Sorbonne Université

École doctorale Informatique, Télécommunications et Électronique (Paris)

Inria / Équipe-projet COSMIQ

Analysis of code-based post-quantum cryptosystems

Thèse de doctorat d'informatique

présentée par

Matthieu Lequesne

dirigée par Nicolas Sendrier

soutenue publiquement le 25 mai 2021

devant un jury composé de

Magali Bardet	Université de Rouen Normandie	Examinatrice
Alain Couvreur	Inria	Examinateur
Philippe Gaborit	Université de Limoges	Rapporteur
Thomas Johansson	Lund University	Rapporteur
Antoine Joux	CISPA Helmholtz Center	Examinateur
Pierre Loidreau	DGA-MI et Université de Rennes 1	Examinateur
Nicolas Sendrier	Inria	Directeur
Annick Valibouze	Sorbonne Université	Examinatrice

Remerciements

Les travaux réunis dans ce manuscrit sont l'aboutissement de quatre années de travail, dont la dernière période a été marquée par la pandémie mondiale de Covid. L'épreuve particulièrement difficile du confinement aura au moins été l'occasion de confirmer, à quiconque en aurait douté, que le travail de recherche est avant tout une entreprise collective.

C'est pourquoi, avant de présenter mes résultats scientifiques, qui ne sont qu'une partie visible de l'aventure humaine que constitue la thèse, je tiens à remercier les nombreuses personnes qui m'ont accompagné, conseillé, soutenu, chacune à leur manière, ces dernières années. Que ce soit dans le cadre professionnel, associatif ou privé, j'ai eu la chance d'être entouré de personnes exceptionnelles. Ces travaux n'auraient jamais abouti sans leur soutien. Qu'elles en soient remerciées.

Mes premières pensées sont évidemment pour Nicolas, mon directeur de thèse. Le sujet que tu m'as proposé pour mon stage de master m'a tout de suite enthousiasmé et m'a permis de me plonger rapidement dans la cryptographie fondée sur les codes. J'ai ensuite eu tout le loisir d'explorer différentes dimensions de ce domaine de recherche. Je te remercie pour la confiance que tu m'as témoignée pendant ces quatre années. À chaque étape, tu t'es montré disponible pour me guider et répondre à mes questions avec un intérêt certain. Merci pour ta patience, ton sens du détail et ton calme à toute épreuve lors de nos discussions.

Je remercie vivement Alain avec qui j'ai pris beaucoup de plaisir à collaborer. Tu m'as transmis une vision nouvelle des attaques possibles sur les codes, qui se retrouve dans ce manuscrit. Malgré tes responsabilités croissantes, tu as toujours trouvé un créneau pour discuter d'une énième tentative (souvent infructueuse) de venir à bout d'un cryptosystème. Que ça soit à Paris, à Saclay, en visio (avec tes enfants) ou sur la terrasse virtuelle d'une plateforme pas très RGPD-compatible, nos échanges réguliers ont nourri l'ensemble de mon travail.

Je souhaite remercier les membres de mon jury, Annick Valibouze, Antoine Joux, Magali Bardet, Pierre Loidreau et tout particulièrement les rapporteurs Philippe Gaborit et Thomas Johansson qui ont pris le temps de lire entièrement

ce manuscrit. J'ai eu l'opportunité de vous croiser régulièrement ces dernières années, vos travaux respectifs ont souvent été une source d'inspiration pour moi et votre présence dans mon jury m'honore.

J'ai eu la chance de réaliser ma thèse entouré par une équipe soudée et dynamique de collègues avec qui j'ai pu partager bien plus qu'un environnement de travail. Je dois beaucoup à Anne, qui m'a dirigée vers Nicolas lorsque je cherchais un stage, m'ouvrant ainsi les portes de l'équipe Secret. Anne, je suis admiratif de ta capacité à combiner un emploi du temps de ministre, une activité scientifique de pointe et un engagement sans faille pour défendre les valeurs scientifiques et éthiques de l'équipe et de l'institut, tout en veillant avec attention au bien-être de tes collègues. Je te remercie également d'avoir toujours soutenu les activités de médiation, n'hésitant jamais à montrer l'exemple en présentant la cryptographie à des jeunes, avec l'enthousiasme communicatif qui te caractérise.

En tant que membre de l'équipe Secret, devenue Cosmiq, je dois également énormément à Jean-Pierre. En m'inscrivant à ton cours de théorie de l'information à l'X, je ne me doutais pas que j'allais passer plusieurs années à travailler à tes côtés. Merci pour tes conseils réguliers, tes remarques franches et toujours bienveillantes, qui incitent à persévérer pour produire un travail scientifique toujours plus rigoureux. Merci également pour ton travail d'animation du séminaire le plus select de Paris, qui est d'autant plus prestigieux qu'il est particulièrement difficile de savoir quand et où aura lieu la prochaine réunion.

J'ai évidemment une pensée particulière pour mes collègues du bureau "Tapdance", avant-poste de l'équipe dans sa conquête du deuxième étage : Valentin et Ferdinand, avec qui j'ai partagé quotidiennement ce bureau, mais aussi les personnes que nous avons temporairement accueillies, Mathilde, Daniel et Pierre, et qui ont chacunes pu contribuer à l'esprit Tapdance.

Je remercie chaleureusement Thomas, qui m'a épaulé dès mes premières semaines dans l'équipe, toujours avide de partager avec moi son intérêt pour les codes, et avec qui les longues discussions ont porté sur des sujets bien plus larges que notre travail académique. Merci à Yann et Xavier qui m'ont accompagné lors de nombreuses pauses café, pour parler de recherche, d'enseignement, de politique et de beaucoup d'autres choses.

Enfin je veux remercier l'ensemble de mes collègues que j'ai pu côtoyer au quotidien et qui font de l'équipe Secret/Cosmiq un cadre de travail particulièrement agréable. Travailler au sein de cette équipe m'a permis de développer des compétences transverses, telles que l'étude de la théologie byzantine ou la courses de canoë entre les crocodiles. Merci à André (le jeune), André (le sage), Andrea, Anthony, Antoine, Antonio, Augustin, Aurélie, Charles, Christina, Christophe, Clara, Clémence, Étienne, Gaëtan, Ivan, Johanna, Jules, Julia, Kaushik, Kevin, Léo, Lucien, María, Maxime, Nicolas, Pascale, Rémi,

Ritam, Rocco, Rodolfo, Sébastien, Shibam, Simona et Vivien. Cette liste ne saurait être exhaustive et ce message s'adresse à l'ensemble des collègues que j'ai croisé·e·s ces dernières années. J'ai une pensée particulière pour Christelle, toujours là pour nous accompagner patiemment dans les méandres administratifs et sans laquelle l'équipe cesserait instantanément de fonctionner. Enfin, je n'oublie pas Thierry, Étienne et Bernard, les verbicrucistes attitrés de l'équipe, y compris en temps de pandémie.

Je souhaite également saluer l'ensemble de mes collègues de la rue Simone Iff, les chercheur·e·s mais aussi les équipes de soutien à la recherche. J'ai évidemment une pensée pour les personnes du service médiation avec lesquelles j'ai régulièrement interagi sur des projets divers. Je remercie également mes collègues du CWI, qui m'ont accueilli alors que je terminais la rédaction de ce manuscrit et avec qui je me réjouis de pouvoir travailler.

Au-delà, du cadre professionnel, j'ai passé une partie significative de mon temps, avant et pendant cette thèse, à monter de nombreux projets associatifs. Je commence évidemment par remercier la grande famille Animath et ses innombrables bénévoles toujours heureux·ses de transmettre au plus grand nombre le plaisir qu'iels trouvent dans la pratique des mathématiques et de l'informatique. Je dois beaucoup à Martin, d'abord parce qu'Animath a joué un rôle important dans mon orientation vers des études en mathématiques, mais aussi et surtout pour la confiance qu'il m'a toujours exprimée, me permettant de mener à bien des projets toujours plus grands, tout en m'encourageant à pousser toujours plus loin mes études. Je remercie également Fabrice, avec lequel, je l'espère, nous avons réussi à faire de cet état d'esprit un marqueur fort de l'identité de l'association.

En remontant quelques années en arrière, je sais que je dois beaucoup à plusieurs personnes avec qui j'ai pris un plaisir particulier à organiser et développer les TFJM. Je remercie en particulier Bernardo, David, Igor, Giancarlo, Vincent et Joon, pour tous ces moments partagés, qui m'ont permis d'avoir assez tôt un aperçu du travail de doctorant et un avant-goût de la recherche. Plus récemment (quoique nous en soyons déjà à la 7e édition), nous avons lancé la grande aventure du concours Alkindi et je remercie Mathias, Razvan, Yann et Mélissa pour les longues séances de réflexion sur la manière la plus pédagogique de faire découvrir la cryptographie à des dizaines de milliers d'élèves. Je remercie au passage tou·te·s les collègues de la communauté française de crypto qui soutiennent le concours. Enfin, Animath a été l'occasion de monter de nombreuses autres actions, toujours plus audacieuses, que je ne saurais toutes citer ici. Je salue les centaines de bénévoles que j'ai croisé·e·s partout en France, et dont certain·e·s sont devenu·e·s des ami·e·s proches. Leur engagement collectif est une force incroyable et une source infinie de motivation.

Sur une autre dimension, bien que l'intersection ne soit pas vide, je remercie toutes les personnes que j'ai pu croiser lors des luttes et mobilisations, au sein de différents mouvements, collectifs et associations. Après de nombreuses années, je suis toujours surpris par la richesse des échanges et la force de l'intelligence collective. De vous tou·te·s, j'ai beaucoup appris et j'ai encore beaucoup à apprendre. À celles et ceux qui se reconnaîtront, sachez que vos actions sont une source d'inspiration et d'espoir. Je salue en particulier mes camarades de la Sphinx et des collectifs associés pour l'ensemble du travail que nous avons mené et dont l'ampleur actuelle dépasse nos espérances initiales.

Je salue mes commerçant·e·s préféré·e·s, Steph, Thierry, Emmy ainsi que Hans, pour leurs conseils toujours avisés et leurs produits de première qualité, qui ont toujours fait l'unanimité auprès de mes ami·e·s.

Merci à mes parents et ma sœur qui m'ont toujours soutenu et encouragé dans mes études.

Merci à tou·te·s les camarades de l'ASA, Clara, Denis, Geoffrey, José et les nombreux·ses autres, avec qui j'ai pu passer des soirées inoubliables et qui sont désormais disséminé·e·s aux quatre coins du globe.

Je remercie infiniment Aurore, Aymeric, Joseph, Léa et Roxane, pour leur présence à mes côtés pendant toute cette thèse et pour l'ensemble des moments partagés ces dix dernières années que je ne tenterai pas de résumer ici.

Enfin mes derniers mots vont à Guillaume, qui éclaire ma vie au quotidien. Merci pour ton soutien sans faille et tes encouragements permanents pendant la longue période de rédaction de ce manuscrit.

À toutes et tous, je vous souhaite le meilleur et j'ai hâte de pouvoir vous retrouver autour d'un café, d'une bière ou d'un whisky.

Contents

List of publications

Publications

[ELPS18] Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier. "QC-MDPC: A Timing Attack and a CCA2 KEM". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. 2018, pp. 47–76.

[LT18] Matthieu Lequesne and Jean-Pierre Tillich. "Attack on the Edon-K Key Encapsulation Mechanism". In: *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. 2018, pp. 981–985.

[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.

[BCDL19] Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. "Ternary Syndrome Decoding with Large Weights". In: *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. Lecture Notes in Computer Science. Springer, 2019, pp. 437–466.

Preprints

[CL20] Alain Couvreur and Matthieu Lequesne. *On the security of subspace subcodes of Reed-Solomon codes for public key encryption*. 2020.

Chapter 1

Introduction

Cryptography aims at ensuring the secrecy and security of communications. Although cryptography is nowadays considered as a part of computer science, the need for confidentiality of information appeared long before computers were invented. Hence, cryptography inherits a long history of practices which were gradually formalized and turned into a scientific field.

In modern-day cryptography, the security of the systems is expected to rely on the hardness of a small amount of well-studied mathematical problems. Code-based cryptography, which is the subject of this work, consists in proposing cryptographic schemes based on problems inherited from coding theory.

Before digging into the definition of code-based cryptography, let us introduce the general notions related to cryptography on one side, and coding theory on the other.

Contents

1.1 Introduction to cryptography

1.1.1 Early days of cryptography

With the appearance of the first written documents emerged a new problem: how to make sure that the information would be understandable only by the intended recipients? For centuries, the language itself was a barrier, since only a very small elite could read and write. In this sense, writing was already a way to make information incomprehensible. But some cases appeared where one wanted to restrict the information to a small group. The first known attempts to hide written information is attributed by David Kahn [Kah67, p. 71] to a scribe in Ancient Egypt (1900 BC) who replaced some hieroglyphs by others. But it seems that this was intended to make the religious content of the message more mysterious and intriguing rather than really secret. Still, this is a first example of a secret code. Note that, here, *secret code* is used in its common acceptation, a *code* meaning that some words are replaced by others in a predefined manner to hide the meaning. In the rest of this work, the word *code* will have a different definition (the mathematical definition of an error-correcting code).

The emergence of alphabetical languages allowed the use of the first *ciphers*, in the sense that the alphabet is used as a mathematical object on which one can apply transformations. The oldest example is the scytale, used in ancient Greece [Kah67, p. 82]. The scytale consists in a wooden cylinder around which one wraps a strip of parchment to write the message. One needs a cylinder of the same diameter to decrypt. An enemy would have to guess the dimensions of the cylinder. This operation exactly amounts to applying a transposition on the letters of the message. Another well known example of primitive cipher is attributed by Suetonius to Caesar [Sue21, 56]. In his private communications, the Roman emperor replaced each letter by the letter three positions further in the alphabet (A becomes D, B becomes E, etc.). Here, each letter is shifted by three positions, but the shift could be any arbitrary number. The number of shifts is the *secret key* of this very primitive substitution cipher.

