\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}}$ contains as a subcode $\mathscr{B}_{\mathbb{F}_{q^m}} = \left(\mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}}, (\boldsymbol{x}_{\check{i}} - x_i)\boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}}$ (Lemma (4.4)) on one hand and $\mathscr{C}_0 \stackrel{\text{def}}{=} \left\langle\, y_i^{q^u} \boldsymbol{y}_{\check{i}}^{q^v} - y_i^{q^v} \boldsymbol{y}_{\check{i}}^{q^u} \;:\; u, v \in [\![0, m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}}$ on the other hand. Actually more is true, namely that the extended shortened is a sum of these two subcodes but we will not need this.

Step 3: By using this, in order to prove (4.8) we will start with an element in $\mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}$ and by adding suitable elements of $\mathscr{B}_{\mathbb{F}_{q^m}} \star \mathscr{C}_0$ and $\mathscr{B}^{\star 2}_{\mathbb{F}_{q^m}}$ we will show that we end with an element in $\mathscr{D}'_{\mathbb{F}_{q^m}} = \mathscr{B}_{\mathbb{F}_{q^m}} \star \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}}$.

Let us now state and prove the lemmas we have mentioned above.

Lemma 4.3.

$$\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}} = \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \oplus \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}.$$

Proof. Note that $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}$ decomposes as the set of codewords $\mathscr{A}_0$ that are zero in i (this is $\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}}$ where we add an extra-position at i which is always 0) plus a space of dimension 1 generated by an element of $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}$ which is not equal to 0 at position i. $\boldsymbol{y}$ is clearly such an element and we can write

$$\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = \mathscr{A}_0 \oplus \langle\, \boldsymbol{y} \,\rangle_{\mathbb{F}_{q^m}}.$$

By puncturing these codes at i we get our lemma. □

Lemma 4.4. *We have for any position i*

$$\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \supseteq \left(\mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}},(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}} \tag{4.9}$$

$$\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \supseteq \mathscr{C}_0 \quad \textit{where} \tag{4.10}$$

$$\mathscr{C}_0 \stackrel{def}{=} \left\langle\, y_i^{q^u}\boldsymbol{y}_{\check{i}}^{q^v} - y_i^{q^v}\boldsymbol{y}_{\check{i}}^{q^u} \;\middle|\; u,v \in [\![0,m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}}. \tag{4.11}$$

Proof. By using Proposition 4.6 we know that

$$\begin{aligned}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} &= \left\langle\, \boldsymbol{x}^{aq^\ell}\boldsymbol{y}^{q^\ell};\ a\in[\![0,r-1]\!],\ \ell\in[\![0,m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}},\\ \left(\mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}},(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}} &= \left\langle\, \boldsymbol{x}_{\check{i}}^{aq^\ell}(\boldsymbol{x}_{\check{i}}-x_i)^{q^\ell}\boldsymbol{y}_{\check{i}}^{q^\ell};\ a\in[\![0,r-2]\!],\ \ell\in[\![0,m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}}.\end{aligned}$$

Observe now that

$$\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} = \mathbf{Sh}_i\left(\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}\right).$$

Clearly $\boldsymbol{x}^{aq^\ell}(\boldsymbol{x}-x_i)^{q^\ell}\boldsymbol{y}^{q^\ell} = (\boldsymbol{x}^a(\boldsymbol{x}-x_i))^{q^\ell}\boldsymbol{y}^{q^\ell}$ vanishes at i and belongs to $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}$ for a in $[\![0,r-2]\!]$. Therefore $\boldsymbol{x}_{\check{i}}^{aq^\ell}(\boldsymbol{x}_{\check{i}}-x_i)^{q^\ell}\boldsymbol{y}_{\check{i}}^{q^\ell}$ belongs to $\mathbf{Sh}_i\left(\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}\right)$. This proves (4.9). Similarly $y_i^{q^u}\boldsymbol{y}^{q^v} - y_i^{q^v}\boldsymbol{y}^{q^u}$ belongs clearly to $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}$ and vanishes at i. Hence $y_i^{q^u}\boldsymbol{y}_{\check{i}}^{q^v} - y_i^{q^v}\boldsymbol{y}_{\check{i}}^{q^u}$ belongs to $\mathbf{Sh}_i\left(\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}\right)$. This proves (4.11). □

Lemma 4.5. *Let $\mathscr{B} \stackrel{def}{=} \mathscr{A}_{r-1}(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}}-x_i))^\perp$ and $\mathscr{D}' \stackrel{def}{=} \mathscr{B}\star\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)$, then if $r\geq q+1$:*

$$\mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}} \subseteq \mathscr{D}'.$$

Proof. We use the same notation as in Lemma 4.4 and observe that $\mathscr{D}_0$ and $\mathscr{D}_1$ defined below are both subcodes of $\mathscr{D}'_{\mathbb{F}_{q^m}}$:

$$\begin{aligned}\mathscr{D}_0 \stackrel{\text{def}}{=} \mathscr{B}_{\mathbb{F}_{q^m}}\star\mathscr{C}_0 &= \left\langle\, \boldsymbol{x}_{\check{i}}^{aq^\ell}(\boldsymbol{x}_{\check{i}}-x_i)^{q^\ell}\left(y_i^{q^u}\boldsymbol{y}_{\check{i}}^{q^v+q^\ell} - y_i^{q^v}\boldsymbol{y}_{\check{i}}^{q^u+q^\ell}\right) \;\middle|\; a\in[\![0,r-2]\!],\ell,u,v\in[\![0,m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}}\\ \mathscr{D}_1 \stackrel{\text{def}}{=} \mathscr{B}^{\star 2}_{\mathbb{F}_{q^m}} &= \left\langle\, \boldsymbol{x}_{\check{i}}^{aq^j+bq^\ell}(\boldsymbol{x}_{\check{i}}-x_i)^{q^j+q^\ell}\boldsymbol{y}_{\check{i}}^{q^j+q^\ell} \;\middle|\; a,b\in[\![0,r-2]\!],\ j,\ell\in[\![0,m-1]\!] \,\right\rangle_{\mathbb{F}_{q^m}}\end{aligned}$$

This is a direct consequence of $\mathscr{D}'_{\mathbb{F}_{q^m}} = \mathscr{B}_{\mathbb{F}_{q^m}} \star \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)_{\mathbb{F}_{q^m}}$ (definition of $\mathscr{D}'$ and Point 5 in Lemma 4.2) and Lemma 4.4. We also observe that

$$\mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}} = \left\langle \left(\boldsymbol{x}_{\check{i}}\right)^{aq^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \left(\boldsymbol{y}_{\check{i}}\right)^{q^\ell+1} \mid a \in [\![0, r-2]\!], l \in [\![0, m-1]\!] \right\rangle_{\mathbb{F}_{q^m}}.$$

Our proof strategy is to start with an element appearing in the vector span above and by suitable additions of elements of $\mathscr{D}_i$ (where $i \in \{0,1\}$), and possibly also by multiplying by some elements in $\mathbb{F}_{q^m}$, results at the end in an element in $\mathscr{D}_i$. This will prove our lemma. We will use here the notation

$$\boldsymbol{u} \xrightarrow{\mathscr{D}_i} \boldsymbol{v}$$

to write that $\boldsymbol{v}$ can be obtained from $\boldsymbol{u}$ by adding a suitable element of $\mathscr{D}_i$ and multiplying by some element in $\mathbb{F}_{q^m}$, i.e. this is equivalent to $\boldsymbol{u} - \lambda\boldsymbol{v} \in \mathscr{D}_i$ for a suitable element λ in $\mathbb{F}_{q^m}$. It is readily seen that for any a in $[\![0, r-2]\!]$, u and v in $[\![0, m-1]\!]$ and any polynomial P in $\mathbb{F}_{q^m}[X]$ of degree $\leq r-2$ we have

$$P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+q^u} \xrightarrow{\mathscr{D}_0} P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+q^v} \tag{4.12}$$

$$P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \xrightarrow{\mathscr{D}_1} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell}. \tag{4.13}$$

The first reduction follows by noticing that

$$\begin{aligned} P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+q^u} &= P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell} y_i^{-q^v} \left(y_i^{q^v} \boldsymbol{y}_{\check{i}}^{q^u} - y_i^{q^u} \boldsymbol{y}_{\check{i}}^{q^v} + y_i^{q^u} \boldsymbol{y}_{\check{i}}^{q^v}\right) \\ &= \boldsymbol{d} + y_i^{q^u-q^v} P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+q^v} \end{aligned}$$

where $\boldsymbol{d} = P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell} y_i^{-q^v} \left(y_i^{q^v} \boldsymbol{y}_{\check{i}}^{q^u} - y_i^{q^u} \boldsymbol{y}_{\check{i}}^{q^v}\right)$ clearly belongs to $\mathscr{D}_0$. The second reduction follows by performing the Euclidean division of $P(X)$ by $(X - x_i)$. We can namely write $P(X) = (X - x_i)Q(X) + P(x_i)$ for a polynomial Q of degree $\deg P - 1$. Therefore

$$\begin{aligned} P(\boldsymbol{x}_{\check{i}})^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} &= \left((\boldsymbol{x}_{\check{i}} - x_i)Q(\boldsymbol{x}_{\check{i}}) + P(x_i)\right)^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \\ &= \left((\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} Q(\boldsymbol{x}_{\check{i}})^{q^\ell} + P(x_i)^{q^\ell}\right) (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \quad (4.14) \\ &= \boldsymbol{d} + P(x_i)^{q^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \end{aligned}$$

where (4.14) follows from the $\mathbb{F}_q$-linearity of the Frobenius action $x \mapsto x^{q^\ell}$ and $\boldsymbol{d} = \left((\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} Q(\boldsymbol{x}_{\check{i}})^{q^\ell}\right) (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell}$ belongs obviously to $\mathscr{D}_1$.

Let us show the inclusion by performing for a generator $\boldsymbol{x}_{\check{i}}^{aq^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+1}$ of $\mathscr{D}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}$ a sequence of reductions

$$\begin{aligned} \boldsymbol{x}_{\check{i}}^{aq^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^\ell+1} &\xrightarrow{\mathscr{D}_0} \boldsymbol{x}_{\check{i}}^{aq^\ell} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \\ &\xrightarrow{\mathscr{D}_1} (\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} \end{aligned}$$

The crucial argument is now the simple observation that

$$(\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} = \left((\boldsymbol{x}_{\check{i}} - x_i)^q\right)^{q^{\ell^-}}$$

where $\ell^- \stackrel{\text{def}}{=} \ell - 1$ if $\ell > 0$ and $\ell^- \stackrel{\text{def}}{=} m - 1$ if $\ell = 0$. This is a consequence of the fact that the entries of $\boldsymbol{x}$ are in $\mathbb{F}_{q^m}$. This suggests the following sequence of reductions

$$\begin{aligned}(\boldsymbol{x}_{\check{i}} - x_i)^{q^\ell} \boldsymbol{y}_{\check{i}}^{2q^\ell} &\xrightarrow{\mathscr{D}_0} \left((\boldsymbol{x}_{\check{i}} - x_i)^q\right)^{q^{\ell^-}} \boldsymbol{y}_{\check{i}}^{q^\ell + q^{\ell^-}} = \left((\boldsymbol{x}_{\check{i}} - x_i)^{q-1}\right)^{q^{\ell^-}} (\boldsymbol{x}_{\check{i}} - x_i)^{q^{\ell^-}} \boldsymbol{y}_{\check{i}}^{q^{\ell^-} + q^\ell} \\ &\xrightarrow{\mathscr{D}_0} \left((\boldsymbol{x}_{\check{i}} - x_i)^{q-1}\right)^{q^{\ell^-}} (\boldsymbol{x}_{\check{i}} - x_i)^{q^{\ell^-}} \boldsymbol{y}_{\check{i}}^{2q^{\ell^-}} = \left((\boldsymbol{x}_{\check{i}} - x_i)^{q-2}\right)^{q^{\ell^-}} (\boldsymbol{x}_{\check{i}} - x_i)^{2q^{\ell^-}} \boldsymbol{y}_{\check{i}}^{2q^{\ell^-}}.\end{aligned}$$

Note that the last reduction could be performed because the degree of the polynomial $(\boldsymbol{x}_{\check{i}} - x_i)^{q-1}$, which is $q-1$, is less than or equal to $r-2$ by assumption on r. For the very same reason ($r \geq q+1$) we observe that the right-hand term $\left((\boldsymbol{x}_{\check{i}} - x_i)^{q-2}\right)^{q^{\ell^-}} (\boldsymbol{x}_{\check{i}} - x_i)^{2q^{\ell^-}} \boldsymbol{y}_{\check{i}}^{2q^{\ell^-}}$ belongs to $\mathscr{D}_1$ which finishes the proof. □

We are ready now to prove Theorem 4.2.

Proof of Theorem 4.2. From Lemma 4.3, we know that $\mathscr{C}_{\mathbb{F}_{q^m}} = \left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}})^\perp\right)_{\mathbb{F}_{q^m}}$ can be decomposed as

$$\mathscr{C}_{\mathbb{F}_{q^m}} = \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \oplus \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}}.$$

This implies that

$$\begin{aligned}\mathscr{B}_{\mathbb{F}_{q^m}} \star \mathscr{C}_{\mathbb{F}_{q^m}} &= \underbrace{\mathscr{B}_{\mathbb{F}_{q^m}} \star \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}}}_{\mathscr{D}'_{\mathbb{F}_{q^m}}} + \mathscr{B}_{\mathbb{F}_{q^m}} \star \langle\, \boldsymbol{y}_{\check{i}} \,\rangle_{\mathbb{F}_{q^m}} \\ &= \mathscr{B}_{\mathbb{F}_{q^m}} \star \left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)\right)_{\mathbb{F}_{q^m}} \quad \text{(by Lemma 4.5)} \\ &= \mathscr{D}'_{\mathbb{F}_{q^m}}.\end{aligned}$$

The equality of the extended codes over $\mathbb{F}_{q^m}$ implies the equalities of the codes over $\mathbb{F}_q$ which ends the proof. □

4.3.2 Complexity of computing the filtration

The core of the first part of the attack consists in the computation of the conductor from Conjecture 4.1. With the same conditions, and using Proposition 4.4, we need to compute a basis for the linear code

$$\left(\mathscr{C} \star \mathscr{D}^\perp\right)^\perp,$$

where $\mathscr{C} \stackrel{\text{def}}{=} \mathscr{A}_r(\boldsymbol{x}_\mathcal{I}, \boldsymbol{y}_\mathcal{I})^\perp$, $\mathscr{D} \stackrel{\text{def}}{=} \mathbf{Sh}_\mathcal{I}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp\right)^{\star 2}$ for $|\mathcal{I}| = 1$, starting from a generator matrix of $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^\perp$. We can choose $\mathcal{I} = \{1\}$ and assume that the matrix is in systematic form. In other words we analyses one iteration of the filtration, assuming that the current alternant code length and degree are n and r respectively.

An upper bound for the total cost can be roughly obtained by multiplying the cost of the first iteration by the number of iterations.

Computing $\mathscr{C}$ can be done in $\mathcal{O}(1)$, since it is enough to drop the first position from $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp$. Moreover $\mathbf{Sh}_1\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)$ can be computed in $\mathcal{O}1$ too, whenever the shortened position belongs to the chosen information set, by removing the first row and the first column from the basis, because the first position belongs to the information set. Then the corresponding matrix is still in systematic form. Producing a basis for a component-wise product of two codes $\mathscr{A}$ and $\mathscr{B}$ needs the computation of all possible pairs of basis element in $\mathscr{A}$ and one in $\mathscr{B}$. Therefore it has a linear cost in the length, as well as a linear cost in the dimensions of the two codes $\mathscr{A}$ and $\mathscr{B}$. In the case of a square code, the number of pairs to consider is roughly halved. As a minor improvement, if the code is in systematic form, we can avoid to compute the products in the positions corresponding to the information set for any pair of elements in the basis, since we know them in advance. In particular, since $\dim_{\mathbb{F}_q}\mathbf{Sh}_1\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right) = rm-1$, the cost for calculating $\mathscr{D}$ is given by

$$\binom{rm}{2}\cdot(n-rm) = \mathcal{O}(r^2m^2n).$$

The basis obtained in this way, however, is not in systematic form. To be more accurate, a row reduction is partially done already, because the set of vectors arising from squares of basis vectors is already reduced. Since we want to compute the dual code of $\mathscr{D}$, we need to row reduce its generator matrix. We remark that this $\mathscr{D}$ has unusual small dimension, being contained into the distinguishable code $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)^{\star 2}$ and its dimension is upper bounded by $\binom{rm+1}{2} - D$, where D quantifies how much the code is dinstiguishable according to Proposition 3.2, i.e. $D = \frac{m}{2}(r-1)\left((2e_{\mathscr{A}}+1)r - 2\frac{q^{e_{\mathscr{A}}+1}-1}{q-1}\right)$. Depending on such a value, a smaller amount of vectors have to be reduced. For simplicity, we can bound the dimension of $\mathscr{D}$ with $\binom{rm+1}{2}$ and consequently the complexity of the row reduction step for any value of D with

$$\binom{rm+1}{2}^2 n = \mathcal{O}(r^4m^4n).$$

Computing $\mathscr{D}^\perp$ then has an additional cost of

$$\left(\binom{rm+1}{2} - D\right)n = \mathcal{O}(r^2m^2n).$$

Now, since $\dim_{\mathbb{F}_q}\mathscr{D}^\perp = n - \binom{rm+1}{2} - D$, the product $\mathscr{C}\star\mathscr{D}^\perp$ can be performed in

$$rm\left(n - \binom{rm+1}{2} + D\right)\cdot n = \mathcal{O}(rmn^2).$$

According to Conjecture 4.1, the component-wise product of codes obtained is an alternant code of degree $(r-1)m$ and length $n-1$, hence of dimension $n-(r-1)m = \mathcal{O}(n)$. Therefore a row reduced echelon form of its generator matrix can be provided on average in

$$\mathcal{O}(n^\omega),$$

where $2 \le \omega < 3$ is the constant of linear algebra. In this way the conductor can be computed as the dual of the obtained product with an additional complexity of $\mathcal{O}(rmn)$.

Therefore the cost for computing the conductor is dominated by the computation of $\mathscr{C} \star \mathscr{D}^{\perp}$ and/or the row reductions. Bearing in mind that we have $r^2m^2 = \mathcal{O}(n)$ in the distinguishable regime, the overall complexity for one iteration of the filtration is upper bounded by

$$\mathcal{O}(n^3).$$

If we add the costs for each iteration from alternant code degree r down to $q+1$ and taking into account that the length decreases by only 1 at each step, we can upper bound the total complexity with

$$\sum_{i=q+1}^{r} (r-i)n^3 = \mathcal{O}(rn^3).$$

4.3.3 What is wrong with Goppa codes?

Before moving to the second part of the attack, we make a short digression on how the arguments explained so far (do not) apply to the Goppa case. The discussion below does not represent a proof that computing a filtration is impossible for Goppa codes, but rather an intuition about what hampers it. Goppa codes behave differently from random alternant codes and provide counterexamples to Heuristic 4.1. The latter should be replaced by

Heuristic 4.1. *Let $\mathscr{G}(\boldsymbol{x}, \Gamma) \stackrel{def}{=} \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be a random Goppa code of degree r, with $r \geq q-1$ and $\left(\mathscr{G}(\boldsymbol{x}, \Gamma)^{\perp}\right)^{\star 2}$ being different from the full code. Choose an arbitrary code position i and let $\mathscr{C} \stackrel{def}{=} \left(\mathbf{Sh}_i\left(\mathscr{G}(\boldsymbol{x}, \Gamma)\right)\right)^{\perp}$ and $\mathscr{D} \stackrel{def}{=} \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)^{\star 2}$. Then, with high probability,*

$$\boldsymbol{Cond}(\mathscr{C}, \mathscr{D}) = \mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}.$$

Obtaining new codes, namely $\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))$ for any $i \in [\![1, n]\!]$, still brings forth the question whether some standard constructions starting from this code (e.g. shortening, squaring, intersecting it with other codes, etc.) can lead to a different filtration. We address the question in this subsection but first, we give an interpretation of this heuristic by looking at the dimension of the intersection of some involved codes. We start by stating a monotonicity result about the conductor, which follows straightforwardly from its definition (as well as from its closed-form expression):

Proposition 4.7. *Let $\mathscr{C}, \mathscr{C}', \mathscr{D}, \mathscr{D}' \in \mathbb{F}_q^n$ be linear codes such that $\mathscr{C} \subseteq \mathscr{C}'$ and $\mathscr{D} \supseteq \mathscr{D}'$. Then*

$$\boldsymbol{Cond}(\mathscr{C}, \mathscr{D}) \supseteq \boldsymbol{Cond}\left(\mathscr{C}', \mathscr{D}'\right).$$

Now we recall the special form of the parity-check subcode of a Goppa code.

Definition 4.3. Let $\mathscr{C} \subseteq \mathbb{F}^n$. The *parity-check subcode* of $\mathscr{C}$ is

$$\widetilde{\mathscr{C}} \stackrel{\text{def}}{=} \{(c_1, \ldots, c_n) \in \mathscr{C} \mid \sum_{i=1}^{n} c_i = 0\}.$$

The linear constraint $\sum_{i=1}^{n} c_i = 0$ appearing in the definition implies that either all the codewords in $\mathscr{C}$ already satisfy it (in this case $\mathscr{C}$ is a parity-check code) or the dimension of $\tilde{\mathscr{C}}$ is one less than the one of $\mathscr{C}$. This is even more evident when translating the result in terms of the dual code, i.e.

$$\widetilde{\mathscr{C}}^{\perp} = \mathscr{C}^{\perp} + \langle \mathbf{1} \rangle_{\mathbb{F}},$$

where $\mathbf{1}$ is the constant length-n word with entries 1.

The fact that the conductor arising from Heuristic 4.1 preserves the same degree is unfortunate, since our approach heavily builds upon the fact that the degree of the conductor decreases. Moreover, it will turn out that the code we obtain as a conductor could have been obtained directly by shortening a suitable code. Actually, it will turn out that for Goppa codes, there are several codes that are very close to each other and which are obtained by various shortenings. This is summarized by the following proposition. We will explain in what follows why this phenomenon is the main obstacle to applying our conductor approach.

Proposition 4.8. *Let $\mathscr{G}(\boldsymbol{x}, \Gamma) \overset{def}{=} \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$ be a Goppa code of degree r. We have for any code positions i and j with $i \neq j$:*

$$\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp} = \mathscr{G}(\boldsymbol{x}, \Gamma)^{\perp} + \langle \mathbf{1} \rangle_{\mathbb{F}_q} \quad \textit{(from Proposition 1, [Ber00])} \tag{4.15}$$

$$\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp} = \mathbf{Sh}_i\left(\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right) \tag{4.16}$$

$$\mathbf{Sh}_j\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}\right) = \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}_{\check{j}}, \boldsymbol{y}_{\check{j}}(\boldsymbol{x}_{\check{j}} - x_j))^{\perp}\right). \tag{4.17}$$

Proof. **Proof of (4.15).** (4.15) is the dual counterpart of Proposition 1, [Ber00].

Proof of (4.16).

Choose an $\mathbb{F}_q$-basis $\{\alpha_1, \cdots, \alpha_m\}$ of $\mathbb{F}_{q^m}$. We are first going to prove that

$$\mathscr{G}(\boldsymbol{x}_{\check{i}}, \Gamma)^{\perp} + \langle \mathbf{1} \rangle_{\mathbb{F}_q} = \mathscr{A}_r(\boldsymbol{x}_{\check{i}}, i_{\check{\boldsymbol{y}}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp} + \langle \mathbf{1} \rangle_{\mathbb{F}_q}. \tag{4.18}$$

$$\begin{aligned}
\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp} &= \left\langle \operatorname{Tr}\left(\alpha_j \boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i)\boldsymbol{y}_{\check{i}}\right) \mid a \in [\![0, r-1]\!], j \in [\![0, m-1]\!] \right\rangle_{\mathbb{F}_q} \\
&= \left\langle \operatorname{Tr}\left(\alpha_j \boldsymbol{x}_{\check{i}}^{a+1} \boldsymbol{y}_{\check{i}}\right) - \operatorname{Tr}\left(\alpha_j x_i \boldsymbol{x}_{\check{i}}^a \boldsymbol{y}_{\check{i}}\right) \mid a \in [\![0, r-1]\!], j \in [\![0, m-1]\!] \right\rangle_{\mathbb{F}_q} \\
&\subseteq \left\langle \operatorname{Tr}\left(\alpha_j \boldsymbol{x}_{\check{i}}^b \boldsymbol{y}_{\check{i}}\right) \mid b \in [\![0, r]\!], j \in [\![0, m-1]\!] \right\rangle_{\mathbb{F}_q} \\
&= \mathscr{A}_{r+1}(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}})^{\perp} \\
&= \mathscr{G}(\boldsymbol{x}_{\check{i}}, \Gamma)^{\perp} + \langle \mathbf{1} \rangle_{\mathbb{F}_q}
\end{aligned}$$

On the other hand, since $\Gamma(x_i) \neq 0$, then $\Gamma(\boldsymbol{x}_{\check{i}}) \notin \left\langle \boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i) \mid a \in [\![0, r-1]\!] \right\rangle_{\mathbb{F}_q}$. Therefore $\left\langle \Gamma(\boldsymbol{x}_{\check{i}}), \boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i) \mid a \in [\![0, r-1]\!] \right\rangle_{\mathbb{F}_q}$ is a vector space of dimension $r+1$ of evaluations of polynomials with degree at most r. Hence

$$\left\langle \Gamma(\boldsymbol{x}_{\check{i}}), \boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i) \mid a \in [\![0, r-1]\!] \right\rangle_{\mathbb{F}_q} = \left\langle \boldsymbol{x}_{\check{i}}^b \mid b \in [\![0, r]\!] \right\rangle_{\mathbb{F}_q}.$$

Since $\mathrm{Tr}\left(\alpha_j \Gamma(\boldsymbol{x}_{\check{i}}) \boldsymbol{y}_{\check{i}}\right) = \mathrm{Tr}\left(\alpha_j \cdot \mathbf{1}\right) \in \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q}$, we get

$$\begin{aligned}
\mathscr{G}(\boldsymbol{x}_{\check{i}}, \Gamma)^{\perp} + \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q} &= \left\langle\, \mathrm{Tr}\left(\alpha_j \boldsymbol{x}_{\check{i}}^b \boldsymbol{y}_{\check{i}}\right) \mid b \in [\![0, r]\!], j \in [\![0, m-1]\!] \,\right\rangle_{\mathbb{F}_q} \\
&\subseteq \left\langle\, \mathrm{Tr}\left(\alpha_j \boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i) \boldsymbol{y}_{\check{i}}\right) \mid a \in [\![0, r-1]\!], j \in [\![0, m-1]\!] \,\right\rangle_{\mathbb{F}_q} + \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q} \\
&= \mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp} + \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q}\,.
\end{aligned}$$

Because of the last point, $\mathbf{Sh}_i\left(\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)$ is the set of codewords of $\mathscr{G}(\boldsymbol{x}, \Gamma) + \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q}$ which evaluate to 0 at position i. Clearly the elements of $\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}$ viewed as polynomial evaluations and extended canonically at position i as the linear space

$$\{\mathrm{Tr}\left(\boldsymbol{y}(\boldsymbol{x} - x_i) P(\boldsymbol{x})\right) :\ P \in \mathbb{F}_{q^m}[X],\ \deg P < r\}$$

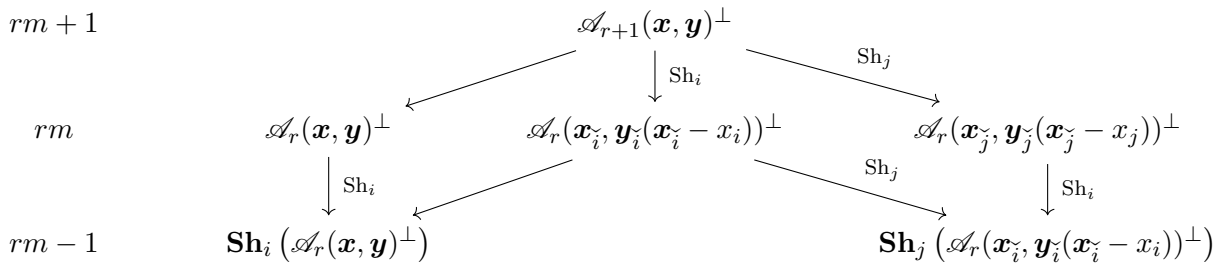
belong to this set. Since the all one vector does not meet this property and because of (4.18) this implies the point (4.16).

Proof of (4.17)**.** This is just a consequence that $\mathbf{Sh}_i\left(\mathbf{Sh}_j\left(\mathscr{C}\right)\right) = \mathbf{Sh}_j\left(\mathbf{Sh}_i\left(\mathscr{C}\right)\right)$ which holds for any code $\mathscr{C}$. Here we apply it to $\mathscr{C} = \mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}$ and apply the previous point:

$$\begin{aligned}
\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}_{\check{j}}, \boldsymbol{y}_{\check{j}}(\boldsymbol{x}_{\check{j}} - x_j))^{\perp}\right) &= \mathbf{Sh}_i\left(\mathbf{Sh}_j\left(\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right) \text{ (by (4.16))} \\
&= \mathbf{Sh}_j\left(\mathbf{Sh}_i\left(\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right) \text{ (by the previous remark)} \\
&= \mathbf{Sh}_j\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}\right) \text{ (by (4.16))}
\end{aligned}$$

□

We can summarize these relationships with the diagram below, where arrows mean an inclusion of the lower code into the upper code and two arrows pointing at the same code represent the intersection. The typical code dimensions are shown too.



The inclusions shown by the picture above turn out to be equalities, due to dimension arguments. Indeed, $\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp}$ contains both $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}$ and $\mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}$. The two latter codes do not coincide and their dimension is only 1 less than the former code. Therefore

$$\mathscr{A}_{r+1}(\boldsymbol{x}, \boldsymbol{y})^{\perp} = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} + \mathscr{A}_r(\boldsymbol{x}_{\check{i}}, \boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}} - x_i))^{\perp}$$

and the same argument shows that

$$\mathscr{A}_{r+1}(\boldsymbol{x},\boldsymbol{y})^{\perp} = \mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\check{i}}(\boldsymbol{x}_{\check{i}}-x_i))^{\perp} + \mathscr{A}_r(\boldsymbol{x}_{\check{j}},\boldsymbol{y}_{\check{j}}(\boldsymbol{x}_{\check{j}}-x_j))^{\perp}$$

On the other hand, $\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_j))^{\perp}\right)$ is contained in both $\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}})^{\perp}$ and $\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_i))^{\perp}$, and its dimension is only one less than the dimensions of the two latter codes. Hence

$$\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}})^{\perp} \cap \mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_i))^{\perp} = \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right).$$

and the same argument shows that

$$\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_i))^{\perp} \cap \mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_j))^{\perp} = \mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}_{\check{i}},\boldsymbol{y}_{\mathcal{I}}(\boldsymbol{x}_{\check{i}}-x_j))^{\perp}\right).$$

We have seen that the codes related to $\mathscr{G}(\boldsymbol{x},\Gamma)$ and that we can compute are very close to each other. This property seems to be inherited by the square codes. Indeed, consider the following heuristic, obtained by experimental computation.

Heuristic 4.2. *Let $\mathscr{G}(\boldsymbol{x},\Gamma) \stackrel{def}{=} \mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be a random Goppa code of degree r, with $r \geq q-1$. Then, with high probability,*

$$\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} \subseteq \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2}. \tag{4.19}$$

While counterexamples to (4.19) can be artificially constructed by choosing appropriately Γ, we never found one in our experiments when $\boldsymbol{x}$ and Γ are sampled at random.

Assuming that the inclusion (4.19) holds, two results readily follow.

Proposition 4.9. *Let $\mathscr{G}(\boldsymbol{x},\Gamma) \stackrel{def}{=} \mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be a Goppa code over $\mathbb{F}_q$, with $r \geq q-1$. Assume that Equation (4.19) is satisfied. Then*

$$\left(\widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}}\right)^{\star 2} = \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + \langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q}.$$

Proof.

$$\begin{aligned}
\left(\widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}}\right)^{\star 2} &= \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} + \langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q}\right)^{\star 2} \\
&= \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} \star \langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q} + \left(\langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q}\right)^{\star 2} \\
&= \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} + \langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q} \\
&= \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} + \langle\,\mathbf{1}\,\rangle_{\mathbb{F}_q} \quad \text{(by (4.19))}
\end{aligned}$$

□

The proposition above implies that $\dim_{\mathbb{F}_{q^m}}\left(\widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}}\right)^{\star 2} \leq \dim_{\mathbb{F}_{q^m}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + 1$.

As a consequence, we also have

Proposition 4.10. *Let $\mathscr{G}(\boldsymbol{x},\Gamma) \stackrel{def}{=} \mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ be a Goppa code over $\mathbb{F}_q$, with $r \geq q-1$. Assume that Equation (4.19) is satisfied. Then*

$$\boldsymbol{Cond}\left(\widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)}^{\perp}, \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2}\right) \supseteq \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}.$$

Proof. We have

$$\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} \star \widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)}^{\perp} = \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} \star \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} + \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} \star \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q} = \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2} + \mathscr{G}(\boldsymbol{x},\Gamma)^{\perp} = \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2},$$

where the last equality follows from Equation (4.19). Hence the largest code $\mathscr{X}$ such that $\mathscr{X} \star \widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)}^{\perp} \subseteq \left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)^{\star 2}$ contains $\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}$. □

Remark 4.4. The inclusion from Proposition 4.10 is tight, meaning that we typically have equality within the distinguishable regime.

It follows from Proposition 4.7 that, in order to get a small conductor $\mathbf{Cond}(\mathscr{C},\mathscr{D})$, we need to choose large $\mathscr{C}$ and small $\mathscr{D}$. We experimented in a systematic way, by choosing between duals of codes we have access to, and by eventually considering shortening and/or intersection. To give the reader an insight of the behavior of this strategy, we just state the following experimental result, which we expect to hold with very high probability when $r \geq q-1$ and $\mathcal{I}$ is small enough (otherwise we have some degenerate behavior):

$$\mathbf{Cond}\left(\mathscr{G}(\boldsymbol{x}_{\check{\mathcal{I}}},\Gamma)^{\perp} + \langle\, \mathbf{1} \,\rangle_{\mathbb{F}_q}, \left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}\right) = \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right).$$

Thus, this kind of conductor does not provide new information.

4.4 Algebraic cryptanalysis

The previous section explains how to obtain, under some conditions, the alternant code $\mathscr{A}_3(\boldsymbol{x}',\boldsymbol{y}')$ with support $\boldsymbol{x}' = \boldsymbol{x}_{\check{\mathcal{I}}}$ and multiplier $\boldsymbol{y}' = \boldsymbol{y}_{\check{\mathcal{I}}}\left(\prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\check{\mathcal{I}}} - x_i)\right)$ for some $\mathcal{I} \subseteq [\![1,n]\!]$ such that $|\mathcal{I}| = r-3$, and with length $n' = n-r+3$ and degree 3, starting from the knowledge of the length-n public code $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$. For the sake of clarity, in this section we perform algebraic cryptanalysis on the alternant code $\mathscr{A}_3(\boldsymbol{x},\boldsymbol{y})$ of length n. Essentially, we can ignore the structure of $\boldsymbol{y}'$ and the decreased length because the filtration preserves the support and multiplier randomness and the code distinguishability. At the end of the analysis, we will see how to get back a support and a multiplier defining $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ (not necessarily $\boldsymbol{x}$ and $\boldsymbol{y}$) from a support and a multiplier defining $\mathscr{A}_3(\boldsymbol{x}',\boldsymbol{y}')$ (not necessarily $\boldsymbol{x}'$ and $\boldsymbol{y}'$). Moreover, we will focus on the case $r=3$ for the system resolution, but the algebraic modeling is more general and makes sense for any $r \geq 3$. We also remark that this section has a more general validity in terms of field size. The full attack needs the filtration to reach degree 3, and therefore works specifically for $q=2$ or $q=3$. On the other hand, taking this part alone, we can claim a polynomial time attack on alternant codes of degree 3 for any field size. This additional result is also original, and to the best of our knowledge, no polynomial time attack was known on non-structured alternant or Goppa codes even for $r=3$. We also remark that in the binary case it does not make sense to reach degree 2 through the filtration. Indeed, the smallest degree for which alternant codes behave differently from random linear codes is $r=3$, and the analysis we are going to present in this section would not be applicable for $r=2$.

4.4.1 The algebraic modeling from [Fau+13]

Given the support $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ and the multiplier $\boldsymbol{y} = (y_1 \ldots, y_n) \in \mathbb{F}_{q^m}^n$, we recall from Chapter 3 the alternative definition of the alternant code $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, which turns out to be more suitable for this section:

$$\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \left\{ \boldsymbol{c} \in \mathbb{F}_q^n \mid \boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y}) \boldsymbol{c}^T = \boldsymbol{0} \right\},$$

where

$$\boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{bmatrix} y_1 & \cdots & y_n \\ y_1 x_1 & \cdots & y_n x_n \\ \vdots & \ddots & \vdots \\ y_1 x_1^{r-1} & \cdots & y_n x_n^{r-1} \end{bmatrix}.$$

We will adopt the notation and follow the description of the algebraic model presented in [Fau+13]. We denote with $\boldsymbol{G} = (g_{i,j}) \in \mathbb{F}_q^{k \times n}$ the $k \times n$ generator matrix of $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$. Equation (1.1) thus becomes

$$\boldsymbol{V}_r(\boldsymbol{x}, \boldsymbol{y}) \boldsymbol{G}^T = \boldsymbol{0},$$

which is equivalent to the following polynomial system:

$$\left\{ \sum_{j=1}^{n} g_{i,j} Y_j X_j^e = 0 \mid i \in [\![1, k]\!], e \in [\![0, r-1]\!] \right\},$$

where $\boldsymbol{X} \stackrel{\text{def}}{=} (X_1, \ldots, X_n)$ and $\boldsymbol{Y} \stackrel{\text{def}}{=} (Y_1, \ldots, Y_n)$ are two blocks of n unknowns, each corresponding to the support and multiplier coordinates respectively. Observe that the sought vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfy indeed the polynomial system.

We can assume, up to a permutation of columns, that $\boldsymbol{G}$ is in systematic form, i.e. $\boldsymbol{G} = [\boldsymbol{I}_k \mid \boldsymbol{P}]$, where $\boldsymbol{I}_k$ is the identity matrix of size k and $\boldsymbol{P} = (p_{i,j})$ for $i \in [\![1, k]\!], j \in [\![k+1, n]\!]$. The polynomial system can be therefore rewritten as

$$\left\{ Y_i X_i^e = - \sum_{j=k+1}^{n} p_{i,j} Y_j X_j^e \mid i \in [\![1, k]\!], e \in [\![0, r-1]\!] \right\}. \tag{4.20}$$

Moreover, when the alternant code is sampled randomly, we expect that $k = n - rm$.

From now on, we focus on the case $r = 3$. The assumptions made above can be summarized in

Assumption 4.1 (Random alternant code)**.** *We assume that $\mathscr{A}_3(\boldsymbol{x}, \boldsymbol{y})$ is in standard form, and that its dimension satisfies $k = n - rm = n - 3m$.*

As explained in [Fau+13], thanks to the systematic form assumption, we can get rid of several variables and consider an algebraic system in only $2(n-k)$ unknowns. For $r = 3$, we can choose in (3.6) the tuple $(a, b, c, d, l) = (2, 0, 1, 1, 0)$ and get the corresponding identity

$$Y_i (Y_i X_i^2) = (Y_i X_i)^2$$

for $i \in [\![1, k]\!]$. With this choice, the algebraic system (3.3) specializes into

Modeling 4.1 (Alternant/Goppa codes modeling [Fau+13], $r = 3$)**.**
System:

$$\mathcal{S} \stackrel{def}{=} \left\{ \sum_{(j,j')\in J} p_{i,j}p_{i,j'}Y_jY_{j'}(X_j - X_{j'})^2 \mid i \in [\![1,k]\!] \right\}, \tag{4.21}$$

where $J = \{(j,j') \in \mathbb{N}^2 \mid k+1 \leq j < j' \leq n\}$.
Unknowns: $2rm = 6m$ *unknowns* X_i, Y_i, $i \in [\![k+1,n]\!]$.
Equations: $k = n - 3m$ *equations over* $\mathbb{F}_q$ *of bidegree* $(2,2)$ *in* $(\boldsymbol{X},\boldsymbol{Y})$.

The rank of System (4.21) is trivially upper bounded by the number of expressions $Y_jY_{j'}(X_j - X_{j'})^2$, i.e. by $|J| = \binom{n-k}{2}$. However, in the high rate regime, the distinguisher from [Fau+13] and revisited in Chapter 3 shows that the upper bound is tighter. We place ourselves within the distinguishable regime and we assume that the upper bound is tight, i.e.

Assumption 4.2 (High rate regime)**.** *We assume that*

- $\mathbf{Rank}(\mathcal{S}) = \binom{3m}{2} - m \leq k$ *if* $q \geq 3$*;*
- $\mathbf{Rank}(\mathcal{S}) = \binom{3m}{2} - 3m \leq k$ *if* $q = 2$.

This implies that, even after the change of variables $Z_{j,j'} \stackrel{\text{def}}{=} Y_jY_{j'}(X_j - X_{j'})^2$, the number of unknowns is larger than the number of independent equations and linearization techniques lead to a solution space of very large dimension, which also includes many wrong solutions. Therefore, in the following, we are going to explain how to tackle this problem with more advanced techniques, namely Gröbner basis.

4.4.2 Reducing the number of solutions

System (4.21) contains many solutions. Some of them, including of course the actual private key, are valid pairs of support and multiplier for the public code. Other solutions are parasitic. The aim is to remove all the "wrong" solutions from the system and reduce the number of good "solutions". First of all, notice that if $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ is a valid support-multiplier pair, then for any $l \in [\![1,m-1]\!]$, $(\bar{\boldsymbol{x}}^{q^l},\bar{\boldsymbol{y}}^{q^l})$ is too. This readily follows from the fact that these pairs are obtained from $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ by applying l times the Frobenius morphism $z \mapsto z^q$ component-wise and that alternant codes are subfield subcodes defined over $\mathbb{F}_q$. This is reflected within Modeling 4.1: $(\bar{\boldsymbol{x}},\bar{\boldsymbol{y}})$ is a solution of System (4.21) if and only if $(\bar{\boldsymbol{x}}^{q^l},\bar{\boldsymbol{y}}^{q^l})$ is. Therefore we can see the space of solutions as made by blocks of m solutions related by the Frobenius map.

Furthermore, the ideal generated by $\mathcal{S}$ is not zero-dimensional and this is not due to the structure highlighted so far. It will be convenient here to reduce to this case by specializing appropriately some variables. The positive dimension of this ideal is due in the first instance to the degrees of freedom for the support and multiplier coordinates. In essence, this is due to the fact that a homography $z \mapsto \frac{az+b}{cz+d}$, $ad - bc \neq 0$, maps the support $\boldsymbol{x}$ of an alternant code to another support describing the same alternant code (but possibly with a different multiplier) at the condition that $cx_i + d$ never vanishes. When there exists a value x_i of the support of the alternant code for which $cx_i + d = 0$, the resulting code is not an alternant code, but

belongs to a slightly larger family of codes: it will be a subfield subcode of a Cauchy code. Let us recall its definition taken from [Dür87]. Given a field $\mathbb{F}$ we can identify the projective line $\mathcal{P}^1(\mathbb{F})$ with $\bar{\mathbb{F}} \stackrel{\text{def}}{=} \mathbb{F} \cup \{\infty\}$, where the symbol ∞ is called *point at infinity*, through the map $\phi : \bar{\mathbb{F}} \to \mathbb{F}^2 \setminus \{0\}, \phi(e) = (e, 1)$ if $e \in \mathbb{F}$ and $\phi(\infty) = (1, 0)$. Moreover, let $\mathbb{F}[W, Z]_l^H$ be the set of homogeneous polynomials of degree l in two variables W, Z. Given $P \in \mathbb{F}[W, Z]_l^H$ and $e \in \bar{\mathbb{F}}$, we define $P(e) \stackrel{\text{def}}{=} P(\phi(e))$. Then

Definition 4.4 (Cauchy code)**.** Let $\boldsymbol{x} \stackrel{\text{def}}{=} (x_1, \ldots, x_n) \in \bar{\mathbb{F}}^n$ be a vector of distinct elements and $\boldsymbol{y} \stackrel{\text{def}}{=} (y_1, \ldots, y_n) \in \mathbb{F}^n$ be a vector of nonzero elements. Let $r \in [\![0, n]\!]$. The **Cauchy code** $\mathscr{C}_r(\boldsymbol{x}, \boldsymbol{y})$ is defined as

$$\mathscr{C}_r(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 P(x_1), \ldots, y_n P(x_n)) \mid P \in \mathbb{F}[W, Z]_{r-1}^H\}.$$

As in the case of generalized Reed-Solomon codes, $\boldsymbol{x}$ is called a *support* and $\boldsymbol{y}$ a *multiplier* of the Cauchy code.

If we assume $x_n = \infty$, a generator matrix of $\mathscr{C}_r(\boldsymbol{x}, \boldsymbol{y})$ is given by

$$\begin{bmatrix} y_1 & \cdots & y_{n-1} & 0 \\ y_1 x_1 & \cdots & y_{n-1} x_{n-1} & 0 \\ \vdots & \ddots & \vdots & \\ y_1 x_1^{r-1} & \cdots & y_{n-1} x_{n-1}^{r-1} & y_n \end{bmatrix}. \tag{4.22}$$

On the other hand, when $\boldsymbol{x} \in \mathbb{F}^n$, i.e. when all the x_i's are different from ∞, the Cauchy code $\mathscr{C}_r(\boldsymbol{x}, \boldsymbol{y})$ can be easily seen as $\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})$. They are also MDS codes. Cauchy codes are then a generalization of GRS codes. Analogously subfield subcodes of Cauchy codes generalize subfield subcodes of GRS codes, i.e. alternant codes.

One of the main results of [Dür87] was to characterize the possible supports and multipliers of Cauchy codes. In particular, it is proven there that

Theorem 4.3. *[Dür87] Let $r \in [\![2, n-2]\!]$. Then $\mathscr{C}_r(\boldsymbol{x}, \boldsymbol{y}) = \mathscr{C}_r(\boldsymbol{x}', \boldsymbol{y}')$ if and only if there exists a homography $f(z) = \frac{az+b}{cz+d}$ ($a, b, c, d \in \mathbb{F}$, $ad - bc \neq 0$) such that $\boldsymbol{x}' = f(\boldsymbol{x})$ and $\boldsymbol{y}' = \lambda \theta(\boldsymbol{x})^{r-1} \boldsymbol{y}$ where $\lambda \in \mathbb{F} \setminus \{0\}$ and*

$$\begin{aligned} \theta(z) = cz + d \quad & \text{if } z \in \mathbb{F} \text{ and } cz + d \neq 0, \\ \theta(z) = (ad - bc)/(-c) \quad & \text{if } z \in \mathbb{F} \text{ and } cz + d = 0, \\ \theta(\infty) = c \quad & \text{if } c \neq 0, \\ \theta(\infty) = a \quad & \text{if } c = 0. \end{aligned}$$

Since the elements a, b, c, d in $f(z) = \frac{az+b}{cz+d}$ are defined up to a multiplication by a nonzero scalar, Theorem 4.3 pragmatically implies that we are allowed to fix three variables in block $\boldsymbol{X}$ and one in block $\boldsymbol{Y}$.

Remark 4.5. A sufficient condition for the homography $z \mapsto \frac{az+b}{cz+d}$ to map a GRS code into another GRS code, regardless of the support vector, is to be a linear map $z \mapsto az + b$, i.e. $c = 0$ (and $d \neq 0$). It is then clear that there are only two degrees of freedom for the support, up to multiplication by a nonzero scalar. This would allow us to fix only 3 variables: two of them in block $\boldsymbol{X}$ and one in $\boldsymbol{Y}$. Moreover, for full-support GRS codes, this condition becomes necessary as well, because $cx_i + d = 0$ for $x_i = -d/c$.

The price to pay for the additional specialization is that now we have to eventually handle the point at infinity. We have seen that the column corresponding to the point at infinity in the generator matrix of a Cauchy code has a special form and this changes for this coordinate the form of the system $\mathcal{S}$ given in (4.21). The problem is that we do not know a priori which x_i will be infinite. To circumvent this problem, we choose the value x_i that will be set to infinity, say x_n.

Concerning which other unknowns to specialize, there exist two different options up to a permutation of columns: either the index of the fixed $\boldsymbol{Y}$ variable also corresponds to a fixed $\boldsymbol{X}$ variable (e.g. we fix X_{n-2}, X_{n-1}, X_n and Y_n), or not (e.g. we fix X_{n-2}, X_{n-1}, X_n and Y_{n-3}). Both these choices lead to very comparable performance and behavior when computing the Gröbner basis algorithm described later in this section. We select the former option, and we fix the $\boldsymbol{Y}$ unknown with the same index as the $\boldsymbol{X}$ variable specialized in ∞, since this choice results in a slightly easier analysis.

Then, we have to decide the remaining three values to fix. Again, we notice that this choice does not affect the behavior of the algorithm nor the shape of the Gröbner basis or the number of degree falls during each step of its computation. However, some useful expedients lead to a good choice of such values. Indeed we notice that, specializing over the subfield $\mathbb{F}_q$ preserves the membership of the coefficients in $\mathcal{S}$ to $\mathbb{F}_q$. This brings two advantages:

1. All the operations among equations in the system become operations among their coefficients, i.e. additions and multiplications over $\mathbb{F}_q$ and not over $\mathbb{F}_{q^m}$. For instance, when computing a Gröbner basis, the computer algebra system Magma applies direct coercion to the subfield, resulting in a practical speed up. Instead, the same does not happen if at least one variable is fixed over $\mathbb{F}_{q^m} \setminus \mathbb{F}_q$.

2. Due to the $\mathbb{F}_q$-linearity of the Frobenius action, the m solutions related by this automorphism have the same specialization. We will be able to obtain an algebraic variety with exactly m elements. With this specialization, all these elements are good pairs of support and multiplier for defining the sought alternant code. This point will become more clear once we will have described the shape of the Gröbner basis. Being able to choose whichever solution will also reduce the linear algebra work in the final linear part of the attack.

We remark that, since the multiplier coordinates must be different from zero, the only choice that makes sense for any field size consists in fixing the block $\boldsymbol{X}$ unknowns in 0 and 1 and the block $\boldsymbol{Y}$ unknown in 1 (e.g. $\mathbb{F}_2 = \{0, 1\}$).

It is also convenient to opt for values that belong to the subfield $\mathbb{F}_q$ over which the alternant code is defined: all the Gröbner bases computations will stay in the subfield and this results in slightly improved computation times. To summarize, we made the following choice (which also simplifies slightly the analysis of the Gröbner basis computations):

$$X_{n-2} = 0, \quad X_{n-1} = 1, \quad X_n = \infty, \quad Y_n = 1, \tag{4.23}$$

which results, for the $r = 3$ case, in the following Vandermonde matrix

$$\boldsymbol{V}_3(\boldsymbol{X}, \boldsymbol{Y}) \stackrel{\text{def}}{=} \left[\begin{array}{ccc|ccc} Y_1 & \dots & Y_{n-3} & Y_{n-2} & Y_{n-1} & 0 \\ Y_1X_1 & \dots & Y_{n-3}X_{n-3} & 0 & Y_{n-1} & 0 \\ Y_1X_1^2 & \dots & Y_{n-3}X_{n-3}^2 & 0 & Y_{n-1} & 1 \end{array}\right].$$

With this specialization, the system $\mathcal{S}$ becomes

Proposition 4.11. *We can choose part of the support and multiplier as $X_{n-2} = 0$, $X_{n-1} = 1$, $X_n = \infty$, $Y_n = 1$ and obtain the following algebraic system*

$$\begin{aligned} \mathcal{S}' \stackrel{def}{=} \Big\{ & \textstyle\sum_{k+1 \le j < j' \le n-3} p_{i,j}p_{i,j'}Y_jY_{j'}(X_j - X_{j'})^2 \\ & + \textstyle\sum_{j=k+1}^{n-3} p_{i,j}p_{i,n-2}Y_jY_{n-2}X_j^2 + \sum_{j=k+1}^{n-3} p_{i,j}p_{i,n-1}Y_jY_{n-1}(X_j - 1)^2 \\ & + p_{i,n-2}p_{i,n-1}Y_{n-2}Y_{n-1} + \textstyle\sum_{j=k+1}^{n-1} p_{i,j}p_{i,n}Y_j \mid i \in [\![1, k]\!] \Big\}. \end{aligned} \tag{4.24}$$

Proof. The fact that we can choose the support and the multiplier in this way follows from the fact that homographies act 3-transitively on the projective plane and from Theorem 4.3. To obtain the algebraic system we proceed similarly to what was done for obtaining the algebraic system $\mathcal{S}$: we write $Y_i(Y_iX_i^2) = (Y_iX_i)^2$ for $i \in [\![1, k]\!]$ and use this time that

$$\begin{aligned} Y_i &= \sum_{k+1 \le j \le n-1} p_{i,j}Y_j \\ Y_iX_i &= \sum_{k+1 \le j \le n-1} p_{i,j}Y_jX_j \\ Y_iX_i^2 &= \sum_{k+1 \le j \le n-1} p_{i,j}Y_jX_j^2 + p_{i,n}. \end{aligned}$$

□

The arising modeling is thus given by *affine* degree-4 equations with 4 less variables:

Modeling 4.2 (Alternant/Goppa codes modeling [Fau+13], specialized, $r = 3$)**.**
System: *$\mathcal{S}'$ as defined in (4.24).*
Unknowns: *$2rm-4 = 6m-4$ unknowns X_i, Y_j, $i \in [\![k+1, n-3]\!], j \in [\![k+1, n-1]\!]$.*
Equations: *$k = n - 3m$ affine equations over $\mathbb{F}_q$ of bidegree $(2, 2)$ in $(\boldsymbol{X}, \boldsymbol{Y})$.*

However, the set of solutions of the system $\mathcal{S}'$ given in (4.24) still contains at least a component of positive dimension $n - k - 1 = rm - 1$, that corresponds to the solutions of $\{Y_j = 0 \mid k + 1 \le j \le n - 1\}$. The classical way to deal with the parasite solutions $Y_j = 0$ is to introduce to the system a new variable T_j together with the polynomial $T_jY_j - 1$. This ensures that $Y_j = 0$ is not a solution to the system. However, this also adds variables to the system and increases the degree of the polynomials during a Gröbner basis computation. The same phenomenon occurs with the constraints $X_j - X_{j'} \neq 0$. We solve these problems in an easier way in Steps (2) and (3) of the algorithm presented in the next section. We conclude this subsection by giving more details about the strategy above and explaining the theory behind it. The reader interested in the actual attack can skip it and directly jump to the next subsection.

Let us recall the definition of *ideal quotient* (not to be confused with the quotient with respect to an ideal) or *colon ideal*:

Definition 4.5 (Colon Ideal)**.** Let $\mathcal{I}, \mathcal{J}$ be two ideals of a commutative ring R, the **colon ideal** $\mathcal{I}\colon\mathcal{J}$ is defined as

$$\mathcal{I}\colon\mathcal{J} = \{r \in R \mid r\mathcal{J} \subseteq \mathcal{I}\}.$$

The colon ideal $\mathcal{I}\colon\mathcal{J}$ is an ideal of R, too.

In the particular case where R is a multivariate polynomial ring in block $\boldsymbol{Z} = (Z_1, \ldots, Z_l)$, we are going to see how the constraints $p_1(\boldsymbol{Z}) \neq 0, \ldots, p_j(\boldsymbol{Z}) \neq 0$ can be taken into account by computing $\mathcal{I}\colon\mathcal{J}$, where $\mathcal{J} \stackrel{\text{def}}{=} \langle p_1, \ldots, p_j \rangle_{\mathbb{K}}$ is the ideal generated by $p_1 \ldots, p_j$.

We also need to recall what the *saturation* of a polynomial ideal with respect to another ideal is.

Definition 4.6. Let $\mathcal{I}, \mathcal{J} \subseteq \mathbb{K}[\boldsymbol{Z}]$ be two ideals. The **saturation** $\mathcal{I}\colon\mathcal{J}^\infty$ of $\mathcal{I}$ with respect to $\mathcal{J}$ is defined as

$$\mathcal{I}\colon\mathcal{J}^\infty = \{f \in \mathbb{K}[\boldsymbol{Z}] \mid \forall\, g \in \mathcal{J},\, \exists n \geq 0 \text{ s.t. } fg^N \in \mathcal{I}\}.$$

From a geometric point of view, saturation roughly purges the parasite solutions that correspond to the zeros of $\mathcal{J}$.

Proposition 4.12 (Theorem 10 p.203, [CLO15])**.** *Let $\mathcal{I}, \mathcal{J} \subseteq \mathbb{K}[\boldsymbol{Z}]$ be two ideals. Then*

- $\overline{\boldsymbol{V}(\mathcal{I}) \setminus \boldsymbol{V}(\mathcal{J})} \subseteq \boldsymbol{V}(\mathcal{I}\colon\mathcal{J}^\infty)$,
- *if $\mathbb{K}$ is algebraically closed, then $\overline{\boldsymbol{V}(\mathcal{I}) \setminus \boldsymbol{V}(\mathcal{J})} = \boldsymbol{V}(\mathcal{I}\colon\mathcal{J}^\infty)$.*

When $\mathcal{I}$ is radical, which we expect to be the case in our application, the result above can be extended to the colon ideal:

Proposition 4.13 (Corollary 11 p. 204, [CLO15])**.** *Let $\mathcal{I}, \mathcal{J} \in \mathbb{K}[\boldsymbol{Z}]$ be two ideals. If $\mathbb{K}$ is algebraically closed and $\mathcal{I}$ is radical, then*

$$\overline{\boldsymbol{V}(\mathcal{I}) \setminus \boldsymbol{V}(\mathcal{J})} = \boldsymbol{V}(\mathcal{I}\colon\mathcal{J}).$$

Note that $\mathcal{J} \stackrel{\text{def}}{=} \langle p_1, \ldots, p_j \rangle = \sum_{i=1}^{j} \langle p_i \rangle_{\mathbb{K}}$. The next proposition reduces the computation of a quotient ideal/saturation respectively to the intersection of quotient ideals/saturations with respect to principal ideals.

Proposition 4.14 (Proposition 13 p. 204, [CLO15])**.** *Let $\mathcal{I}, \mathcal{J}_1, \ldots, \mathcal{J}_j \subseteq \mathbb{K}[\boldsymbol{Z}]$ be polynomial ideals. Then*

$$\mathcal{I}\colon\sum_{i=1}^{j} \mathcal{J}_i = \bigcap_{i=1}^{j} (\mathcal{I}\colon\mathcal{J}_i),$$

$$\mathcal{I}\colon\left(\sum_{i=1}^{j} \mathcal{J}_i\right)^\infty = \bigcap_{i=1}^{j} (\mathcal{I}\colon\mathcal{J}_i^\infty).$$

Applying the proposition above to $\mathcal{J}_i = \langle p_i \rangle$, we readily obtain

$$\mathcal{I}\colon\mathcal{J} = \bigcap_{i=1}^{j} (\mathcal{I}\colon\langle p_i \rangle).$$

Finally, the next theorem links the basis of an ideal to the basis of its saturation with respect to a principal ideal.

Proposition 4.15 (Theorem 14 p. 205, [CLO15]). *Let $\mathcal{I} = \langle f_1, \ldots, f_m \rangle \subseteq \mathbb{K}[\boldsymbol{Z}]$ and $p \in \mathbb{K}[\boldsymbol{Z}]$. Let $\tilde{\mathcal{I}} = \langle f_1, \ldots, f_m, 1 - T \cdot p \rangle \subseteq \mathbb{K}[\boldsymbol{Z}, T]$ where T is a new variable.*

- *If $\mathcal{I} \cup \langle p \rangle = \langle h_1, \ldots, h_l \rangle$, then*

$$\mathcal{I} : \langle p \rangle = \langle h_1/p, \ldots, h_l/p \rangle .$$

- *Let $\tilde{\mathcal{I}} = \langle f_1, \ldots, f_m, 1 - T \cdot p \rangle \subseteq \mathbb{K}[\boldsymbol{Z}, T]$. Then*

$$\mathcal{I} : \langle p \rangle_{\mathbb{F}_q}^{\infty} = \tilde{\mathcal{I}} \cup \mathbb{K}[\boldsymbol{Z}].$$

Moreover, if G is a Gröbner basis of $\tilde{\mathcal{I}}$ with respect to an elimination order such that $T > Z_i$, then $G \cup \mathbb{K}[\boldsymbol{Z}]$ is a basis of $\mathcal{I} : \langle p \rangle_{\mathbb{F}_q}^{\infty}$.

It is possible to devise an algorithm for computing a basis of a colon ideal (or analogously of the saturation) of $\mathcal{I} = \langle f_1, \ldots, f_m \rangle$ with respect to $\mathcal{J} = \langle p_1, \ldots, p_j \rangle$ in the following way:

1. For all $i \in [\![1, j]\!]$ compute a basis of $\mathcal{I} \cup \langle p_i \rangle$. We recall that the computation of an intersection ideal can be performed by exploiting the so-called *elimination theory*. More precisely, the intersection can be expressed as

$$\mathcal{I} \cap \langle p_i \rangle = T_i \cdot \mathcal{I} + (1 - T_i) \langle p_i \rangle \cap \mathbb{K}[\boldsymbol{Z}],$$

   where T_i is a new variable. Hence, it is enough to compute an elimination basis of $T_i \cdot \mathcal{I} + (1 - T_i) \langle p_i \rangle$ such that $T_i > Z_l$ for any l, and take only the elements in the basis that do not contain the variable T_i.

2. Compute separately the bases of the ideal quotients with $\mathcal{I} : \langle p_i \rangle$ using Proposition 4.15.

3. Using elimination theory, calculate a basis of $\mathcal{I} : \mathcal{J}$ by computing recursively a basis of:

   - $\mathcal{I} : \langle p_1, p_2 \rangle = (\mathcal{I} : \langle p_1 \rangle) \cup (\mathcal{I} : \langle p_2 \rangle)$.
   - $\mathcal{I} : \langle p_1, p_2, p_3 \rangle = (\mathcal{I} : \langle p_1, p_2 \rangle) \cup (\mathcal{I} : \langle p_1 \rangle)$.
   - ...

The algorithm depicted here always allows to compute a basis of $\mathcal{I} : \mathcal{J}$ in a finite number of steps. However, this construction entails a growth of the degree in the equations and computing a basis for an elimination order in our application is practically out of reach.

As a workaround, we propose a method that searches for multiples of some polynomials in the vector space generated by a system of equations instead of in the ideal generated by the system itself. This construction does not guarantee finding such multiples, even if they belong to the generated ideal. Nevertheless, we will show that for our system and parameters, this strategy finds a non-trivial vector space of multiples and produces new equations.