These two examples of early ciphers were used in a military context. For centuries long, the military usage has been the principal use of cryptography. For this reason, the history of cryptography is often closely tied to the most significant military events.

With the first cryptosystems came the first attacks on cryptosystems, also known as *cryptanalysis*. Given the low security of the ciphers described above, one was certainly able to break them by exhaustively trying all the keys (there are 26 possible keys in the case of Caesar's cipher). But the first known systematic approach to break a cipher is due to the Arab scientist Al-Kindi. Al-Kindi was a philosopher, mathematician and physician living in Bagdad in the 9^{th}

century. In his book *Manuscript on Deciphering Cryptographic Messages* [Kin09], he describes the frequency analysis method to find the key of a text encrypted using Ceasar's cipher. This approach combines mathematics and the use of the structural properties of language to break the cryptosystem faster than with an exhaustive search.

Over the years, new cryptosystems were proposed, such as the popular Vigenère cipher, which can be seen as a generalisation of Ceasar's cipher. Scientists dedicated time and energy to try to break them [Kah67; Sin00]. It is interesting to note that one of the most prominent of them, the British scientist Charles Babbage, realised in the 19$^{\text{th}}$ century that it could be useful to automate most computations, hence giving birth to the concept of computers. Indeed, the security of cryptosystems depends on the ability of the enemy to perform quickly a large number of computations. In this sense, it is not surprising that the concept of computers first appeared as a tool for cryptanalysis.

This "classical" way of manually designing and applying cryptosystems continued until World War II, with interesting examples using transpositions and substitutions, such as the Playfair cryptosystem or the ADFGVX cryptosystem [Sin00]. The well-known Enigma machine, used by the German army during World War II, was the first large-scale use of an electro-mecanic device to perform the encryption and decryption operations. This opened the way for new family of cryptosystems, using more complex operations and a much larger key space. The role of Turing's cryptanalysis of Enigma in the outcome of the conflict proved the crucial need of a rigorous scientific approach in the design of cryptosystems and the study of their security.

1.1.2 Theorisation of cryptography

In the 19$^{\text{th}}$ century, the invention of the telegraph induced a radical change in the way messages are convoyed. This transformed the nature of the problem. Instead of having to ensure the security of a private conversation between two persons, one now had to think of cryptography as a way to ensure the privacy of all possible messages that could potentially be exchanged between two individuals using the telegraphic network. Moreover, before being transmitted, the messages are now encoded (whether using Morse code, Baudot code or, later, bit encoding). This new definition of cryptography required a more systematic, hence more abstract approach of the problem [Dur14].

Kerckhoffs' principle. The first step towards theorisation of cryptography is often attributed to the Dutch cryptographer August Kerckhoffs. In his essay *La Cryptographie Militaire, ou, Des chiffres usités en temps de guerre*, he lists some fundamental principles to design cryptosystems. The most famous one, known as Kerckhoffs's principle, states that the security of the system should not rely

on the hypothesis that the enemy ignores how the cryptosystem works [Ker83, p. 8]. Kerckhoffs design principle comes as a response to the military usage of cryptography. With Kerckhoffs' approach, the enemy may well capture a soldier and learn how the cryptosystem works. As the security relies in a small secret (the *key*) and not in the whole description of the system, if the key is compromised, one only needs to use another key, and not change the whole cryptosystem as it was the case before.

In his article *Communication Theory of Secrecy Systems* published in 1949 [Sha49] (though a classified version had already been published in 1945), Shannon explicitly expresses the need of abstraction. *"As a first step in the mathematical analysis of cryptography, it is necessary to idealize the situation suitably, and to define in a mathematically acceptable way what we shall mean by a secrecy system."* Influenced by his recent work on information theory [Sha48], Shannon considers the plaintext as a sequence of symbols, regardless of their meaning. Shannon reasserts Kerckhoff's principle and makes a strong distinction between *steganography*, which consists in concealing a message (using invisible ink or any method where the enemy does not know that the message exist), and *cryptography*, which he defines as "'*true' secrecy systems where the meaning of the message is concealed by cipher, code etc., although its existence is not hidden*". Shannon insists that contrary to steganography, cryptography is a technological problem.

One-time pad. In 1882, Miller proposed a cryptosystem corresponding to Ceasar's cipher but where one would change the shift number for each letter. Miller's definition included some warning when choosing the predefined list of numbers that would be used to shift each letter [Mil82]. *"The differences between such numbers must not be regular. When a shift-number has been applied, or used, it must be erased from the list and not used again."*

Miller's cipher was later rediscovered and patented by Vernam in 1919 and is better known as Vernam's cipher, or the *one-time pad*. This cipher is not very convenient, since it requires a secret key that is as long as the message itself. But Shannon proved that, for this cryptosystem, it is (mathematically) impossible to recover the message without knowing the public key [Sha49]. In modern terms, this system is *perfectly*, or *information-theoretically secure*. Moreover, Shannon proved that this property is only achievable if the key is at least as long as the message.

1.1.3 Modern cryptography

Computational security. While the general idea of cryptography is that it should be *impossible* to decrypt the ciphertext without the secret key, the one-time pad solution is not fit for practical usage (apart very special cases like

the Moscow–Washington hotline during the Cold War). Therefore, Shannon introduced the weaker notion of *computational security* (or *practical secrecy* in his own terms [Sha49]). Following his work, instead of designing systems where it would be *impossible* (in the sense of information theory) to decrypt without the secret key, modern cryptography (only) requires that attacking the cryptosystem is *computationally hard*. This means that one can design an algorithm to decrypt without the secret key, but this algorithm has exponential complexity and it requires a number of operations that is impossible to achieve, even with access to a lot of resources (think of a national intelligence agency) and a lot of time (hundreds of years).

In practice, the minimal attack cost (measured in number of operations) required to consider that a cryptosystem is secure evolves in time and depends of the threat model. One used to consider that 2^{80} operations is unreachable, but now the standard is 2^{128}, or even 2^{256} for a more conservative approach. The exponent is called the *security level*, or *number of security bits*. This can be thought as a parameter of a cryptosystem: the cryptosystem should be available in different sizes to match different security levels.

Hard problems. Modern-days cryptography relies on the notion of *hard problem*. This idea can again be attributed to Shannon: *"We may construct our cipher in such a way that breaking it is equivalent to* (*or requires at some point in the process*) *the solution of some problem known to be laborious"* [Sha49, p. 704]. The idea is that instead of considering the algorithmic description of the cryptosystem as a whole, and asking if it is secure as a whole, it is easier to narrow down the critical part. The principle of a security reduction is to say that *if an enemy is able to* (*efficiently*) *decrypt a message without the secret key, then this person is able to* (*efficiently*) *solve the problem* P, where P is a well-defined mathematical problem, unanimously regarded as hard to solve by the community of mathematicians and computer scientists. By this manner, we can design a large portfolio of cryptographic primitives (fitting different needs) relying on a very small number of mathematical problems. A strong attention is given to these problems, to make sure that they are indeed hard to solve.

Key distribution and secret sharing. In the 1960s, banks and companies started using computers. They needed to encrypt data to securely communicate between them. But cryptosystems required that the two parties had previously agreed on the value of a shared secret key. This key could not be transmitted using an insecure channel, hence it had to be convoyed physically by a trusted person. And one had to use a different key for each interlocutor. Hence, when the number of businesses using cryptography grew, the number

of keys to securely distribute grew quadratically. This key distribution procedure had a cost, and it soon became the main limitation of the civil use of cryptography.

In 1976, Diffie and Hellman (with the help of Merkle) proposed a new protocol to resolve this issue. They proved the following counter-intuitive result: two persons can agree on a shared secret within an entirely public discussion [DH76a]. With this scheme, people did not have to physically meet anymore to agree on a secret key. This drastically reduced the cost of key distribution. Today, the Diffie–Hellman secret sharing scheme is still widely used between computers and servers to agree on a secret key.

Public key cryptography. One remaining issue was that, before sending a private information to someone, one had to enter a (public) discussion to agree on a secret key. One could not just straightforwardly send encrypted data to a recipients without running the secret sharing procedure first. In another article, Diffie and Hellman imagined a way to overcome this [DH76b]. In previous cryptosystems, the secret key used to encrypt and to decrypt is the same. Diffie and Hellman proposed a family of cryptosystems where the encryption key is different from the decryption key. Hence, the encryption key can be made public (and is therefore called the *public key*). Of course, the decryption key should remain secret (and is called the *secret key*). In this setting, each person has a public key, associated with a secret key. Anyone willing to securely communicate with this person can use the public key to encrypt messages and send them. Only the owner of the secret key can decrypt. This is known as *asymetric* or *public-key* cryptography.

However, Diffie and Hellman's proposal was relying on the existence of a trapdoor one-way function, *i.e.* a function that is easy to compute in one direction and difficult to inverse, unless one knows a secret information (the trapdoor). But they could not find any example of a function with this property. In 1978, Rivest, Shamir and Adleman proposed to use exponentiation modulo a product of large prime numbers to instantiate the trapdoor function [RSA78]. This became the famous RSA cryptosystem, which is still widely used today.

The GCHQ later revealed that similar ideas had been published by British military cryptographers [Sin00, p. 279]. The principle of public-key cryptography was discovered by Ellis in 1969. Cocks proposed a cryptosystem similar to RSA in 1973. Finally, Williamson proposed a secret-sharing scheme in 1975. This information remained classified until 1997.

Standardisation. With the key distribution problem solved, any two individuals could suddenly communicate securely using any communication channel. Still, they need to use the same encryption method. Moreover, people who

are not experts in cryptography need advice to decide which cryptosystem they can safely use. For these reasons, it was decided that some cryptosystem would become standards. This is all the more important since, nowadays, cryptography does not only cover human to human interactions but the whole network of computers and servers constantly exchanging information over the internet. This could not be possible without common standards.

In 1975, America's National Bureau of Standard announced the standardisation of a first cryptosystem, known as the Data Encryption Standard (or DES). This symmetric cryptosystem was soon used by most businesses. But criticism from the academic community appeared, regarding the NSA's involvement in the design choice. Indeed, the choice of a standard is a sensitive issue, and some actors may be tempted to influence the decision towards a cryptosystem that they know how to break. For this reason, and to prevent suspicion, the recent standardisation procedures strongly involve the international academic community.

When DES became obsolete, America's National Institute of Standards and Technologies (NIST) announced in 1997 that it would organise an open competition to decide of its successor. Researchers were asked to submit different cryptosystems. After two rounds of competition, the new standard (AES) was announced in 2001. Another similar competition was organised by the NIST to standardise SHA3 in 2007. Recently, the NIST launched two standardisation procedures, one for post-quantum cryptography in 2016 and one for light-weight cryptography in 2018. Both are still ongoing.

Since the AES standardisation process, the academic community of cryptographers plays a key role in proposing and auditing cryptosystems for standardisation. Still, in 2013, Snowden revealed the existence of a backdoor in a pseudorandom number generator (Dual EC DRBG) standardised by the NIST and other international organisations. This proves that the transparency of the standardisation process is not enough to ensure the absence of external influence in the decision. Cryptographers should remain particularly careful that the decisions are only motivated by the will to offer the best possible security.

1.1.4 New trends in cryptography

Today, cryptography is widely used to ensure the *confidentiality* of communications, *i.e.* preventing an adversary to have access to the content of a message. But with the digitalisation of communication came two new classes of problems for cryptography to solve: *authenticity* and *integrity*. *Authenticity* is making sure that a message was issued by the right person. If a banker receives a transfer request, he needs to know that it was indeed issued by the account owner. The cryptographic primitives to solve this problem are called digital

signatures. *Integrity* is making sure that the data was not altered during the transmission and that the file that is received corresponds exactly to the file that was sent. Hash functions can be used to detect if someone has tempered with a message.

1.1.4.1 New challenges

The development of internet in the 21st century gave birth to plenty of new applications of computer science, each with specific constraints. Cryptography has to adapt to ensure confidentiality, authenticity and integrity in all contexts. Here is a short and non-exhaustive list of challenges that 21st century cryptography has to face.

Light-weight cryptography. Microprocessors are now everywhere: whether in access badges or connected objects. These objects require cryptographic algorithms to run on very small circuits with limited energy consumption. Think of a pacemaker: it should achieve the highest security level, but one cannot expect to change the batteries every week. Therefore, cryptographers have to come up with specially designed primitives to use the minimal possible amount of resources. This is called light-weight cryptography.

Privacy-preserving computation. The amount of data gathered by computers grows exponentially. Exploiting these data can yield tremendous results, for instance for biomedical research. But this should not be achieved at the cost of a loss of confidentiality. For this reason, cryptographers came up with the concept of homomorphic encryption, which is an idealised solution to this problem. The basic concept is that one should be able to collect encrypted data from different sources, perform a computation with the encrypted data, and only decrypt the result of the computation, without ever having access to the input. In practice, this property is difficult to achieve with reasonable efficiency, but research in the last decade made a lot of progress in this direction.

Multiparty computation. Nowadays, most of our communications involve multiple users and the collaborations are not necessarily bilateral. Therefore, one has to design cryptosystem involving multiple users, such that the system remains secure even if some of these users are malicious. In this paradigm, the enemy is not external but is part of the legitimate users of the system. This field is known as secure multiparty computation.

Quantum computers. Finally, the potential existence of (large and reliable) quantum computers in a few decades impacts the field of cryptography. On

one hand, properties of quantum physics (such as the no-cloning theorem) could be used to design cryptosystem with properties that can not be achieved by classical systems. Quantum key distribution could provide a solution to the key distribution problem that is information-theoretically secure. On the other hand, the novel properties of quantum computing could be used to attack the classical cryptosystems. Therefore, cryptographers have to come up with cryptosystems (for classical computers) that remain secure, even if the enemy has access to a quantum computer. This particular issue is developed in the next section.

1.1.4.2 Post-quantum cryptography

Today, asymetric cryptography is widely used and relies on the hardness of two mathematical problems: the discrete logarithm problem (for Diffie–Hellman secret sharing scheme [DH76a]) and the factorization of a product of two large prime numbers (for RSA [RSA78]). These two problems come from number theory and can be seen as particular instances of a larger problem, known as the hidden subgroup problem [Joz01].