4.4.3 The algorithm for q odd

Let $\boldsymbol{X}' = (X_{k+1}, \dots, X_{n-3})$ and $\boldsymbol{Y}' = (Y_{k+1}, \dots, Y_{n-1})$, i.e. the two blocks of variables of System (4.24), i.e. after specialization. For the rest of the section, we consider the grevlex order of the monomials of $\mathbb{F}_q[\boldsymbol{X}', \boldsymbol{Y}']$ where the variables are ordered like $X_{k+1} > X_{k+2} > \cdots X_{n-3} > Y_{k+1} > Y_{k+2} > \cdots > Y_{n-1}$. We present here the algorithm for q odd used to compute the solutions of the system $\mathcal{S}'$ given in (4.24) that satisfy $X_j - X_{j'} \neq 0$ and $Y_j \neq 0$, under the assumption that $\mathbf{Rank}(\mathcal{S}') = \binom{3m}{2} - m \leq k$. This is equivalent to Assumption 4.2, because, even after specialization, the terms $Z_{j,j'}$ are still linearly independent.

Generic Gröbner basis algorithms are not expected to solve efficiently systems with the same degree and same number of unknowns and equations as the one described before. Here, however, some expedients can be taken into account to exploit the very strong algebraic structure and specific shape of the equations involved. Hence, an ad hoc algorithm based on Gröbner basis can be designed to recover the secret key in this case. In particular, we agglomerate in our strategy the constraints $Y_j \neq 0$ and $X_j - X_{j'} \neq 0$.

We will give a full and detailed explanation of this approach for the odd case, i.e. when q is the power of an odd prime. This outline covers for instance the case $q = 3$, for which a full key recovery can be achieved, thanks to the filtration of alternant codes. The even case requires some modifications and we will briefly mention which adjustments are needed at the end.

1. (Echelonizing step at degree 4) We compute a basis of the $\mathbb{F}_q$-vector space $\mathscr{S}'$ generated by $\mathcal{S}'$. It contains $2m - 1$ homogeneous linear polynomials in Y, that come from the choice $X_n = \infty$ and $Y_n = 1$ (see Proposition 4.17 for the proof). This can be done in $O(m^{2\omega})$ operations in $\mathbb{F}_q$ by linear algebra on the Macaulay matrix $\mathrm{Mac}(\{g_1, \cdots, g_k\}, 4)$, where ω is the linear algebra constant.

2. (Removing the $Y_j = 0$ component) For each $j \in [\![k+1, n-1]\!]$, we prove that there exists a set of $2m - 1$ linearly independent polynomials in $\mathscr{S}'$ that are multiple of Y_j. As we know that our solution satisfies $Y_j \neq 0$, we add to the system the set V_j of these polynomials divided by Y_j (see Proposition 4.18 for details and proof). This has the effect to add $(2m-1)(3m-1)$ linearly independent polynomials of degree 3 to the system, and to remove the nonzero-dimensional component from the solution set. Note that Step (1) corresponds to the computation of V_n, as $Y_n = 1$. The cost for all j is $O(m^{\omega+1})$.

3. (Finding low-degree equations from the constraint $X_{j_1} \neq X_{j_2}$) For each $j \in [\![k+1, n-1]\!] \setminus \{n-2\}$, we consider the vector spaces $\mathscr{U}_{j,n-2}$ formed by the polynomials p such that $X_j p \in \mathscr{V}_j + \mathscr{V}_{n-2}$, where $\mathscr{V}_j$ is the $\mathbb{F}_q$-vector space generated by V_j. We prove in Proposition 4.19 that $\dim_{\mathbb{F}_q}(\mathscr{U}_{j,n-2}) \geq m$. Experimentally, this set has dimension exactly m. As we know that our solution satisfies $x_j \neq 0 = x_{n-2}$, we add to the system a basis $U_{j,n-2}$ of $\mathscr{U}_{j,n-2}$. This has the effect to add $m(3m-2)$ linearly independent polynomials of degree 2 to the system, and to remove the last spurious solutions. The cost for all j is $O(m^{\omega+1})$.

4. (Eliminating $2m - 1$ variables Y_j using the linear polynomials from Step (1)) We now eliminate $2m - 1$ variables Y from the polynomials in $U_{j,n-2}$ using

the $2m-1$ homogeneous linear polynomials in Y from Step (1). This step is heuristic, but verified experimentally: we get a basis that contains an additional affine linear polynomial in Y, so that we get in total all $2m$ linearly independent linear polynomials in Y_j that we can expect (see Proposition 4.16). The other polynomials in the basis express all monomials $Y_jX_{j'}$ linearly in terms of the Y_j's. The cost of this step is $O(m^{2\omega})$.

5. (Computing linear polynomials in the X variables) This step is proven, provided that Step (4) occurred as described. From the coefficients of the affine linear polynomial in Y, we can compute for each j one affine linear polynomial expressing X_j in terms of the Y's: these polynomials belong to the augmented system at degree $(2,1)$ in (X,Y), but we provide a trick to compute them directly, see Proposition 4.21. The cost is $O(m^2)$.

6. (Computing the Gröbner basis) By eliminating the X_i's from the polynomials for $Y_jX_{j'}$, we get the final grevlex Gröbner basis of the system, that is

$$\begin{cases} Y_jY_{j'} - L'_{j,j'}(Y_i : i \in I_1) & j,j' \in I_1 \\ X_j - L'_{X_j}(Y_i : i \in I_1, 1) & j \in [\![k+1, n-3]\!] \\ Y_j - L'_{Y_j}(Y_i : i \in I_1, 1) & j \in [\![k+1, n-1]\!] \setminus I_1 \end{cases}$$

for a set $I_1 \subset [\![k+1, n-1]\!]$ of size $m-1$, where the functions L' are affine linear functions. This describes a variety of dimension 0 with m solutions, that are exactly the m solutions obtained by applying the Frobenius morphism. The cost is $O(m^{2\omega})$.

7. (Computing the solutions) The lex basis can be obtained using the FGLM Algorithm from [Fau+93] with $O(m^4)$ operations in $\mathbb{F}_q$, and allows to retrieve the m solutions by factorization of a polynomial over $\mathbb{F}_q$ of degree $\leq m$.

8. The final step consists in retrieving separately the values for $Y_1, \ldots, Y_k$ and $X_1, \ldots, X_k$ from (4.20) for $e=0$ and $e=1$. This costs $O(nm)$ operations in $\mathbb{F}_q$. This needs to be done only once, since with the chosen specialization any of the m solutions is a valid pair of support and multiplier coordinates, thus we can choose arbitrarily any of them.

Note that all steps are just linear algebra, and the total complexity is polynomial in m and n, the global cost being $O(m^{2\omega} + nm)$ as $m, n \to \infty$.

4.4.4 Theoretical and experimental validation of the algebraic algorithm

We start with a property that will be useful to determine the number of linearly independent polynomials in $\mathcal{S}'$. Recall that we assumed without loss of generality that $y_n = 1$.

Proposition 4.16. *Let $\mathscr{C}$ be the $\mathbb{F}_{q^m}$ linear code generated by $(y_{k+1}, \ldots, y_{n-1})$ in $\mathbb{F}_{q^m}^{mr-1}$. Then, under Assumption 4.1, we have*

$$\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathscr{C})) = m,$$
$$\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathscr{C})^\perp) = (r-1)m - 1.$$

As any $\mathbb{F}_q$-linear combination of the y_j's that is equal to zero provides a codeword in $\mathrm{Tr}(\mathscr{C})^\perp$, therefore there cannot be more than $(r-1)m-1$ linearly independent homogeneous $\mathbb{F}_q$-linear polynomials in $Y_{k+1},\dots,Y_{n-1}$ which cancel on $y_{k+1},\dots,y_{n-1}$, and no more than $(r-1)m$ linearly independent affine $\mathbb{F}_q$-linear polynomials in $Y_{k+1},\dots,Y_{n-1}$ that cancel on $(y_{k+1},\dots,y_{n-1},1)$.

Equivalently, for all $j \in [\![k+1, n-1]\!]$, the code $\mathscr{C}'_j \subset \mathbb{F}_{q^m}^{mr-1}$ generated by $(y_{k+1}(x_{k+1}-x_j)^{r-1},\dots,y_{n-1}(x_{n-1}-x_j)^{r-1},1) \in \mathbb{F}_{q^m}^{mr}$ and punctured in position $j-k$ satisfies

$$\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathscr{C}'_j)) = m,$$
$$\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathscr{C}'_j)^\perp) = (r-1)m-1.$$

Proof. If the code $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})$ has dimension $k = n-mr$ and is in standard form, then the last $n-k=mr$ columns of its parity-check matrix must be an information set, i.e. the last mr columns of $\boldsymbol{V}_r(\boldsymbol{x},\boldsymbol{y})$ must generate a trace code with dimension mr. This means in particular that the first row $(y_{k+1},\dots,y_{n-1},0)$ must have rank weight m, and this is the same for all r rows. Then, by elementary combination of rows, the trace code of the following code must still have dimension m:

$$\begin{pmatrix} 0 & y_{k+2}(x_{k+2}-x_{k+1})^{r-1} & \dots & y_{n-1}(x_{n-1}-x_{k+1})^{r-1} & 1 \end{pmatrix}$$

and this can be done for any x_j instead of x_{k+1}, hence the proposition. □

Step (1): linearizing at degree 4

We linearize the set of polynomials (4.24), by replacing "polynomials" by variables, instead of classically replacing any monomial by a new variable. The variables we consider are:

$$Z_{j,j'} = \begin{cases} Y_jY_{j'}(X_j-X_{j'})^2 & \forall j,j' \in [\![k+1,n-3]\!],\ j<j' \\ Y_jY_{n-2}X_j^2 & \forall j \in [\![k+1,n-3]\!],\ j'=n-2 \\ Y_jY_{n-1}(X_j-1)^2 & \forall j \in [\![k+1,n-3]\!],\ j'=n-1 \\ Y_{n-2}Y_{n-1} & j=n-2,\ j'=n-1 \\ Y_j & \forall j \in [\![k+1,n-3]\!],\ j'=n. \end{cases}$$

It is easy to verify that, under Assumption 4.2, they are linearly independent.

Proposition 4.17. *Let $q \geq 3$. Under Assumptions 4.1 and 4.2, for any set $I \subset [\![k+1,n-1]\!]$ of size m such that $\dim_{\mathbb{F}_q}(\langle y_\ell : \ell \in I\rangle_{\mathbb{F}_q}) = m$, a basis of $\mathscr{S}'$ is given by*

$$\begin{cases} Y_jY_{j'}(X_j-X_{j'})^2 + L_{j,j'}(Y_\ell : \ell \in I) & \forall k+1 \leq j < j' \leq n-1 \\ Y_{n-2}Y_{n-1} + L_{n-2,n-1}(Y_\ell : \ell \in I) & \\ Y_j - L_{j,n}(Y_\ell : \ell \in I) & \forall k+1 \leq j \leq n-1, j \notin I. \end{cases} \tag{4.25}$$

where the $L_{j,j'}$ are linear functions of the Y_ℓ's, $\ell \in I$ (note that $L_{j,j'}$ implicitly depends on I). This basis can be computed in time $O(m^{2\omega})$ where ω is the constant of linear algebra.

Proof. We have $\binom{3m}{2} - m$ polynomials in $\binom{3m}{2}$ variables. Among the variables, $3m-1$ are of degree 1 (the Y_j's for $k+1 \le j \le n-1$), one is of degree 2 ($Z_{n-2,n-1} = -Y_{n-2}Y_{n-1}$, as $X_{n-2} = 0$ and $X_{n-1} = 1$) and the last $\binom{3m}{2} - 3m$ are of degree 4. We can eliminate from the system all terms of degree 4 and 2. As the polynomials are linearly independent, we get at least $2m-1$ linear polynomials in the Y_j's.

By Proposition 4.16, we have at most $2m-1$ linear relations between the $Y_{k+1}, \dots, Y_{n-1}$, hence we have exactly $2m-1$ linear polynomials in the Y_j's expressing any Y_j in terms of the $\{Y_\ell \mid \ell \in I\}$ for some $I \subset [\![k+1, n-1]\!]$ of size m, and all other polynomials express the terms of degree ≥ 2 in terms of the $\{Y_\ell \mid \ell \in I\}$.

To compute the basis it is enough to compute an echelon form of a matrix of size $(\binom{3m}{2} - m) \times \binom{3m}{2}$, the cost is $O(m^{2\omega})$. □

Step (2): removing the $Y_j = 0$ component

The linear polynomials we get come from the fact that we have specialized the nth component to $x_n = \infty$ and $y_n = 1$. Here we show that it is equivalently possible to introduce the constraint $Y_j \neq 0$ for all $j \in [\![k+1, n-1]\!]$. We define the vector spaces

$$\mathscr{V}_j = \frac{1}{Y_j}\left(\mathscr{S}' \cap Y_j \cdot \mathbb{F}_q[\boldsymbol{X}', \boldsymbol{Y}']_{\le 3}\right), \qquad j \in [\![k+1, n-1]\!] \tag{4.26}$$

that is $\mathscr{V}_j \stackrel{\text{def}}{=} \langle \frac{h_1}{Y_j}, \dots, \frac{h_\ell}{Y_j} \rangle_{\mathbb{F}_q}$ where $\{h_1, \dots, h_\ell\}$ is a basis of $\mathscr{S}' \cap (Y_j \cdot \mathbb{F}_q[\boldsymbol{X}, \boldsymbol{Y}]_{\le 3})$. We also define

$$\mathscr{V}_n = \langle Y_j - L_{j,n}(Y_\ell : \ell \in I) \rangle_{j \in [\![k+1, n-1]\!] \setminus I}. \tag{4.27}$$

Proposition 4.18. *Let $q \ge 3$. Under Assumptions 4.1 and 4.2,*

$$\dim_{\mathbb{F}_q}(\mathscr{V}_j) = 2m - 1 \qquad \forall k+1 \le j \le n \tag{4.28}$$

and any polynomial in $\mathscr{V}_j$ is a linear combination of the $3m-1$ terms

$$\begin{cases} Y_{j'}(X_j - X_{j'})^2, & j' \in [\![k+1, n-3]\!] \setminus \{j\} \\ 1, \end{cases}$$

where the variables are specialized as in (4.23).

We also have, for each $j_1, j_2 \in [\![k+1, n-1]\!]$ with $j_1 \neq j_2$:

$$\dim_{\mathbb{F}_q}(\mathscr{V}_{j_1} + \mathscr{V}_{j_2}) = 4m - 2.$$

A basis V_j of $\mathscr{V}_j$ can be computed in time $O(m^\omega)$ from the basis (4.25) of $\mathscr{S}'$ and the set of all V_j's can be computed in time $O(m^{\omega+1})$.

Proof. Choose any set $I \subset [\![k+1, n-1]\!]$ of size m such that $\dim_{\mathbb{F}_q} \langle (y_\ell)_{\ell \in I} \rangle_{\mathbb{F}_{q^m}} = m$. Consider first the case where $j \notin I$, and $j \le n-3$. To compute $\mathscr{V}_j$, we just take the polynomials in (4.25) that contain Y_j, there are $3m-1$ polynomials:

$$\begin{cases} Y_j Y_{j'}(X_j - X_{j'})^2 + L_{j,j'}(Y_\ell : \ell \in I) & \forall k+1 \le j' \le n-1, j' \neq j \\ Y_j - L_{j,n}(Y_\ell : \ell \in I). \end{cases}$$

We have $3m-1$ linearly independent polynomials in m variables Y_ℓ, $\ell \in I$, and $3m-1$ variables $Y_jY_{j'}(X_j - X_{j'})^2$ and Y_j. By eliminating the $Y_\ell : \ell \in I$ we get at least $2m-1$ polynomials that are multiples of Y_j.

If $j \in I$, we have $\dim_{\mathbb{F}_q} \left\langle (y_i)_{i\in[\![k+1,n-1]\!]\setminus\{j\}} \right\rangle_{\mathbb{F}_{q^m}} \in \{m-1, m\}$. If the dimension over $\mathbb{F}_q$ is m, then we can take a different set I that generates a vector space $\langle (y_\ell)_{\ell\in I} \rangle_{\mathbb{F}_{q^m}}$ of dimension m over $\mathbb{F}_q$ such that $j \notin I$. If the dimension over $\mathbb{F}_q$ is $m-1$, then the linear polynomials $Y_{j'} - L_{j',n}(Y_\ell : \ell \in I)$ does not involve Y_j (or we would have a linear polynomial expressing Y_j in terms of $Y_{j'}$ and the Y_ℓ, $\ell \in I \setminus \{j\}$, which is impossible considering that $\dim_{\mathbb{F}_q} \left\langle (y_i)_{i\in[\![k+1,n-1]\!]} \right\rangle_{\mathbb{F}_{q^m}} = m$). In this case, we take the $3m-2$ polynomials involving $Y_jY_{j'}(X_j - X_{j'})^2$ for all $j' \neq j$, they contains those $3m-2$ terms, the variable Y_j and $m-1$ variables $Y_\ell : \ell \in I, \ell \neq j$. By eliminating the $Y_\ell, \ell \in I, \ell \neq j$ we get at least $2m-1$ linear polynomials multiple of Y_j.

For $j \in [\![n-2, n-1]\!]$, it is exactly the same but with one more polynomial involving one more variable $Y_{n-2}Y_{n-1}$.

In any case, we get at least $2m-1$ polynomials involving the monomials $Y_{j'}(X_{j'} - X_j)^2$ for $j' \in [\![k+1, n-1]\!]$, $j' \neq j$ and 1, and all those polynomials evaluate to zero on the support and the multiplier of the code. Now, according to Proposition 4.16 with $r = 3$, $\dim_{\mathbb{F}_q} \langle y_{k+1}(x_{k+1} - x_j)^2, \ldots, y_{n-1}(x_{n-1} - x_j)^2, 1 \rangle_{\mathbb{F}_{q^m}} = m$, hence there can be at most $2m-1$ linear polynomials between the $3m-1$ terms 1 and $Y_{j'}(X_{j'} - X_j)^2$, $j' \in [\![k+1, n-1]\!], j' \neq j$.

Finally, the polynomials in two different $\mathscr{V}_j$'s are linearly independent, as the only common monomial appearing in two different $\mathscr{V}_j$'s is 1, and the ideal is not generated by 1. □

Step (3): finding low-degree equations from the constraint $X_{j_1} \neq X_{j_2}$

The system given by the union of $\mathcal{S}'$ and the $(3m-1)(2m-1)$ cubic equations $V_j, j \in [\![k+1, n-1]\!]$ determined at Step (2) generates a zero-dimensional ideal, whose variety contains exactly m solutions, related by the Frobenius automorphism. It would be enough to run a Gröbner basis for this new system, in order to retrieve the support and multiplier. However, specifically for the q odd case, we are able to deepen the analysis and introduce efficiently the constraints about support coordinates, i.e. $X_{j_1} - X_{j_2} \neq 0$, by computing efficiently a set of bilinear equations. The latter does not refine the variety, nor the ideal, this one being already 0-dimensional, but their prediction allows to speed up the computation.

Proposition 4.19. *Let q be of odd characteristic. The vector space $\mathscr{U}_{n-2,n-1} \stackrel{def}{=} (\mathscr{V}_{n-2} + \mathscr{V}_{n-1}) \cap \mathbb{F}_q[\boldsymbol{X}, \boldsymbol{Y}]_{\leq 2}$ contains more than m linearly independent polynomials of degree 2, that are linear combination of the terms $Y_j(2X_j - 1)$ for $j \in [\![k+1, n-1]\!]$.*

Moreover, $\mathscr{V}_{n-2} + \mathscr{V}_{n-1}$ contains an additional polynomial of degree 2 expressing the monomial 1 in terms of the $Y_j(2X_j - 1)$ for $j \in [\![k+1, n-1]\!]$. We denote by $U_{n-2,n-1}$ this set of $m+1$ polynomials, and by $u_{n-2,n-1}$ the polynomial containing the monomial 1.

Proof. The terms appearing in the polynomials in V_{n-2} are $1, Y_{n-1}$ and $Y_jX_j^2$ for $j \in [\![k+1, n-3]\!]$. The ones in V_{n-1} are $1, Y_{n-2}$ and $Y_jX_j^2 + Y_j(1-2X_j)$, $j \in$

$[\![k+1, n-3]\!]$. This means that the polynomials in $\mathscr{U}_{n-2,n-1}$ can all be expressed as linear combination of the $3m-3$ monomials $Y_j X_j^2$ of degree 3, the monomial 1 of degree 0, plus $3m-1$ terms of degree at most 2: the monomials Y_{n-1}, Y_{n-2} and the $Y_j(2X_j-1)$'s, $j \in [\![k+1, n-3]\!]$. The dimension of the vector space $\mathscr{U}_{n-2,n-1}$ is $4m-2$, so that we get at least m linearly independent polynomials of degree 2 in $\mathscr{U}_{n-2,n-1}$ that are combination of $3m-1$ terms $(Y_{k+1}(2X_{k+1}-1), \ldots, Y_{n-3}(2X_{n-3}-1), Y_{n-2}, Y_{n-1})$. In all cases, we also get an additional polynomial of degree 2 involving these $3m-1$ terms and the monomial 1, and this gives in characteristic 2 an additional affine linear polynomial in Y. □

It will turn out that the polynomial $u_{n-2,n-1}$ will be relevant in the next steps.

This can be generalized to the following vector spaces. For any $k+1 \leq j_1 < j_2 \leq n-1$, define the vector space

$$\mathscr{U}_{j_1,j_2} = \frac{1}{X_{j_1} - X_{j_2}} \left((\mathscr{V}_{j_1} + \mathscr{V}_{j_2}) \cap (X_{j_1} - X_{j_2}) \cdot \mathbb{F}_q[\boldsymbol{X}', \boldsymbol{Y}']_{\leq 2} \right) \tag{4.29}$$

that consists of the polynomials p such that $p \cdot (X_{j_1} - X_{j_2}) \in \mathscr{V}_{j_1} + \mathscr{V}_{j_2}$.

Proposition 4.20. *For any $k+1 \leq j_1 < j_2 \leq n-1$, $(j_1, j_2) \neq (n-2, n-1)$, we have $\dim_{\mathbb{F}_q}(\mathscr{U}_{j_1,j_2}) \geq m$, and the polynomials in $\mathscr{U}_{j_1,j_2}$ are linear combination of the following terms:*

$$Y_j(2X_j - X_{j_1} - X_{j_2}), \qquad j \in [\![k+1, n-1]\!].$$

Proof. The $2m-1$ polynomials in V_{j_1} contains the following $3m-1$ terms:

$$\begin{cases} Y_j(X_j - X_{j_1})^2, & j \in [\![k+1, n-1]\!] \setminus \{j_1\} \\ 1 \end{cases}$$

It is the same for V_{j_2}, but we can rewrite, for $j \in [\![k+1, n-1]\!] \setminus \{j_1, j_2\}$:

$$Y_j(X_j - X_{j_2})^2 = Y_j(X_j - X_{j_1})^2 + Y_j(2X_j - X_{j_1} - X_{j_2})(X_{j_1} - X_{j_2}),$$

so that the $4m-2$ polynomials generating $\mathscr{V}_{j_1} + \mathscr{V}_{j_2}$ can be written in terms of the following terms:

$$\begin{cases} Y_j(X_j - X_{j_1})^2 & j \in [\![k+1, n-1]\!] \setminus \{j_1, j_2\}, \\ 1 & \\ Y_{j_1}(X_{j_1} - X_{j_2})^2 & \\ Y_{j_2}(X_{j_1} - X_{j_2})^2 & \\ Y_j(2X_j - X_{j_1} - X_{j_2})(X_{j_1} - X_{j_2}), & j \in [\![k+1, n-1]\!] \setminus \{j_1, j_2\}. \end{cases}$$

If we eliminate the $3m-2$ first terms that are not multiple of $X_{j_1} - X_{j_2}$, we get at least m linearly independent polynomials that are multiple of $X_{j_1} - X_{j_2}$. After division by $X_{j_1} - X_{j_2}$, the polynomials are linear combination of the $3m-1$ terms $Y_j(2X_j - X_{j_1} - X_{j_2})$ for $j \in [\![k+1, n-1]\!]$. □

Remark 4.6. One could try to adapt to remove the $X_{j_1} = X_{j_2}$ component, by defining the analogous of $\mathscr{V}_j$ for support constraints, i.e. by computing the vector space

$$\frac{1}{X_{j_1} - X_{j_2}} \left(\mathscr{S}' \cap (X_{j_1} - X_{j_2}) \cdot \mathbb{F}_q[\boldsymbol{X}', \boldsymbol{Y}']_{\leq 3} \right).$$

However, it can be readily verified that such vector space is simply $\{\mathbf{0}\}$. This is not surprising: the combinatorial arguments used to derive the dimension of $\mathscr{V}_j$ do not apply here, because there are not enough terms that are divisible by $X_{j_1} - X_{j_2}$, regardless of the change of basis adopted. In other words, we can not take out the $X_{j_1} = X_{j_2}$ component directly from $\mathscr{S}'$ if we restrict the computation to degree 4. On the other hand, we have shown that such component can be removed if we consider equations that are already deprived of some Y_j components.

Despite the existence of a quadratic number of vector spaces U_{j_1,j_2}, in practice, it is sufficient to exploit the equations derived from the $3m-2$ subspaces $U_{\ell,n-2}$, $\ell \in [\![k+1, n-1]\!] \setminus \{n-2\}$, thus reducing the complexity of this step.

Heuristic 4.3. *Experimentally,* $\dim(\mathscr{U}_{j_1,j_2}) = m$ *and for odd* q*,*

$$\dim_{\mathbb{F}_q} \bigcup_{\ell \in [\![k+1,n-1]\!] \setminus \{n-2\}} \mathscr{U}_{\ell,n-2} = m(3m-2).$$

Step (4): eliminating $2m-1$ variables Y_j using the linear polynomials from Step (1)

This part is specific to the case q odd. Assuming Fact 4.3, the system $\oplus_{j_1 \in [\![k+1,n-3]\!]} \mathcal{U}_{j_1,n-2}$ contains $m(3m-2)$ linearly independent polynomials, and they can all be expressed as linear combination of the monomials $Y_j, j \in [\![k+1, n-1]\!]$ and $Y_j X_{j'}$ for $j \in [\![k+1, n-1]\!]$ and $j' \in [\![k+1, n-3]\!]$.

The system $\mathcal{V}_n$ contains $2m-1$ homogeneous linear polynomials in the Y_j's, expressing the Y_i's for $i \notin I$ in term of the Y_ℓ's for $\ell \in I$. If we use them to eliminate the $2m-1$ variables Y_i's ($i \notin I$) from the polynomials in $\oplus_{j_1 \in [\![k+1,n-3]\!]} \mathcal{U}_{j_1,n-2}$, we are left with polynomials that are linear combinations of m linear monomials $\{Y_\ell \mid \ell \in I\}$, and $m(3m-3)$ quadratic monomials $Y_j X_{j'}$ for $j \in I$ and $j' \in [\![k+1, n-3]\!]$. This means that we have as many polynomials as monomials. However, the polynomials now have no reason to remain linearly independent, and in fact they are not.

Experimentally, after linearization, we get one polynomial expressing each quadratic term $Y_j X_{j'}$ in terms of the m independent $\{Y_\ell : \ell \in I\}$ and m reductions to zero, as we cannot get more than $2m-1$ linear polynomials relating the Y_j's: a basis $\mathcal{U}$ of $\oplus_{j_1} \mathscr{U}_{j_1,n-2}$ modulo $\mathcal{V}_n$ has the shape

$$\mathcal{U} \stackrel{\text{def}}{=} \{Y_j X_{j'} + L_{j,j'}(Y_\ell : \ell \in I) \mid j \in I, j' \in [\![k+1, n-3]\!]\}.$$

We can now use the polynomial $u_{n-2,n-1}$ from Proposition 4.19, which is a linear combination of the monomials $1, Y_{n-2}, Y_{n-1}$ and the $Y_j(2X_j - 1)$ for $j \in [\![k+1, n-3]\!]$. We eliminate the Y_j's, $j \notin I$ using equations in $\mathcal{V}_n$ and the $Y_j X_{j'}$ for $j \in I, j' \in [\![k+1, n-3]\!]$ using $\mathcal{U}$ and obtain a linear polynomial in the Y_j's and 1. Note that, as we already have $2m-1$ homogeneous polynomials between the Y_j's, we cannot have another homogeneous polynomial, hence this polynomial contains a nonzero constant term.

To perform the elimination and the linearization, we can perform linear algebra on a matrix where the columns are the $Y_j X_{j'}$, hence $\mathcal{O}m^2$ columns, and the rows are the basis for $\mathscr{U}_{j,n-2}$ and the $X_i L_j$ with L_j a linear polynomial in $\mathscr{V}_n$. Hence, the matrix has $\mathcal{O}(m^2)$ rows and the complexity becomes $\mathcal{O}(m^{2\omega})$.

Step (5): computing linear polynomials for the X variables

Proposition 4.21. *Assume that a basis of $\cup_{j\neq n-2}\mathscr{U}_{j,n-2}$ where the linear polynomials from $\mathcal{V}_n$ have been eliminated is given by*

$$\begin{cases} Y_j X_{j'} + L_{j,j'}(Y_\ell : \ell \in I, 1), & \text{for all } j \in I, j' \in [\![k+1, n-3]\!], \\ \sum_{j\in I} a_j Y_j - 1, & \text{with } a_j \in \mathbb{F}_q. \end{cases} \tag{4.30}$$

Then the vector space generated by the polynomials (4.30) contains the polynomials

$$X_{j'} + \sum_{j\in I} a_j L_{j,j'}(Y_\ell : \ell \in I, 1), \qquad j' \in [\![k+1, n-3]\!]. \tag{4.31}$$

Proof. We have

$$\sum_{j\in I} a_j (Y_j X_{j'} + L_{j,j'}(Y_\ell : \ell \in I, 1)) = X_{j'} + X_{j'}(\sum_{j\in I} a_j Y_j - 1) + \sum_{j\in I} a_j L_{j,j'}(Y_\ell : \ell \in I, 1)$$

so that we get in the ideal generated by (4.30) one affine linear polynomial expressing each $X_{j'}$ in terms of the monomials in $\{1\} \cup \{Y_\ell \mid \ell \in I\}$. □

Step (6): the final Gröbner basis

Now, if we use the polynomials in (4.31) to eliminate the X_i's from the polynomials in (4.30), we get one linear polynomial for each term of degree 2 in $\boldsymbol{Y}$. Let I_1 be the set I minus one element $i \in I$ such that $a_i \neq 0$. The final basis has the shape

$$\begin{cases} Y_j Y_{j'} - L'_{j,j'}(Y_i : i \in I_1) & j, j' \in I_1 \\ X_j - L'_{X_j}(Y_i : i \in I_1, 1) & j \in [\![k+1, n-3]\!] \\ Y_j - L'_{Y_j}(Y_i : i \in I_1, 1) & j \in [\![k+1, n-1]\!] \setminus I_1 \end{cases}$$

This describes a variety of dimension 0. The Hilbert series is $H(t) = (m-1)t + 1$ as $\#I_1 = m - 1$, only m monomials $\{1\} \cup \{Y_i \mid i \in I_1\}$ are not leading term of a polynomial in the ideal. Thus, by Remark 1.27, the system has exactly m solutions, which are precisely the m solutions obtained by applying the Frobenius morphism. This proves that the basis is a Gröbner basis. It coincides with the basis that would have been computed from $\mathscr{S}'$ plus the cubic equations from the V_j's. However, our approach is more efficient, because it avoids unnecessary calculations.

Step (7): finding the solutions

Since the ideal is zero-dimensional, we can obtain the Gröbner basis for the lexicographic order from the grevlex one in polynomial time, using the FGLM algorithm [Fau+93]. The destination basis contains a polynomial where only the smallest (non-fixed) variable, i.e. Y_{n-1}, appears. By factoring this polynomial, we

get the m (counted with multiplicity) possible values for Y_{n-1} and then recover the other variables too. In particular, when the smallest extension on $\mathbb{F}_q$ to which the $(n-1)$-th coordinates of the m actual $\bar{\boldsymbol{y}}$ solutions belong is $\mathbb{F}_{q^m}$, the lexicographic basis is given by

$$\begin{cases} X_j + Q_{X_j}(Y_{n-1}) = 0, & j \in [\![k+1, n-3]\!] \\ Y_j + Q_{Y_j}(Y_{n-1}) = 0, & j \in [\![k+1, n-2]\!] \\ P(Y_{n-1}) = 0 \end{cases} \tag{4.32}$$

where P is a polynomial of degree m and Q_{X_j}, Q_{Y_j} are polynomials of degree smaller than the degree of P (compare with Lemma 1.11). From the factorization of the univariate polynomial P we get its roots. We can pick one of them, replace Y_{n-1} with its value in the other polynomials of the lex basis and retrieve all the other unknowns by equating those polynomials to 0.

Remark 4.7. Up to reordering the Y variables in a different way, we can always assume to obtain the lexicographic basis as in (4.32). Indeed, if this is not the case, then all the actual $\bar{\boldsymbol{y}}$'s solutions would be defined over a proper subfield of $\mathbb{F}_{q^m}$. But this is in contradiction with Proposition 4.16 and, consequently, with our assumptions.

4.4.5 Differences in the $q = 2^s$ case

As anticipated, when the field characteristic is 2 we point out some discrepancies with respect to the analysis just given. First of all, in this case the system $\mathcal{S}'$ can be rewritten as

$$\mathcal{S}' = \left\{ \sum_{k+1 \le j < j' \le n-1} p_{i,j} p_{i,j'} Y_j Y_{j'} (X_j^2 + X_{j'}^2) + \sum_{j=k+1}^{n-1} p_{i,j} p_{i,n} Y_j \mid i \in [\![1, k]\!] \right\}.$$

Since the X_j's variables appear in the system with power 2 only, we can perform a change of variables by defining $W_j \stackrel{\text{def}}{=} X_j^2$, so that the system becomes

$$\mathcal{S}'_2 = \left\{ \sum_{k+1 \le j < j' \le n-1} p_{i,j} p_{i,j'} Y_j Y_{j'} (W_j + W_{j'}) + \sum_{j=k+1}^{n-1} p_{i,j} p_{i,n} Y_j \mid i \in [\![1, k]\!] \right\}. \tag{4.33}$$

Therefore, equations in $\mathcal{S}'_2$ have bidegree $(1, 2)$ in $\boldsymbol{W} \stackrel{\text{def}}{=} \{W_1, \ldots, W_n\}$ and $\boldsymbol{Y}$ respectively. This simple trick decreases the maximum degree reached by the Gröbner basis, thus providing an effective speed up to the resolution. With this change of variables, the adopted specialization (4.23) becomes

$$W_{n-2} = 0, \quad W_{n-1} = 1, \quad W_n = \infty, \quad Y_n = 1. \tag{4.34}$$

Other differences depend on whether $q = 2$ or $q = 2^s,\ s > 1$. We therefore split the discussion into two subcases.

Overview for $q = 2^s,\ s > 1$

1. (Echelonizing step at degree 3) The initial echelonization is analogous to the one for q odd, with the only difference that equations have bidegree $(1, 2)$ in $(\boldsymbol{W}, \boldsymbol{Y})$, i.e. total degree 3. In particular, Proposition 4.17 still holds, recalling that the terms $(X_j - X_{j'})^2$'s can be replaced by $W_j + W_{j'}$'s.

2. (Removing the $Y_j = 0$ component) we can compute the V_j's similarly to what was done for the odd case. Proposition 4.18 still applies, in particular $\dim_{\mathbb{F}_q} \mathscr{V}_{j_1} = 2m - 1$ and $\dim_{\mathbb{F}_q} \mathscr{V}_{j_1} + \mathscr{V}_{j_2} = 4m - 2$ for all $j_1, j_2 \in [\![k + 1, n]\!]$, $j_1 \neq j_2$. The polynomials in V_j's, however, are now affine bilinear in $(\boldsymbol{W}, \boldsymbol{Y})$. On the other hand, since the mixed products $X_{j_1} X_{j_2}$'s disappear from $\mathcal{S}'_2$, Proposition 4.19 must be modified in the following way.

   Proposition 4.22. *Let $q = 2^s, s > 1$. The vector space $\mathscr{U}_{n-2,n-1} \stackrel{def}{=} (\mathscr{V}_{n-2} + \mathscr{V}_{n-1}) \cap \mathbb{F}_q[\boldsymbol{X}, \boldsymbol{Y}]_{\leq 1}$ contains at least $m + 1$ linearly independent affine linear polynomials in Y_j's. We denote by $U_{n-2,n-1}$ a (sub)set of $m + 1$ of these polynomials, and by $u_{n-2,n-1}$ one of the polynomials containing the monomial 1. Moreover, $\mathscr{U}_{n-2,n-1} + \mathscr{V}_n$ contains a subspace of affine linear polynomials of dimension at least $2m$.*

   Proof. The terms appearing in the polynomials in V_{n-2} are $1, Y_{n-1}$ and $Y_j W_j^2$ for $j \in [\![k + 1, n - 3]\!]$. The ones in V_{n-1} are $1, Y_{n-2}$ and $Y_j W_j^2 + Y_j$, $j \in [\![k+1, n-3]\!]$. This means that the polynomials in $\mathscr{U}_{n-2,n-1}$ can all be expressed as linear combination of the $3m - 3$ monomials $Y_j X_j^2$'s, $j \in [\![k + 1, n - 3]\!]$, of degree 3, the monomial 1 and the $3m-1$ lienar monomials Y_j's, $j \in [\![k+1, n-1]\!]$. The dimension of the vector space $\mathscr{U}_{n-2,n-1}$ is $4m - 2$, so that we get at least $m + 1$ linearly independent affine linear polynomials in $\mathscr{U}_{n-2,n-1}$. Since all the elements in $\mathscr{V}$ are homogeneous, $\mathscr{U}_{n-2,n-1} \not\subseteq \mathscr{V}_n$. Hence $\mathscr{V}_n + \mathscr{U}_{n-2,n-1}$ has at least $\dim_{\mathbb{F}_q} \mathscr{V}_n + 1 = 2m$ linearly independent linear polynomials in Y_j's. □

   Remark 4.8. Experimentally, $\dim_{\mathbb{F}_q} \mathscr{U}_{n-2,n-1} = m + 1$. Moreover, the subspace of homogeneous polynomials in $\mathscr{U}_{j_1,j_2}$ is contained in $\mathscr{V}_n$. In other words,

$$\dim_{\mathbb{F}_q}(\mathscr{V}_n + \mathscr{U}_{n-2,n-1}) = \dim_{\mathbb{F}_q} \mathscr{V}_n + 1 = 2m.$$

   Remark 4.9 explains why the other subspaces $\mathscr{U}_{j_1,j_2}$'s have dimension m instead.

3. (eliminating $2m$ variables Y_j's using linear polynomials) Let I be a set of cardinality m such that all the Y_j's variables are expressed in terms of $\{Y_i \mid i \in I\}$ and $I_1 \subset I$, $|I_1| = m - 1$, such that all the Y_j's variables are expressed in terms of $\{1\} \cup \{Y_i \mid i \in I_1\}$. After the elimination of all Y_j in $[\![k+1, n-1]\!] \setminus I_1$, the polynomials in the V_l's can be written as a linear combination of the $(3m - 3)(m - 1)$ bilinear monomials $W_i Y_j$, $i \in [\![k + 1, n - 3]\!]$, $j \in I_1$, the $(3m-3)+(3m-1)$ linear variables W_i, $i \in [\![k+1, n-3]\!]$ and Y_j, $j \in [\![k+1, n-1]\!]$ and 1. Note that the monomials W_j's arise from the elimination of linear variables because there exist *affine* linear equations in $\mathscr{V}_n + \mathscr{U}_{n-2,n-1}$. This would not happen by eliminating only the variables from the *homogeneous*

linear equations in $\mathscr{V}_n$. Heuristically, after linearization of the $(2m-1)(3m-1)$ polynomials in $\bigcup_{j=k+1}^{n-1} V_j$, we obtain the following set of polynomials:

$$\begin{cases} Y_j W_{j'} + L_{j,j'}(Y_\ell : \ell \in I_1, 1), & \text{for all } j \in [\![k+1, n-3]\!] \setminus I_1, j' \in [\![k+1, n-3]\!], \\ W_j + L_{W_j}(Y_\ell : \ell \in I_1, 1), & \text{for all } j \in [\![k+1, n-3]\!], \\ Y_j + L_{Y_j}(Y_\ell : \ell \in I_1, 1), & \text{for all } j \in [\![k+1, n-3]\!] \setminus I_1. \end{cases}$$

4. (computing the Gröbner basis) This step is analogous to Steps 6. By eliminating the W_j's from the polynomials with leading monomials $Y_j W_{j'}$, we get the final Gröbner basis of the system, that is

$$\begin{cases} Y_j Y_{j'} + L'_{j,j'}(Y_\ell : \ell \in I_1, 1), & j, j' \in [\![k+1, n-3]\!] \backslash, j < j', \\ W_j + L_{W_j}(Y_\ell : \ell \in I_1, 1), & j' \in [\![k+1, n-3]\!], \\ Y_j + L_{Y_j}(Y_\ell : \ell \in I_1), & j \in [\![k+1, n-1]\!] \setminus I_1. \end{cases}$$

   Basically, the shape coincides with the grevlex basis obtained for the q odd case, with the only exception that the X_i are replaced by $W_i = X_i^2$'s.

5. (Computing the solutions) Similarly to the q odd case, a lex basis is then obtained through the FGLM algorithm and has the following shape:

$$\begin{cases} W_j + Q_{W_j}(Y_{n-1}) = 0, & j \in [\![k+1, n-3]\!] \\ Y_j + Q_{Y_j}(Y_{n-1}) = 0, & j \in [\![k+1, n-2]\!] \\ P(Y_{n-1}) = 0 \end{cases}$$

6. (Recovering the support and multiplier) Since every element is a square in a finite field of characteristic 2, we get

$$W_i + a_i = X_i^2 + a_i = (X_i + \sqrt{a_i})^2,$$

   i.e. we can uniquely determine the value $X_i = \sqrt{a_i}$ from the equation $W_i + a_i = 0$. This means that we can recover the last coordinates of a valid support from the lex basis. Thus we can retrieve the whole support and multiplier by linear algebra, as done in the q odd case.

Remark 4.9. At Step (3), in the odd characteristic case, we have derived low-degree polynomials by requiring that the unknowns in the block $\boldsymbol{X}$ must differ. Not only this step is not necessary when $q = 2^s$, but its adaptation does not provide *new* degree falls. Due to the change of variables $W_i = X_i^2$'s introduced, finding subspaces of multiples of $X_j - X_{j'}$ translates into searching for multiples of $W_j + W_{j'}$. Retracing the proof of Proposition 4.19, we get the $4m-2$ polynomial generating $\mathscr{V}_{j_1} + \mathscr{V}_{j_2}$ cam be written in terms of:

$$\begin{cases} Y_j(W_j + W_{j_1}) & j \in [\![k+1, n-1]\!] \setminus \{j_1, j_2\}, \\ 1 & \\ Y_{j_1}(W_{j_1} + W_{j_2}) & \\ Y_{j_2}(W_{j_1} + W_{j_2}) & \\ Y_j(W_{j_1} + W_{j_2})^2, & j \in [\![k+1, n-1]\!] \setminus \{j_1, j_2\}. \end{cases}$$

By eliminating the first $3m-2$ terms that are not multiple of $W_{j_1}+W_{j_2}$, we find a subspace of dimension m of linearly independent polynomials that are multiple of $W_{j_1}+W_{j_2}$. After division by the latter term, we obtain a space of dimension m of homogeneous linear polynomials in Y_j's. However, this subspace is experimentally contained in $\mathscr{V}_n$, thus it does not add any useful information for the algorithm.

Overview for $q=2$.

When $q=2$, Assumption 4.2 asserts that the rank of $\mathcal{S}'_2$ (or equivalently $\mathcal{S}'$) is smaller than for all the other cases, namely $\mathbf{Rank}(\mathcal{S}')=\mathbf{Rank}(\mathcal{S}'_2)=\binom{3m}{2}-3m$. This invalidates all the combinatorial arguments for the dimensions of V_j's and for the number of degree falls. In this case, we have

Proposition 4.23. *Let $q=2$. Under Assumptions (4.1) and (4.2),*

$$\dim_{\mathbb{F}_q}(\mathscr{V}_j)\geq m-1 \qquad\qquad \forall k+1\leq j\leq n \tag{4.35}$$

Proof. This proof requires a result that will be proved in Remark 5.4, Chapter 5, namely that

$$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_j\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}\leq\binom{3m}{2}-m.$$

Taking $\mathscr{C}\stackrel{\text{def}}{=}\mathbf{Pct}_j\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})\right)$, so that $\mathscr{C}^{\perp}\stackrel{\text{def}}{=}\mathbf{Sh}_j\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)$, it follows from Proposition 3.2 that the subspace of $\mathscr{S}$ whose elements can be written without the terms corresponding to $Z_{j,j'}$, for $j'\in[\![k+1,n]\!]\setminus\{j\}$, has rank

$$\begin{aligned}&\binom{3m-1}{2}-\binom{3m}{2}+\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_j\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}\\ \leq&\binom{3m-1}{2}-\binom{3m}{2}+\binom{3m}{2}-m=\binom{3m-1}{2}-m.\end{aligned}$$

Equivalently,

$$\dim_{\mathbb{F}_q}(\mathscr{V}_j)\geq\mathbf{Rank}(\mathcal{S})-\left(\binom{3m-1}{2}-m\right)=\left(\binom{3m}{2}-3m\right)-\left(\binom{3m-1}{2}-m\right)=m-1.$$

□

Remark 4.10. Empirically, the lower bound from Proposition 4.23 is tight and $\dim_{\mathbb{F}_q}(\mathscr{V}_j)=m-1$ with high probability. We will assume that the equality holds from now on. In particular, by taking $j=n$, we get that the number of independent linear equations in Y_j's coming from row-reduction of $\mathcal{S}$ is $m-1$.

While computing a Gröbner basis of $\mathcal{S}'_2$ plus the equations obtained from all the V_j's still seems viable, we propose an alternative approach, which refines the ideal (not the corresponding variety though) and is also more efficient.

More precisely, if we do not perform the change of variables $W_j=X_j^2$, we can symbolically obtain new low-degree equations from V_j, by taking into account the identity:

$$(X_i^2Y_i)(Y_i)^2=(Y_i)(X_iY_i)^2.$$

After expansion, this gives the system of bidegree $(2,3)$

$$\mathcal{S}'' \stackrel{\text{def}}{=} \left\{ \sum_{(j,j')\in J} p_{i,j}p_{i,j'}Y_jY_{j'}(Y_j+Y_{j'})(X_j+X_{j'})^2 = 0 \mid i \in [\![1,k]\!] \right\},$$

where, for clarity, we do not take into account the usual specialization. In a similar way to what done for $\mathcal{S}'$, we can define

$$\mathscr{T}_j = \frac{1}{Y_j}\left(\mathscr{S}'' \cap Y_j \cdot \mathbb{F}_q[\boldsymbol{X}', \boldsymbol{Y}']_{\leq 3}\right), \qquad j \in [\![k+1, n-1]\!]. \tag{4.36}$$

Comparing the equations in $\mathcal{S}'$ and $\mathcal{S}''$, it is clear that

$$\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell)^2 \in \mathscr{V}_\ell \iff \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(Y_j+Y_\ell)(X_j+X_\ell)^2 \in \mathscr{T}_\ell,$$

hence

$$\dim_{\mathbb{F}_q} \mathscr{T}_\ell = \dim_{\mathbb{F}_q} \mathscr{V}_\ell = m-1.$$

We can split an equation in $\mathscr{T}_\ell$ in the following way:

$$\begin{aligned}
&\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(Y_j+Y_\ell)(X_j+X_\ell)^2 \\
&= \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j^2(X_j+X_\ell)^2 + Y_\ell \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell)^2.
\end{aligned}$$

Since $Y_\ell \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell)^2$ is in the ideal generated by $\mathcal{S}'$, we obtain the equation

$$\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j^2(X_j+X_\ell)^2 = 0.$$

Since the coefficients $v_j \in \mathbb{F}_2$, by applying the Frobenius map $m-1$ times, we derive

$$\begin{aligned}
&\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j^2(X_j+X_\ell)^2 = 0 \\
\iff &\left(\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j^2(X_j+X_\ell)^2 = 0\right)^{2^{m-1}} = 0 \\
\iff &\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell) = 0,
\end{aligned}$$

i.e. we produce $(m-1)(3m-1)$ bilinear equations that are experimentally linearly independent. We define

$$\mathscr{U}_\ell \stackrel{\text{def}}{=} \{ \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell) \mid \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_jY_j(X_j+X_\ell)^2 \in \mathscr{V}_\ell \}.$$

This is trivially a vector space and $\dim_{\mathbb{F}_q} \mathscr{U}_\ell = \dim_{\mathbb{F}_q} \mathscr{V}_\ell = m-1$.

We are now going to describe some bilinear degree falls that are crucial in solving the system. Let us fix $\ell \in [\![k+1, n-3]\!]$ and consider the polynomials

$$\sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_j^{(\ell)}(X_j^2+X_\ell^2)Y_j \in \mathscr{V}_\ell \quad \text{and} \quad \sum_{j\in[\![k+1,n]\!]\setminus\{\ell\}} v_j^{(\ell)}(X_j+X_\ell)Y_j \in \mathscr{U}_\ell.$$

By equating to 0 a polynomial combination of the two, we obtain

$$\begin{aligned} 0 &= \left(\sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(\ell)}(X_j^2+X_\ell^2)Y_j + v_{n-2}^{(\ell)}X_\ell^2Y_{n-2} + v_{n-1}^{(\ell)}(X_\ell^2+1)Y_{n-1} + v_n^{(\ell)}\right) \\ &\quad + X_\ell\left(\sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(\ell)}(X_j+X_\ell)Y_j + v_{n-2}^{(\ell)}X_\ell Y_{n-2} + v_{n-1}^{(\ell)}(X_\ell+1)Y_{n-1}\right) \\ &= \sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(\ell)}X_j(X_j+X_\ell)Y_j + v_{n-1}^{(\ell)}(X_\ell+1)Y_{n-1} + v_n^{(\ell)}. \end{aligned}$$

Then, we consider the polynomials

$$\sum_{j\in[\![k+1,n]\!]\setminus\{n-2\}} v_j^{(n-2)}X_j^2Y_j \in \mathscr{V}_{n-2} \quad \text{and} \quad \sum_{j\in[\![k+1,n]\!]\setminus\{n-2\}} v_j^{(n-2)}X_jY_j \in \mathscr{U}_{n-2}.$$

By equating to 0 a polynomial combination of the two, we obtain

$$\begin{aligned} 0 &= \left(\sum_{j\in[\![k+1,n-3]\!]} v_j^{(n-2)}X_j^2Y_j + v_{n-1}^{(n-2)}Y_{n-1} + v_n^{(n-2)}\right) \\ &\quad + X_\ell\left(\sum_{j\in[\![k+1,n-3]\!]} v_j^{(n-2)}X_jY_j + v_{n-1}^{(n-2)}Y_{n-1}\right) \\ &= \sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(n-2)}X_j(X_j+X_\ell)Y_j + v_{n-1}^{(n-2)}(X_1+1)Y_{n-1} + v_n^{(n-2)}. \end{aligned}$$

Analogously, from the polynomials

$$\sum_{j\in[\![k+1,n]\!]\setminus\{n-1\}} v_j^{(n-1)}(X_j+1)^2Y_j \in \mathscr{V}_{n-1} \quad \text{and} \quad \sum_{j\in[\![k+1,n]\!]\setminus\{n-1\}} v_j^{(n-1)}(X_j+1)Y_j \in \mathscr{U}_{n-1},$$

we get

$$\begin{aligned} 0 &= \left(\sum_{j\in[\![k+1,n-3]\!]} v_j^{(n-1)}(X_j^2+1)Y_j + v_{n-2}^{(n-1)}Y_{n-2} + v_n^{(n-1)}\right) \\ &\quad + X_\ell\left(\sum_{j\in[\![k+1,n-3]\!]} v_j^{(n-1)}(X_j+1)Y_j + v_{n-2}^{(n-1)}Y_{n-2}\right) \\ &= \sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(n-1)}X_j(X_j+X_\ell)Y_j + \sum_{j\in[\![k+1,n-3]\!]\setminus\{\ell\}} v_j^{(n-1)}(1+X_\ell)Y_j + v_n^{(n-1)}. \end{aligned}$$

Hence, from $\mathscr{V}_\ell + \mathscr{V}_{n-2} + \mathscr{V}_{n-1} + X_\ell\mathscr{U}_\ell + X_\ell\mathscr{U}_{n-2} + X_\ell\mathscr{U}_{n-1}$, we get a set of $3(m-1)$ (experimentally linearly independent) cubic polynomial, whose cubic part can be written using only $3m-4$ terms: $X_j(X_j+X_\ell)Y_j,\ j \in [\![k+1, n-3]\!] \setminus \{\ell\}$. Therefore , by linearization, we get $(3m-3)-(3m-4)=1$ bilinear equation with the following shape:

$$\sum_{j\in[\![k+1,n-1]\!]\setminus\{\ell,n-2\}} d_j(X_\ell+1)Y_j + 1 = 0.$$

So, for any $\ell \in [\![k+1, n-3]\!]$, we predicted one bilinear degree fall occurring from the linearization of degree 3 polynomials. Therefore, we get a total of $3m-3$ bilinear equations that are clearly linearly independent, since each variable X_ℓ appears in only one of them. We call $\mathscr{D}$ the space generated by them. These degree falls are critical in the resolution. Indeed, it is now sufficient to compute a Gröbner basis in degree 3 of $\sum_j \mathscr{V}_j + \sum_j \mathscr{U}_j + \mathscr{D}$ to get a basis. From the very first step, $2(m-2)$ linear degree falls in $\boldsymbol{X}$ are found. After this, the recursive elimination of variables terminates the algorithm very quickly.

The Gröbner basis shape is the same as for the $q > 2$ case, hence finding the solution can be done analogously. In particular, the corresponding solutions have multiplicity 1. This differs from the Gröbner basis that would have been computed with the change of variables $W_i = X_i^2$'s and without exploiting the identities $(X_i^2Y_i)(Y_i)^2 = (Y_i)(X_iY_i)^2$'s. In that case, the multiplicity of each solution would have been 2, thus doubling the degree of the polynomials in the lex basis.

4.4.6 Limitations of the algebraic cryptanalysis approach: higher orders and Goppa codes

We have shown in depth how to retrieve an equivalent secret key in the regime where the filtration step permits to reduce the difficulty of the problem. On the other side, the algebraic cryptanalysis detailed in this section illustrates why a filtration down to degree $r = 3$ is needed for the algorithm to work at several steps. Indeed, our analysis relies on estimating the dimension of subspaces and counting the expected number of degree falls. However, these are consequences of the number of available independent equations at a certain step, which in its turn depends on the square code dimension. This can be observed from the very beginning of the algorithm. Indeed, consider a random alternant code. The dimension of the vector space $\mathscr{S}$ over $\mathbb{F}_q$ can be smaller than $\binom{3m}{2} - m$, leading to an equally smaller subspace $\mathscr{V}_l$. Even in the best scenario where $e_{\mathscr{A}}$ as defined in (3.11) is null, i.e. when $r < q+1$, a generalization of Proposition 4.18 leads to

$$\begin{aligned}
\dim_{\mathbb{F}_q} \mathscr{V}_j &= \max\left(\left(\binom{rm}{2} - \frac{(r-1)(r-2)}{2}m\right) - \binom{rm-1}{2}, 0\right) \\
&= \max\left(rm - 1 - \frac{(r-1)(r-2)}{2}m, 0\right) \\
&= \begin{cases} 2m-1 & \text{if } r=3 \\ m-1 & \text{if } r=4 \\ 0 & \text{if } r \geq 5 \end{cases}.
\end{aligned}$$

In other terms, the subspace $\mathscr{V}_j$ degenerates for $r \geq 5$, and even for $r = 4$ all the arguments of our analysis are invalidated. This does not necessarily mean that solving these instances becomes of exponential complexity, but the solving degree becomes undoubtedly bigger and further steps and expedients will be required. This is why the generalization of the whole attack to subfields of size $q > 3$ is not straightforward.

Similarly, even if we assume that a filtration for Goppa codes exists, the Gröbner basis algorithm requires modifications as well, unless $r < q - 1$, which is the critical point above which Goppa codes behave differently from random alternant codes in terms of the square of the dual code. Since we are not aware of any filtration, it is not even self-evident what the starting code of the algebraic attack could be. One could argue that, if a filtration exists, this will reasonably provide an alternant code that is not a Goppa code anymore and the additional problem of a different algebraic cryptanalysis does not arise. For instance if $\mathscr{G}(\boldsymbol{x}, \Gamma) = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, then $\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x}_{\check{\mathcal{I}}}, \boldsymbol{y}_{\check{\mathcal{I}}} \prod_{\mathcal{I}}(\boldsymbol{x}_{\check{\mathcal{I}}} - x_i))$ is not a Goppa code. Indeed the coordinates of the multiplier

$$\boldsymbol{y}_{\check{\mathcal{I}}} \prod_{\mathcal{I}} (\boldsymbol{x}_{\check{\mathcal{I}}} - x_i) = \frac{\prod_{\mathcal{I}}(\boldsymbol{x}_{\check{\mathcal{I}}} - x_i)}{\Gamma(\boldsymbol{x}_{\check{\mathcal{I}}})}$$

are the evaluation of a rational function in the coordinates of the support $\boldsymbol{x}_{\check{\mathcal{I}}}$, with a numerator of degree $|\mathcal{I}|$ and a denominator of degree r.

Interestingly enough, we can unveil an analogy with the transformation characterizing Cauchy codes and their subfield subcodes when applied to Goppa codes. To that end, let us consider the Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma) = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{F}_q^n$ and define

$$\boldsymbol{x}' = \frac{a\boldsymbol{x} + b}{c\boldsymbol{x} + d} \quad \text{and} \quad \boldsymbol{y}' = \lambda(c\boldsymbol{x} + d)^{r-1}\boldsymbol{y}, \tag{4.37}$$

for some $a, b, c, d \in \mathbb{F}_{q^m}$, $ad - bc \neq 0$, and $\lambda \in \mathbb{F}_{q^m}^*$. For simplicity, we also assume $cx_i + d \neq 0$ for all $i \in [\![1, n]\!]$. Then $\mathscr{A}_r(\boldsymbol{x}', \boldsymbol{y}') = \mathscr{G}(\boldsymbol{x}, \Gamma)$ is clearly a Goppa code, however the support and multiplier representatives $\boldsymbol{x}'$ and $\boldsymbol{y}'$ do not reflect the Goppa polynomial relation, i.e. $\boldsymbol{y}'$ is not guaranteed to be the evaluation in $\boldsymbol{x}'$ of the inverse of a polynomial Γ' of degree r. While the Goppa polynomial relation is preserved for all linear transformations $\boldsymbol{x}' = a\boldsymbol{x} + b$, $\boldsymbol{y}' = \lambda\boldsymbol{y}$, we are now going to show that it is not, in general, an invariant. Indeed, by definition of $\boldsymbol{x}'$, we can invert the relation with $\boldsymbol{x}$ and obtain

$$\boldsymbol{x} = \frac{a'\boldsymbol{x}' + b'}{c'\boldsymbol{x}' + d'},$$

where

$$\begin{bmatrix} a' & b' \\ c' & d' \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \begin{bmatrix} d & -b \\ -c & a \end{bmatrix},$$

thus leading to

$$\boldsymbol{x} = \frac{d\boldsymbol{x}' - b}{-c\boldsymbol{x}' + a}.$$

Note that

$$c\boldsymbol{x} + d = \frac{cd\boldsymbol{x}' - bc}{-c\boldsymbol{x}' + a} + d = \frac{ad - bc}{-c\boldsymbol{x}' + a}.$$

Hence, the coordinates of the new multiplier $\boldsymbol{y}'$ can be formulated as the evaluation of a rational function in the coordinates of the new support $\boldsymbol{x}'$ as

$$\begin{aligned}\boldsymbol{y}' =&\lambda(c\boldsymbol{x}+d)^{r-1}\boldsymbol{y} = \frac{\lambda(c\boldsymbol{x}+d)^{r-1}}{\Gamma\left(\frac{d\boldsymbol{x}'-b}{-c\boldsymbol{x}'+a}\right)} = \frac{\lambda(c\boldsymbol{x}+d)^{r-1}}{\sum_{i=0}^{r}\gamma_i\left(\frac{d\boldsymbol{x}'-b}{-c\boldsymbol{x}'+a}\right)^i} = \frac{\lambda(c\boldsymbol{x}+d)^{r-1}(-c\boldsymbol{x}'+a)^r}{\sum_{i=0}^{r}\gamma_i(d\boldsymbol{x}'-b)^i(-c\boldsymbol{x}'+a)^{r-i}}\\ =&\frac{\lambda(ad-bc)^{r-1}(-c\boldsymbol{x}'+a)}{\sum_{i=0}^{r}\gamma_i(d\boldsymbol{x}'-b)^i(-c\boldsymbol{x}'+a)^{r-i}}\end{aligned}$$

More specifically, the reduced form of such rational function has in general a numerator of degree 1 and a denominator of degree r, i.e.

$$\boldsymbol{y}' = \frac{A\boldsymbol{x}'+B}{\sum_{i=0}^{r}\gamma_i'(\boldsymbol{x}')^i},$$

with $A=-\lambda(ad-bc)^{r-1}c$ and $B=\lambda(ad-bc)^{r-1}a$. In particular, the condition for $\boldsymbol{x}'$ and $\boldsymbol{y}'$ being related by a degree-r Goppa polynomial is $A=0 \iff c=0$. In this way

$$\boldsymbol{x}' = \frac{a}{d}\boldsymbol{x}+\frac{b}{d} \quad \text{and} \quad \boldsymbol{y}' = \lambda d^{r-1}\boldsymbol{y},$$

and we get back the subset of linear transformations.

This has some consequences on some of the results provided before for alternant codes. First of all, the property about the parity-check subcode does not hold anymore, i.e.

$$\mathscr{A}_{r+1}(\boldsymbol{x}',\boldsymbol{y}') \neq \widetilde{\mathscr{A}_r(\boldsymbol{x}',\boldsymbol{y}')},$$

and $\mathscr{A}_{r+1}(\boldsymbol{x}',\boldsymbol{y}')$ is expected to have dimension $n-(r+1)m$ and not $n-rm-1$.

This same remark has an extra effect on the algebraic cryptanalysis of the system originated by a Goppa code. Assume we are given a Goppa code of low degree r, so that we can hope to model the key-recovery problem and solve it directly, without first computing a filtration. Since we have easily access to $\widetilde{\mathscr{G}(\boldsymbol{x},\Gamma)} = \mathscr{A}_{r+1}(\boldsymbol{x},\boldsymbol{y})$, we might be tempted to exploit identities involving the monomial $Y_iX_i^r$, such as

$$(Y_iX_i^r)(Y_i) = (Y_iX_i^{r-1})(Y_iX_i).$$

While the secret vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are a solution of the equation above, the same cannot be said about $\boldsymbol{x}'$ and $\boldsymbol{y}'$ as in (4.37), but only for those corresponding to linear transformation, i.e. when $c=0$. Making use of identities involving $Y_iX_i^r$ thus implies being allowed to specialize only 2 (instead of 3) variables in the $\boldsymbol{X}$ block (and 1 in the $\boldsymbol{Y}$ block).

4.5 Interlacing the algebraic recovering with the filtration

We now get back to distinguishing between the full-length vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and their shortening due to the filtration attack. We can restore the information lost from the filtration shortening, by simply repeating the attack twice on disjoint sets $\mathcal{I}_1, \mathcal{I}_2$.