In an article published in 1994, Shor proved that a quantum computer could solve this problem in polynomial time [Sho94]. This means that if an enemy one manages to build a (large and reliable) quantum computer, one can break Diffie-Hellman and RSA. This is a huge threat for cryptography. Fortunately, such large quantum computers do not exist. But in the last decade, several companies (Google, Microsoft, IBM, etc.) launched important research programs to develop quantum computers, while academic research keeps making progress. There is no certainty regarding the fact that large quantum computers will ever reach a state where they can be used to break RSA, but it this were to happen, it would have tremendous consequences. Today, researchers argue that there is a possibility that this would happen within the next decades. Even if the probability is small, the risk is too high not to be taken into account.

Moreover, there are two other factors to have in mind. First, the complexity of classical (non-quantum) algorithms to solve the factorisation problem has been improved in the last decade. Hence, independently of the quantum threat, it is important to have cryptosystems relying on different mathematical problems and not only factorisation. Secondly, one has to take into account the time scale. If a secret communication is encrypted at time T_0 and should remain secret for a period δ, then one has to make sure that no technology is able to break the encryption before $T_0 + \delta$. Otherwise, an enemy can record the communication at time T_0 and decrypt it later using modern technology. For some applications, δ is equal to twenty or even fifty years. Hence it is crucial to anticipate potential future improvements of cryptanalysis. This is all the

more important since deciding of new standards and deploying them takes at least ten years.

For all these reasons, the NIST announced in 2017 its decision to launch a process to standardise public-key cryptosystems that resist quantum attacks. The NIST issued a call for proposals and researchers were invited to submit their ideas. The NIST received 82 submissions, 64 were accepted to compete in the first round: 19 signature schemes and 45 encryption schemes (and key encapsulation schemes) [Moo19].

We can identify five families of hard problems on which most post-quantum schemes submitted at the NIST rely [BBD09].

	Lattice	Code	Multivariate	Other
Round 1 [2017]	21	17	2	5
Round 2 [2019]	9	7	0	1
Round 3 [2020] (finalists)	3	1	0	0
Round 3 [2020] (alternate)	2	2	0	1

Figure 1.1: Number of candidates (encryption and key encapsulation schemes) in the different rounds of the NIST standardisation process [Moo19; Moo20]. Starting from round 2, the only remaining scheme in the "other" category is the SIKE scheme based on isogenies)

Code-based cryptography. Code-based cryptography is the oldest alternative to number-theoretic encryption schemes. Indeed, in 1978, the year of publication of the RSA cryptosystem, Robert McEliece proposed another hard problem which could be used to build a one-way trapdoor function, and hence a public key cryptosystem [McE78]. His idea came from the field of information theory, and more specifically error correction. McEliece was designing codes, where redundant information was added to the message so that potential errors due to the transmission could be corrected. He remarked that correcting errors in a random code was particularly hard, and that this could be used as a hard problem to design cryptographic schemes. Still, his proposition suffered from a serious drawback: the size of the public key. Indeed, McEliece's cryptosystem requires a public key of a few millions of bits, whereas the RSA key is only of thousands of bits long. This is probably why it did not receive a lot of attention at the time.

Lattice-based cryptography. A lattice is a discrete subgroup of $\mathbf{R}^n$. These objects have many interesting properties and several hard problems can be derived from them. For instance, the shortest vector problem: given

a basis of a lattice Λ, what is the smallest vector of Λ? Or the closest vector problem: given a point $x \in \mathbf{R}^n$ and a basis of lattice Λ, which point of Λ is the closest to x? One of the first cryptosystems based of lattice problems is the NTRU cryptosystem, proposed in 1998 [HPS98].

Hash-based cryptography. Hash-based cryptography is based on the hardness of inverting hash functions. This branch is limited to signature schemes. The seminal idea is due to Lamport [Lam79] and was later extended using hash-tree by Merkle [Mer87].

Multivariate cryptography. Multivariate cryptography is based on the difficulty of solving a system of multivariate (often quadratic) polynomial equations over a finite field. The first multivariate signature scheme was introduced by Matsumoto and Imai [MI88] and generalised by Patarin [Pat96].

Isogeny-based cryptography. This is the most recent family of cryptosystems. It relies on the hardness of finding isogenies between elliptic curves [JD11].

In this work, our attention will be focused on code-based public-key encryption schemes. Before considering the code-based cryptosystems, let us introduce the definition and main properties of error-correcting codes.

1.2 Introduction to coding theory

In his landmark article *A Mathematical Theory of Communication*, Shannon describes the fundamental problem of communication as *"reproducing at one point either exactly or approximately a message selected at another point"* [Sha48]. The problem is that, whatever the communication channel, the message may be altered in the transmission. The notion of *noise* describes the difference between what is sent and what is received. A typical situation is a noisy radio channel: to spell a word, one tends to use the NATO phonetic alphabet: "Alpha" for "A", "Bravo" for "B", "Charlie" for "C" etc. This is all the more useful when the message has no particular meaning (eg. a flight number, a license plate, etc.) so the semantic cannot be used to correct the noise.

More abstractly, the principle is the following: instead of directly transmitting the message (say, a flight number), the sender *encodes* the message using a predefined code (here the NATO phonetic alphabet). This adds *redundancy* to the message. This encoded message is sent over the channel (here, radio) and some noise is added, which alters the message. The receiver interprets the received message and *decodes* it to obtain something as close as possible in

the list of possible codewords. For instance, if the received message sounds like "Novemba, Mango, Biskey", the original message was likely "November, Tango, Whiskey", and will therefore be decoded as "NTW".

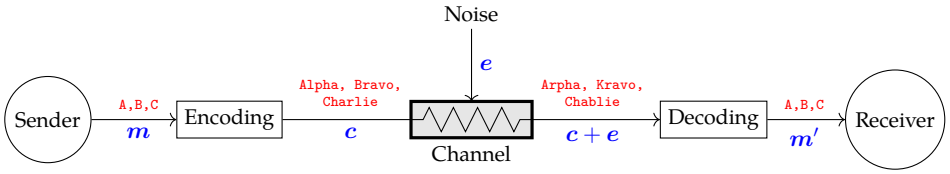


Figure 1.2: Channel coding

Of course, if the noise is too high and the communication channels completely changes the information, the original message can not be retrieved. The goal of error-correction is to design codes such that, if the error noise $\boldsymbol{e}$ is *small enough* (for some precise definition of *small* to be determined), we have a guarantee that $\boldsymbol{m} = \boldsymbol{m}'$.

Let us introduce definitions to formalise this concept.

1.2.1 Error-correcting codes

Definition. A *code* of length n over $\mathbf{F}_q$ is a subset of $\mathbf{F}_q^n$. This subset can be represented by an exhaustive list of all codewords. But this representation is not very convenient. Hence, we will restrict our attention to *linear codes*.

Definition 1.1 (Linear codes)**.** An $[n, k]$-linear code (or $[n, k]$-code) $\mathscr{C}$ over $\mathbf{F}_q$ is a linear subspace of $\mathbf{F}_q^n$ of dimension k.

The parameter n is called the *length* of the code and k is its *dimension*. Note that necessarily $k \leqslant n$.

The information rate of the code corresponds to the average number of bits necessary to encode one bit of information of the message. It measures expansion induced by the code.

Definition 1.2 (Information rate)**.** The information rate of an $[n, k]$-code $\mathscr{C}$ is defined as the ratio between the dimension and the length

$$R \stackrel{\text{def}}{=} k/n.$$

Representation. A linear code can be represented by a basis of codewords. A basis of codewords (represented in rows) defines a *generator matrix* of the code.

Definition 1.3 (Generator matrix)**.** A matrix $\boldsymbol{G} \in \mathbf{F}_q^{k\times n}$ is a generator matrix of the $[n,k]$-linear code $\mathscr{C}$ if its rowspan is $\mathscr{C}$, that is if

$$\mathscr{C} = \left\{ \boldsymbol{x}\boldsymbol{G} \,|\, \boldsymbol{x} \in \mathbf{F}_q^k \right\}.$$

Equivalently, a code can be defined as the kernel of a linear application. A matrix of this application is called a *parity-check matrix* of the code.

Definition 1.4 (Parity-check matrix)**.** A matrix $\boldsymbol{H} \in \mathbf{F}_q^{(n-k)\times n}$ is a parity-check matrix of the $[n,k]$-linear code $\mathscr{C}$ if $\mathscr{C}$ is the kernel of $\boldsymbol{H}$, that is if

$$\mathscr{C} = \left\{ \boldsymbol{y} \in \mathbf{F}_q^n \,|\, \boldsymbol{H}\boldsymbol{y}^\intercal = \mathbf{0}_k{}^\intercal \right\}.$$

Note that a code can be represented by different generator or parity-check matrices. Two generator (or parity-check) matrices are said to be *equivalent* if they represent the same code. The generator (resp. parity-check) matrix of a code is unique up to left multiplication by a $k \times k$ (resp. an $(n-k)\times(n-k)$) invertible matrix over $\mathbf{F}_q$.

Performing a Gaussian elimination on a matrix amounts to left-multiplying it by a non-singular matrix. Hence, any code admits a unique generator matrix such that the left $k \times k$ sub-matrix corresponds to the identity. This generator matrix is said to be in *systematic form*. Same thing applies to parity-check matrices.

Duality. We can define the dual of a code.

Definition 1.5. The dual of a code $\mathscr{C} \subseteq \mathbf{F}_q^n$, denoted $\mathbf{Dual}(\mathscr{C})$ is the set of all vectors of $\mathbf{F}_q^n$ that are orthogonal to all the codewords of $\mathscr{C}$.

$$\boldsymbol{y} \in \mathbf{Dual}(\mathscr{C}) \qquad \Longleftrightarrow \qquad \forall \boldsymbol{x} \in \mathscr{C}, \boldsymbol{x}\boldsymbol{y}^\intercal = 0 \in \mathbf{F}_q^n.$$

If $\mathscr{C}$ is an $[n,k]$-code then $\mathbf{Dual}(\mathscr{C})$ is an $[n,n-k]$-code.

Proposition 1.6. *Let $\boldsymbol{G}$ and $\boldsymbol{H}$ be a generator matrix and a parity-check of $\mathscr{C}$ respectively. Then $\boldsymbol{G}$ is a parity-check matrix of $\mathbf{Dual}(\mathscr{C})$ and $\boldsymbol{H}$ is a generator matrix of $\mathbf{Dual}(\mathscr{C})$.*

Hamming metric. In this work, we will mainly focus on codes defined using the Hamming metric.

Definition 1.7 (Support)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{F}_q^n$. The support of $\boldsymbol{x}$, denoted $\mathbf{Support}(\boldsymbol{x})$, is the set of non-zero indices of $\boldsymbol{x}$.

$$\mathbf{Support}(\boldsymbol{x}) \stackrel{\text{def}}{=} \{i \in [\![1,n]\!], x_i \neq 0\}.$$

Definition 1.8 (Hamming weight)**.** The Hamming weight $\mathbf{w}_H(\boldsymbol{x})$ of a vector $\boldsymbol{x} \in \mathbf{F}_q^n$ is the size of its support, *i.e.* the number of its non-zero components.

$$\mathbf{w}_H(\boldsymbol{x}) \stackrel{\text{def}}{=} |\mathbf{Support}(\boldsymbol{x})|.$$

The Hamming distance between two vectors is the weight of their difference.

Definition 1.9 (Hamming distance)**.** The Hamming distance $d_H(\boldsymbol{x}, \boldsymbol{y})$ between two vector $\boldsymbol{x}$ and $\boldsymbol{y} \in \mathbf{F}_q^n$ is defined as

$$d_H(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{w}_H(\boldsymbol{y} - \boldsymbol{x}).$$

This distance is natural in this context, since it corresponds to the minimal number of symbols that one has to change to transform the vector $\boldsymbol{x}$ in the vector $\boldsymbol{y}$.

Remark 1.10. *There exist other metrics that can be used to define error-correcting codes, especially the* rank metric*. Most definition can be adapted straightforwardly from Hamming metric to rank metric. Rank-metric codes play an important role in code-based cryptography. An example of rank-metric code-based scheme is introduced in Chapter 4.*

An important characteristic of a code is the minimal distance between two codewords.

Definition 1.11 (Minimal distance)**.** The minimal distance of a code $\mathscr{C}$ is defined as

$$d \stackrel{\text{def}}{=} \min \left\{ d_H(\boldsymbol{x}, \boldsymbol{y}) \,|\, \boldsymbol{x}, \boldsymbol{y} \in \mathscr{C}, \boldsymbol{x} \neq \boldsymbol{y} \right\}.$$

If $\mathscr{C}$ is a linear code, then

$$d = \min \left\{ \mathbf{w}_H(\boldsymbol{c}) \,|\, \boldsymbol{c} \in \mathscr{C}, \boldsymbol{c} \neq \mathbf{0} \right\}.$$

A linear code of length n, dimension k and minimal distance d is called an $[n, k, d]$-code.

The minimal distance measures the (theoretical) error-correction capacity of a linear code. Indeed, let $t = \lfloor (d-1)/2 \rfloor$, the balls of radius t centered on codewords are all disjoint. Hence, any vector of $\mathbf{F}_q^n$ at distance at most t of a codeword has a unique closest codeword. In other words, if an error of Hamming weight at most t is added to a codeword, one can recover the original message by finding the closest codeword. This formalises the property that was expected at the end of the introductory section. Note that this is a purely theoretic result since it does not provide an efficient way to find the closest codeword.

1.2.2 Encoding, decoding

Encoding consists in mapping each vector of $\mathbf{F}_q^k$ to a codeword.

Definition 1.12 (Encoder)**.** Let $\mathscr{C}$ be an $[n,k]$-code and $\boldsymbol{G}$ a generator matrix of $\mathscr{C}$.

$$\mathrm{Enc}: \quad \begin{cases} \mathbf{F}_q^k & \longrightarrow \mathscr{C} \subseteq \mathbf{F}_q^n \\ \boldsymbol{x} & \longmapsto \quad \boldsymbol{xG} \end{cases}$$

is an encoder of $\mathscr{C}$.