This is possible because $|\mathcal{I}_1 \cup \mathcal{I}_2| = 2(r-3)$ is very small compared to n. Indeed, if we shorten the positions corresponding to $\mathcal{I}_1 \stackrel{\text{def}}{=} [\![1, r-3]\!]$ (the order is irrelevant) during the filtration attack, at the end of the algebraic recovering we have access to m pairs of vectors in $\mathbb{F}_{q^m}^{n-(r-3)}$:

$$\bar{\boldsymbol{x}}_{\mathcal{I}_1}\text{'s} \quad \text{and} \quad \bar{\boldsymbol{y}}_{\mathcal{I}_1}\left(\prod_{i\in\mathcal{I}_1}(\bar{\boldsymbol{x}}_{\mathcal{I}_1} - x_i)\right)\text{'s}.$$

Analogously, if we shorten the positions corresponding to $\mathcal{I}_2 \stackrel{\text{def}}{=} [\![(r-3)+1, 2(r-3)]\!]$ during the filtration attack, at the end of the algebraic recovering we have access to m pairs of vectors in $\mathbb{F}_{q^m}^{n-(r-3)}$:

$$\bar{\boldsymbol{x}}_{\mathcal{I}_2}\text{'s} \quad \text{and} \quad \bar{\boldsymbol{y}}_{\mathcal{I}_2}\left(\prod_{i\in\mathcal{I}_2}(\bar{\boldsymbol{x}}_{\mathcal{I}_2} - x_i)\right)\text{'s}.$$

In particular, if the same specialization has been chosen, we can couple m pairs $(\bar{\boldsymbol{x}}_{\mathcal{I}_1}, \bar{\boldsymbol{x}}_{\mathcal{I}_2})$'s such that the two vectors of each pair coincide on the last $n-2(r-3)$ coordinates. We can easily detect them from the last $3m$ coordinates, so we do not need to solve $2m$ linear systems but it is sufficient to choose one pair and solve only the 2 corresponding linear systems. In this way, we obtain a full solution $\bar{\boldsymbol{x}}$ for the original problem as

$$\bar{\boldsymbol{x}} = (\underbrace{\bar{x}_1, \ldots, \bar{x}_{r-3}}_{\text{first } r-3 \text{ coordinates of } \bar{\boldsymbol{x}}_{\mathcal{I}_1}}, \underbrace{\bar{x}_{r-2}, \ldots, \bar{x}_{2(r-3)}}_{\text{first } r-3 \text{ coordinates of } \bar{\boldsymbol{x}}_{\mathcal{I}_2}}, \underbrace{\bar{x}_{2(r-3)+1}, \ldots, \bar{x}_{n-3}, 0, 1, \infty}_{\text{last common coordinates of } \bar{\boldsymbol{x}}_{\mathcal{I}_1} \text{ and } \bar{\boldsymbol{x}}_{\mathcal{I}_2}}).$$

By replacing the found values in the corresponding $\bar{\boldsymbol{y}}_{\mathcal{I}_1}\left(\prod_{i\in\mathcal{I}_1}(\bar{\boldsymbol{x}}_{\mathcal{I}_1} - x_i)\right)$ and $\bar{\boldsymbol{y}}_{\mathcal{I}_2}\left(\prod_{i\in\mathcal{I}_2}(\bar{\boldsymbol{x}}_{\mathcal{I}_2} - x_i)\right)$, we retrieve $\bar{\boldsymbol{y}}_{\mathcal{I}_1}$ and $\bar{\boldsymbol{y}}_{\mathcal{I}_2}$. Similarly to what was done for the support, we can put together the information of these two vectors and get

$$\bar{\boldsymbol{y}} = (\underbrace{\bar{y}_1, \ldots, \bar{y}_{r-3}}_{\text{first } r-3 \text{ coordinates of } \bar{\boldsymbol{y}}_{\mathcal{I}_1}}, \underbrace{\bar{y}_{r-2}, \ldots, \bar{y}_{2(r-3)}}_{\text{first } r-3 \text{ coordinates of } \bar{\boldsymbol{y}}_{\mathcal{I}_2}}, \underbrace{\bar{y}_{2(r-3)+1}, \ldots, \bar{y}_{n-1}, 1}_{\text{last common coordinates of } \bar{\boldsymbol{y}}_{\mathcal{I}_1} \text{ and } \bar{\boldsymbol{y}}_{\mathcal{I}_2}}).$$

So, a pair of valid support and multiplier has been recovered. However, $\bar{\boldsymbol{x}} \notin \mathbb{F}_{q^m}^n$, because $\bar{x}_n = \infty$. The last question is therefore how to get a valid pair of support and multiplier such that both are defined over $\mathbb{F}_{q^m}$, i.e. how to get the alternant code representation. In other words, we need to determine some $f \in GL_2(\mathbb{F}_{q^m})$ and $\lambda \in \mathbb{F}_{q^m}^*$ such that

$$\bar{x}'_i = f(\bar{x}_i) \in \mathbb{F}_{q^m}, \quad \forall i \in [\![1, n]\!]$$

and

$$\bar{y}'_i = \lambda\theta(f, \bar{x}_i)^{r-1}\bar{y}_i \in \mathbb{F}_{q^m}, \quad \forall i \in [\![1, n]\!],$$

with θ defined as in Theorem 4.3. We observe that, since there are only $n-1$ coordinates of $\bar{\boldsymbol{x}}$ in $\mathbb{F}_{q^m}$ and $n-1 < q^m$, there exists at least one element $\hat{x} \in \mathbb{F}_{q^m}$ that is different from all $\bar{\boldsymbol{x}}$ coordinates. We also remark that $\hat{x} \neq 0$, since $\bar{x}_{n-2} = 0$, so the map f on $\bar{\mathbb{F}}_{q^m}$

$$f \stackrel{\text{def}}{=} \frac{z}{z - \hat{x}}$$

is induced by an element of the linear group. We have $\theta(f, z) = z - \hat{x}$ if $z \in \mathbb{F}_{q^m}$ and $\theta(f, \infty) = 1$ and we choose $\lambda = 1$. Therefore

$$\begin{aligned}
\bar{x}'_i &= \tfrac{\bar{x}_i}{\bar{x}_i - \hat{x}}, && i \in [\![1, n-1]\!], \\
\bar{x}'_n &= 1, \\
\bar{y}'_i &= (\bar{x}_i - \hat{x})^{r-1} \bar{y}_i, && i \in [\![1, n-1]\!], \\
\bar{y}'_n &= \bar{y}_n = 1.
\end{aligned}$$

We finally obtained a support and a multiplier with coordinates over $\mathbb{F}_{q^m}$ that define the public code. This concludes the key-recovery attack on high-rate random alternant codes.

Remark 4.11. We have seen that both the filtration and the algebraic cryptanalysis can be performed in polynomial time. In order to interlace them, we need to repeat the full attack twice. Thus, this does not change the order of the complexity. Finally, we move from the private key corresponding to the subfield subcode of a Cauchy code to the one of an alternant code. This task has a negligible cost with respect to the rest of the algorithm, requiring only $\mathcal{O}(n)$ operations. Therefore the total cost of the key recovery is polynomial too.

4.6 Conclusions

We have presented a polynomial time key-recovery attack on unstructured alternant codes of high rate. This is the first time that this family has been cryptanalyzed. We have shown how the key-recovery problem can be reduced to that of an alternant code of smaller code. This was done by iteratively computing conductor codes that provided a filtration of alternant codes of decreasing order, which stops at an order equal to the field size. This method made extensive use of the results got in the previous chapter about the structure of the product and square of dual of alternant codes. Once obtained the filtration, we tackled the easier version of the key-recovery problem, namely with respect to alternant code order 3, through algebraic cryptanalysis. More precisely, we have studied a polynomial system modeling it. Such a system was already known and its unusually small rank explains why alternant and Goppa codes are distinguishable in the high rate setting. However, we have been able to solve it efficiently by designing an original method based on Gröbner bases. The strategy and its explanation partially differ depending on the field size. We provided a detailed description of the algorithm for odd characteristic fields and briefly cover the tweaks needed when the field size is 2 or a higher power of 2. Finally, we explained how the two parts of the attack can be merged as long as the field size is small enough, namely 2 or 3. Overall, the attack succeeds with high probability against binary and ternary random distinguishable alternant codes.

One possible research line consists in extending the algebraic attack to orders larger than 3. This would allow to break instances with larger field sizes, because the filtration could stop at a previous step in that case. The most interesting open problem, however, is to adapt such an attack to the Goppa case, for which our strategy fails in computing the filtration. This would completely break the CFS digital signature, which indeed relies on high-rate binary Goppa codes and essentially resisted more than 20 years of cryptanalysis.

Chapter 5
Enhancing the distinguisher by shortening the dual code

In this chapter, which is meant more to raise questions rather than provide answers, we tackle the problem of improving the distinguisher presented in Chapter 3. More precisely, the aim is to decrease the minimum code rate for which an alternant code is distinguishable. Following the construction presented in Chapter 3, our method also exploits the notion of square code. However, before the computation of the dimension of the latter, the dual of the alternant code is shortened in a set of coordinates. The complexity of the distinguishing algorithm remains polynomial. For some parameters, this procedure allows to enhance the distinguisher. As a demonstration of the effectiveness of our method, we experimentally exhibit instances that would not be distinguishable according to the approach presented in Chapter 3, but that become such using the shortening tweak. We also illustrate how to produce other parameters with this property and give a partial explanation of the behavior observed empirically, based on a direct sum decomposition of the shortened dual code.

Contents

5.1 Introduction

The cryptanalysis presented in Chapter 4 turns the distinguisher for high-rate alternant codes into a polynomial time attack. We already discussed the challenges behind its adaptation to Goppa codes. However, another question arises from this result. Indeed, we have revealed a weakness for non-structured subfield subcodes of GRS codes, but within a limited range of parameters. Although the techniques adopted, namely the computation of a filtration of alternant codes and the use of Gröbner bases to solve multivariate algebraic systems, harshly challenge and put in doubt the security of McEliece-like schemes based on high-rate codes, such as CFS signature [CFS01], it barely affects the confidence in schemes like Classic McEliece [Alb+20], where the code rate $R \in [0.7, 0.8]$ for all security levels. Indeed, the attack in Chapter 4 is strictly connected to the distinguisher from [Fau+13] and Chapter 3, and it is subject to the same limitations. It is therefore natural to wonder whether the distinguishable range can be extended.

Our question then downsizes to the preliminary issue of decreasing the distinguishable threshold for the code rate. In this chapter, we propose an elementary strategy to tackle this problem. It makes use of a standard construction in coding theory, namely the shortening of a code, together with the computation of square codes, the latter being already used in Chapter 3 and 4. We will show that this technique is effective, providing parameters that can be distinguished using this strategy and that were not known to be distinguishable before. This is the first improvement of the approximately 10 years old distinguisher from [Fau+13].

In Chapter 4, we have also recalled the extension field formalism for linear codes and we proved some results concerning how $\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)$ (or equivalently $\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}$) decomposes in the direct sum of two codes, one being the dual of an alternant code of order $r-1$. With these tools at hand, we can now study the effect of shortening the dual of an alternant code, with respect to its square code. We will experimentally illustrate that this technique allows in some cases to extend the regime for which an alternant code is distinguishable and we will give some insight into the underlying algebraic structure.

More precisely, let us fix some parameters q, r, m and $n \leq q^m$. From now on, in order to avoid confusion, we say that an $[n, n-rm]$ alternant code $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y}) \subseteq \mathbb{F}_{q^m}$ is *classically distinguishable* if and only if

$$\dim_{\mathbb{F}_q}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)^{\star 2} < \min\left(n, \binom{rm+1}{2}\right).$$

In other words, an alternant code is classically distinguishable if it is with respect to the distinguisher proposed in [Fau+13] and largely analyzed in Chapter 4.

Indeed, for an $[n,k]$ random code $\mathscr{R}$ over $\mathbb{F}$, we expect, with overwhelming probability,

$$\dim_{\mathbb{F}} \mathscr{R}^{\star 2} = \min\left(n, \binom{k+1}{2}\right).$$

As a consequence, given a set of positions $\mathcal{I} \subseteq [\![1, n]\!]$,

$$\dim_{\mathbb{F}}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{R}\right)\right)^{\star 2} \geq \min\left(n - |\mathcal{I}|, \binom{k-|\mathcal{I}|+1}{2}\right)$$

with overwhelming probability. Moreover, equality holds with high probability if $\mathcal{I}$ is a subset of an information set for $\mathscr{R}$.

In the following, we are going to illustrate that there exist $[n, n-rm]$ alternant codes $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{F}_q$ for which

$$\dim_{\mathbb{F}_q} \left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)^{\star 2} = \min\left(n, \binom{rm+1}{2}\right),$$

i.e. that are not classically distinguishable, but such that

$$\dim_{\mathbb{F}_q} \left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)^{\star 2} < \min\left(n - |\mathcal{I}|, \binom{rm - |\mathcal{I}| + 1}{2}\right)$$

for appropriate choices of $\mathcal{I}$, or more simply of $|\mathcal{I}|$. In other words, the alternant code becomes distinguishable, through the square code construction, after shortening the dual code at a proper amount of coordinates. Note that, thanks to Proposition 1.2, one can equivalently puncture the primal code and then consider the square of the dual code.

We also remark that this simple distinguishing algorithm only requires shortening a code and computing a square code. All these operations are well known to have polynomial complexity.

The effectiveness of such an approach strongly depends on the parameters, being more powerful for a small field extension m. As usual, Goppa codes may require an ad hoc analysis, which is not given in this chapter. For this subclass of codes, we will only show here some experimental results and notice that, since the distinguishable region is larger, the improvement given by shortening the dual code has a more limited impact.

5.2 Experimental results

We compute the dimension of the square code of the shortening of the dual code with respect to sets of positions $\mathcal{I}$ of different cardinalities and for several choices of the 4-tuple (q, r, m, n). We recall that the latter uniquely determines the value $e_{\mathscr{A}}$ and $e_{\mathscr{G}}$, whose definition, first given in Equations (3.11) and (3.12), is recalled here:

$$e_{\mathscr{A}} \stackrel{\text{def}}{=} \max\{i \in \mathbb{N} \mid r \geq q^i + 1\} = \left\lfloor \log_q(r-1) \right\rfloor$$
$$e_{\mathscr{G}} \stackrel{\text{def}}{=} \min\{i \in \mathbb{N} \mid r \leq (q-1)^2 q^i\} + 1 = \left\lceil \log_q\left(\frac{r}{(q-1)^2}\right) \right\rceil + 1.$$

In the tables that follow we will also print the numbers $n - |\mathcal{I}|$ and $\binom{rm-|\mathcal{I}|+1}{2}$. The smallest between the two is written in bold, this represents the expected dimension for a random code with the same parameters as the alternant code. We also highlight in red the values of the square code dimension that are distinguishable, i.e. that are strictly smaller than the bold value in the same column.

Let us start with a small extension field, $m = 2$. We choose $q = 17$ and different values of r. In Table 5.1, we choose $(q, r, m, n) = (17, 15, 2, 289)$. Since $r < q - 1$, alternant and Goppa codes have the same behavior.

The first column of Table 5.1 (corresponding to $|\mathcal{I}| = 0$) shows that the code is classically distinguishable. Therefore shortening the code is not useful for these

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$n-\lvert\mathcal{I}\rvert$	**289**	**288**	**287**	**286**	**285**	**284**	**283**	282	281	280	279	278	277	276	275	274
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	465	435	406	378	351	325	300	**276**	**253**	**231**	**210**	**190**	**171**	**153**	**136**	**120**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	283	279	274	268	261	253	244	234	223	211	198	184	169	153	136	120
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	283	279	274	268	261	253	244	234	223	211	198	184	169	153	136	120

Table 5.1: Square code dimensions. Parameters: $(q,r,m,n)=(17,15,2,289)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$n-\lvert\mathcal{I}\rvert$	**278**	**277**	**276**	**275**	**274**	**273**	**272**	**271**	270	269	268	267	266	265	264	263
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	465	435	406	378	351	325	300	276	**253**	**231**	**210**	**190**	**171**	**153**	**136**	**120**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	278	277	274	268	261	253	244	234	223	211	198	184	169	153	136	120
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	278	277	274	268	261	253	244	234	223	211	198	184	169	153	136	120

Table 5.2: Square code dimensions. Parameters: $(q,r,m,n)=(17,15,2,278)$

parameters. However, it is already possible to discern the key point of this strategy: shortening in $|\mathcal{I}|$ positions the dual code decreases the square code dimension by more than $|\mathcal{I}|$. Therefore, we can easily see the impact of shortening in play, by keeping the same 3-tuple (q,r,m) but choosing $n<\dim_{\mathbb{F}_q}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)^{\star 2}$ instead of the full-length code. This is illustrated in Table 5.2 for $n=278$. We also remark that for $|\mathcal{I}|\geq 13$, the square code behaves like a random code and thus the code loses its distinguishable property.

Several observations are worth to be made regarding Table 5.2. First of all, we notice that, since the code is shorter, the minimal value of $|\mathcal{I}|$ for which $\binom{rm-|\mathcal{I}|+1}{2}<|\mathcal{I}|$ has increased from 7 to 8. More importantly, the alternant/Goppa code is not classically distinguishable, as the square of the dual code coincides with the whole ambient space $\mathbb{F}_q^n$. The same occurs after shortening in one position: the square code is simply $\mathbb{F}_q^{n-1}$. However, due to the phenomenon observed before, starting from $|\mathcal{I}|=2$, the square code dimension decreases below $\min\left(n-|\mathcal{I}|,\binom{rm-|\mathcal{I}|+1}{2}\right)$, thus making the alternant/Goppa code distinguishable.

In Table 5.3 we increase r in such a way that still $e_{\mathscr{A}}=0$, but Goppa codes feature new quadratic relationships, i.e. $e_{\mathscr{G}}$ is defined (and strictly positive). More precisely, we consider the tuple of parameters $(q,r,m,n)=(17,17,2,289)$. The different behavior is evident from the table. Both random alternant and Goppa codes with these parameters are not distinguishable at the beginning, and they become such if shortened in the right amount of positions. However, the "red window" is wider for Goppa codes (it starts before and ends after) and, within this interval, the square code dimension is strictly smaller than for random alternant codes.

In Tables 5.4 and 5.5, instances corresponding to the parameters $(q,r,m,n)=(17,18,2,289)$ and $(q,r,m,n)=(17,19,2,289)$ respectively are shown. Here $r\geq q+1$, hence they are the first example where $e_{\mathscr{A}}>0$ (in particular $e_{\mathscr{A}}=1$).

Now, we provide some examples with a larger extension field degree m. This allows to pick a smaller field size q and therefore increase the integer $e_{\mathscr{A}}$. In order to better realize how fast the square code dimension drops with respect to $|\mathcal{I}|$, we choose classically distinguishable parameters. Nevertheless, we recall that starting

$\lvert\mathcal{I}\rvert$	0	1	...	7	8	9	10	11	12	13	14	15	16	17
$n-\lvert\mathcal{I}\rvert$	**289**	**288**	...	**282**	**281**	**280**	**279**	278	277	276	275	274	273	272
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	595	561	...	378	351	325	300	**276**	**253**	**231**	**210**	**190**	**171**	**153**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	289	288	...	282	281	280	270	256	241	225	208	190	171	153
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	289	288	...	282	279	269	258	246	233	219	204	188	171	153

Table 5.3: Square code dimensions. Parameters: $(q,r,m,n)=(17,17,2,289)$

$\lvert\mathcal{I}\rvert$	0	1	...	9	10	11	12	13	14	15	16	17	18
$n-\lvert\mathcal{I}\rvert$	**289**	**288**	...	**280**	**279**	**278**	**277**	**276**	275	274	273	272	271
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	666	630	...	378	351	325	300	**276**	**253**	**231**	**210**	**190**	**171**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	289	288	...	280	279	278	270	256	241	225	208	190	171
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	289	288	...	280	279	269	258	246	233	219	204	188	171

Table 5.4: Square code dimensions. Parameters: $(q,r,m,n)=(17,18,2,289)$

$\lvert\mathcal{I}\rvert$	0	1	...	11	12	13	14	15	16	17	18	19	20
$n-\lvert\mathcal{I}\rvert$	**289**	**288**	...	**278**	**277**	**276**	**275**	**274**	273	272	271	270	269
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	741	703	...	378	351	325	300	276	**253**	**231**	**210**	**190**	**171**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	289	288	...	278	277	276	270	256	241	225	208	190	171
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	289	288	...	278	277	269	258	246	233	219	204	188	171

Table 5.5: Square code dimensions. Parameters: $(q,r,m,n)=(17,19,2,289)$

from them, it is then possible to derive non-classically distinguishable parameters by decreasing n, as exemplified in Table 5.2.

Tables 5.6, 5.7, 5.8, 5.9, 5.10, 5.11 illustrate the dimensions corresponding to full-length alternant/Goppa codes with pair of extension field degree and field size $(q,m)=(3,7)$ and with order $r\in[\![5,10]\!]$.

We remark that, in these examples, $e_{\mathscr{A}}>0$ ($e_{\mathscr{A}}=2$ for $r=10$ and $e_{\mathscr{A}}=1$ otherwise). In Table 5.1 we observe a drop of $4=2m$ of the dimension, after shortening in a first position. In all the tables referring to the pair $(q,m)=(3,7)$, that quantity decreases by $7=m$ in the case of random alternant codes. We will see in the next section that this is indeed related to the number $e_{\mathscr{A}}$. Moreover, we experience here the first evidence of a more restricted impact of the shortening technique on Goppa codes. Indeed, as the square code dimension is already far below the classically distinguishable threshold, shortening the dual codes decreases the

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	2187	2186	2185	2184	2183	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**630**	**595**	**561**	**528**	**496**	**465**	**435**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	532	525	518	507	489	465	435
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	420	419	418	417	416	415	414

Table 5.6: Square code dimensions. Parameters: $(q,r,m,n)=(3,5,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	2187	2186	2185	2184	2183	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**903**	**861**	**820**	**780**	**741**	**703**	**666**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	728	721	714	707	699	682	659
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	420	419	418	417	416	415	414

Table 5.7: Square code dimensions. Parameters: $(q,r,m,n)=(3,6,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	2187	2186	2185	2184	2183	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**1225**	**1176**	**1128**	**1081**	**1035**	**990**	**946**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	952	945	938	931	924	917	904
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	686	685	684	683	682	681	680

Table 5.8: Square code dimensions. Parameters: $(q,r,m,n)=(3,7,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	2187	2186	2185	2184	2183	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**1596**	**1540**	**1485**	**1431**	**1378**	**1326**	**1275**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	1204	1197	1190	1183	1176	1169	1162
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	840	839	838	837	836	835	834

Table 5.9: Square code dimensions. Parameters: $(q,r,m,n)=(3,8,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	2187	2186	2185	2184	2183	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**2016**	**1953**	**1891**	**1830**	**1770**	**1711**	**1653**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	1484	1477	1470	1463	1456	1449	1442
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	1008	1007	1006	1005	1004	1003	1002

Table 5.10: Square code dimensions. Parameters: $(q,r,m,n)=(3,9,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n-\lvert\mathcal{I}\rvert$	**2187**	**2186**	**2185**	**2184**	**2183**	2182	2181
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	2485	2415	2346	2278	2211	**2145**	**2080**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	1729	1722	1715	1708	1701	1694	1687
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	1190	1189	1188	1187	1186	1185	1184

Table 5.11: Square code dimensions. Parameters: $(q,r,m,n)=(3,10,7,2187)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n - \lvert\mathcal{I}\rvert$	8192	8191	8190	8189	8188	8187	8186
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**4186**	**4095**	**4005**	**3916**	**3828**	**3741**	**3655**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	3367	3354	3341	3328	3315	3302	3289
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	2093	2092	2091	2090	2089	2088	2087

Table 5.12: Square code dimensions. Parameters: $(q, r, m, n) = (2, 7, 13, 8192)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n - \lvert\mathcal{I}\rvert$	8192	8191	8190	8189	8188	8187	8186
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**5460**	**5356**	**5253**	**5151**	**5050**	**4950**	**4851**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	4277	4264	4251	4238	4225	4212	4199
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	2600	2599	2598	2597	2596	2595	2594

Table 5.13: Square code dimensions. Parameters: $(q, r, m, n) = (2, 8, 13, 8192)$

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n - \lvert\mathcal{I}\rvert$	8192	8191	8190	8189	8188	8187	8186
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	**6903**	**6786**	**6670**	**6555**	**6441**	**6328**	**6216**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	5187	5174	5161	5148	5135	5122	5109
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	3042	3041	3040	3039	3038	3037	3036

Table 5.14: Square code dimensions. Parameters: $(q, r, m, n) = (2, 9, 13, 8192)$

square code dimension by $|\mathcal{I}|$ only.

We conclude this first section of examples with some binary instances within the classically distinguishable regime. In particular, we choose full-length codes from the pair $(q, m) = (2, 13)$ and of order $r \in [\![7, 10]\!]$. We remark that $e_{\mathscr{A}}$ increases again: it is equal to 2 for $r \in [\![7, 8]\!]$ and 3 for $r \in [\![9, 10]\!]$. The square code dimension fall is analogous to the ternary instances of the previous tables. More precisely, it decreases by m for each position shortened in the random alternant code case, and by 1 for each position shortened in the Goppa code case.

$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6
$n - \lvert\mathcal{I}\rvert$	**8192**	**8191**	**8190**	8189	8188	8187	8186
$\binom{rm-\lvert\mathcal{I}\rvert+1}{2}$	8515	8385	8256	**8128**	**8001**	**7875**	**7750**
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)^{\star 2}$	6175	6162	6149	6136	6123	6110	6097
$\dim_{\mathbb{F}_q}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{G}(\boldsymbol{x},\Gamma)^{\perp}\right)\right)^{\star 2}$	3510	3509	3508	3507	3506	3505	3504

Table 5.15: Square code dimensions. Parameters: $(q, r, m, n) = (2, 10, 13, 8192)$

5.3 A direct sum decomposition of the shortened dual code

Although this chapter has mainly an experimental nature, we provide in this section a partial justification about the dimension drop for the square code, caused by shortening the dual code. In particular, we will present the dimension analysis for some subcodes involved in the case where the code is shortened in one position only, i.e. $\mathcal{I} = \{i\}$, and for $e_{\mathscr{A}} \leq 2$.

We first show that $\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)$ (or equivalently $\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)\right)_{\mathbb{F}_{q^m}}$) decomposes in the direct sum of two codes, one being the dual of an alternant code of order $r - |\mathcal{I}|$. More specifically, we prove that

Theorem 5.1. *Let $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y}) \subseteq \mathbb{F}_q^n$ be an alternant code and let $\mathcal{I} \subseteq [\![1,n]\!]$ be a non-empty set of cardinality at most r. Then*

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}} \supseteq \mathscr{B}_{\mathbb{F}_{q^m}} + \mathscr{C}_0,$$

where

$$\mathscr{B} \stackrel{def}{=} \mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x}_{\breve{\mathcal{I}}}, \boldsymbol{y}_{\breve{\mathcal{I}}} \prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\breve{\mathcal{I}}} - x_i))^{\perp}$$

and

$$\mathscr{C}_0 \stackrel{def}{=} \sum_{j\in\mathcal{I}} \left\langle \left(\prod_{i\in\mathcal{I}\setminus\{j\}} (x_j - x_i)^{q^u-1}(\boldsymbol{x}_{\breve{\mathcal{I}}} - x_i)\right) y_j^{q^u} \boldsymbol{y}_{\breve{\mathcal{I}}} - y_j \left(\prod_{i\in\mathcal{I}\setminus\{j\}} (\boldsymbol{x}_{\breve{\mathcal{I}}} - x_i)^{q^u}\right) \boldsymbol{y}_{\breve{\mathcal{I}}}^{q^u} \mid u \in [\![1, m-1]\!] \right\rangle_{\mathbb{F}_{q^m}}.$$

This theorem generalizes Lemma 4.4, to sets $\mathcal{I}$ that are not necessarily singletons. Indeed, although Lemma 4.4 is already enough for understanding the partial explanation that follows, we believe that this structural result is of interest in itself. Moreover, it would anyway represent the first step for a complete analysis. The next subsection is devoted to the proof of Theorem 5.1.

Proof of Theorem 5.1

We start by decomposing a GRS code in the direct sum of two linear codes in such a way that one is identically zero over a set of positions $\mathcal{I}$. With some abuse of notation, we allow here the multiplier of GRS codes and dual of alternant codes to have some zero coordinates. In this case, the codes are merely meant to be identically zero in the corresponding positions.

Lemma 5.1. *Let $\boldsymbol{GRS}_r(\boldsymbol{x},\boldsymbol{y}) \subseteq \mathbb{F}_{q^m}^n$ be a GRS code and let $\mathcal{I} \subseteq [\![1,n]\!]$ be a non-empty set of cardinality at most r. Then*

$$\boldsymbol{GRS}_r(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x} - x_i)) \oplus \boldsymbol{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}).$$

Proof. We prove separately both the inclusions of equality.

Proof of "$\supseteq$": we first show that the two linear codes on the right-hand side are subcodes of $\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})$. For the latter code on the right-hand side,

i.e. $\mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})$, this is obvious, since it is a GRS code that shares the same support and multiplier with $\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})$ and has a smaller degree. Regarding the code, the definition of GRS code implies that

$$\mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i)) = \left\langle \boldsymbol{x}^a\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i)\boldsymbol{y} \mid a\in[\![0,r-|\mathcal{I}|-1]\!]\right\rangle_{\mathbb{F}_{q^m}}.$$

For any $a\in[\![0,r-|\mathcal{I}|-1]\!]$, we have

$$\deg(\boldsymbol{x}^a\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))\leq r-1,$$

hence $\boldsymbol{x}^a\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i)\boldsymbol{y}\in\ \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})$ from which we obtain

$$\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})\supseteq\ \mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i)).$$

Proof of "$\subseteq$": Any codeword in $\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})$ can be written as $P(\boldsymbol{x})\boldsymbol{y}$ for some polynomial $P\in\mathbb{F}_{q^m}[z]$ of degree at most $r-1$, with the usual notation that extends to vectors the evaluation of a function. Let Q,R be respectively the quotient and remainder of P with respect to the polynomial

$$\prod_{i\in\mathcal{I}}(z-x_i)\in\mathbb{F}_{q^m}[z],$$

so that

$$P(z)=Q(z)\cdot\prod_{i\in\mathcal{I}}(z-x_i)+R(z),$$

with $\deg(Q)=\deg(P)-\deg(\prod_{i\in\mathcal{I}}(z-x_i))\leq(r-1)-|\mathcal{I}|$ and $\deg(R)<|\mathcal{I}|$. Therefore we have

$$P(\boldsymbol{x})\boldsymbol{y}=\underbrace{Q(\boldsymbol{x})\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i)\boldsymbol{y}}_{\in\ \mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))}+\underbrace{R(\boldsymbol{x})\boldsymbol{y}}_{\in\ \mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})},$$

which proves the "$\subseteq$" inclusion.

The fact that the sum in $\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})=\ \mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))+\mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})$ is direct follows from a dimension argument: since

$$\begin{aligned}\dim_{\mathbb{F}_{q^m}}\ \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y}) &= r\\ &= (r-|\mathcal{I}|)+|\mathcal{I}|\\ &= \dim_{\mathbb{F}_{q^m}}\ \mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))+\dim_{\mathbb{F}_{q^m}}\ \mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}),\end{aligned}$$

the two GRS codes on the right-hand side must have a trivial intersection. □

Lemma 5.1 translates, in terms of alternant codes, into

Lemma 5.2. *Let $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})\subseteq\mathbb{F}_{q^m}^n$ be an alternant code and let $\mathcal{I}\subseteq[\![1,n]\!]$ be of cardinality at most r. Then*

$$\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}=\left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^\perp\right)_{\mathbb{F}_{q^m}}+\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}.$$

Proof. Lemma 5.1 readily implies, for any non negative integer i,

$$\mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})^{q^i} = \mathbf{GRS}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^{q^i} \oplus \mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{q^i},$$

It is then enough to sum over $i \in [\![0, m-1]\!]$ the codes in the equalities above to get the thesis. □

Remark 5.1. Under the standard assumption that $\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = \bigoplus_{i\in[\![0,m-1]\!]} \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})^{q^i}$, the sum from Lemma 5.2 is direct:

$$\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = \left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^\perp\right)_{\mathbb{F}_{q^m}} \bigoplus \left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}}.$$

The next proposition explains how the shortening operation behaves on a sum of two codes, one of which is identically zero in the shortened positions.