Definition 1.13 (Decoder)**.** Let $\mathscr{C}$ be an $[n,k]$-code. A decoder for $\mathscr{C}$ is a function $\mathrm{Dec} : \mathbf{F}_q^n \to \mathscr{C} \cup \{\perp\}$ such that

$$\forall \boldsymbol{c} \in \mathscr{C}, \quad \mathrm{Dec}(\boldsymbol{c}) = \boldsymbol{c}.$$

The symbol $\perp$ here denotes the fact that the decoding may fail to return a codeword. It should be interpreted as "I do not know how to decode this vector". Sometimes it is useful to know that the decoder failed.

This definition of a decoder is not very interesting. For a decoder to be useful, it has to respect some properties.

Definition 1.14. Let $\mathscr{C}$ be an $[n,k]$-code and Dec be a decoder of $\mathscr{C}$. Let t be an integer. We say that Dec is a t-bounded decoder (or that Dec can *correct up to t errors in $\mathscr{C}$*) if

$$\forall \boldsymbol{c} \in \mathscr{C}, \forall \boldsymbol{y} \in \mathbf{F}_q^n, \quad d_H(\boldsymbol{c}, \boldsymbol{y}) \leqslant t \quad \Longrightarrow \quad \mathrm{Dec}(\boldsymbol{y}) = \boldsymbol{c},$$

or equivalently

$$\forall \boldsymbol{c} \in \mathscr{C}, \forall \boldsymbol{e} \in \mathbf{F}_q^n, \quad \mathbf{w}_H(\boldsymbol{e}) \leqslant t \quad \Longrightarrow \quad \mathrm{Dec}(\boldsymbol{c}+\boldsymbol{e}) = \boldsymbol{c}.$$

Note that, formally (to be used as in Figure 1.2), the decoder should not return a codeword $\boldsymbol{c} \in \mathscr{C}$ but the element $\boldsymbol{x} \in \mathbf{F}_q^k$ such that $\boldsymbol{c} = \boldsymbol{xG}$. But for our later use of error-correcting codes in the context of cryptography, we prefer to stick to this definition (which corresponds to a *corrector* rather than a decoder properly speaking). Anyway, the interesting part of the decoding lies in the error-correction. Once the codeword is obtained, it is easy to invert the encoding.

Another notion that is very useful when considering the decoding of a code is that of syndrome. Given the parity-check matrix $\boldsymbol{H}$ of a code $\mathscr{C}$, by definition, a vector $\boldsymbol{c}$ belongs to $\mathscr{C}$ if and only if $\boldsymbol{Hc}^\intercal = \boldsymbol{0}_{n-k}{}^\intercal$. Hence, let $\boldsymbol{y}$ be a vector of $\mathbf{F}_q^n$ such that $\boldsymbol{y} = \boldsymbol{c}+\boldsymbol{e}$ with $\boldsymbol{c} \in \mathscr{C}$. Then, $\boldsymbol{Hy}^\intercal = \boldsymbol{Hc}^\intercal + \boldsymbol{He}^\intercal = \boldsymbol{He}^\intercal$. This quantity is called the *syndrome*.

Definition 1.15 (Syndrome)**.** The syndrome $\boldsymbol{s} \in \mathbf{F}_q^{n-k}$ of a vector $\boldsymbol{e} \in \mathbf{F}_q^n$ is $\boldsymbol{s}^\intercal = \boldsymbol{H}\boldsymbol{e}^\intercal$.

Given a noisy codeword $\boldsymbol{y} \in \mathscr{C}$ of syndrome $\boldsymbol{s} \in \mathbf{F}_q^{n-k}$, the set of vectors that have the same syndrome as $\boldsymbol{y}$ are exactly the cosets $\boldsymbol{y} + \mathscr{C}$ and form a partition of $\mathbf{F}_q^n$. Hence, we can define a decoder that takes as input a syndrome (instead of a noisy codeword) and returns the associated error.

Definition 1.16 (Syndrome decoder)**.** Let $\mathscr{C}$ be an $[n, k]$-code and t an integer. A t-bounded syndrome-decoder for $\mathscr{C}$ is a function $\mathrm{SynDec} : \mathbf{F}_q^{n-k} \to \mathbf{F}_q^n$ such that

$$\forall \boldsymbol{e} \in \mathbf{F}_q^n, \quad \mathbf{w}_H(\boldsymbol{e}) \leqslant t \quad \Longrightarrow \quad \mathrm{SynDec}(\boldsymbol{H}\boldsymbol{e}^\intercal) = \boldsymbol{e}.$$

The two kinds of decoders are equivalent.

Proposition 1.17. *A t-bounded decoder and a t-bounded syndrome decoder are equivalent.*

Proof. Let $\mathscr{C}$ be a code and $\boldsymbol{H}$ a parity-check matrix of $\mathscr{C}$.

- Given a t-bounded syndrome decoder SynDec of $\mathscr{C}$, let us construct a t-bounded decoder. Let $\boldsymbol{y} \in \mathbf{F}_q^n$ be a noisy codeword. Suppose that there is $\boldsymbol{c} \in \mathscr{C}$ and $\boldsymbol{e} \in \mathbf{F}_q^n$ of weight $\mathbf{w}_H(\boldsymbol{e}) \leqslant t$ such that $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$. We want to find $\boldsymbol{c}$. First we compute the syndrome $\boldsymbol{s}$ such that $\boldsymbol{H}\boldsymbol{y}^\intercal = \boldsymbol{s}^\intercal$. Hence $\mathrm{SynDec}(\boldsymbol{s})$ returns $\boldsymbol{e}$ and $\boldsymbol{y} - \boldsymbol{e}$ yields $\boldsymbol{c}$.
- Given a t-bounded decoder Dec of $\mathscr{C}$, let us construct a t-bounded syndrome decoder. Let $\boldsymbol{s} \in \mathbf{F}_q^{n-k}$ be a syndrome. Suppose that there is an element $\boldsymbol{e} \in \mathbf{F}_q^n$ of weight $\mathbf{w}_H(\boldsymbol{e}) \leqslant t$ such that $\boldsymbol{H}ev^\intercal = \boldsymbol{s}^\intercal$. We want to find $\boldsymbol{e}$. We can first consider any element $\boldsymbol{y} \in \mathbf{F}_q^n$ such that $\boldsymbol{H}\boldsymbol{y}^\intercal = \boldsymbol{s}^\intercal$ (which is easily achievable using basic linear algebra). $\boldsymbol{y}$ and $\boldsymbol{e}$ have the same syndrome, hence there exists $\boldsymbol{c} \in \mathscr{C}$ such that $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$. Hence, $\boldsymbol{y} - \mathrm{Dec}(\boldsymbol{y}) = \boldsymbol{e}$.

□

1.2.3 Decoding problems

Decoding a noisy codeword (that is, a codeword to which an error has been added) is in general not an easy task. We can define the problem as follows.

Problem 1.18 (General Decoding Problem - $\mathrm{GD}(q, R, W)$)**.**

Instance: $\boldsymbol{G} \in \mathbf{F}_q^{k \times n}$ *of full rank,*
$\boldsymbol{y} \in \mathbf{F}_q^n$.

Output: $\boldsymbol{c}, \boldsymbol{e} \in \mathbf{F}_q^n$ *such that* $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, $\boldsymbol{c} \in \mathscr{C}$ *and* $\mathbf{w}_H(\boldsymbol{e}) = w$,

where $k \stackrel{def}{=} \lceil Rn \rceil$ *and* $w \stackrel{def}{=} \lceil Wn \rceil$.

Here, $\mathscr{C}$ denotes the code generated by the matrix $\boldsymbol{G}$. An instance of the problem consists of randomly choosing $\boldsymbol{G}$ and $\boldsymbol{y}$. A solution to the problem does not necessarily exist. A decisional version of the problem consists in deciding whether such a solution exists, without explicitly outputing a solution.

Equivalently, we can define the syndrome decoding problem.

Problem 1.19 (Syndrome Decoding - $\mathrm{SD}(q, R, W)$)**.**
Instance: $\boldsymbol{H} \in \mathbf{F}_q^{(n-k)\times n}$ *of full rank,*
$\boldsymbol{s} \in \mathbf{F}_q^{n-k}$ (*usually called the* syndrome).
Output: $\boldsymbol{e} \in \mathbf{F}_q^n$ *such that* $\mathbf{w}_H(\boldsymbol{e}) = w$ *and* $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$,
where $k \stackrel{def}{=} \lceil Rn \rceil$ *and* $w \stackrel{def}{=} \lceil Wn \rceil$.

Again, the two problems are equivalent, because of Proposition 1.17.

In an article published in 1978, Berlekamp, Massey and van Tilborg proved that for almost every code, the syndrome decoding problem is NP-hard and the decisional version of the problem is NP-complete [BMT78].

This result shows that the problem is hard in the worst case. However, the syndrome decoding problem is also believed to be hard in the average case. In [Ale03], Alekhnovich conjectured that decoding even an error of weight $w = n^\epsilon$ in a code of length n is hard on average for any $\epsilon > 0$.

The best algorithms designed to solve the syndrome decoding problem are called *information set decoding algorithms*. Their complexity is exponential in time. These algorithms are details in Chapter 8.

Because decoding is a difficult problem for a random code, the whole point of coding theory is to design special families of codes for which there exist efficient decoding algorithms up to some distance. For instance, there exists families of codes for which one can decode $\mathcal{O}(n)$ errors in polynomial time.

In such a case, the decoding algorithm exploits the special structure of the code. Here, we see the pattern of a problem which is difficult in general but becomes hard for some particular instances. The structure of these particular codes, necessary to decode efficiently, can be kept secret and serve as a trapdoor. Hence, the decoding problem can be used as a one way trapdoor function to create cryptographic primitives. This will be detailed in Section 1.3.

1.2.4 Bounds on codes

We have seen that we can encode a message of $\mathbf{F}_q^k$ in a codeword of $\mathbf{F}_q^n$ such that we can correct errors on up to t positions of the vector. Hence there are two parameters that we want to optimise. First, we do not want the encoded message to be too long compared to the original message. This corresponds to

the rate $R \stackrel{\text{def}}{=} k/n$ which should remain as close as possible to 1. But we also want to be able to correct as many errors as possible. We know that we can decode unambiguously at most $t \leqslant \lfloor (d-1)/2 \rfloor$ errors, where d is the minimal distance of the code. Hence, let $D \stackrel{\text{def}}{=} d/n$, we also want D to be as close as possible to 1. In this case, we could theoretically correct errors on up to half of the positions. In information theory, the communication channel is modeled as a binary symmetric channel, which means that it adds errors independently on each position of the vector with probability p. Hence, if $D \to 1$ it means that we can correctly decode almost all messages as soon as $p < 1/2$.

On one hand, we can easily build a family of codes such that $R \to 0$ and $D \to 1$. Think of the repetition code of dimension 1 and length n, where 0 is encoded in $\mathbf{0}_n$ and 1 is encoded in $\mathbf{1}_n$. In this case, decoding amounts to a majority vote on the bits. On the other hand, the parity-check code of dimension $n-1$ and length n (*i.e.* adding a bit at the end of each vector such that the sum of all bits is even) yields $R \to 1$ but $D \to 0$. But can we get both R and D close to 1? Intuitively, these goals are opposite, since it is the redundancy of information that enables the correction. Coding theory is about finding "good" codes, *i.e.* codes which attain a good trade-off between their rate and their decoding radius. But not all combinations of R and D are reachable. Here are some constrains on these parameters.

Entropy. To state asymptotic results, we need to introduce the notion of entropy, which is central in information theory. This notion will be useful to study the asymptotic complexity of generic decoding algorithms. Entropy is defined as follows.

Definition 1.20 (Entropy function)**.** The q-ary entropy function is defined as

$$h_q \begin{cases} [0,1] \to \mathbf{R} \\ x \quad \mapsto x\log_q(q-1) - x\log_q x - (1-x)\log_q(1-x). \end{cases}$$

The Singleton bound. The simplest bound is knows as the Singleton bound.

Theorem 1.21 (Singleton bound)**.** *Let $\mathscr{C}$ be an $[n,k,d]$-code over $\mathbf{F}_q$, then*

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

Proof. An algorithmic approach to this result is to think of it as a consequence of Gaussian elimination. Indeed, consider $\boldsymbol{G} \in \mathbf{F}_q^{k\times n}$ a full-rank generator matrix of the code $\mathscr{C}$. All rows of $\boldsymbol{G}$ are codewords. We can apply a Gaussian elimination to $\boldsymbol{G}$ to put it in row echelon form. The operations are just linear operations on the rows so the new matrix is still a generator matrix of $\mathscr{C}$. The

new matrix is in systematic form, and hence the first row (which is a codeword) has weight $\leqslant n - k + 1$. □

A codes that reaches the Singleton bound is called a *maximum distance separable* (MDS) code. We will see in Chapter 5 that generalised Reed–Solomon codes are MDS codes.

We can easily extend the singleton bound to obtain an asymptotic result.

Corollary 1.22 (Asymptotic Singleton bound)**.** *Given a family of $[n_i, k_i, d_i]$-codes of increasing size ($n_i \to \infty$) and such that $k_i/n_i \to R$ and $d_i/n_i \to D$, we have*

$$R + D \leqslant 1.$$

The Hamming bound. The Hamming bound, or sphere-packing bound, is a tighter upper bound on the minimum distance given the length and dimension of a code [Ham50]. It derives from the fact that the balls for which we can decode unanbiguously to a given codeword are disjoint.

Theorem 1.23 (Hamming bound)**.** *Let $\mathscr{C}$ be an $[n, k, d]$-code over $\mathbf{F}_q$. Then*

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i \leqslant q^{n-k},$$

where $t = \lfloor (d-1)/2 \rfloor$.