Proposition 5.1. *Let $\mathscr{C},\mathscr{D}\subseteq\mathbb{F}^n$ be two codes. If $\mathbf{Pct}_\mathcal{I}(\mathscr{C}) = \mathbf{Sh}_\mathcal{I}(\mathscr{C})$ then*

$$\mathbf{Sh}_\mathcal{I}(\mathscr{C}+\mathscr{D}) = \mathbf{Pct}_\mathcal{I}(\mathscr{C}) + \mathbf{Sh}_\mathcal{I}(\mathscr{D}).$$

Proof. Proof of "$\supseteq$": From $\mathbf{Sh}_\mathcal{I}(\mathscr{C}+\mathscr{D}) \supseteq \mathbf{Sh}_\mathcal{I}(\mathscr{C})$ and $\mathbf{Sh}_\mathcal{I}(\mathscr{C}+\mathscr{D}) \supseteq \mathbf{Sh}_\mathcal{I}(\mathscr{D})$ it follows that

$$\begin{aligned}\mathbf{Sh}_\mathcal{I}(\mathscr{C}+\mathscr{D}) &\supseteq \mathbf{Sh}_\mathcal{I}(\mathscr{C}) + \mathbf{Sh}_\mathcal{I}(\mathscr{D})\\ &= \mathbf{Pct}_\mathcal{I}(\mathscr{C}) + \mathbf{Sh}_\mathcal{I}(\mathscr{D}).\end{aligned}$$

Proof of "$\subseteq$": Let $\boldsymbol{s}\in\mathbf{Sh}_\mathcal{I}(\mathscr{C}+\mathscr{D})$. There exists a vector $\boldsymbol{a}=\boldsymbol{c}+\boldsymbol{d}\in\mathbb{F}^n$, with $\boldsymbol{c}\in\mathscr{C}$ and $\boldsymbol{d}\in\mathscr{D}$, such that $\boldsymbol{s}=\boldsymbol{a}_{\overset{\vee}{\mathcal{I}}}$ and $a_i=0$ for all $i\in\mathcal{I}$. Since $\mathbf{Pct}_\mathcal{I}(\mathscr{C}) = \mathbf{Sh}_\mathcal{I}(\mathscr{C})$, any codeword in $\mathscr{C}$ is zero over $\mathcal{I}$, in particular $c_i=0$ for all $i\in\mathcal{I}$. Hence, for any position $i\in\mathcal{I}$,

$$d_i = a_i - c_i = 0 - 0 = 0,$$

which implies that $\boldsymbol{d}_{\overset{\vee}{\mathcal{I}}}\in\mathbf{Sh}_\mathcal{I}(\mathscr{D})$. Thus,

$$\boldsymbol{s} = \boldsymbol{a}_{\overset{\vee}{\mathcal{I}}} = \boldsymbol{c}_{\overset{\vee}{\mathcal{I}}} + \boldsymbol{d}_{\overset{\vee}{\mathcal{I}}} \in \mathbf{Sh}_\mathcal{I}(\mathscr{C}) + \mathbf{Sh}_\mathcal{I}(\mathscr{D}) = \mathbf{Pct}_\mathcal{I}(\mathscr{C}) + \mathbf{Sh}_\mathcal{I}(\mathscr{D}).$$

□

We are now ready to prove the main result.

Proof. Proof of Theorem 5.1: Observe that, since $\left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^\perp\right)_{\mathbb{F}_{q^m}}$ is identically zero over $\mathcal{I}$,

$$\begin{aligned}\mathbf{Sh}_\mathcal{I}\left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^\perp\right)_{\mathbb{F}_{q^m}} &= \mathbf{Pct}_\mathcal{I}\left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^\perp\right)_{\mathbb{F}_{q^m}}\\ &= \mathbf{Sh}_\mathcal{I}\left(\mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))\right)^\perp_{\mathbb{F}_{q^m}} \quad \text{(from Proposition 1.2)}\\ &= \mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}},\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}}\prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\overset{\vee}{\mathcal{I}}}-x_i))^\perp_{\mathbb{F}_{q^m}} \quad \text{(from Proposition 4.1).}\end{aligned}$$

Hence, by shortening with respect to $\mathcal{I}$ both sides of

$$\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}} = \mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y}\prod_{i\in\mathcal{I}}(\boldsymbol{x}-x_i))^{\perp}_{\mathbb{F}_{q^m}} + \mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}},$$

we obtain

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right) = \mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x}_{\breve{\mathcal{I}}},\boldsymbol{x}_{\breve{\mathcal{I}}}\prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\breve{\mathcal{I}}}-x_i))^{\perp}_{\mathbb{F}_{q^m}} + \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right).$$

It remains to prove that $\mathscr{C}_0 \subseteq \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right)$. Take a basis element of $\mathscr{C}_0$ as in the definition of Theorem 5.1, i.e. fix $j\in\mathcal{I}$ and $u\in[\![1,m-1]\!]$, and consider

$$\boldsymbol{c}_0 \stackrel{\text{def}}{=} \left(\prod_{i\in\mathcal{I}\setminus\{j\}}(x_j-x_i)^{q^u-1}(\boldsymbol{x}_{\breve{\mathcal{I}}}-x_i)\right)y_j^{q^u}\boldsymbol{y}_{\breve{\mathcal{I}}} - y_j\left(\prod_{i\in\mathcal{I}\setminus\{j\}}(\boldsymbol{x}_{\breve{\mathcal{I}}}-x_i)^{q^u}\right)\boldsymbol{y}_{\breve{\mathcal{I}}}^{q^u} \in \mathscr{C}_0.$$

It follows from

$$\left(\prod_{i\in\mathcal{I}\setminus\{j\}}(x_j-x_i)^{q^u-1}(\boldsymbol{x}_{\breve{\mathcal{I}}}-x_i)\right)y_j^{q^u}\boldsymbol{y}_{\breve{\mathcal{I}}} \in \mathbf{Pct}_{\mathcal{I}}\left(\ \mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})\right)$$

and

$$y_j\left(\prod_{i\in\mathcal{I}\setminus\{j\}}(\boldsymbol{x}_{\breve{\mathcal{I}}}-x_i)^{q^u}\right)\boldsymbol{y}_{\breve{\mathcal{I}}}^{q^u} \in \mathbf{Pct}_{\mathcal{I}}\left(\ \mathbf{GRS}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})\right)^{q^u}$$

that

$$\boldsymbol{c}_0 \in \mathbf{Pct}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right).$$

Moreover, a direct evaluation shows that the restriction of $\boldsymbol{c}_0$ to $\mathcal{I}$ is the null vector. Therefore

$$\mathscr{C}_0 \subset \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right)$$

and the proof is concluded. □

Remark 5.2. Experimental computations show that the equality between $\mathscr{C}_0$ and $\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right)$ is expected with high probability and a dimension inspection supports this. On one side, $\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right)$ is the shortening in $|\mathcal{I}|$ positions of a code of dimension $|\mathcal{I}|m$. On the other hand, we wrote $\mathscr{C}_0$ using $|\mathcal{I}|(m-1) = |\mathcal{I}|m-|\mathcal{I}|$ generators. Although we have not proved their linear independence, the latter is an extremely reasonable assumption if the alternant code has not an additional structure.

From now on, we will make the following assumption

Assumption 5.1. *In the setting of Theorem 5.1, we assume that*

$$\mathscr{C}_0 = \mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_{|\mathcal{I}|}(\boldsymbol{x},\boldsymbol{y})^{\perp}_{\mathbb{F}_{q^m}}\right)$$

and consequently

$$\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}} = \mathscr{B}_{\mathbb{F}_{q^m}} + \mathscr{C}_0.$$

5.4 A decomposition for the square code

As in the proof of Lemma 4.5, we define the codes

$$\mathscr{D}_0 \stackrel{\text{def}}{=} \mathscr{B}_{\mathbb{F}_{q^m}} \star \mathscr{C}_0, \qquad \mathscr{D}_1 \stackrel{\text{def}}{=} \left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2},$$

thus extending the definition for any $|\mathcal{I}|$. Moreover, we denote with $\mathscr{D}_2$ the square of $\mathscr{C}_0$:

$$\mathscr{D}_2 \stackrel{\text{def}}{=} \mathscr{C}_0^{\star 2}.$$

Thus we have

Proposition 5.2. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{F}_{q^m}^n$ be an alternant code and $\mathcal{I} \subset [\![1, n]\!]$ a set of cardinality at most r. Under Assumption 5.1, we have*

$$\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2} = \mathscr{D}_0 + \mathscr{D}_1 + \mathscr{D}_2.$$

Proof. The thesis immediately follows by using the decomposition of the shortened code:

$$\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2} = \left(\mathscr{B}_{\mathbb{F}_{q^m}} + \mathscr{C}_0\right)^{\star 2} = \mathscr{D}_0 + \mathscr{D}_1 + \mathscr{D}_2.$$

□

The dimension of $\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2}$ can not be computed using the inclusion-exclusion principle, as it does not hold for the sum of three or more vector spaces. Nevertheless, we can write

$$\begin{aligned}\dim_{\mathbb{F}_{q^m}}\left(\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)\right)_{\mathbb{F}_{q^m}}\right)^{\star 2} &= \dim_{\mathbb{F}_{q^m}}(\mathscr{D}_0 + \mathscr{D}_1 + \mathscr{D}_2) \\ &= \dim_{\mathbb{F}_{q^m}}(\mathscr{D}_0 + \mathscr{D}_1) + \dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 - \dim_{\mathbb{F}_{q^m}}((\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2) \\ &= \dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 + \dim_{\mathbb{F}_{q^m}} \mathscr{D}_1 + \dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 \\ &\quad - \dim_{\mathbb{F}_{q^m}}(\mathscr{D}_0 \cap \mathscr{D}_1) - \dim_{\mathbb{F}_{q^m}}((\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2)\,.\end{aligned} \tag{5.1}$$

Understanding the dimensions of $\mathscr{D}_0$, $\mathscr{D}_1$ and $\mathscr{D}_2$ and of their mutual intersections is therefore at the core of the comprehension of $\dim_{\mathbb{F}_{q^m}}\left(\mathbf{Sh}_{\mathcal{I}}\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2}$. We will discuss these quantities in some special cases and derive upper bounds for $\dim_{\mathbb{F}_{q^m}}\left(\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2}$. Before that, we start with some examples.

5.4.1 Empirical dimensions of the square code summands and their intersections

We come back to the examples shown in the previous section and specify the dimensions of $\mathscr{D}_0$, $\mathscr{D}_1$ and $\mathscr{D}_2$ and of their intersections.

We start with the 3-tuple $(q, m) = (17, 2)$ and $r \in \{15, 17, 19\}$, in Tables 5.16, 5.17, 5.18, 5.19.

In the case where $r < q - 1$, the analogy between alternant and Goppa codes for what concerns square codes is mirrored in their decomposition, too. In particular, in Table 5.16, we observe several phenomena in both cases:

	$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alternant/Goppa	$\mathscr{B}_{\mathbb{F}_{q^m}}$	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	$\mathscr{D}_0$	0	28	52	72	88	100	108	112	112	108	100	88	72	52	28	0
	$\mathscr{D}_1$	283	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	3	6	10	15	21	24	14	3	0	0	0	0	0	0

Table 5.16: Dimensions of codes for the square code decomposition. Parameters: $(q, r, m, n) = (17, 15, 2, 289)$

- In the example of Table 5.16, $\dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 = \binom{|\mathcal{I}|+1}{2}$. More generally,

$$\dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 \leq \binom{|\mathcal{I}|(m-1)+1}{2}.$$

  Since $\mathscr{D}_2$ is the square of $\mathscr{C}_0$, and the latter has dimension equal to $|\mathcal{I}|(m-1)$, if the equality holds it means that $\mathscr{C}_0$ behaves like a random code in terms of the square code.

- The product code $\mathscr{D}_0$ is such that

$$\dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 = \dim_{\mathbb{F}_{q^m}} \mathscr{B}_{\mathbb{F}_{q^m}} \cdot \dim_{\mathbb{F}_{q^m}} \mathscr{C}_0,$$

  hence it behaves as a random componentwise product.

- the intersection $\mathscr{D}_0 \cap \mathscr{D}_1$ is trivial at least in the range of parameters where the shortened code is distinguishable.

- At first, $\dim_{\mathbb{F}_{q^m}} (\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$ has a quadratic growth with respect to $|\mathcal{I}|$, but then it starts to decrease towards 0. This can be qualitatively explained in the following way. For small $|\mathcal{I}|$, the code $\mathscr{D}_0 + \mathscr{D}_1$ is a code with very big dimension. When the code is not classically distinguishable, $\mathscr{D}_0 + \mathscr{D}_1$ can even be the whole ambient space $\mathbb{F}_{q^m}^{n-|\mathcal{I}|}$. Such a code contains $\mathscr{D}_2$, hence $(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2 = \mathscr{D}_2$. With $|\mathcal{I}|$ increasing, $\mathscr{D}_0 + \mathscr{D}_1$ becomes smaller and smaller until, for these parameters, the only common codewords with $\mathscr{D}_2$ is the null vector.

If instead, the shortened code is not distinguishable, the intersection $\mathscr{D}_0 + \mathscr{D}_1$ is not degenerate. Table 5.17 shows this fact for the non-classically distinguishable parameters $(q, r, m, n) = (17, 15, 2, 278)$. In more detail, we have seen in the previous section that shortening the code in one position is not sufficient for these parameters. As a result, $\dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 \cap \mathscr{D}_1 = 1 > 0$.

Let us now increase r. Table 5.18 refers to parameters $(q, r, m, n) = (17, 17, 2, 289)$. In this case, random alternant codes behave, as expected, differently from Goppa codes, because $r \geq q - 1$, thus we split the table into two parts.

Let us focus on $\mathscr{D}_0 \cap \mathscr{D}_1$. In both cases, such intersection is not trivial for some values of $|\mathcal{I}|$. For instance, this occurs when the shortened code is not distinguishable. The corresponding dimensions are written in blue. For the values of $|\mathcal{I}|$ above the

	$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Alternant/Goppa	$\mathscr{B}_{\mathbb{F}_{q^m}}$	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	$\mathscr{D}_0$	0	28	52	72	88	100	108	112	112	108	100	88	72	52	28	0
	$\mathscr{D}_1$	278	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 5.17: Dimensions of codes for the square code decomposition. Parameters: $(q, r, m, n) = (17, 15, 2, 278)$

	$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Alternant	$\mathscr{B}_{\mathbb{F}_{q^m}}$	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	$\mathscr{D}_0$	0	32	60	84	104	120	132	140	144	144	140	132	120	104	84	60	32	0
	$\mathscr{D}_1$	289	288	283	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	32	56	48	38	26	12	0	0	0	0	0	0	0	0	0	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	3	6	10	15	21	24	14	3	0	0	0	0	0	0	0	0
Goppa	$\mathscr{B}_{\mathbb{F}_{q^m}}$	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
	$\mathscr{D}_0$	0	32	60	84	104	120	132	140	144	144	140	132	120	104	84	60	32	0
	$\mathscr{D}_1$	289	288	283	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	32	56	48	38	26	20	18	16	14	12	10	8	6	4	2	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	3	6	10	15	13	6	0	0	0	0	0	0	0	0	0	0

Table 5.18: Dimensions of codes for the square code decomposition. Parameters: $(q, r, m, n) = (17, 17, 2, 289)$

distinguishability threshold, the pattern becomes more clear. In the alternant case, the intersection degenerates, and this is related to the fact that $e_{\mathscr{A}} = 0$. Regarding Goppa codes,

$$\dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 \cap \mathscr{D}_1 = 2(r - 1 - |\mathcal{I}|).$$

Finally, we take $r = 19$, which implies $e_{\mathscr{A}} = 1$. This time, both in the alternant and Goppa cases, the dimension of $\mathscr{D}_0 \cap \mathscr{D}_1$ has a quadratic growth for small values of $|\mathcal{I}|$ and then a linear decrease. Furthermore, $(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$ has positive dimension even for higher values of $|\mathcal{I}|$.

As done before, we now increase the extension field degree m (and decrease the field size q). Table 5.20 gives the target dimensions for the 4-tuple $(q, r, m, n) = (3, 8, 7, 2187)$, which determines $e_{\mathscr{A}} = 1$. We remark that here and more in general for high values of m, the dimension $(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$ is not non-negligible with respect to the total square code dimension, even for $|\mathcal{I}|$ close to r.

	$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Alternant	$\mathscr{B}_{\mathbb{F}_{q^m}}$	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	$\mathscr{D}_0$	0	36	67	96	120	140	156	168	176	180	180	176	168	156	140	120	96	68	36	0
	$\mathscr{D}_1$	289	288	287	286	283	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153	171	190
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	36	67	96	118	106	92	76	58	38	32	28	24	20	16	12	8	4	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	39	30	20	9	2	2	2	2	2	0
Goppa	$\mathscr{B}_{\mathbb{F}_{q^m}}$	38	36	34	32	30	28	26	24	22	20	18	16	14	12	10	8	6	4	2	0
	$\mathscr{C}_0$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
	$\mathscr{D}_0$	0	36	67	96	120	140	156	168	176	180	180	176	168	156	140	120	96	68	36	0
	$\mathscr{D}_1$	289	288	287	286	283	250	219	190	163	138	115	94	75	58	43	30	19	10	3	0
	$\mathscr{D}_2$	0	1	3	6	10	15	21	28	36	45	55	66	78	91	105	120	136	153	171	190
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	36	67	96	118	106	92	76	60	54	48	42	36	30	24	18	12	6	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	1	3	6	10	15	21	28	34	29	23	16	8	6	6	6	6	6	6	0

Table 5.19: Dimensions of codes for the square code decomposition. Parameters: $(q, r, m, n) = (17, 19, 2, 289)$

	$\lvert\mathcal{I}\rvert$	0	1	2	3	4	5	6	7	8
Alternant	$\mathscr{B}_{\mathbb{F}_{q^m}}$	56	49	42	35	28	21	14	7	0
	$\mathscr{C}_0$	0	6	12	18	24	30	36	42	48
	$\mathscr{D}_0$	0	294	497	616	644	616	490	294	0
	$\mathscr{D}_1$	1204	952	728	532	364	224	105	28	0
	$\mathscr{D}_2$	0	21	78	171	300	465	659	882	1134
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	70	105	112	84	56	21	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	0	8	24	48	80	71	50	0
Goppa	$\mathscr{B}_{\mathbb{F}_{q^m}}$	56	49	42	35	28	21	14	7	0
	$\mathscr{C}_0$	0	6	12	18	24	30	36	42	48
	$\mathscr{D}_0$	0	294	496	581	609	581	490	294	0
	$\mathscr{D}_1$	840	734	642	515	364	224	105	28	0
	$\mathscr{D}_2$	0	21	78	171	300	465	638	728	832
	$\mathscr{D}_0 \cap \mathscr{D}_1$	0	196	328	322	249	145	55	0	0
	$(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$	0	14	50	108	188	290	344	217	0

Table 5.20: Dimensions of codes for the square code decomposition. Parameters: $(q, r, m, n) = (3, 8, 7, 2187)$

5.4.2 A partial explanation for the square of the shortened code

As already mentioned before, we focus on the case $\mathcal{I} = \{i\}$ for our explanation. In this setting, Theorem 5.1 simplifies into the inclusion

$$\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} \supseteq \mathscr{B}_{\mathbb{F}_{q^m}} + \mathscr{C}_0,$$

where

$$\mathscr{B} \stackrel{\text{def}}{=} \mathscr{A}_{r-1}(\boldsymbol{x}_{\tilde{i}}, (\boldsymbol{x}_{\tilde{i}} - x_i)\boldsymbol{y}_{\tilde{i}})^\perp$$

and

$$\mathscr{C}_0 \stackrel{\text{def}}{=} \left\langle y_i^{q^u}\boldsymbol{y}_{\tilde{i}} - y_i\boldsymbol{y}_{\tilde{i}}^{q^u} \mid u \in [\![1, m-1]\!] \right\rangle_{\mathbb{F}_{q^m}}.$$

In this case, a dimensional argument even guarantees that the inclusion from Theorem 5.1 is an equality and there is no need to make any assumption, except the standard one that the dual code $\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp$ has dimension rm.

Proposition 5.3.

$$\mathbf{Sh}_i\left(\mathscr{A}_r(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = \mathscr{B}_{\mathbb{F}_{q^m}} + \mathscr{C}_0,$$

where

$$\mathscr{B} \stackrel{def}{=} \mathscr{A}_{r-1}(\boldsymbol{x}_{\tilde{i}}, (\boldsymbol{x}_{\tilde{i}} - x_i)\boldsymbol{y}_{\tilde{i}})^\perp$$

and

$$\mathscr{C}_0 \stackrel{def}{=} \left\langle y_i^{q^u}\boldsymbol{y}_{\tilde{i}} - y_i\boldsymbol{y}_{\tilde{i}}^{q^u} \mid u \in [\![1, m-1]\!] \right\rangle_{\mathbb{F}_{q^m}}.$$

Proof. From the proof of Theorem 5.1, we know that

$$\mathscr{C}_0 \subseteq \mathbf{Sh}_i\left(\mathscr{A}_1(\boldsymbol{x},\boldsymbol{y})^\perp_{\mathbb{F}_{q^m}}\right)$$

Since the multiplier $\boldsymbol{y}$'s coordinates are non-zero, $\mathscr{A}_1(\boldsymbol{x},\boldsymbol{y})^\perp_{\mathbb{F}_{q^m}}$ has not identically zero coordinates and therefore, when shortened in only one position i, its dimension must decrease by 1. Hence

$$\dim_{\mathbb{F}_{q^m}} \mathbf{Sh}_i\left(\mathscr{A}_1(\boldsymbol{x},\boldsymbol{y})^\perp\right)_{\mathbb{F}_{q^m}} = m - 1 = \dim \mathscr{C}_0,$$

which, given one inclusion, shows the equality between the two codes and concludes the proof. □

Remark 5.3. Note that the linear code $\mathscr{C}_0$ is the same as the one in Lemma 4.4, despite the different definitions. Indeed,

$$\dim_{\mathbb{F}_{q^m}} \mathscr{C}_0 = m - 1,$$

and a set of independent generators for $\mathscr{C}_0$ could have been obtained, for instance, by fixing one between u or v and forcing the other index to be different. This coincides, in fact, with the definition just adopted here.

The generic upper bounds for square and product of codes, coming from Proposition 1.6 can be used to estimate the dimensions of $\mathscr{D}_0, \mathscr{D}_1$ and $\mathscr{D}_2$. In particular

$$\dim_{\mathbb{F}_{q^m}} \mathscr{C}_0 = m-1 \quad \Rightarrow \quad \dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 \leq \binom{m}{2}$$

and

$$\begin{cases} \dim_{\mathbb{F}_{q^m}} \mathscr{B}_{\mathbb{F}_{q^m}} = (r-1)m \\ \dim_{\mathbb{F}_{q^m}} \mathscr{C}_0 = m-1 \end{cases} \quad \Rightarrow \quad \dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 \leq (r-1)m(m-1).$$

Regarding $\mathscr{D}_1$, the upper bounds about alternant codes, i.e. Theorem 3.2 provides a finer estimate. Being careful to replace r with $r-1$ in the formula, we obtain

$$\dim_{\mathbb{F}_{q^m}} \mathscr{D}_1 \leq \binom{(r-1)m+1}{2} - \frac{m}{2}(r-2)\left((2e'+1)(r-1) - 2\frac{q^{e'+1}-1}{q-1}\right),$$

with $e' \stackrel{\text{def}}{=} \left\lfloor \log_q(r-2) \right\rfloor$.

In the rest of the section, we focus on the analysis of $\dim_{\mathbb{F}_{q^m}}(\mathscr{D}_0 \cap \mathscr{D}_1)$. In the following technical lemma, we exhibit sets of codewords that belong to $\mathscr{D}_0 \cap \mathscr{D}_1$, thus providing a lower bound on its dimension, assuming such codewords to be linearly independent.

Lemma 5.3. *Let $\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y}) \subseteq \mathbb{F}_q$ be an alternant code, $i \in [\![1, n]\!]$ and $\mathscr{D}_1$ be defined as above. Let $e_{\mathscr{A}} = \left\lfloor \log_q(r-1) \right\rfloor$. Then*

$$\left[\left(\boldsymbol{x}_i^a (\boldsymbol{x}_{\check{i}} - x_i)^2 \boldsymbol{y}_{\check{i}} \right)^{q^{m-u}} \star (y_i^{q^{m-u}} \boldsymbol{y}_{\check{i}} - y_i \boldsymbol{y}_{\check{i}}^{q^{m-u}}) \right]^{q^\ell} \in \mathscr{D}_0 \cap \mathscr{D}_1 \tag{5.2}$$

and

$$\left[\left(\boldsymbol{x}_{\check{i}}^b (\boldsymbol{x}_{\check{i}} - x_i)^{q^u+1} \boldsymbol{y}_{\check{i}} \right)^{q^{m-u}} \star (y_i^{q^{m-u}} \boldsymbol{y}_{\check{i}} - y_i \boldsymbol{y}_{\check{i}}^{q^{m-u}}) \right]^{q^\ell} \in \mathscr{D}_0 \cap \mathscr{D}_1 \tag{5.3}$$

for all $u \in [\![1, e_{\mathscr{A}}]\!], a \in [\![0, r-3]\!], b \in [\![0, r-q^u-2]\!], \ell \in [\![0, m-1]\!]$.

Proof. Proof of (5.2). Let $P(z) \stackrel{\text{def}}{=} z^a(z - x_i) \in \mathbb{F}_{q^m}[z]$. For any $a \in [\![0, r-3]\!]$, $\deg(P) = a+1 \leq r-2$. Therefore

$$\boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i)^2 \boldsymbol{y}_{\check{i}} = (\boldsymbol{x}_{\check{i}} - x_i) P(\boldsymbol{x}_{\check{i}}) \boldsymbol{y}_{\check{i}} \in \mathbf{Sh}_i\left(\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) \right) \subseteq \mathscr{B}_{\mathbb{F}_{q^m}},$$

from which we obtain, by applying $\ell + m - u$ times the Frobenius morphism $z \to z^q$,

$$\left(\boldsymbol{x}_{\check{i}}^a (\boldsymbol{x}_{\check{i}} - x_i)^2 \boldsymbol{y}_{\check{i}} \right)^{q^{l+m-u}} \in \mathbf{Sh}_i\left(\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y}) \right)^{q^{\ell+m-u}} = \mathbf{Sh}_i\left(\mathbf{GRS}_r(\boldsymbol{x}, \boldsymbol{y})^{q^{\ell+m-u}} \right) \subseteq \mathscr{B}_{\mathbb{F}_{q^m}}.$$

On the other hand, for any $\ell \in [\![0, m-1]\!]$ and $u \in [\![1, e_{\mathscr{A}}]\!]$,

$$\begin{aligned} &(y_i^{q^{m-u}} \boldsymbol{y}_{\check{i}} - y_i \boldsymbol{y}_{\check{i}}^{q^{m-u}})^{q^\ell} \\ &= y_i^{q^{m-u+\ell}} \boldsymbol{y}_{\check{i}}^{q^\ell} - y_i^{q^\ell} \boldsymbol{y}_{\check{i}}^{q^{m-u+\ell}} \in \mathscr{C}_0. \end{aligned}$$

Then, by definition of the component-wise product of codes,

$$\begin{aligned}&\left[\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star\left(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\right)\right]^{q^\ell}\\ =&\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\right)^{q^{\ell+m-u}}\star\left(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\right)^{q^\ell}\in\mathscr{B}_{\mathbb{F}_{q^m}}\star\mathscr{C}_0.\end{aligned}$$

We now have to show that the target vector belongs to $\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}$, too. We observe that, since alternant codes are $\mathbb{F}_q$-stable and the square code operator preserves such stability, it is enough to prove that

$$\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}})\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2},$$

i.e. we can ignore the Frobenius morphism $z\mapsto z^{q^\ell}$.

We split the star-product of codewords above in the following difference of two star-products

$$\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\star(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}})=\ \boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\star y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\star y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}$$

and treat them separately. We have

$$\begin{aligned}&\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\star y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}\\ =&y_i^{q^{m-u}}\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\star(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\right)\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}\end{aligned}$$

and

$$\begin{aligned}&\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)^2\boldsymbol{y}_{\check{i}}\star y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\\ =&y_i\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\star(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}^{q^{m-u}}\right)\\ =&y_i\left(\boldsymbol{x}_{\check{i}}^{a}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\star\left((\boldsymbol{x}_{\check{i}}-x_i)^{q^u}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\right)\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2},\end{aligned}$$

where the last membership follows from the fact that $q^u\le q^{e_{\mathscr{A}}}\le r-1$ and therefore

$$(\boldsymbol{x}_{\check{i}}-x_i)^{q^u}\boldsymbol{y}_{\check{i}}\in\mathscr{B}_{\mathbb{F}_{q^m}}.$$

Hence the target vector belongs to the intersection of $\mathscr{B}_{\mathbb{F}_{q^m}}\star\mathscr{C}_0$ and $\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}$.

Proof of (5.3). For any $b\in[\![0,r-q^u-2]\!]$, $b+q^u+1\le r-1$, thus

$$\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\in\mathbf{Sh}_i\left(\ \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})\right)\subseteq\mathscr{B}_{\mathbb{F}_{q^m}},$$

from which we obtain, by applying $\ell+m-u$ times the Frobenius morphism $z\to z^q$,

$$\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{\ell+m-u}}\in\mathbf{Sh}_i\left(\ \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})\right)^{q^{\ell+m-u}}=\mathbf{Sh}_i\left(\ \mathbf{GRS}_r(\boldsymbol{x},\boldsymbol{y})^{q^{\ell+m-u}}\right)\subseteq\mathscr{B}_{\mathbb{F}_{q^m}}.$$

As already shown in the proof of (5.2), for any $\ell\in[\![0,m-1]\!]$ and $u\in[\![1,m-1]\!]\supseteq[\![1,e_{\mathscr{A}}]\!]$,

$$(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}})^{q^\ell}\in\mathscr{C}_0.$$

Then, by definition of the component-wise product of codes,

$$\begin{aligned}&\left[\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star\left(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\right)\right]^{q^\ell}\\ =&\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{\ell+m-u}}\star\left(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\right)^{q^\ell}\in\mathscr{B}_{\mathbb{F}_{q^m}}\star\mathscr{C}_0.\end{aligned}$$

We now have to show that the target vector belongs to $\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}$, too. We observe that, since alternant codes are $\mathbb{F}_q$-stable, it is enough to prove that

$$\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}})\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2},$$

i.e. we can ignore the Frobenius morphism $z\to z^{q^\ell}$.