Proof. If the minimal distance of $\mathscr{C}$ is d, then, by triangular inequality, the balls of radius t and centered in each codewords are disjoint. There are q^k such balls, each of volume

$$\sum_{i=0}^{t} \binom{n}{i} (q-1)^i.$$

The sum of these disjoint balls is contained in $\mathbf{F}_q^n$, which is of size q^n. □

From this bound, we can derive the following asymptotic result [Cou20, Lemma 4.5].

Corollary 1.24 (Asymptotic Hamming bound)**.** *Given a family of $[n_i, k_i, d_i]$-codes of increasing size ($n_i \to \infty$) and such that $k_i/n_i \to R$ and $d_i/n_i \to D$, we have*

$$R \leqslant 1 - h_q\left(\frac{D}{2}\right).$$

The Gilbert-Varshamov bound. We have introduced two upper bounds on d (for fixed n and k). We now state a lower bound: the Gilbert-Varshamov bound [Gil52; Var57].

Theorem 1.25 (Gilbert-Varshamov bound)**.** *Let q, n, d be integers such that $2 \leqslant d \leqslant n$. Then, there exists a code $\mathscr{C} \subseteq \mathbf{F}_q^n$ of length n and minimum distance d such that*

$$|\mathscr{C}| \sum_{i=0}^{d-1} \binom{n}{i} (q-1)^i \geqslant q^n.$$

Proof. Among the codes of length n and minimal distance d over $\mathbf{F}_q$, let $\mathscr{C}$ have the maximal number of codewords. Consider all the balls of radius $d-1$ and centered in the codewords of $\mathscr{C}$. These balls must cover the whole space $\mathbf{F}_q^n$. Indeed, suppose that there exists one element $\boldsymbol{x} \in \mathbf{F}_q^n$ that does not belong to any of these balls, this element is at distance at least d of each codeword of $\mathscr{C}$ and hence the code $\mathscr{C}' \stackrel{\text{def}}{=} \mathscr{C} \cup \{\boldsymbol{x}\}$ is also a code of length n and minimal distance d. But then $|\mathscr{C}'| > |\mathscr{C}|$, which contradicts the hypothesis on $\mathscr{C}$. □

This result can be extended to prove that there always exist a linear code verifying this bound, using a counting argument on the number of generator matrices [Dem97, Proposition 8.9].

Again, we can derive an asymptotic result [Cou20, Theorem 4.10].

Corollary 1.26 (Asymptotic Gilbert-Varshamov bound)**.** *There exists a sequence of $[n_i, k_i, d_i]$-codes over $\mathbf{F}_q$ of increasing size ($n_i \to \infty$) and such that $k_i/n_i \to R$ and $d_i/n_i \to D$, and such that*

$$R \geqslant 1 - h_q(D).$$

An important result (which is particularly useful for code-based cryptography) is that, with high probability, the minimal distance of a random code is close to the Gilbert-Varshamov bound. This results is formally stated as follows [Cou20, Theorem 4.11].

Theorem 1.27. *For $0 < D < 1 - \frac{1}{q}$, let $\varepsilon > 0$ and let $\mathscr{C}$ be a random code of dimension $k < (1 - h_q(D) - \varepsilon)n$. Then, let d denote the minimum distance of $\mathscr{C}$,*

$$\mathbb{P}[d > Dn] \geqslant 1 - q^{\varepsilon n}.$$

Hence, for a fixed rate R, the typical value D such that the minimum distance of a random code of rate R is almost always $\geqslant D$ verifies $h_q(D) = 1 - R$. This is called the relative Gilbert-Varshamov distance.

Shannon's theorem. Finally, let us conclude by a fundamental theorem of coding theory. This result, due to Shannon [Sha48], answers the following question: which amount of redundancy (hence which rate) is necessary to (almost always) correctly recover the message? Of course, this depends on the properties of the channel (how often errors occur etc.). Shannon defines the notion of *capacity* of the channel, which reflects the maximal rate of a code able to correct almost all errors added by this channel.

Shannon's result can be informally stated as follows. Given a channel that transmits elements of $\mathbf{F}_q$ and adds an error independently on each symbol with probability p, one needs a minimal proportion of $h_q(p)$ redundant symbols to (almost always) correct the errors, where h_q denotes the q-ary entropy function. Hence, for a message of k symbols, we need $n > k + h_q(p)n$, *i.e.* $R > 1 - h_q(p)$. This yields the formula for the capacity of the channel: $C \stackrel{\text{def}}{=} 1 - h_q(p)$. Then,

1. for $R < C$, there always exists a code of rate R such that the probability of a wrong decoding is exponentially small ;
2. for $R > C$, all codes of rate R yield a probability for a vector to be incorrectly decoded of at least one half.

More formally, the theorem can be stated as follows (from [Cou20], Theorem 3.9).

Theorem 1.28 (Shannon's theorem)**.** *For all $0 < p < 1 - 1/q$ and all $0 < \varepsilon < 1 - 1/q - p$, let $C \stackrel{def}{=} 1 - h_q(p)$, the following statements holds.*

1. *There exists $\delta > 0$ such that, for any large enough n, there exists a code $\mathscr{C}$ of length n and rate $R = C - \varepsilon$ a decoder Dec such that*
$$\mathbb{P}_{\text{fail}}\left[\mathscr{C}, \mathrm{Dec}\right] < q^{-\delta n}.$$
2. *For all large enough n and all pairs $(\mathscr{C}, \mathrm{Dec})$ where $\mathscr{C}$ is a code of length n and rate $R = C + \varepsilon$, and Dec is a decoder,*
$$\mathbb{P}_{\text{fail}}\left[\mathscr{C}, \mathrm{Dec}\right] \geqslant \frac{1}{2}.$$

Here, $\mathbb{P}_{\text{fail}}$ denotes the probability, over all codewords $\boldsymbol{c} \in \mathscr{C}$ chosen uniformly at random and all errors $\boldsymbol{e} \in \mathbf{F}_q^n$ chosen among a Bernoulli distribution of parameter p, that $\mathrm{Dec}(\boldsymbol{c} + \boldsymbol{e}) \neq \boldsymbol{c}$, and h_q denotes the q-ary entropy function defined as

$$h_q(p) \stackrel{\text{def}}{=} p\log_q(q-1) - p\log_q p - (1-p)\log_q(1-p).$$

This result also holds for non-linear codes.

Note that Shannon's theorem is a theoretic result and (concerning the first statement) does not explain how to construct such codes and the associated decoding algorithms. It does not even say that such decoders are efficient, they may well have exponential time and space complexity. A part of coding theory is dedicated to finding families of codes with rates as close as possible to the channel capacity but with efficient decoders.

1.3 Introduction to code-based cryptography

1.3.1 The McEliece scheme

In 1978, McEliece was working on coding theory. He was designing special families of codes for which there exists an efficient decoding algorithm. After reading Diffie and Hellman's work introducing public key cryptography [DH76a], he proposed to use such special codes as a one-way function to design public-key cryptographic schemes [McE78]. Let us first present his idea in an abstract way (as it is done in [Cou19]).

1.3.1.1 McEliece's idea

As we have seen, in general, given a random linear code, decoding in this code is a hard problem. But for certain families of codes (having some special property), there exists a decoding algorithm that makes use of this special property to decode efficiently. Moreover, the fact that a code has this property is not necessarily apparent. A generator matrix of such a code might well look exactly like the generator matrix of any other linear code.

McEliece's idea is to use this as a one way function. It uses a family of codes $\mathcal{F}$, for which there exists an efficient (*i.e.* polynomial) decoding algorithm Dec, and a function to construct these codes. This function takes some input s and yields a code $\mathscr{C}(s) \in \mathcal{F}$ that has the expected special property. Here, s will serve as a secret. Let us denote $\mathcal{S}$ the space of s. The idea is that

$$\begin{array}{ccc} \mathcal{S} & \longrightarrow & \mathcal{F} \\ s & \longmapsto & \mathscr{C}(s) \end{array}$$

is a one-way function, *i.e.* it is hard to find s given $\mathscr{C}(s)$. Moreover, the important property is that Dec only works if s is known. Here, s serves as a trapdoor. From this, we can design a cryptosystem in the following way.

Key generation.

- First, randomly pick a secret $s \in \mathcal{S}$.

- Compute the corresponding code $\mathscr{C} \stackrel{\text{def}}{=} \mathscr{C}(s)$ an $[n, k]$-code.
- Use s as the secret key and $(\boldsymbol{G}_{\text{pub}}, t)$ as the public key, where $\boldsymbol{G}_{\text{pub}}$ is a generator matrix of $\mathscr{C}$ and t is the maximal number of error that Dec can decode in $\mathscr{C}$.

Encryption.

- The message $\boldsymbol{m} \in \mathbf{F}_q^k$ in encrypted as

$$\boldsymbol{y} \stackrel{\text{def}}{=} \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e} \in \mathbf{F}_q^n,$$

  where $\boldsymbol{e}$ is a random vector of $\mathbf{F}_q^n$ of weight t.

Decryption.

- First correct the errors using the decoding algorithm: $\boldsymbol{x} \stackrel{\text{def}}{=} \mathrm{Dec}(s, \boldsymbol{y}) \in \mathscr{C}$.
- Then recover $\boldsymbol{m} \in \mathbf{F}_q^k$ such that $\boldsymbol{m}\boldsymbol{G} = \boldsymbol{x}$ using linear algebra.

In his article [McE78], McEliece did not present the scheme for an abstract family of codes $\mathcal{F}$ but for a particular instance. His idea was to use the family of *Goppa codes*, which are known to have an efficient decoding algorithm. Following his work, many proposals were made to replace Goppa codes with other kinds of codes. All these proposals follow the same steps as McEliece, just changing the family of codes (and the associated function to construct a code from a secret input). Hence, they can all be generalised by presenting the scheme in this way. All these cryptosystems are said to follow the McEliece scheme.

1.3.1.2 The Niederreiter variant

Niederreiter proposed a variant of McEliece's scheme in [Nie86]. The idea stays the same: the ciphertext is a noisy codeword and the decryption phase corresponds to decoding. The decoding is possible only if one knows the secret. In McEliece's approach, the plaintext is encoded in a codeword, to which a random error weight t is added. Niederreiter proposes to encode the plaintext in the error pattern: first a codeword is picked at random, to which an error (encoding the plaintext) is added. For this, one needs a function φ from the message space (say $\mathbf{F}_q^\ell$) to the space of vectors of $\mathbf{F}_q^n$ of weight t. This function should be invertible. Such functions exist (see [Sen02]).

To decrypt a ciphertext $\boldsymbol{y}$ of the Niederreiter scheme, first one uses the decoding algorithm to compute $\boldsymbol{c} = \mathrm{Dec}(s, \boldsymbol{y}) \in \mathscr{C}$, and then one can recover the message $\boldsymbol{m} = \varphi^{-1}(\boldsymbol{y} - \boldsymbol{c})$.

In terms of security, the Niederreiter scheme is equivalent to the McEliece scheme when used with the same code [LDW94]. The advantage of Niederreiter's approach is that it reduces the public key size. Indeed, the public key of the McEliece scheme is a generator matrix $\boldsymbol{G}_{\text{pub}}$ of the code, of size $k \times n$. In the Niederreiter setting, one can choose a particular generator matrix of the code, namely the generator matrix that is in systematic form. Hence one only needs to send the $k \times (n-k)$ sub-matrix corresponding to the non-identity part. Because the codeword is chosen randomly, having access to a systematic generator matrix does not change anything to the security of the system. This trick can not be used in the McEliece setting because the public generator matrix is used to encode the plaintext.

Because the security is equivalent, most of the time, we will not make a distinction between the McEliece and Niederreiter settings and simply refer to the McEliece scheme, to simplify the notations. However, most cryptosystem that we will refer to as "following the McEliece scheme" in fact use the Niederreiter setting in practice to obtain shorter public keys.

1.3.1.3 Security of the McEliece scheme

To decrypt a ciphertext encrypted using the McEliece scheme, an attacker has two possibilities.

Message security. The first one is to try to decode the noisy codeword, independently of the special properties induced by the fact that $\mathscr{C} \in \mathcal{F}$. This amounts to having the ability to decode t errors in a random $[n, k]$-code. This corresponds exactly to the general decoding problem introduced in Section 1.2.3. As we have seen, this problem is considered to be intractable. This is the fundamental security hypothesis of code-based cryptography. The best knows algorithms to perform such attacks are the *information set decoding algorithms*, such as Prange's algorithm (see Chapter 8). Hence the parameters should be chosen such that the best such algorithm takes 2^{κ} operations to solve the problem, with κ being the security parameter. This is known as the *message security*.

Key security. The other approach consists in using the special properties of the code, due to the fact that $\mathscr{C} \in \mathcal{F}$. The most straightforward approach would be to find s such that $\mathscr{C} = \mathscr{C}(s)$ and then decrypt using the decoder Dec. This should not be feasible. More generally, the second security hypothesis on which the security relies is that, given a public generator matrix $\boldsymbol{G}_{\text{pub}}$, an attacker should not be able to distinguish if it corresponds to a code $\mathscr{C} \in \mathcal{F}$ or a random $[n, k]$-code. Note that this hypothesis also covers the case where the

attacker would not recover the value of s but use a weaker property of $\mathcal{F}$ to make the decoding more efficient. This is known as the *key security*.

Remark 1.29. *In general, the function $\mathscr{C} : s \mapsto \mathscr{C}(s) \in \mathcal{F}$ is not injective so the attacker tries to find any value s' such that $\mathscr{C} = \mathscr{C}(s')$.*

Note that there is a fundamental difference between these two security hypothesis. The first hypothesis, the hardness of general decoding, does not depend on the choice of a family $\mathcal{F}$ of codes to instantiate the scheme. On the contrary, the second hypothesis, the indistinguishability of the code family $\mathcal{F}$, is specific to a particular choice of $\mathcal{F}$. Hence, there are families of codes for which this hypothesis holds, others for which it does not. Attacks exploiting a flaw in this hypothesis on a family $\mathcal{F}$ are also referred to as *structural attacks* because they exploit the (supposedly hidden) structure of the codes in $\mathcal{F}$. Examples of such attacks are presented in Chapters 4, 5, 6 and 7.

Under these two hypothesis, the McEliece scheme is proven secure [Sen11b]. Note that the security of the textbook scheme holds in the one-way chosen-plaintext attack model (OW-CPA) but that some additional layers of security transformations are necessary to obtain security in stronger attack models.

1.3.1.4 Good codes to instantiate McEliece

A good part of code-based cryptography is dedicated to finding the best possible family of codes $\mathcal{F}$ to instantiate the McEliece scheme such that the key security hypothesis is fulfilled, while achieving the best possible performances, both in terms of encryption/decryption time and in terms of key size. The key size of often the most significant issue, since it constitutes the major drawback of code-based cryptosystems, compared to other post-quantum schemes.

We present here several attempts to instantiate the McEliece scheme. A natural idea is to use codes with algebraic properties. These codes often enjoy a large decoding radius (reaching the Singleton bound) with efficient decoders. This yields a good transmission rate. Indeed, in the Niederreiter setting, the plaintext is encoded in the error, so a code that corrects a larger number of errors convoys more secret information.

Reed–Solomon codes. The (generalised) Reed–Solomon (GRS) codes are enumeration codes. Each codeword corresponds to a polynomial. Hence, they have all the necessary properties to make an efficient cryptosystem. Niederreiter was the first person to suggest their use [Nie86]. But Sidelnikov and Shestakov proved that the indistinguishability hypothesis does not hold for these codes [SS92]. Different attempts using variants of GRS codes were proposed, hoping to counter this attack, such as the RLCE [Wan17] and XGRS

cryptosystems [KRW21]. The study of GRS-based cryptosystems is the subject of the second part of this work.

Goppa codes. The Goppa codes are a special family of subfield subcodes of GRS codes. Binary Goppa codes are the codes proposed by McEliece to instantiate his scheme [McE78]. This proposal is still considered secure today [BLP08]. The *Classic McEliece* submission at the NIST standardisation process uses such codes [BCLMM+19]. The only known weakness of Goppa codes is the existence of a distinguisher on high-rate Goppa codes [FGOPT11]. Cryptosystems using q-ary Goppa codes were also proposed [BLP10]. A partial attack exists on these codes [COT14b].

Reed–Muller codes. Sidelnikov proposed to use Reed–Muller codes to instantiate the scheme [Sid94] but this was proven insecure in [MS07].

Concatenated codes. Sendrier proposed to use concatenated codes for McEliece [Sen94] but he later found a weakness in this scheme [Sen98].

Geometric codes. In [JM96], Janwa and Moreno suggested to use McEliece with algebraic-geometric codes. This corresponds to a generalisation of GRS codes to a higher genus. This was proved insecure, first for small genus [FM08] and later for all curves [CMP17].

MDPC codes. In 2013, Misoczki, Tillich, Sendrier and Barreto proposed to use a new class of codes, moderate-density parity-check codes [MTSB13]. This proposal differs significantly from the others, because these codes do not have any algebraic structure. They do have a particularly efficient decoding algorithm, but this algorithm is probabilistic. The lack of structure is an advantage, as it leaves less possibilities for structural attacks, but the probabilistic nature of the decoding algorithm give rise to new kind of problems, as we will see in Chapters 2 and 3. These codes, in their quasi-cyclic setting, are used in the BIKE submission for the NIST standardisation process [ABBBB+17].

This list is not exhaustive and is restricted to Hamming-metric cryptosystem. Most families of codes cited above have rank-metric equivalents, which were also proposed to instantiate the McEliece scheme.

1.3.2 Other code-based encryption schemes

The McEliece scheme corresponds to the seminal article on code-based cryptography and, as we have seen, can be instantiated using different families of

codes. The security of the scheme relies on two security hypothesis: the hardness of the general decoding problem, and a security hypothesis that depends on the family of codes. The first hypothesis corresponds to a very generic and well-studied mathematical problem, believed to be intractable, whereas the second hypothesis is specific to an instantiation choice. This second hypothesis appears to be weaker (although in the case of Goppa codes it has received a lot of attention). Hence, there are proposals to build code-based cryptosystem relying only on the hardness of the general decoding problem.

The main idea was proposed by Alekhnovich in [Ale03]. It uses the fact that the scalar product of two binary vectors of length n and weight $\sqrt{n}$ is biased. Hence, using a clever protocol, it is possible to encrypt a plaintext as an codeword, to which a mask is added. Each bit of this mask is computed as a scalar product of two vectors of moderate weight. Hence the total weight of the mask is not too high and the error can be corrected.

This idea was generalised and used in a quasi-cyclic setting to yield the HQC cryptosystem [AABBB+17b] in Hamming metric, and its rank-metric equivalent the RQC cryptosystem [AABBB+17a]. Both were submitted to the NIST standardisation process.

1.3.3 Code-based signature schemes

For a long time, the existence of code-based digital signature schemes has been an open problem. The first code-based signature, referred to as CFS, was proposed in 2001 [CFS01]. But the parameters of this signature become unreasonably large (public key of a few gigabytes) to reach 128 security bits. Moreover, this signature uses Goppa codes in a high rate regime, for which there exists a distinguisher [FGOPT11]. Although this does not provide a way to reconstruct the secret key, it constitutes a significant weakness. Hence CFS does not provide good candidate for secure code-based signature schemes. Over the last decade, new signatures were proposed. We can classify these attempts among different approaches.

Hash and sign. On one hand, there is the *hash and sign* approach. Just like McEliece encryption, this relies on the fact that one can use encoding (or equivalently computing a syndrome) as a trapdoor one-way function. Given a parity-check matrix $\boldsymbol{H}$ of a code $\mathscr{C}$, the function that takes as input a vector $\boldsymbol{e} \in \mathbf{F}_q^n$ of weight $\mathbf{w}_H(\boldsymbol{e}) = t$ and returns $\boldsymbol{H}\boldsymbol{e}^\intercal$ is hard to invert, if $\boldsymbol{H}$ is a random matrix. But if one knows the special structure of the code $\mathscr{C}$, it is possible to efficiently decode the syndrome (hence finding a pre-image $\boldsymbol{e}$ for a given output).

Hence, to sign a message $\boldsymbol{m}$, the signature consists in displaying a vector $\boldsymbol{e} \in \mathbf{F}_q^n$ of weight t such that $\boldsymbol{H}\boldsymbol{e}^\intercal = \mathcal{H}(\boldsymbol{m})$, where $\mathcal{H}$ denotes some hash

function. The matrix $\boldsymbol{H}$ is public and hence anyone can verify that the signature is valid, but only a person who knows the secret structure of the code can produce a valid signature. Such signature schemes have large public keys but rather compact signatures.

This approach is used in the CFS proposal [CFS01] with Goppa codes. The RankSign signature [GRSZ14] also uses this paradigm, in rank metric. This scheme was attacked in [DT18]. In lattice-based cryptography, this approach in developed in [GPV08]. More recently, the Wave signature [DST19] was introduced. This scheme uses ternary codes in Hamming metric, but the goal is to decode errors of very large weight rather than small weight. This problem is discussed in Chapter 9.

Fiat–Shamir. In 1986, Fiat and Shamir introduced a protocol to transform an identification scheme into a signature. An identification scheme works as follows. A prover wants to prove to a verifier that he knows a secret. The prover first sends some initial information, a commitment. Then the verifier sends him a challenge. The prover returns his response to the challenge. The verifier can check that the response is consistent with the challenge and with the initial commitment.

The idea of the Fiat–Shamir transform is to get rid of the interaction with the verifier. Instead of having the verifier send a challenge, the challenge is derived from the commitment using a hash function. Hence, the prover does not choose the value of the challenge. This yields a signature scheme.

We can distinguish two families of signatures using the Fiat–Shamir transform. First, the signatures using a zero-knowledge identification scheme. The first code-based signature using this idea was proposed by Stern [Ste93], followed by Veron [Vér96]. Such signatures enjoy a small public key, but the main drawback is that each run of the identification protocol only proves that the prover is the legitimate user with constant probability, *e.g.* with probability $2/3$ in the case of Stern's protocol. Hence the protocol has to be repeated numerous times to amplify the result and ensure a negligible soundness error. As a consequence, even with some improvements, this yields large signatures (tenths of kilobytes for 128 security bits) [AGS11].

To overcome this pitfall, Lyubachevsky proposed that instead of trying to obtain a signature as a series of independent binary challenges, one should try to use the lattice structure to combine these into one single challenge, awaiting for one single response, hence saving communication cost, which yields more compact keys [Lyu09]. His framework works well for lattice-based signatures. There has been several attempts to adapt this idea to Hamming-metric code-based schemes [Per12; FRXKM+17; Per18; SHMWW20; LXY20]. All of these have been subject of attacks. There seems to be an inherent difficulty

to "rerandomize" the instances. Hence, whether it is possible to adapt this framework to create Hamming-metric code-based signature schemes remains an open problem. For now, the only successful adaptation using codes in the Durandal scheme [ABGHZ19] using rank-metric. This approach is promising but the security relies on an *ad hoc* problem which requires further study.

Code-based cryptography is a rich field, both for the design and analysis of cryptographic primitives. In the rest of this document, we will address different aspects of the security of code-based encryption schemes.

Part I

Key-recovery attacks on post-quantum cryptosystems: analysis of probabilistic constructions

Chapters

Chapter 2

Moderate density parity-check codes

In the previous chapter, we have seen how error-correcting codes can be used to create public key encryption schemes. The most important construction is the McEliece scheme. This scheme relies on the choice of a particular family of codes, having some internal structure to allow efficient decoding. The original proposal is to instantiate the McEliece scheme using Goppa codes. But the main drawback is the large size of the public key. Therefore, there has been several proposals to instantiate the McEliece scheme with other families of codes.

In this chapter, we introduce one of the most promising choices to replace Goppa codes in the McEliece scheme: moderate density parity-check (MDPC) codes. We explain how these codes are constructed, how to decode in these codes, and the properties that make them good candidates to build efficient post-quantum cryptosystems. In the next chapter, we will conduct a thorough analysis of a cryptosystem relying on MDPC codes and study some possible weakness of this scheme.

Contents

2.1 Introduction to MDPC codes

2.1.1 LDPC codes

MDPC codes find their origin in low density parity-check (LDPC) codes, which are very similar. Like most families of error-correcting codes, LDPC codes originally appeared in the context of information theory. They were introduced by Gallager in [Gal63]. Their name is pretty self-explanatory: LDPC codes are linear codes admitting a particularly sparse parity-check matrix.

Definition 2.1 (LDPC codes [Gal63])**.** A low density parity-check (LDPC) code is a binary linear code admitting a parity-check matrix for which the number of 1's on each line and column is bounded by some constants (very small compared to the length of the code, usually $\leqslant 10$).

The simplicity of this definition is striking, as well as the lack of structure, compared to algebraic codes which were more usual at the time. From a cryptographic point of view, this is certainly an advantage (we will come back to that) but this was probably too unusual and certainly explains the lack of interest for LDPC codes in the decades following their definition.

Despite their very competitive error-correction capacity and their efficient decoding algorithm, LDPC codes were almost forgotten, until MacKay and Neal "rediscovered" them in the 90's [MN96]. Even at this time, they did not draw a lot of attention, as most of the research of this field was dedicated to the recently introduced turbo-codes [BGT93], that were used in new telecommunication standards. It is only in the early 2000's that LDPC were found more appealing than turbo-codes. Since then, LDPC codes found numerous applications in telecommunication.

The idea of using such codes to instantiate the McEliece scheme is due to Monico, Rosenthal and Shokrollahi in a 2000 paper [MRA00]. However, in this paper, the authors note that if the rows of the parity-check matrix are too sparse (and it is indeed the case for LDPC codes used in telecommunication), it means that there exist codewords of extremely low weight in the dual of the code, and hence can serve as a distinguisher.

2.1.2 MDPC codes

Following this unsuccessful attempt, two proposals were made to thwart this attack [BC07; BBC08]. The general idea is to multiply the parity-check matrix of the LDPC code by a sparse invertible matrix, hence increasing the weight of the dual code. An unfortunate design choice for this invertible matrix led to a cryptanalysis of the first such proposal in [OTD08] but the general idea remains correct.

In 2013, Misoczki, Tillich, Sendrier and Barreto [MTSB13] proposed to replace this two-step process (first generating a low density parity-check matrix and then multiplying it by a sparse invertible matrix) by directly considering the code admitting a parity-check matrix with rows that would be sparse, but still dense enough to avoid the attack. Their computation shows that the row weight should scale in $\mathcal{O}(\sqrt{n \log n})$ for a code of length n, whereas the parity-check matrix of LDPC codes has constant row weight. They name this construction *medium density parity-check* (MDPC) codes.

Definition 2.2 (MDPC codes [MTSB13])**.** A moderate density parity-check (MDPC) code is a binary linear code of length n admitting a parity-check matrix with constant row weight w where $w = \mathcal{O}(\sqrt{n \log n})$.

Remark 2.3. *In this work we will only consider binary MDPC codes, although the definition could well be generalised to q-ary codes for a larger value of q.*

It is important to note that the choice of increasing the row weight lowers the decoding capacity, compared to LDPC codes which achieve the best possible trade-off. Hence, MDPC codes are not very interesting for telecommunication. In such a sense, MDPC codes are the first family of error-correcting codes designed exclusively for their application in cryptography.

Compared to "enhanced LDPC" codes proposed in [BBC08], generating the parity-check matrix directly leaves less structure, and hence less room for potential attacks (see for instance [APRS20] exploiting this weakness).

2.1.3 The quasi-cyclic structure

Remember that the main drawback of the McEliece scheme instantiated with Goppa codes is the large size of the public key. To make MDPC codes a competitive alternative, one needs to achieve significantly better public key size. Here, the public key corresponds to a parity-check matrix of the code that does not reveal enough structure to decode efficiently.

The interest of MDPC (and before them LDPC) codes relies on the fact that one can use quasi-cyclic instances of these codes. Here, quasi-cyclic means that its generator (resp. parity-check) matrix can be represented as the concatenation of several circulant matrices.