We split the star-product of codewords above in the following difference of two star-products

$$\begin{aligned}&\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star(y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}-y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}})\\ =&\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}\\ &-\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\end{aligned}$$

and treat them separately. We have

$$\begin{aligned}&\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}\\ =&\boldsymbol{x}_{\check{i}}^{bq^{m-u}}(\boldsymbol{x}_{\check{i}}-x_i)^{1+q^{m-u}}\boldsymbol{y}_{\check{i}}^{q^{m-u}}\star y_i^{q^{m-u}}\boldsymbol{y}_{\check{i}}\\ =&y_i^{q^{m-u}}\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}\end{aligned}$$

and

$$\begin{aligned}&\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)^{q^u+1}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star y_i\boldsymbol{y}_{\check{i}}^{q^{m-u}}\\ =&y_i(\boldsymbol{x}_{\check{i}}-x_i)\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star\left((\boldsymbol{x}_{\check{i}}-x_i)^{q^u}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\\ =&y_i\left(\boldsymbol{x}_{\check{i}}^{b}(\boldsymbol{x}_{\check{i}}-x_i)\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\star\left((\boldsymbol{x}_{\check{i}}-x_i)^{q^u}\boldsymbol{y}_{\check{i}}\right)^{q^{m-u}}\in\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}.\end{aligned}$$

Hence the target vector belongs to the intersection of $\mathscr{B}_{\mathbb{F}_{q^m}}\star\mathscr{C}_0$ and $\left(\mathscr{B}_{\mathbb{F}_{q^m}}\right)^{\star 2}$. □

The total number of linear relationships found is

$$m\sum_{u=1}^{e_{\mathscr{A}}}((r-2)+(r-q^u-1))=m\sum_{u=1}^{e_{\mathscr{A}}}(2r-q^u-3).$$

Remark 5.4. Let $\mathscr{A}_3(\boldsymbol{x},\boldsymbol{y})\subseteq\mathbb{F}_2$ be a classically distinguishable random binary alternant code of order 3. Then $e_{\mathscr{A}}=\lfloor\log_2(3-1)\rfloor=1$. Let us consider the square

code of the dual code shortened in one position and let $\mathscr{D}_0, \mathscr{D}_1, \mathscr{D}_2$ be the codes defined from the decomposition given before. Then we have

$$\begin{cases} \dim_{\mathbb{F}_q} \mathscr{D}_0 \leq (r-1)m(m-1) = 2m(m-1), \\ \dim_{\mathbb{F}_q} \mathscr{D}_1 \leq \binom{(r-1)m+1}{2} - \frac{m}{2}(r-2)(r-3) = \binom{2m+1}{2}, \\ \dim_{\mathbb{F}_q} \mathscr{D}_2 \leq \binom{(m-1)+1}{2} = \binom{m}{2}, \\ \dim_{\mathbb{F}_q} (\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2 \geq 0. \end{cases} \tag{5.4}$$

Note that, with these parameters, the vectors proved in Lemma 5.3 to lie in $\mathscr{D}_1$ only come from (5.2), since $b \in [\![0, r-q^u-2]\!]$, but $r-q^u-2 \leq r-q-2 < 0$. If we assume that the $m(r-2)$ linear relationships from (5.2) are linearly independent, we obtain

$$\dim_{\mathbb{F}_q} \mathscr{D}_0 \cap \mathscr{D}_1 \geq (r-2)m. \tag{5.5}$$

We can thus upper bound the square code dimension as

$$\begin{aligned} \dim_{\mathbb{F}_{q^m}} \left(\left(\mathbf{Sh}_{\mathcal{I}} \left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp} \right) \right)_{\mathbb{F}_{q^m}} \right)^{\star 2} &= \dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 + \dim_{\mathbb{F}_{q^m}} \mathscr{D}_1 + \dim_{\mathbb{F}_{q^m}} \mathscr{D}_2 \\ &\quad - \dim_{\mathbb{F}_{q^m}} (\mathscr{D}_0 \cap \mathscr{D}_1) - \dim_{\mathbb{F}_{q^m}} ((\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2) \\ &\leq \binom{2m+1}{2} + 2m(m-1) + \binom{m}{2} - m \\ &= \binom{3m}{2} - m. \end{aligned}$$

Experimentally, the upper bound turns out to be tight and equality holds in the distinguishable regime, meaning that all the bounds from (5.4) and (5.5) are tight too. This result is related to the rank of $\mathcal{S}'$ in the analysis of Section 4.4.5 for the binary case (compare with the proof of Proposition 4.23).

Other linear relationships for $e_{\mathscr{A}} \geq 2$

The analysis becomes more complicated as $e_{\mathscr{A}}$ increases, as other linear relationships turn out. For instance, starting from $e_{\mathscr{A}} = 2$, the dimension of $\mathscr{D}_0$ becomes systematically smaller than $(r-1)(m-1)m$. Indeed, for any $\ell \in [\![0, m-1]\!]$, consider the three following elements of $\mathscr{D}_0$ that are component-wise products of basis elements of $\mathscr{B}_{\mathbb{F}_{q^m}}$ and $\mathscr{C}_0$:

- $\left[\left((\boldsymbol{x}_{\check{i}} - x_i)^{q^2} \boldsymbol{y}_{\check{i}} \right) \star (y_i^{q^2} \boldsymbol{y}_{\check{i}}^{q} - y_i^{q} \boldsymbol{y}_{\check{i}}^{q^2}) \right]^{q^\ell}$,
- $-\left[\left((\boldsymbol{x}_{\check{i}} - x_i)^{q} \boldsymbol{y}_{\check{i}} \right)^{q} \star (y_i^{q^2} \boldsymbol{y}_{\check{i}} - y_i \boldsymbol{y}_{\check{i}}^{q^2}) \right]^{q^\ell}$,
- $\left[\left((\boldsymbol{x}_{\check{i}} - x_i) \boldsymbol{y}_{\check{i}} \right)^{q^2} \star (y_i^{q} \boldsymbol{y}_{\check{i}} - y_i \boldsymbol{y}_{\check{i}}^{q}) \right]^{q^\ell}$.

It is straightforward to check that these three vectors sum to 0, providing m linear relationships (one for each choice of ℓ) among the standard set of generators of $\mathscr{D}_0$. Thus

$$\dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 \leq (r-1)m(m-1) - m.$$

Moreover, for $e_{\mathscr{A}} > 2$, these linear relationships still do not capture all the structure.

What is missing?

We have seen from the experimental results of this section that the impact of $(\mathscr{D}_0 + \mathscr{D}_1) \cap \mathscr{D}_2$ is not negligible. The exact formula or at least a good estimate of the latter code would help in the comprehension of the main square code. Furthermore, as already pointed out several times, the analysis provided here is far from covering the whole casuistry of the number $\dim_{\mathbb{F}_{q^m}} \mathscr{D}_0 \cap \mathscr{D}_1$. In particular, it would be interesting to achieve a better understanding of the dimension of the square of the shortened dual code in the following cases:

- when $e_{\mathscr{A}}$ is larger than 2,
- when the dual code is shortened on several positions,
- when the alternant code is also a Goppa code and the set of quadratic relationships is not the same, i.e. when the Goppa polynomial degree r is $\geq q - 1$.

For what concerns the latter case, even from an empirical viewpoint, it is not clear whether specific upper bounds could be derived. Indeed, we recall that, given the Goppa code $\mathscr{G}(\boldsymbol{x}, \Gamma) = \mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})$, the code $\mathscr{B}^{\perp} = \mathscr{A}_{r-|\mathcal{I}|}(\boldsymbol{x}_{\overline{\mathcal{I}}}, \prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\overline{\mathcal{I}}} - x_i)\boldsymbol{y}_{\overline{\mathcal{I}}})$ is not a Goppa code. Indeed, as highlighted in Chapter 4, $\prod_{i\in\mathcal{I}}(\boldsymbol{x}_{\overline{\mathcal{I}}} - x_i)\boldsymbol{y}_{\overline{\mathcal{I}}}$ is a vector of evaluations of a rational function that is not the inverse of a polynomial (of degree $r - |\mathcal{I}|$). Therefore, it remains to characterize in which range of parameters, if any, it is possible to upper bound $\dim_{\mathbb{F}_{q^m}} \left(\mathbf{Sh}_{\mathcal{I}} \left(\mathscr{A}_r(\boldsymbol{x}, \boldsymbol{y})^{\perp}\right)_{\mathbb{F}_{q^m}}\right)^{\star 2}$ with a number smaller than

$$\dim_{\mathbb{F}_{q^m}} \left(\mathscr{G}(\boldsymbol{x}, \Gamma)^{\perp}\right)^{\star 2} - |\mathcal{I}|,$$

the latter not improving the classical distinguisher.

5.5 Conclusions

In this chapter, we have presented a simple technique that allows, for some parameters, to enhance the classical distinguisher for high-rate alternant and Goppa codes from [Fau+13]. Our strategy exploits the viewpoint of the distinguisher in terms of the dimension of the square of the dual code and therefore it builds upon the structural results from Chapter 3. The strategy consists in shortening the dual code and then computing its square code. Then, the arising dimension has to be compared with that obtained by replacing the original alternant/Goppa code with a random one with the same parameters. If the former is smaller, then the dimension of the square code represents a distinguisher. This procedure can be done in polynomial time.

We proposed a possible way to decompose the shortened code in such a way that part of the algebraic structure characterizing these families of codes is preserved. Starting from this decomposition, we provided a partial description of the dimension of the summands in the square code. Although the analysis is not complete, this is a first step for understanding and assessing the impact of this strategy. The chapter is also with numerous empirical examples. In particular, we found instances of random alternant code that are not distinguishable with the original approach form [Fau+13], but that become such with our strategy. We also showed how to produce

non-full-length parameters for which there exists an improvement, by decreasing the code length. Hence, to the best of our knowledge, this is the first time the high-rate distinguishable regime from [Fau+13] has been extended. Similar results have been obtained for Goppa codes, but only in the case where the order is small enough compared to the field size. In particular, no improvement is obtained for binary Goppa codes.

We leave as an open problem the complete analysis of the square code dimension after shortening. Solving it would shed light more rigorously on which parameters are really distinguishable. Moreover, it might also help to develop an even wider range distinguisher, for both alternant and Goppa codes, using more advanced techniques.

Conclusion

In this thesis, we have addressed two questions related to algebraic codes, namely the decoding of Reed-Solomon codes through algebraic techniques and the security of alternant and Goppa codes for cryptographic applications.

The first problem has a long history and extensive literature associated with it. Several efficient decoders for Reed-Solomon codes have been designed. Their ancestor is the Berlekamp-Welch algorithm that decodes uniquely up to the minimum code distance, which, RS codes being MDS codes, is the maximum possible for fixed parameters. This decoding radius was then improved by list decoding algorithms; first Sudan's and then Guruswami-Sudan's that reach Sudan's and Johnson's error-correction radii, respectively. Later, alternative methods were developed, including the so-called power-decoding algorithm [SSB10]. It is characterized by some algebraic equations that model the decoding problem and whose resolution leads to the correction of errors present in the received message. In its most advanced version [Nie18], the power-decoding strategy achieves correction performance similar to that of the Guruswami-Sudan algorithm. In Chapter 2, we further studied these multivariate equations. Instead of solving them by linearization, as done in the above algorithm, we analyze them by Gröbner bases techniques. These are the main tool for studying and solving multivariate algebraic systems, and their basics have been recalled in the introductory Chapter 1. All key equations used by the power decoding algorithm [Nie18] can be obtained from a subset of bilinear equations, i.e. linear with respect to two blocks of variables. The first is given by the coefficients of the error-locator polynomial, and the second by the coefficients of the polynomial associated with the sent codeword. Estimates of the complexity of computing a Gröbner basis for bilinear systems are better than those existing for random systems. However, the system in question can be solved even more efficiently than a generic bilinear system with the same number of equations and variables. Indeed, we have shown that it is possible to correct in polynomial time a number of errors up to Sudan's radius. The algorithm is appreciable in its simplicity: all necessary key equations are produced recursively and automatically by computing the Gröbner basis. We have pushed our analysis beyond Sudan's bound and through empirical observations have brought evidence that our method allows in some cases to correct up to and even slightly beyond Johnson's radius, which represents a barrier for power decoding, instead. By doing so, we derived new algebraic identities involving a single block of variables, even trying to insert them directly into the initial system. These appeared to be useful for the computation of degree falls crucial to solve the system. We have thus initiated a kind of algebraic decoding of Reed-Solomon codes based on the key equations of the power decoding algorithm. We see two possible directions to investigate this strategy further. First, it would be interesting to fully understand

the results obtained experimentally regarding the decoding when the number of errors reaches and exceeds Johnson's bound. A thorough study could indicate what is the optimal strategy in choosing the initial system, i.e. which equations to choose in the various cases in order to achieve the best efficiency in computing a Gröbner basis. On the other hand, the discovery of new polynomial identities suggests that different algebraic manipulations could lead to other key equations and consequently enhance the decoding by Gröbner basis techniques.

The other topic we have addressed in this manuscript concerns the security of Goppa codes (or more generally of alternant codes) in a cryptographic context. Goppa codes have been used in the original proposal of the first scheme based on codes, the McEliece cryptosystem [McE78], as well as in the CFS digital signature [CFS01]. Approximately 45 years after its invention, McEliece's encryption is still considered extremely secure and the attraction for it has even increased since it is considered a quantum-safe alternative. Currently, Classic McEliece [Alb+20] has been admitted to the fourth round of the NIST post-quantum competition and is considered ready for standardization. While generic decoding algorithms used to design parameters in code-based cryptography remain of exponential complexity, there is no guarantee that the special structure of Goppa codes could not be taken into account to break the cryptosystem. Our study draws inspiration from the high-rate distinguisher presented in [Fau+13], which, for the first time, challenges the widely believed view that Goppa codes are indistinguishable from random codes, only for sufficiently high rates, though. In particular, the parameters affected by this distinguisher do not include those of Classic McEliece (whereas those of CFS are). Furthermore, a distinguisher does not necessarily imply the existence of a structural attack.

In this context, our work goes in several directions. First, in Chapter 3, we revisited the distinguisher, proposing an alternative viewpoint, i.e. in terms of Schur's product and square of codes. While the distinguisher of [Fau+13] relies on credible heuristics, we rigorously proved upper bounds on the size of the square code of the dual of an alternant/Goppa code, which turns out to be smaller than that of a random code. Moreover, these upper bounds are tight for both random alternant and Goppa codes and for each field size, while [Fau+13] covers the non-binary Goppa codes case only empirically.

The big open problem, however, is to convert the distinguisher into an attack. The square code construction had previously been used to attack variants of McEliece's scheme based on generalized RS codes. More precisely, a filtration of Reed-Solomon codes of decreasing dimension was computed through the conductor code with respect to a product code. On the other hand, the subfield subcode structure of alternant and Goppa codes poses further difficulties and hinders a direct adaptation of such an attack. Thanks to a better understanding of the square code structure given in Chapter 3, we have been able to find a good conductor that leads to a filtration of random alternant codes of decreasing order. This is the first part of the cryptanalysis described in Chapter 4. Because of technical reasons, the filtration does not succeed when the underlying code is Goppa, though. In any case, the attack is not finished: we are left with an alternant code of small order, down to an order equal to field size, where the private structure, i.e. its support and multiplier, are still hidden. Once again, Gröbner bases techniques came to the aid. We exploited them to solve the algebraic system from [Fau+10a], which models the key-recovery problem for

alternant/Goppa codes. The key point was that decreasing the alternant code order made the system easier to be solved. Overall, our cryptanalysis is successful against binary or ternary random alternant codes. The small field size is needed in order to compute a filtration down to alternant code order 3, for which the algebraic attack becomes practical. Even if limited to the distinguishable regime, we thus have presented the first attack on unstructured alternant codes. Previous attacks only worked for alternant/Goppa codes with some additional properties that eventually and crucially revealed vulnerabilities. Within, this first achievement opens the road to future works. An obvious target is to adapt the attack to Goppa codes: such a result would break for instance the CFS signature. To this extent, the biggest hurdle consists in computing differently a filtration of Goppa codes or finding an alternative way to simplify the key-recovery problem. Moreover, the Gröbner basis algorithm would also need to be adjusted. On the other hand, even in the case of random alternant codes, it would be interesting to extend the algebraic cryptanalysis to orders higher than 3. This would allow to stop the filtration at a previous step and therefore attack codes with a larger field size.

The ultimate goal remains to corroborate or dismantle the assumption of indistinguishability of Goppa codes for rates compatible with the McEliece cryptosystem. In Chapter 5, we addressed this topic, trying to enhance the classical distinguisher. We have been able to decrease the minimal distinguishable rate in some cases, by keeping the procedure of polynomial complexity. More precisely, before computing the square of the dual of an alternant/Goppa code, we shorten it in some amount of positions. The distinguisher is still given by the dimension of the resulting square code. This strategy is more effective for random alternant codes rather than Goppa codes and for big rather than small field sizes. In particular, from our experiments, binary Goppa codes do not seem to be affected by any improvement. This is a very preliminary work, both in terms of the theoretical evidence provided and the magnitude of the amelioration. However, it clearly shows that the distinguishable rate barrier is not insurmountable. Could related ideas and more advanced techniques lead to an efficient algorithm capable of distinguishing Goppa codes used in Classic McEliece? This question draws vital lymph from the work presented in this manuscript. In the same way, as we have been able to devise a distinguisher-based attack after approximately a decade since [Fau+13] was published, a structural attack on McEliece's scheme could eventually emerge from an improved distinguisher.

Bibliography

[AGS11] Carlos Aguilar, Philippe Gaborit, and Julien Schrek. *A new zero-knowledge code based identification scheme with reduced communication.* In: *Proc. IEEE Inf. Theory Workshop- ITW 2011*. IEEE, Oct. 2011, pp. 648–652.

[Agu+18] Carlos Aguilar-Melchor, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. *Efficient Encryption From Random Quasi-Cyclic Codes.* In: *IEEE Trans. Inform. Theory* 64.5 (2018), pp. 3927–3943.

[Agu+20] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Maxime Bros, Alain Couvreur, Jean-Christophe Deneuville, Philippe Gaborit, Gilles Zémor, and Adrien Hauteville. *Rank Quasi Cyclic (RQC).* Second Round submission to NIST Post-Quantum Cryptography call. Apr. 2020.

[Agu+21] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, and Jurjen Bos. *HQC.* Round 3 Submission to the NIST Post-Quantum Cryptography Call. https://pqc-hqc.org/doc/hqc-specification_2021-06-06.pdf. June 2021.

[AK11] Mortuza Ali and Margreta Kuijper. *A Parametric Approach to List Decoding of Reed-Solomon Codes Using Interpolation.* In: *IEEE Transactions on Information Theory* 57.10 (2011), pp. 6718–6728.

[Ala+22] Gorjan Alagic, Daniel Apon, David Cooper, Quynh Dang, Thinh Dang, John Kelsey, Jacob Lichtinger, Carl Miller, Dustin Moody, Rene Peralta, et al. *Status report on the third round of the NIST post-quantum cryptography standardization process.* In: *US Department of Commerce, NIST* (2022).

[Alb+20] Martin Albrecht, Daniel J. Bernstein, Tung Chou, Carlos Cid, Jan Gilcher, Tanja Lange, Varun Maram, Ingo von Maurich, Rafael Mizoczki, Ruben Niederhagen, Edoardo Persichetti, Kenneth Paterson, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, Cen Jung Tjhai, Martin Tomlinson, and Wang Wen. *Classic McEliece (merger of Classic McEliece and NTS-KEM.* https://classic.mceliece.org. Third round finalist of the NIST post-quantum cryptography call. Oct. 2020.

[Ale03] Alekhnovich, Michael. *More on Average Case vs Approximation Complexity*. In: *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Cambridge, MA, USA, Proceedings*. IEEE Computer Society, 2003, pp. 298–307.

[Apo+20] Daniel Apon, Ray A. Perlner, Angela Robinson, and Paolo Santini. *Cryptanalysis of LEDAcrypt*. In: *Advances in Cryptology - CRYPTO 2020, Part III*. Ed. by Daniele Micciancio and Thomas Ristenpart. Vol. 12172. Lecture Notes in Computer Science. Springer, 2020, pp. 389–418.

[Ara+17] N. Aragon, P. Barreto, S. Bettaieb, Loic Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, S. Gueron, T. Güneysu, C. Aguilar Melchor, R. Misoczki, E. Persichetti, N. Sendrier, J.-P. Tillich, and G. Zémor. *BIKE*. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Ara+19] Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. *Durandal: a rank metric based signature scheme*. In: *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*. Vol. 11478. LNCS. Springer, 2019, pp. 728–758.

[Ara+21] Nicolas Aragon, Marco Baldi, Jean-Christophe Deneuville, Karan Khathuria, Edoardo Persichetti, and Paolo Santini. *Cryptanalysis of a code-based full-time signature*. In: *Designs, Codes and Cryptography* 89.9 (2021), pp. 2097–2112.

[Ars+04] Gwénolé Ars, Jean-Charles Faugère, Hiroshi Imai, Mitsuru Kawazoe, and Makoto Sugita. *Comparison Between XL and Gröbner Basis Algorithms*. In: *ASIACRYPT*. 2004.

[Bae+21] John Baena, Pierre Briaud, Daniel Cabarcas, Ray A. Perlner, Daniel Smith-Tone, and Javier A. Verbel. *Improving Support-Minors rank attacks: applications to GeMSS and Rainbow*. In: *IACR Cryptol. ePrint Arch., accepted for publication in CRYPTO 2022* (2021), p. 1677.

[Bal+11] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. *Enhanced public key security for the McEliece cryptosystem*. submitted. arxiv:1108.2462v2[cs.IT]. 2011.

[Bal+13] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. *Using LDGM Codes and Sparse Syndromes to Achieve Digital Signatures*. In: *Post-Quantum Cryptography 2013*. Vol. 7932. LNCS. Springer, 2013, pp. 1–15.

[Bal+19a] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. *A finite regime analysis of information set decoding algorithms*. In: *Algorithms* 12.10 (2019), p. 209.

[Bal+19b] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. *LEDAcrypt*. Second round submission to the NIST post-quantum cryptography call. Jan. 2019.

[Bal+21] Marco Baldi, Jean-Christophe Deneuville, Edoardo Persichetti, and Paolo Santini. *Cryptanalysis of a code-based signature scheme without trapdoors*. In: *Cryptology ePrint Archive* (2021).

[Ban+17] Gustavo Banegas, Paulo S.L.M Barreto, Brice Odilon Boidje, Pierre-Louis Cayrel, Gilbert Ndollane Dione, Kris Gaj, Cheikh Thiécoumba Gueye, Richard Haeussler, Jean Belo Klamti, Ousmane N'diaye, Duc Tri Nguyen, Edoardo Persichetti, and Jefferson E. Ricardini. *DAGS : Key Encapsulation for Dyadic GS Codes*. https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/DAGS.zip. First round submission to the NIST post-quantum cryptography call. Nov. 2017.

[Bar+05] Magali Bardet, Jean-Charles Faugère, Bruno Salvy, and Bo-Yin Yang. *Asymptotic expansion of the degree of regularity for semi-regular systems of equations*. In: *MEGA'05 – Effective Methods in Algebraic Geometry*. 2005, pp. 1–14.

[Bar+17] Magali Bardet, Élise Barelli, Olivier Blazy, Rodolfo Canto Torres, Alain Couvreur, Phillipe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich. *BIG QUAKE*. https://bigquake.inria.fr. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Bar+20a] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, Vincent Neiger, Olivier Ruatta, and Jean-Pierre Tillich. *An Algebraic Attack on Rank Metric Code-Based Cryptosystems*. In: *Advances in Cryptology - EUROCRYPT 2020 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020. Proceedings*. 2020.

[Bar+20b] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier Verbel. *Improvements of Algebraic Attacks for solving the Rank Decoding and MinRank problems*. In: *Advances in Cryptology - ASIACRYPT 2020, International Conference on the Theory and Application of Cryptology and Information Security, 2020. Proceedings*. 2020, pp. 507–536.

[Bar+21] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. *LESS-FM: fine-tuning signatures from the code equivalence problem*. In: *Internationa@MiscGrassl:codetables, author = "Grassl, Markus", title = "Bounds on the minimum distance of linear codes and quantum codes", howpublished = "Online available at http://www.codetables.de", year = "2007", note = "Accessed on 2022-09-27" l Conference on Post-Quantum Cryptography*. Springer. 2021, pp. 23–43.

[Bar04] Magali Bardet. *Étude des systèmes algébriques surdéterminés. Applications aux codes correcteurs et à la cryptographie*. http://tel.archives-ouvertes.fr/tel-00449609/en/. PhD thesis. Université Paris VI, Dec. 2004.

[Bar94] S Barg. *Some new NP-complete coding problems*. In: *Problemy Peredachi Informatsii* 30.3 (1994), pp. 23–28.

[BBC08] Marco Baldi, Marco Bodrato, and Franco Chiaraluce. *A New Analysis of the McEliece Cryptosystem Based on QC-LDPC Codes*. In: *Proceedings of the 6th international conference on Security and Cryptography for Networks*. SCN '08. Amalfi, Italy: Springer-Verlag, 2008, pp. 246–262.

[BBD] Daniel J Bernstein, Johannes Buchmann, and Erik Dahmen. *Post-Quantum Cryptography.–2009*. In: *DOI: https://doi. org/10.1007/978-3-540-88702-7* ().

[BC18] Élise Barelli and Alain Couvreur. *An Efficient Structural Attack on NIST Submission DAGS*. In: *Advances in Cryptology - ASIACRYPT'18*. Ed. by Thomas Peyrin and Steven Galbraith. Vol. 11272. LNCS. Springer, Dec. 2018, pp. 93–118.

[BCD22] Maxime Bombar, Alain Couvreur, and Thomas Debris–alazard. *On codes and learning with errors over function fields*. 2022.

[Bec+12] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. *Decoding Random Binary Linear Codes in $2^{n/20}$: How $1 + 1 = 0$ Improves Information Set Decoding*. In: *Advances in Cryptology - EUROCRYPT 2012*. LNCS. Springer, 2012.

[Bee+18] Peter Beelen, Martin Bossert, Sven Puchinger, and Johan Rosenkilde. *Structural Properties of Twisted Reed–Solomon Codes with Applications to Cryptography*. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. 2018, pp. 946–950.

[Ber+09] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani. *Reducing Key Length of the McEliece Cryptosystem*. In: *Progress in Cryptology - AFRICACRYPT 2009*. Ed. by Bart Preneel. Vol. 5580. LNCS. Gammarth, Tunisia, June 2009, pp. 77–97.

[Ber+19] Daniel J. Bernstein, Tung Chou, Tanja Lange, Ingo von Maurich, Rafael Mizoczki, Ruben Niederhagen, Edoardo Persichetti, Christiane Peters, Peter Schwabe, Nicolas Sendrier, Jakub Szefer, and Wang Wen. *Classic McEliece: conservative code-based cryptography*. https://classic.mceliece.org. Second round submission to the NIST post-quantum cryptography call. Mar. 2019.

[Ber00] Thierry P. Berger. *On the cyclicity of Goppa codes, parity-check subcodes of Goppa codes and extended Goppa codes*. In: *Finite Fields Appl.* 6.3 (2000), pp. 255–281.

[Beu22] Ward Beullens. *Breaking Rainbow Takes a Weekend on a Laptop*. In: *Advances in Cryptology - CRYPTO 2022*. LNCS. Springer-Verlag, 2022.

[BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. *On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations*. In: *Proceedings of the International Conference on Polynomial System Solving*. 2004, pp. 71–74.

[BFS15] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy. *On the complexity of the F_5 Gröbner basis algorithm*. In: *J. Symbolic Comput.* 70 (2015), pp. 49–70.

[Bid+22] Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Victor Mateu. *Code-based Signatures from New Proofs of Knowledge for the Syndrome Decoding Problem*. In: *arXiv preprint arXiv:2201.05403* (2022).

[BL05] Thierry P. Berger and Pierre Loidreau. *How to Mask the Structure of Codes for a Cryptographic Use*. In: *Des. Codes Cryptogr.* 35.1 (2005), pp. 63–79.

[BL11] Andrej Bogdanov and Chin Ho Lee. *Homomorphic encryption from codes*. In: *arXiv preprint arXiv:1111.4301* (2011).

[BLM11] Paulo Barreto, Richard Lindner, and Rafael Misoczki. *Monoidic codes in cryptography*. In: *Post-Quantum Cryptography 2011*. Vol. 7071. LNCS. Springer, 2011, pp. 179–199.

[BLP10] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. *Wild McEliece*. In: *Selected Areas in Cryptography*. Ed. by Alex Biryukov, Guang Gong, and Douglas R. Stinson. Vol. 6544. LNCS. 2010, pp. 143–158.

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. *Wild McEliece Incognito*. In: *Post-Quantum Cryptography 2011*. Ed. by Bo-Yin Yang. Vol. 7071. LNCS. Springer Berlin Heidelberg, 2011, pp. 244–254.

[BMS11] Paulo S.L.M Barreto, Rafael Misoczki, and Marcos A. Jr. Simplicio. *One-time signature scheme from syndrome decoding over generic error-correcting codes*. In: *Journal of Systems and Software* 84.2 (2011), pp. 198–204.

[BMT21] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. *Decoding Reed-Solomon codes by solving a bilinear system with a Gröbner basis approach*. In: *2021 IEEE International Symposium on Information Theory (ISIT)*. Melbourne, Australia, July 2021, pp. 872–877.

[BMT23] Magali Bardet, Rocco Mora, and Jean-Pierre Tillich. *Polynomial time key-recovery attack on high rate random alternant codes*. In: *preprint* (2023).

[BMT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. *On the inherent intractability of certain coding problems*. In: *IEEE Trans. Inform. Theory* 24.3 (May 1978), pp. 384–386.

[Buc65] Bruno Buchberger. *Ein Algorithmus zum Auffinden der Basiselemente des Restklassenringes nach einem nulldimensionalen Polynomideal*. PhD thesis. Universitat Innsbruck, 1965.

[Car+22] Kevin Carrier, Thomas Debris-Alazard, Charles Meyer-Hilfiger, and Jean-Pierre Tillich. *Statistical Decoding 2.0: Reducing Decoding to LPN*. In: *Advances in Cryptology - ASIACRYPT 2022*. LNCS. Springer, 2022.

[Cas+15] Igniacio Cascudo, Ronald Cramer, Diego Mirandola, and Gilles Zémor. *Squares of Random Linear Codes*. In: *IEEE Trans. Inform. Theory* 61.3 (Mar. 2015), pp. 1159–1173. ISSN: 0018-9448.

[CD22] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH (preliminary version)*. In: *Cryptology ePrint Archive* (2022).

[CDE21] André Chailloux, Thomas Debris-Alazard, and Simona Etinski. *Classical and Quantum Algorithms for Generic Syndrome Decoding Problems and Applications to the Lee Metric.* In: *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20-22, 2021, Proceedings.* Ed. by Jung Hee Cheon and Jean-Pierre Tillich. Vol. 12841. Lecture Notes in Computer Science. Springer, 2021, pp. 44–62.

[CF17] Edern Christian and Jean-Charles Faugère. *A survey on signature-based algorithms for computing Gröbner bases.* In: *J. Symbolic Comput.* 80 (2017), pp. 719–784.

[CFS01] Nicolas Courtois, Matthieu Finiasz, and Nicolas Sendrier. *How to Achieve a McEliece-based Digital Signature Scheme.* In: *Advances in Cryptology - ASIACRYPT 2001.* Vol. 2248. LNCS. Gold Coast, Australia: Springer, 2001, pp. 157–174.

[Cha+20] Melissa Chase, David Derler, Steven Goldfeder, Jonathan Katz, Vladimir Kolesnikov, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Xiao Wang, et al. *The picnic signature scheme.* 2020.

[Cho+15] Muhammad FI Chowdhury, Claude-Pierre Jeannerod, Vincent Neiger, Eric Schost, and Gilles Villard. *Faster algorithms for multivariate interpolation with multiplicities and simultaneous polynomial approximations.* In: *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2370–2387.

[CL22] Alain Couvreur and Matthieu Lequesne. *On the Security of Subspace Subcodes of Reed–Solomon Codes for Public Key Encryption.* In: *IEEE Trans. Inform. Theory* 68.1 (2022), pp. 632–648.

[CLO15] David Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and algorithms: an Introduction to Computational Algebraic Geometry and Commutative Algebra.* Undergraduate Texts in Mathematics, Springer-Verlag, New York., 2015. ISBN: 978-3-319-16720-6.

[CLT19] Alain Couvreur, Matthieu Lequesne, and Jean-Pierre Tillich. *Recovering short secret keys of RLCE in polynomial time.* In: *Post-Quantum Cryptography 2019.* Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. LNCS. Chongquing, China: Springer, May 2019, pp. 133–152.