Definition 2.4 (Circulant matrix)**.** An $m \times m$ square matrix is a circulant matrix if for any $i \in [\![2, m]\!]$, the i-th line is a cyclic shift of the $(i-1)$-th line.

Due to this definition, a circulant matrix is entirely defined by its first line.

Definition 2.5 (Quasi-cyclic matrix)**.** An $n \times k$ matrix $\boldsymbol{M}$ is a quasi-cyclic matrix of order m if it can be written as a block-matrix, where each block is a circulant matrix of size $m \times m$, *i.e.* if there exist $m \times m$ circulant matrices $(\boldsymbol{M}_{i,j})$ such that

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{M}_{1,1} & \boldsymbol{M}_{1,2} & \cdots & \boldsymbol{M}_{1,n_0} \\ \boldsymbol{M}_{2,1} & \boldsymbol{M}_{2,2} & \cdots & \boldsymbol{M}_{2,n_0} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{M}_{k_0,1} & \boldsymbol{M}_{k_0,2} & \cdots & \boldsymbol{M}_{k_0,n_0} \end{bmatrix}.$$

Definition 2.6 (Quasi-cyclic code)**.** A linear code is a quasi-cyclic (QC) code of order m if it admits a parity-check matrix that is a quasi-cyclic matrix of order m.

Proposition 2.7. *Admitting a quasi-cyclic parity-check matrix is equivalent to admitting a quasi-cyclic generator matrix. This can be reformulated as: the dual of a quasi-cyclic code is a quasi-cyclic code.*

Remark 2.8. *Note that, originally, the definition of a quasi-cyclic of order m is a code such that any cyclic shift of a codeword by m places is again a codeword. But, we can easily see that, up to permutation of the columns, such a code has the quasi-cyclic property stated in our definition.*

A quasi-cyclic code can be represented by the first line of each circulant submatrix. Hence, keeping in mind that the public key of a McEliece cryptosystem is a parity-check matrix of the code, the size of the public key is now linear in the size of the code, and not quadratic as it is when you have to provide the full matrix. This properties is a clear motivation to use quasi-cyclic codes in code-based cryptosystems. Moreover, the algebra of $m \times m$ binary circulant matrices is isomorphic to the algebra of polynomials modulo $X^m - 1$ over $\mathbf{F}_2$, which yields efficient computations.

The idea of using quasi-cyclic (QC) codes in cryptography was first stated by Gaborit in [Gab05] using subcodes of BCH codes, but was attacked in [OTD08]. Quasi-cyclic alternant codes were proposed in [BCGO09], but an attack was proposed in [FOPT10]. This shows that the quasi-cyclic structure can create some weakness if applied to the wrong family of codes, especially when used with algebraic codes. The first paper suggesting to use quasi-cyclic LDPC codes is [BC07]. We already stated that this proposal is insecure but this is due to the choice of LDPC codes and not to the quasi-cyclic structure.

The later proposal [BBC08] and the seminal article introducing MDPC codes [MTSB13] both consider quasi-cyclic codes to reduce the key size. Indeed, the quasi-cyclic structure does not seem to affect the security reduction for MDPC codes [Sen10].

Remark 2.9. *A similar idea studied at the time in order to reduce the public key size is to use quasi-dyadic codes, for instance in* [*MB09*]*. But this does not apply to MDPC codes.*

Notation 2.10. *Quasi-cyclic MDPC* (*resp. LDPC*) *codes are denoted QC-MDPC* (*resp. QC-LDPC*) *codes.*

2.2 Decoding MDPC codes

We have seen that MDPC codes enjoy very little algebraic structure, together with a good correction capacity. Moreover, in the quasi-cyclic setting, their parity-check matrix can be described with a small amount of information, *i.e.* they enjoy short public key.

This seems promising for potential use in a public key cryptosystem. But we have seen that a key property for a family of codes to be used to instantiate the McEliece scheme is to have an efficient decoding algorithm relying on the structure of the code, that serves as trapdoor 1.3.1.

LDPC and MDPC codes have such a decoding algorithm, which we will describe here.

2.2.1 The bit-flipping algorithm

The mail algorithm used to decode LDPC codes (and later MDPC) codes is introduced by Gallager in [Gal63] and known as the *bit-flipping* algorithm. The main idea is the following.

2.2.1.1 General idea

Given a noisy codeword $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e} \in \mathbf{F}_2^n$, where $\boldsymbol{c} \in \mathscr{C}$ and $\boldsymbol{e}$ is a vector or small Hamming weight w, and a sparse parity-check matrix $\boldsymbol{H} \in \mathbf{F}_2^{r \times n}$ of the code $\mathscr{C}$, the goal is to find $\boldsymbol{e}$. Here, we consider that $\boldsymbol{H}$ meets the criterion of an LDPC code, *i.e.* each row has a very small weight.

We compute the value $\boldsymbol{s} \in \mathbf{F}_2^r$ such that $\boldsymbol{s}^\intercal = \boldsymbol{H}\boldsymbol{y}^\intercal$. This is called the *syndrome*. Let us denote s_i the i-th bit of $\boldsymbol{s}$. The value of s_i corresponds to the scalar product $\langle \boldsymbol{h}^{(i)}, \boldsymbol{y} \rangle$, where $\boldsymbol{h}^{(i)}$ denotes the i-th row of $\boldsymbol{H}$.

Note that because $\boldsymbol{c} \in \mathscr{C}$, $\boldsymbol{s}^\intercal = \boldsymbol{H}\boldsymbol{y}^\intercal = \boldsymbol{H}\boldsymbol{e}^\intercal$. Hence s_i is equal to $\langle \boldsymbol{h}^{(i)}, \boldsymbol{e} \rangle$. If $\boldsymbol{e} = 0$, then $\boldsymbol{s} = 0$. Hence, each bit set to 1 in $\boldsymbol{s}$ is due to some errors in $\boldsymbol{e}$. The goal is to find these errors.

For each i, the value of s_i indicates how many errors bits in $\boldsymbol{e}$ are contained in the support of the i-th row $\boldsymbol{h}^{(i)}$ of $\boldsymbol{H}$. More exactly, because the scalar product is computed in $\mathbf{F}_2$, it indicates the parity of the number of error. But two elements make this information useful. First, by definition of LDPC codes, the support of each row of $\boldsymbol{H}$ is very small ($\leqslant 10$). Moreover, the error $\boldsymbol{e}$ is sparse. Hence, if $s_i = \langle \boldsymbol{h}^{(i)}, \boldsymbol{e} \rangle = 1$, with high probability this indicates that one of the bits in the support of $\boldsymbol{h}^{(i)}$ corresponds to an error in $\boldsymbol{e}$. It could always be three, or five, or any other odd number, but this is less likely. Anyway, at least one of them contains an error. On the other hand, if $s_i = 0$, it means that there is an even number of error positions in $\boldsymbol{e}$ contained in the support of $\boldsymbol{h}^{(i)}$. This number is likely zero, although it could still be a positive even number. Therefore, when $\boldsymbol{H}$ and $\boldsymbol{e}$ are sparse enough, if $s_i = 1$, it is more likely that a position of $\boldsymbol{e}$ in the support of $\boldsymbol{h}^{(i)}$ is equal to 1.

Because we know the value of s_i for every i, we can use all these indicators to assign to each bit of $\boldsymbol{e}$ a number measuring the likelihood that this position is an error. The most straightforward way to use this information is by counting, for each position, how many times it is involved in the support of an unsatisfied equation, *i.e.* computing for each $j \in [\![1, n]\!]$ the quantity

$$\sigma_j \stackrel{\text{def}}{=} \#\{i \in [\![1, r]\!] \mid s_i = 1 \text{ and } j \in \mathbf{Support}(\boldsymbol{h}^{(i)})\}.$$

The higher the value σ_j is, the more likely it is that the j-th bit of $\boldsymbol{e}$ is equal to 1. This is the key element of the bit-flipping algorithm. Then, there are different variants depending on how to use this information. The most basic form of the algorithm is the following:

1. compute the syndrome $\boldsymbol{s}$;
2. compute σ_j for all $j \in [\![1, n]\!]$,
3. flip the bit of $\boldsymbol{e}$ corresponding to the highest observed value of σ_j,
4. repeat from step 1 until $\boldsymbol{s} = \mathbf{0}_r$.

More formally, we obtain Algorithm 1

Remark 2.11. *Note that these algorithms is presented here for general LDPC/MDPC codes, hence the entire parity-check matrix is provided as input, but when used for quasi-cyclic codes, one only needs to provide the first line of the matrix as input, and the rest of the matrix can be deduced by performing cyclic shifts.*

Algorithm 1: Step-by-step bit-flipping decoding algorithm

Input: $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbf{F}_2^n$, $\boldsymbol{H} = (\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(n)}) \in \mathbf{F}_2^{r \times n}$

$\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{c}^\intercal$ `// compute the syndrome`

while $\boldsymbol{s} \neq \mathbf{0}_r$ **do**

  for $j \in [\![1, n]\!]$ **do**

    $\sigma_j \leftarrow 0$

  for $i \in [\![1, r]\!]$ *such that* $s_i = 1$ **do**

    for $j \in \mathbf{Support}(\boldsymbol{h}^{(i)})$ **do**

      $\sigma_j \leftarrow \sigma_j + 1$ `// compute the counters` σ_j

  for $j \in [\![1, n]\!]$ *such that* $\sigma_j = \max_\ell\{\sigma_\ell\}$ **do**

    $y_j \leftarrow y_j \oplus 1$ `// flip the` j^{th} `bit`

  $\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{y}^\intercal$

return $\boldsymbol{y}$

This description explains the name of the *bit-flipping* algorithm. It turns out that this very simple algorithm performs extremely well on LDPC codes. Moreover, the same algorithm works to decode MDPC codes.

Here are a few remarks about this algorithm.

1. This algorithm is extremely simple to understand and implement.

2. This is a probabilistic algorithm. There is *a priori* no certainty that the algorithm will end and output the smallest possible $\boldsymbol{e}$ such that $\boldsymbol{y} - \boldsymbol{e} \in \mathscr{C}$. Moreover, the algorithm may never terminate. We can only observe (or prove under some hypothesis) that the algorithm terminates most of the time with the expected output, under some sparseness conditions on the input.

3. This is an iterative algorithm, the number of iterations is unknown and differs depending on the entry.

The first remark is an advantage in the context of cryptography. But the second (and to a lesser extent third) remark is a serious issue, which will be addressed in the next section and extensively discussed in the next chapter.

Many variants of this algorithm exist, to improve its performance. For instance, computing all the values s_i to flip only one bit at the end of the loop does not seem very efficient. We present here a variant that flips multiple bits in the same loop.

2.2.1.2 Threshold algorithms

The idea is to flip all bits corresponding to positions for which σ_j is higher than some threshold b, instead of only flipping the bit corresponding to the maximal value of σ_j. The value of this threshold b is given as a parameter of the algorithm. See Algorithm 2.

Algorithm 2: Threshold bit-flipping decoding algorithm

Input: $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbf{F}_2^n$, $\boldsymbol{H} = (\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(n)}) \in \mathbf{F}_2^{r \times n}$, $b \in \mathbf{N}$

$\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{c}^\intercal$ `// compute the syndrome`

while $\boldsymbol{s} \neq \boldsymbol{0}_r$ **do**

  for $j \in [\![1, n]\!]$ **do**

    $\sigma_j \leftarrow 0$

  for $i \in [\![1, r]\!]$ *such that* $s_i = 1$ **do**

    for $j \in \mathbf{Support}(\boldsymbol{h}^{(i)})$ **do**

      $\sigma_j \leftarrow \sigma_j + 1$ `// compute the counters` σ_j

  for $j \in [\![1, n]\!]$ *such that* $\sigma_j \geqslant b_t\}$ **do**

    $y_j \leftarrow y_j \oplus 1$ `// flip the` j^{th} `bit`

  $\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{y}^\intercal$

return $\boldsymbol{y}$

The threshold value should not be considered as a fixed value but more as a function that may depend on every information available at the time. In such a sense, the initial step-by-step can be considered as a special case, where the threshold at each step is defined as the maximal value of the counters σ_j. This corresponds to a rather conservative choice.

Another approach suggested in [MTSB13] is to take at each iteration the maximal value of the counters, minus a small constant (the proposed value is 5) to speed up the process.

Another approach is to have a fixed precomputed threshold value (given as input), which depends only on the iteration count. Indeed, during the first iteration of the algorithm, most errors are corrected and the number of incorrect equations drops significantly. Hence for the second iteration, it is necessary to consider a lower threshold value. Therefore, it makes sense to have the value of the threshold depend on the number of iterations. This configuration is known as *fixed threshold* decoding. It is the configuration used in [MTSB13] and [Cho16].

Another interesting possibility is to make the value of the threshold depend on the iteration counter but also on the syndrome weight. This choice, referred

to as *variable threshold* decoding, improves the efficiency of the algorithm and lowers the decoding failure rate, as explained in [CS16b].

2.2.2 The decoding failure rate

As we can see, the decoding algorithms of LDPC/MDPC codes include a while loop, and the number of iterations to decode a word is uncertain. In fact, the decoding of a word could even create an infinite loop. In practice, to avoid infinite loops, after a certain number of iterations, the decoding algorithm stops and returns an error.