[CMP14] Alain Couvreur, Irene Márquez–Corbella, and Ruud Pellikaan. *A polynomial time attack against algebraic geometry code based public key cryptosystems.* In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014.* June 2014, pp. 1446–1450.

[CMP17] Alain Couvreur, Irene Márquez–Corbella, and Ruud Pellikaan. *Cryptanalysis of McEliece Cryptosystem Based on Algebraic Geometry Codes and Their Subcodes.* In: *IEEE Trans. Inform. Theory* 63.8 (Aug. 2017), pp. 5404–5418.

[COT14a] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. *New Identities Relating Wild Goppa Codes.* In: *Finite Fields Appl.* 29 (2014), pp. 178–197.

[COT14b] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. *Polynomial Time Attack on Wild McEliece over Quadratic Extensions*. In: *Advances in Cryptology - EUROCRYPT 2014*. Ed. by Phong Q. Nguyen and Elisabeth Oswald. Vol. 8441. LNCS. Springer Berlin Heidelberg, 2014, pp. 17–39.

[COT17] Alain Couvreur, Ayoub Otmani, and Jean–Pierre Tillich. *Polynomial Time Attack on Wild McEliece over Quadratic Extensions*. In: *IEEE Trans. Inform. Theory* 63.1 (Jan. 2017), pp. 404–427. ISSN: 0018-9448.

[Cou+00] Nicolas Courtois, Alexander Klimov, Jacques Patarin, and Adi Shamir. *Efficient Algorithms for Solving Overdefined Systems of Multivariate Polynomial Equations*. In: *Advances in Cryptology - EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 392–407. ISBN: 978-3-540-45539-4.

[Cou+13] Alain Couvreur, Philippe Gaborit, Valérie Gautier, Ayoub Otmani, and Jean-Pierre Tillich. *Distinguisher-Based Attacks on Public-Key Cryptosystems Using Reed-Solomon Codes*. In: *International Workshop on Coding and Cryptography - WCC 2013*. Bergen, Norway, Apr. 2013, pp. 181–193.

[Cou+14] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich. *Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes*. In: *Des. Codes Cryptogr.* 73.2 (2014), pp. 641–666.

[Cou19] Alain Couvreur. *Codes algébriques et géométriques, applications à la cryptographie et à l'information quantique*. Accreditation to supervise research. Université Paris Diderot, Dec. 2019.

[COV07] Pierre-Louis Cayrel, Ayoub Otmani, and Damien Vergnaud. *On Kabatianskii-Krouk-Smeets Signatures*. In: *Arithmetic of Finite Fields - WAIFI 2007*. Vol. 4547. LNCS. Madrid, Spain, June 2007, pp. 237–251.

[CR21] Alain Couvreur and Hugues Randriambololona. *Algebraic geometry codes and some applications*. In: *A concise encyclopedia of coding theory*. CRC press, 2021. Chap. 15, pp. 307–361.

[CS96] Florent Chabaud and Jacques Stern. *The Cryptographic Security of the Syndrome Decoding Problem for Rank Distance Codes*. In: *Advances in Cryptology - ASIACRYPT 1996*. Vol. 1163. LNCS. Kyongju, Korea: Springer, Nov. 1996, pp. 368–381.

[DDW22] Thomas Debris-Alazard, Leo Ducas, and Wessel PJ van Woerden. *An Algorithmic Reduction Theory for Binary Codes: LLL and more*. In: *IEEE Transactions on Information Theory* 68.5 (2022), pp. 3426–3444.

[Deb+22] Thomas Debris-Alazard, Léo Ducas, Nicolas Resch, and Jean-Pierre Tillich. *Smoothing Codes and Lattices: Systematic Study and New Bounds*. In: *CoRR* abs/2205.10552 (2022). arXiv: 2205.10552.

[Deb19] Thomas Debris-Alazard. *Cryptographie fondée sur les codes : nouvelles approches pour constructions et preuves ; contribution en cryptanalyse*. Theses. Sorbonne Université, Dec. 2019.

[Del75] Philippe Delsarte. *On subfield subcodes of modified Reed-Solomon codes.* In: *IEEE Trans. Inform. Theory* 21.5 (1975), pp. 575–576.

[DH76] Whitfield Diffie and Martin Hellman. *New directions in cryptography.* In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. *Wave: A New Family of Trapdoor One-Way Preimage Sampleable Functions Based on Codes.* In: *Advances in Cryptology - ASIACRYPT 2019.* LNCS. Kobe, Japan: Springer, Dec. 2019.

[DT18] Thomas Debris-Alazard and Jean-Pierre Tillich. *A polynomial attack on a NIST proposal: RankSign, a code-based signature in rank metric.* preprint. IACR Cryptology ePrint Archive. Apr. 2018.

[Dum91] Ilya Dumer. *On minimum distance decoding of linear codes.* In: *Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory.* Moscow, 1991, pp. 50–52.

[Dür87] Arne Dür. *The automorphism groups of Reed-Solomon codes.* In: *Journal of Combinatorial Theory, Series A* 44 (1987), pp. 69–82.

[Fau+10a] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. *A Distinguisher for High Rate McEliece Cryptosystems.* IACR Cryptology ePrint Archive, Report2010/331. http://eprint.iacr.org/. 2010.

[Fau+10b] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. *Algebraic Cryptanalysis of McEliece Variants with Compact Keys.* In: *Advances in Cryptology - EUROCRYPT 2010.* Vol. 6110. LNCS. 2010, pp. 279–298.

[Fau+11] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. *A Distinguisher for High Rate McEliece Cryptosystems.* In: *Proc. IEEE Inf. Theory Workshop- ITW 2011.* Paraty, Brasil, Oct. 2011, pp. 282–286.

[Fau+13] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich. *A Distinguisher for High Rate McEliece Cryptosystems.* In: *IEEE Trans. Inform. Theory* 59.10 (Oct. 2013), pp. 6830–6844.

[Fau+14a] Jean-Charles Faugère, Pierrick Gaudry, Louise Huot, and Guénaël Renault. *Sub-cubic Change of Ordering for Gröner Basis: A Probabilistic Approach.* In: *ISSAC '14 - 39th International Symposium on Symbolic and Algebraic Computation.* Kobe, Japan: ACM, July 2014, pp. 170–177.

[Fau+14b] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. *Structural weakness of compact variants of the McEliece cryptosystem.* In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014.* Honolulu, HI, USA, July 2014, pp. 1717–1721.

[Fau+93] Jean-Charles Faugère, Patrizia M. Gianni, Daniel Lazard, and Teo Mora. *Efficient Computation of Zero-Dimensional Gröbner Bases by Change of Ordering.* In: *J. Symbolic Comput.* 16.4 (1993), pp. 329–344.

[Fau02] Jean-Charles Faugère. *A New Efficient Algorithm for Computing Gröbner Bases without Reduction to Zero: F5*. In: *Proceedings ISSAC'02*. ACM press, 2002, pp. 75–83.

[Fau99] Jean-Charles Faugère. *A New Efficient Algorithm for Computing Gröbner Bases (F4)*. In: *J. Pure Appl. Algebra* 139.1-3 (1999), pp. 61–88.

[FJR21] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. *Shared Permutation for Syndrome Decoding: New Zero-Knowledge Protocol and Code-Based Signature*. In: *IACR Cryptol. ePrint Arch.* (2021), p. 1576.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. *Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs*. In: *IACR Cryptol. ePrint Arch.* (2022), p. 188.

[FL10] Jean-Charles Faugère and Sylvain Lachartre. *Parallel Gaussian Elimination for Gröbner bases computations in finite fields*. In: *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation*. 2010, pp. 89–97.

[FPP14] Jean-Charles Faugère, Ludovic Perret, and Frédéric de Portzamparc. *Algebraic Attack against Variants of McEliece with Goppa Polynomial of a Special Form*. In: *Advances in Cryptology - ASIACRYPT 2014*. Vol. 8873. LNCS. Kaoshiung, Taiwan, R.O.C.: Springer, Dec. 2014, pp. 21–41.

[Frö85] Ralf Fröberg. *An inequality for Hilbert series of graded algebras*. In: *Mathematica Scandinavica* 56.2 (1985), pp. 117–144.

[FS87] Amos Fiat and Adi Shamir. *How to Prove Yourself: Practical Solutions to Identification and Signature Problems*. In: *Advances in Cryptology - CRYPTO '86*. Ed. by A.M. Odlyzko. Vol. 263. LNCS. Springer, 1987, pp. 186–194.

[FSS11] Jean-Charles Faugère, Mohab Safey El Din, and Pierre-Jean Spaenlehauer. *Gröbner bases of bihomogeneous ideals generated by polynomials of bidegree (1,1): Algorithms and complexity*. In: *J. Symbolic Comput.* 46.4 (2011), pp. 406–437.

[Fuk+17] Kazuhide Fukushima, Partha Sarathi Roy, Rui Xu, Shinsaku Kiyomoto, Kirill Morozov, and Tsuyoshi Takagi. *RaCoSS (Random Code-based Signature Scheme)*. First round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[Gab+13] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. *Low Rank Parity Check codes and their application to cryptography*. In: *Proceedings of the Workshop on Coding and Cryptography WCC'2013*. Bergen, Norway, 2013.

[Gab+14] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. *RankSign: An Efficient Signature Algorithm Based on the Rank Metric (extended version on arXiv)*. In: *Post-Quantum Cryptography 2014*. Vol. 8772. LNCS. Springer, 2014, pp. 88–107.

[Gab+16] Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. *Identity-based Encryption from Rank Metric*. IACR Cryptology ePrint Archive, Report2017/623. http://eprint.iacr.org/. May 2016.

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes*. Cambridge, Massachusetts: M.I.T. Press, 1963.

[Gao03] Shuhong Gao. *A New Algorithm for Decoding Reed-Solomon Codes*. In: *Communications, Information and Network Security*. Ed. by Vijay K. Bhargava, H. Vincent Poor, Vahid Tarokh, and Seokho Yoon. Boston, MA: Springer US, 2003, pp. 55–68. ISBN: 978-1-4757-3789-9.

[Gen09] Craig Gentry. *Fully homomorphic encryption using ideal lattices*. In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 169–178.

[Gio+91] Alessandro Giovini, Teo Mora, Gianfranco Niesi, Lorenzo Robbiano, and Carlo Traverso. *"One sugar cube, please" or selection strategies in the Buchberger algorithm*. In: *Proceedings of the 1991 international symposium on Symbolic and algebraic computation*. 1991, pp. 49–54.

[GL09] Valérie Gauthier-Umaña and Gregor Leander. *Practical Key Recovery Attacks On Two McEliece Variants*. IACR Cryptology ePrint Archive, Report2009/509. 2009.

[Gli+14] Danilo Gligoroski, Simona Samardjiska, Håkon Jacobsen, and Sergey Bezzateev. *McEliece in the world of Escher*. IACR Cryptology ePrint Archive, Report2014/360. http://eprint.iacr.org/. 2014.

[Gop71] Valerii D. Goppa. *Rational representation of codes and (L, g) codes*. In: *Problemy Peredachi Informatsii* 7.3 (1971). In Russian, pp. 41–49.

[GPS22] Shay Gueron, Edoardo Persichetti, and Paolo Santini. *Designing a Practical Code-Based Signature Scheme from Zero-Knowledge Proofs with Trusted Setup*. In: *Cryptography* 6.1 (2022), p. 5.

[GPT91] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. *Ideals over a non-commutative ring and their applications to cryptography*. In: *Advances in Cryptology - EUROCRYPT'91*. LNCS 547. Brighton, Apr. 1991, pp. 482–489.

[GPV08] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. *Trapdoors for hard lattices and new cryptographic constructions*. In: *Proceedings of the fortieth annual ACM symposium on Theory of computing*. ACM. 2008, pp. 197–206.

[GR06] Venkatesan Guruswami and Atri Rudra. *Explicit Capacity-achieving List-decodable Codes*. In: *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*. STOC '06. Seattle, WA, USA: ACM, 2006, pp. 1–10. ISBN: 1-59593-134-1.

[Gra07] Markus Grassl. *Bounds on the minimum distance of linear codes and quantum codes*. Online available at http://www.codetables.de. Accessed on 2022-09-27. 2007.

[GS12] Philippe Gaborit and Julien Schrek. *Efficient code-based one-time signature from automorphism groups with syndrome compatibility*. In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2012*. Cambridge, MA, USA, July 2012, pp. 1982–1986.

[GS98] Venkatesan Guruswami and Madhu Sudan. *Improved decoding of Reed–Solomon and algebraic-geometric codes*. In: *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*. IEEE. 1998, pp. 28–37.

[GS99] Venkatesan Guruswami and Madhu Sudan. *Improved decoding of Reed-Solomon and algebraic-geometry codes*. In: *IEEE Trans. Inform. Theory* 45.6 (1999), pp. 1757–1767.

[Han18] Antonia Wachter-Zeh Hannes Bartz. *Efficient decoding of interleaved subspace and Gabidulin codes beyond their unique decoding radius using Gröbner bases*. In: *Advances in Mathematics of Communications* 12.4 (2018), pp. 773–804.

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, Cambridge, 2003, pp. xviii+646. ISBN: 0-521-78280-5.

[HPS98] Jeffrey Hoffstein, Jill Pipher, and Joseph H. Silverman. *NTRU: A Ring-Based Public Key Cryptosystem*. In: *Algorithmic Number Theory, Third International Symposium, ANTS-III, Portland, Oregon, USA, June 21-25, 1998, Proceedings*. Ed. by Joe Buhler. Vol. 1423. LNCS. Springer, 1998, pp. 267–288.

[HW20] Anna-Lena Horlemann-Trautmann and Violetta Weger. *Information set decoding in the Lee metric with applications to cryptography*. In: *Advances in Mathematics of Communications* 0 (2020). online version, to appear. ISSN: 1930-5346.

[Ish+07] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. *Zero-knowledge from secure multiparty computation*. In: *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*. 2007, pp. 21–30.

[Jab01] Abdulrahman Al Jabri. *A statistical decoding algorithm for general linear block codes*. In: *Cryptography and coding. Proceedings of the 8^{th} IMA International Conference*. Ed. by Bahram Honary. Vol. 2260. LNCS. Cirencester, UK: Springer, Dec. 2001, pp. 1–8.

[JF11] David Jao and Luca De Feo. *Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies*. In: *International Workshop on Post-Quantum Cryptography*. Springer. 2011, pp. 19–34.

[JM96] Heeralal Janwa and Oscar Moreno. *McEliece Public Key Cryptosystems Using Algebraic-Geometric Codes*. In: *Des. Codes Cryptogr.* 8.3 (1996), pp. 293–307.

[Joz01] Richard Jozsa. *Quantum factoring, discrete logarithms, and the hidden subgroup problem*. In: *Computing in science & engineering* 3.2 (2001), pp. 34–43.

[KKS05] Gregory Kabatianskii, Evgenii Krouk, and Sergei Semenov. *Error Correcting Coding and Security for Data Networks: Analysis of the Superchannel Concept*. John Wiley & Sons, 2005.

[KKS97] Gregory Kabatianskii, Evgenii Krouk, and Ben. J. M. Smeets. *A Digital Signature Scheme Based on Random Error-Correcting Codes*. In: *IMA Int. Conf.* Vol. 1355. LNCS. Springer, 1997, pp. 161–167.

[KPG99] Aviad Kipnis, Jacques Patarin, and Louis Goubin. *Unbalanced oil and vinegar signature schemes*. In: *International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 1999, pp. 206–222.

[KRW19] Karan Khathuria, Joachim Rosenthal, and Violetta Weger. *Encryption Scheme Based on Expanded Reed–Solomon Codes*. In: *Adv. Math. Commun.* (2019). In Press.

[KT17] Ghazal Kachigar and Jean-Pierre Tillich. *Quantum Information Set Decoding Algorithms*. In: *Post-Quantum Cryptography 2017*. Vol. 10346. LNCS. Utrecht, The Netherlands: Springer, June 2017, pp. 69–89.

[Lam79] Leslie Lamport. *Constructing digital signatures from a one way function*. Tech. rep. CSL-98. SRI International, Oct. 1979.

[Laz83] D. Lazard. *Gröbner bases, Gaussian elimination and resolution of systems of algebraic equations*. In: *Computer algebra*. Vol. 162. LNCS. Proceedings Eurocal'83, London, 1983. Berlin: Springer, 1983, pp. 146–156.

[LDW94] Yuan Xing Li, Robert H. Deng, and Xin Mei Wang. *On the equivalence of McEliece's and Niederreiter's public-key cryptosystems*. In: *IEEE Trans. Inform. Theory* 40.1 (1994), pp. 271–273.

[Lee+17] Wijik Lee, Young-Sik Kim, Yong-Woo Lee, and Jong-Seon No. *Post quantum signature scheme based on modified Reed-Muller code pqsigRM*. First round submission to the NIST post-quantum cryptography call. NIST Round 1 submission for Post-Quantum Cryptography. Nov. 2017.

[LJ12] Carl Löndahl and Thomas Johansson. *A New Version of McEliece PKC Based on Convolutional Codes*. In: *Information and Communications Security, ICICS*. Vol. 7168. LNCS. Springer, 2012, pp. 461–470.

[LO06] Kwankyu Lee and Michael E. O'Sullivan. *An Interpolation Algorithm using Gröbner Bases for Soft-Decision Decoding of Reed-Solomon Codes*. In: *2006 IEEE International Symposium on Information Theory*. 2006, pp. 2032–2036.

[LO08] Kwankyu Lee and Michael E. O'Sullivan. *List decoding of Reed–Solomon codes from a Gröbner basis perspective*. In: *Journal of Symbolic Computation* 43.9 (2008), pp. 645–658. ISSN: 0747-7171.

[LR20] Julien Lavauzelle and Julian Renner. *Cryptanalysis of a system based on twisted Reed–Solomon codes*. In: *Designs, Codes and Cryptography* 88.7 (2020), pp. 1285–1300.

[LS01] Pierre Loidreau and Nicolas Sendrier. *Weak keys in the McEliece public-key cryptosystem*. In: *IEEE Trans. Inform. Theory* 47.3 (2001), pp. 1207–1211.

[LT13] Grégory Landais and Jean-Pierre Tillich. *An efficient attack of a McEliece cryptosystem variant based on convolutional codes*. In: *Post-Quantum Cryptography'13*. Ed. by P. Gaborit. Vol. 7932. LNCS. Springer, June 2013, pp. 102–117.

[LT20] Terry Shue Chien Lau and Chik How Tan. *MURAVE: A New Rank Code-Based Signature with MUltiple RAnk VErification*. In: *Code-Based Cryptography Workshop*. Springer. 2020, pp. 94–116.

[LX04] San Ling and Chaoping Xing. *Coding theory: a first course*. Cambridge University Press, 2004.

[LXY20] Zhe Li, Chaoping Xing, and Sze Ling Yeo. *A new code based signature scheme without trapdoors*. In: *Cryptology ePrint Archive* (2020).

[Lyu09] Vadim Lyubashevsky. *Fiat-Shamir with aborts: Applications to lattice and factoring-based signatures*. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2009, pp. 598–616.

[Mac94] Francis Sowerby Macaulay. *The algebraic theory of modular systems*. Vol. 19. Cambridge University Press, 1994.

[MB09] Rafael Misoczki and Paulo Barreto. *Compact McEliece Keys from Goppa Codes*. In: *Selected Areas in Cryptography*. Calgary, Canada, Aug. 2009.

[McE78] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*. In: DSN Progress Report 44. Jet Propulsion Lab, 1978, pp. 114–116.

[MI88] Tsutomu Matsumoto and Hideki Imai. *Public quadratic polynomial-tuples for efficient signature-verification and message-encryption*. In: *Workshop on the Theory and Application of of Cryptographic Techniques*. Springer. 1988, pp. 419–453.

[Mis+13] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. *MDPC-McEliece: New McEliece variants from Moderate Density Parity-Check codes*. In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. 2013, pp. 2069–2073.

[MM22] Luciano Maino and Chloe Martindale. *An attack on SIDH with arbitrary starting curve*. In: *Cryptology ePrint Archive* (2022).

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae. *Decoding random linear codes in $O(2^{0.054n})$*. In: *Advances in Cryptology - ASIACRYPT 2011*. Ed. by Dong Hoon Lee and Xiaoyun Wang. Vol. 7073. LNCS. Springer, 2011, pp. 107–124.

[MO15] Alexander May and Ilya Ozerov. *On Computing Nearest Neighbors with Applications to Decoding of Binary Linear Codes*. In: *Advances in Cryptology - EUROCRYPT 2015*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 203–228.

[MP12] Irene Márquez-Corbella and Ruud Pellikaan. *Error-correcting pairs for a public-key cryptosystem*. CBC 2012, Code-based Cryptography Workshop. Available on http://www.win.tue.nl/~ruudp/paper/59.pdf. 2012.

[MP16] Dustin Moody and Ray A. Perlner. *Vulnerabilities of "McEliece in the World of Escher"*. In: *Post-Quantum Cryptography 2016*. LNCS. Springer, 2016.

[MRA00] Chris Monico, Joachim Rosenthal, and Amin A. Shokrollahi. *Using low density parity check codes in the McEliece cryptosystem*. In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. Sorrento, Italy, 2000, p. 215.

[MS07] Lorenz Minder and Amin Shokrollahi. *Cryptanalysis of the Sidelnikov cryptosystem*. In: *Advances in Cryptology - EUROCRYPT 2007*. Vol. 4515. LNCS. Barcelona, Spain, 2007, pp. 347–360.

[MS86] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. Fifth. Amsterdam: North–Holland, 1986.

[MT22] Rocco Mora and Jean-Pierre Tillich. *On the dimension and structure of the square of the dual of a Goppa code*. In: *Designs, Codes and Cryptography* (2022), pp. 1–22.

[NC02] Michael A Nielsen and Isaac Chuang. *Quantum computation and quantum information*. 2002.

[Nie14] Johan Sebastian Rosenkilde Nielsen. *Power Decoding of Reed-Solomon Codes Revisited*. In: *Coding Theory and Applications, 4th International Castle Meeting, ICMCTA 2014, Palmela Castle, Portugal, September 15-18, 2014*. Ed. by Raquel Pinto, Paula Rocha Malonek, and Paolo Vettori. Vol. 3. CIM Series in Mathematical Sciences. Springer, 2014, pp. 297–305.

[Nie18] Johan Sebastian Rosenkilde Nielsen. *Power decoding Reed-Solomon codes up to the Johnson radius*. In: *Advances in Mathematics of Communications* 12.1 (2018), p. 81.

[Nie86] Harald Niederreiter. *Knapsack-type cryptosystems and algebraic coding theory*. In: *Problems of Control and Information Theory* 15.2 (1986), pp. 159–166.

[OJ02] Alexei V. Ourivski and Thomas Johansson. *New Technique for Decoding Codes in the Rank Metric and Its Cryptography Applications*. English. In: *Problems of Information Transmission* 38.3 (2002), pp. 237–246. ISSN: 0032-9460.

[OT11] Ayoub Otmani and Jean-Pierre Tillich. *An Efficient Attack on All Concrete KKS Proposals*. In: *Post-Quantum Cryptography 2011*. Vol. 7071. LNCS. 2011, pp. 98–116.

[OTD08] Ayoub Otmani, Jean-Pierre Tillich, and Léonard Dallot. *Cryptanalysis of McEliece Cryptosystem Based on Quasi-Cyclic LDPC Codes*. In: *Proceedings of First International Conference on Symbolic Computation and Cryptography*. LMIB Beihang University. Beijing, China, Apr. 2008, pp. 69–81.

[Ove08] Raphael Overbeck. *Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes*. In: *J. Cryptology* 21.2 (2008), pp. 280–301.

[Pat75] N. Patterson. *The algebraic decoding of Goppa codes*. In: *IEEE Trans. Inform. Theory* 21.2 (1975), pp. 203–207.

[Pat95] Jacques Patarin. *Cryptanalysis of the Matsumoto and Imai public key scheme of Eurocrypt'88*. In: *Annual International Cryptology Conference*. Springer. 1995, pp. 248–261.

[Pat96] Jacques Patarin. *Hidden Fields Equations (HFE) and Isomorphisms of Polynomials (IP): Two New Families of Asymmetric Algorithms*. In: *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*. Ed. by Ueli M. Maurer. Vol. 1070. LNCS. Springer, 1996, pp. 33–48.

[Pat97] Jacques Patarin. *The oil and vinegar signature scheme*. In: *Dagstuhl Workshop on Cryptography September, 1997*. 1997.

[Pei14] Chris Peikert. *Lattice cryptography for the internet*. In: *International workshop on post-quantum cryptography*. Springer. 2014, pp. 197–219.

[Per18] Edoardo Persichetti. *Efficient one-time signatures from quasi-cyclic codes: A full treatment*. In: *Cryptography* 2.4 (2018), p. 30.

[Pra62] Eugene Prange. *The use of information sets in decoding cyclic codes*. In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9.

[PT16] Aurélie Phesso and Jean-Pierre Tillich. *An Efficient Attack on a Code-Based Signature Scheme*. In: *Post-Quantum Cryptography 2016*. Vol. 9606. LNCS. Fukuoka, Japan: Springer, Feb. 2016, pp. 86–103.

[Ran15] Hugues Randriambololona. *On products and powers of linear codes under componentwise multiplication*. In: *Algorithmic arithmetic, geometry, and coding theory*. Vol. 637. Contemp. Math. Amer. Math. Soc., Providence, RI, 2015, pp. 3–78.

[Ras13] Roohallah Rastaghi. *An Efficient CCA2-Secure Variant of the McEliece Cryptosystem in the Standard Model*. submitted. 2013.

[Reg05] Oded Regev. *On lattices, learning with errors, random linear codes, and cryptography*. In: *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, Baltimore, MD, USA, May 22-24, 2005*. 2005, pp. 84–93.

[Rob22] Damien Robert. *Breaking SIDH in polynomial time*. In: *Cryptology ePrint Archive* (2022).

[RS16] Johan Rosenkilde né Nielsen and Arne Storjohann. *Algorithms for simultaneous Padé approximations*. In: *Proceedings of the ACM on International Symposium on Symbolic and Algebraic Computation*. 2016, pp. 405–412.

[RS60] Irving S. Reed and Gustave Solomon. *Polynomial codes over certain finite fields*. In: *Journal of the society for industrial and applied mathematics* 8.2 (1960), pp. 300–304.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*. In: *Commun. ACM* 21.2 (1978), pp. 120–126.

[SBC19] Paolo Santini, Marco Baldi, and Franco Chiaraluce. *Cryptanalysis of a one-time code-based digital signature scheme*. In: *2019 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2019, pp. 2594–2598.

[Sen00] Nicolas Sendrier. *Finding the permutation between equivalent linear codes: The support splitting algorithm*. In: *IEEE Trans. Inform. Theory* 46.4 (2000), pp. 1193–1203.

[Sen10] Nicolas Sendrier. *On the use of structured codes in code based cryptography*. In: *Coding Theory and Cryptography III*. Ed. by L. Storme S. Nikova B. Preneel. The Royal Flemish Academy of Belgium for Science and the Arts. 2010, pp. 59–68.

[Sha48] Claude E. Shannon. *A Mathematical Theory of Communication*. In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/j.1538-7305.1948.tb01338.x.

[Sho94] Peter W. Shor. *Algorithms for quantum computation: Discrete logarithms and factoring*. In: *FOCS*. Ed. by S. Goldwasser. 1994, pp. 124–134.

[Sho97] Peter W. Shor. *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*. In: *SIAM J. Comput.* 26.5 (1997), pp. 1484–1509.

[Sid94] Vladimir Michilovich Sidelnikov. *A public-key cryptosytem based on Reed-Muller codes*. In: *Discrete Math. Appl.* 4.3 (1994), pp. 191–207.

[Son+20] Yongcheng Song, Xinyi Huang, Yi Mu, Wei Wu, and Huaxiong Wang. *A code-based signature scheme from the Lyubashevsky framework*. In: *Theoretical Computer Science* 835 (2020), pp. 15–30.

[Spa12] Pierre-Jean Spaenlenhauer. *Résolution de systèmes multi-homogènes et determinantiels*. PhD thesis. Univ. Pierre et Marie Curie- Paris 6, Oct. 2012.

[SS92] Vladimir Michilovich Sidelnikov and S.O. Shestakov. *On the insecurity of cryptosystems based on generalized Reed-Solomon codes*. In: *Discrete Math. Appl.* 1.4 (1992), pp. 439–444.

[SSB10] Georg Schmidt, Vladimir Sidorenko, and Martin Bossert. *Syndrome decoding of Reed-Solomon codes beyond half the minimum distance based on shift-register synthesis*. In: *IEEE Trans. Inf. Theory* 56.10 (2010), pp. 5245–5252.

[Ste88] Jacques Stern. *A method for finding codewords of small weight*. In: *Coding Theory and Applications*. Ed. by G. D. Cohen and J. Wolfmann. Vol. 388. LNCS. Springer, 1988, pp. 106–113.

[Ste93] Jacques Stern. *A New Identification Scheme Based on Syndrome Decoding*. In: *Advances in Cryptology - CRYPTO'93*. Ed. by D.R. Stinson. Vol. 773. LNCS. Springer, 1993, pp. 13–21.

[Sud97] Madhu Sudan. *Decoding of Reed–Solomon Codes beyond the Error–Correction Bound*. In: *J. Complexity* 13.1 (1997), pp. 180–193.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. *Efficient Key Recovery for All HFE Signature Variants*. In: *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part I*. Ed. by Tal Malkin and Chris Peikert. Vol. 12825. Lecture Notes in Computer Science. Springer, 2021, pp. 70–93.

[Tri10] Peter V. Trifonov. *Efficient Interpolation in the Guruswami–Sudan Algorithm*. In: *IEEE Transactions on Information Theory* 56.9 (2010), pp. 4341–4349.

[TV13] Ido Tal and Alexander Vardy. *How to Construct Polar Codes*. In: *IEEE Trans. Inform. Theory* 59.10 (2013), pp. 6562–6582.

[TVZ82] Michael A. Tsfasman, Sergei G. Vlăduţ, and T. Zink. *Modular curves, Shimura curves, and Goppa codes, better than Varshamov-Gilbert bound*. In: *Math. Nach.* 109.1 (1982), pp. 21–28.

[Var97] Alexander Vardy. *The Intractability of Computing the Minimum Distance of a Code*. In: *IEEE Trans. Inform. Theory* 43.6 (Nov. 1997), pp. 1757–1766.

[Vér96] Pascal Véron. *Improved identification schemes based on error-correcting codes*. In: *Appl. Algebra Eng. Commun. Comput.* 8.1 (1996), pp. 57–69.

[Wag02] David Wagner. *A generalized birthday problem*. In: *Advances in Cryptology - CRYPTO 2002*. Ed. by Moti Yung. Vol. 2442. LNCS. Springer, 2002, pp. 288–303. ISBN: 978-3-540-44050-5.

[WB86] Lloyd R Welch and Elwyn R Berlekamp. *Error correction for algebraic block codes*. US Patent 4,633,470. Dec. 1986.

[Wie06] Christian Wieschebrink. *Two NP-complete Problems in Coding Theory with an Application in Code Based Cryptography*. In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. 2006, pp. 1733–1737.

[Wie10] Christian Wieschebrink. *Cryptanalysis of the Niederreiter Public Key Scheme Based on GRS Subcodes*. In: *Post-Quantum Cryptography 2010*. Vol. 6061. LNCS. Springer, 2010, pp. 61–72.

[Xag18] Keita Xagawa. *Practical attack on racoss-r*. In: *Cryptology ePrint Archive* (2018).

[ZS10] Alexander Zeh and Christian Senger. *A link between Guruswami-Sudan's list-decoding and decoding of interleaved Reed-Solomon codes*. In: *2010 IEEE International Symposium on Information Theory*. 2010, pp. 1198–1202.