Algorithm 3: Threshold bit-flipping decoding algorithm, with a bounded number of iterations

Input: $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbf{F}_2^n$, $\boldsymbol{H} = (\boldsymbol{h}^{(1)}, \ldots, \boldsymbol{h}^{(n)}) \in \mathbf{F}_2^{r \times n}$, $b \in \mathbf{N}$, $N \in \mathbf{N}$

1 $\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{c}^\intercal$ `// compute the syndrome`
2 **for** $t \in [\![1, N]\!]$ **do**
3 Compute the σ_j's `// See lines 3 to 7 in Alg. 2`
4 **for** $j \in [\![1, n]\!]$ *such that* $\sigma_j \geqslant b_t\}$ **do**
5 $y_j \leftarrow y_j \oplus 1$ `// flip the` j^{th} `bit`
6 $\boldsymbol{s}^\intercal \leftarrow \boldsymbol{H} \cdot \boldsymbol{y}^\intercal$
7 **if** $\boldsymbol{s} = \boldsymbol{0}_m$, **then**
8 **return** $\boldsymbol{y}$
9 **return** $\perp$

The parameters (weight of the error and thresholds) are chosen in such a way that this event, which is called a *decoding failure*, is very rare. But one should clarify the meaning of *rare* here. Indeed, from the user's point of view, an algorithm that would fail one out of a million times could be considered as good enough in most use-cases. For instance, with the parameters proposed in the first version of the BIKE submission claim a decoding failure rate (DFR) of 10^{-7}. This claim is based on experimental results.

Moreover, allowing a large number of loops reduces the decoding failure rate (DFR) but makes the decoding algorithm more time consuming. Therefore, one tries to reach a low DFR while keeping the number of iterations as small as possible.

However, as we will see in the next chapter, decoding failures can lead to serious security issues. Hence, to claim a security level of λ security bits for a scheme, one should require a DFR of $2^{-\lambda}$. This is (by definition) out of reach

of simulations. Hence one needs a theoretical estimate, if not a proof, of the low DFR of a cryptosystem.

2.2.3 Other decoders

We will not go into the details but there is a long line of work on improving the decoding algorithms for LDPC/MDPC codes.

In his original paper on LDPC codes [Gal63], Gallager also proposes a soft-decision algorithm, *i.e.* instead of taking the (hard) decision to flip a bit or not, the likelihood that a certain position of $\boldsymbol{e}$ is an error is represented by a real value between 0 and 1. This is natural in this context of statistical decoding. One of these soft-decision algorithms is the *sum-product* algorithm, but there exist other ways to evaluate this likelihood. However, such soft-decision algorithm have not been used in the context of code-based cryptography.

Nevertheless, the latest attempts to improve the MDPC decoders for a cryptographical use (especially in order to reduce the decoding failure rate and improve the efficiency) make some use of the idea of soft-decoding. Depending the value of the threshold they partition the positions in different groups: certain positions are most certainly to be flipped, others are considered as probable errors but for which the decision is postponed. Some of these new variants of decoders are discussed at the end of Chapter 3.

Finally, there are a lot of details regarding the implementation of these algorithms that can have consequences. For instance the choice to update the syndrome directly when a bit is flipped (*in place* decoding) or to flip the bits and update the syndrome later (*out of place* decoding, as it is the case in the examples of this section). We will see in the next chapter that this subtle differences sometimes matter.

2.3 QC-MDPC schemes

2.3.1 QC-MDPC McEliece

We describe here the McEliece scheme using QC-MDPC codes. This cryptosystem was first introduced in [MTSB13], then further studied in [BGGMP+17]. The "BIKE" [ABBBB+17] submission for the NIST post-quantum standardisation process mainly relies on this cryptosystem.

The general idea is to instantiate the McEliece scheme presented in Chapter 1 with QC-MDPC codes of rate 1/2, *i.e.* codes for which there exists a parity-check matrix consisting in the concatenation of two circulant block-matrices. We provide here details about this cryptosystem.

2.3.1.1 Parameters.

The QC-MDPC McEliece scheme uses four parameters:

- n the length of the code;
- k the dimension of the code;
- w the weight of each row of the sparse parity-check matrix of the code;
- t the number of errors.

The code is chosen such that $n = 2k$. The notation $r = n - k$ is sometimes used and has been used previously in this chapter, but in this case, $r = k$. We have already stated that for MDPC codes, the row weight is usually chosen such that $w = \mathcal{O}(\sqrt{n})$. The number of errors t must be chosen such that the bit-flip decoder can efficiently decode t errors. This usually leads to $tw = \mathcal{O}(n)$, hence $t = \mathcal{O}(\sqrt{n})$. Note that k should be prime to prevent attacks exploiting non-prime quasi-cyclicity such as [FL08]. Finally, w will be chosen to be an even integer so that the weight of the rows can split evenly in both parts of the quasi-cyclic parity-check matrix.

Table 2.1 shows the parameters suggested in [MTSB13].

Table 2.1: Parameters proposed for QC-MDPC McEliece [MTSB13]

security level	n	k	w	t
80	9602	4801	90	84
128	20326	10163	142	134
256	65542	32771	274	264

2.3.1.2 Key Generation

Note that in practice, there is no need to compute the full matrices $\boldsymbol{H}$ and $\boldsymbol{Q}$. The vector $\boldsymbol{q}$ can directly be computed from $\boldsymbol{h}_0$ and $\boldsymbol{h}_1$. The public key $\boldsymbol{q}$ (resp. the secret key $(\boldsymbol{h}_0, \boldsymbol{h}_1)$) is enough to describe the matrix $\boldsymbol{G}$ (resp. $\boldsymbol{H}$), but considerably shorter. Here, the size of the public key is exactly n.

Moreover, $\boldsymbol{G}$ is a generator matrix of the code defined by the parity-check matrix $\boldsymbol{H}$. Hence both matrices describe exactly the same code. However, the matrix $\boldsymbol{H}$ is sparse (it follows the definition of an MDPC code) whereas the matrix $\boldsymbol{G}$ is dense.

2.3.1.3 Encryption

The encryption is exactly like any McEliece-based cryptosystem: we encode the message and add a random error.

Algorithm 4: QC-MDPC key generation

Input: Parameters (n, k, w, t) corresponding to the desired security parameter.
Output: Public key pk, secret key sk.

1 Randomly generate $\boldsymbol{h}_0, \boldsymbol{h}_1 \in \mathbf{F}_2^k$, both of weight $w/2$.
2 Let $\boldsymbol{H} = [\boldsymbol{H}_0|\boldsymbol{H}_1]$ where $\boldsymbol{H}_i$ denotes the $k \times k$ circulant matrix obtained cyclically shifting $\boldsymbol{h}_i$.
3 Let $\boldsymbol{G} = [\boldsymbol{I}_k|\boldsymbol{Q}]$, where $\boldsymbol{I}_k$ is the $k \times k$ identity matrix and $\boldsymbol{Q} = (\boldsymbol{H}_1^{-1}\boldsymbol{H}_0)^\intercal$. The matrix $\boldsymbol{Q}$ is a circulant matrix. Let $\boldsymbol{q}$ denote its first row.
4 **return** *Secret key* $= \boldsymbol{q}$, *Private key* $= (\boldsymbol{h}_0, \boldsymbol{h}_1)$

Algorithm 5: QC-MDPC encryption

Input: Public key $\boldsymbol{q}$, message $\boldsymbol{m} \in \mathbf{F}_2^k$.
Output: Ciphertext $\boldsymbol{y} \in \mathbf{F}_2^n$.

1 Reconstruct $\boldsymbol{G} = [\boldsymbol{I}_k|\boldsymbol{Q}]$ from $\boldsymbol{q}$.
2 Generate a random error vector $\boldsymbol{e}$ of weight t.
3 Set $\boldsymbol{y} = \boldsymbol{m}\boldsymbol{G} + \boldsymbol{e}$.
4 **return** $\boldsymbol{y}$

2.3.1.4 Decryption

Just like in any McEliece-based cryptosystem, the decryption part consists in decoding. Here, we use the bit-flipping algorithm to decode.

Algorithm 6: QC-MDPC decryption

Input: Public key $\boldsymbol{q}$, private key $(\boldsymbol{h}_0, \boldsymbol{h}_1)$, ciphertext $\boldsymbol{y} \in \mathbf{F}_2^n$.
Output: Message $\boldsymbol{m} \in \mathbf{F}_2^k$.

1 Reconstruct $\boldsymbol{H} = [\boldsymbol{H}_0|\boldsymbol{H}_1]$ and $\boldsymbol{G} = [\boldsymbol{I}_k|\boldsymbol{Q}]$ from the private and public keys.
2 Run the bit-flipping algorithm to decode $\boldsymbol{y}$ with the MDPC matrix $\boldsymbol{H}$ as input.
3 Let $\boldsymbol{c}$ be the output of the decoder.
4 **return** $\boldsymbol{m}$ *such that* $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}$.

Note that the knowledge of the private key is necessary to decrypt since the bit-flipping algorithm requires the sparse public key to efficiently decode. This acts as the trapdoor for this public-key encryption scheme.

2.3.2 KEM vs. PKE

In the previous section, the QC-MDPC McEliece cryptosystem has been presented as a public key encryption scheme (PKE). However, [BGGMP+17] and the NIST submission [ABBBB+17] describe a key encapsulation mechanism (KEM) rather than a PKE. The difference is that a KEM is not used to directly transmit a message chosen by the user but to agree on a random secret (usually a key that will serve to transmit the message using symmetric encryption). Hence, the two schemes are very similar, but in a KEM, the content of the message is random by design and each key is only used once.

There is not much difference between what is described in [BGGMP+17; ABBBB+17] and our description above, apart from the fact that they use the Niederreiter setting to reduce the communication cost (see § 1.3.1.2).

But the cryptosystem as we have described it presents several weakness. The first and foremost is the fact that it is not resistant to key reuse: encrypting the same message several time with the same key leads to a trivial attack. Moreover, it does not achieve CCA security. Indeed, one could change only a few bits of the ciphertext (which is exactly equivalent to adding a few errors), use the oracle to decode and recover the message.

To circumvent this problem, one can use the Fujisaki-Okamoto transformation [FO99; HHK17] to transform the CPA-secure PKE into a CCA-secure KEM, hence achieving CCA security. This transformation forces the messages to be random, so this yields a KEM, not a PKE. This is exactly what is done in [BGGMP+17; ABBBB+17] and explains why these are key encapsulation mechanisms. However, the security proof requires that the decoding failure rate (DFR) should be $< 2^{-\kappa}$ where κ denotes the security parameter [SV20]. Therefore, lowering the DFR is particularly important to achieve CCA security. This will be discussed further in the next chapter.

2.3.3 Other MDPC-based schemes

The QC-MDPC McEliece cryptosystem that we presented is the straightforward instantiation of the McEliece scheme with QC-MDPC codes. However, there exist other similar code-based cryptosystems .

- The *Ouroboros* cryptosystem [DGZ17] is a variant of this scheme, using a quasi-cyclic matrix with three circulant matrices instead of two. In the initial version of the BIKE [ABBBB+17] submission, this was presented as an alternative version of the cryptosystem (known as BIKE-III) but the authors were then encouraged to propose a single scheme.

- The *LEDAcrypt* cryptosystem [BBCPS19] submitted to the NIST uses QC-LDPC codes multiplied by a sparse matrix (as described in § 2.1.2). An attack on this cryptosystem was proposed in [APRS20].
- There exist rank-metric equivalents to MDPC codes, named *low rank parity-check* (LRPC) codes [GMRZ13]. They can be used in a cryptosystem such as the ROLLO scheme [ABDGH+19] submitted to the NIST. These codes will be introduced in Chapter 4.
- Finally, there exist cryptosystems using random (non-MDPC) quasi-cyclic codes, as in the *Hamming Quasi-Cyclic* (*HQC*) submission to the NIST [AABBB+17b].

This list is non-exhaustive. For different reasons, all these submissions were eliminated after the first or second round of the NIST standardisation process. Only the BIKE submission using a QC-MDPC McEliece-based KEM was selected to the third round.

2.4 Security of QC-MDPC schemes

Let us discuss the security of the QC-MDPC McEliece cryptosystem as we presented it. The security of cryptosystems following the McEliece scheme has already been addressed in § 1.3.1.3. In this chapter, we explained that the security relies on two notions [Sen10]:

- the *message security*, *i.e.* the fact that decoding the message in the public code (without knowing anything particular about its structural properties) is computationally hard;
- the *key security*, *i.e.* the fact that the public key does not reveal anything about the structure of the code.

2.4.1 Message security

As explained in the introduction (§ 1.3.1.3), the message security is independent of the family of structured codes at stake. Here, we assume that one does not use anything specific to the secret structure of the code, hence the code is considered to be a random code. Hence, the message security means that decoding t errors in a random $[n, k]$-code is hard.

This is exactly the definition of the Syndrome Decoding problem introduced in Section 1.2.3. We have seen that this problem is known to be NP-hard [BMT78], and is conjectured to be hard on average [Ale11]. The most efficient

algorithm known to solve this problem are called *information set decoding* (ISD) algorithm and are extensively discussed in Chapter 8. For given parameters, one can compute the workfactor and check that the correct security level is reached.

However, in the case of QC-MDPC code, the MDPC structure is hidden, but the quasi-cyclic structure of the code is entirely public. Hence, the message security means that decoding in a random *quasi-cyclic* code is hard. Therefore, the security relies on a quasi-cyclic variation of the syndrome decoding problem.

In the quasi-cyclic case, there is no known proof of NP-hardness. Still, the only known difference with the general problem is that, in the quasi-cyclic setting, one can try not only to decode the ciphertext but to decode any on the cyclic shifts of the ciphertext. A solution to any of these attempts yields the message. Such a situation is known as *decoding one out of many* (DOOM) [Sen11a]. In this case, because there are k instances to solve, this provides a speedup factor of $\sqrt{k}$ in the exponent.

2.4.2 Key security

The key security is the fact that, given the public key, it is computationally hard to distinguish whether the matrix $\boldsymbol{G}$ is taken from a distribution of QC-MDPC code or from a distribution of random quasi-cyclic codes.

Problem 2.12 (Distinguishability of QM-MDPC codes - Decisional)**.** *Given an instance $\boldsymbol{q} \in \mathbf{F}_2^k$, does there exist vectors $\boldsymbol{h}_0, \boldsymbol{h}_1 \in \mathbf{F}_2^k$, with $|\boldsymbol{h}_0| = |\boldsymbol{h}_1| = w/2$, such that the quasi-cyclic matrix $\boldsymbol{H} = [\boldsymbol{H}_0|\boldsymbol{H}_1]$ is a parity-check matrix of the code generated by the matrix $\boldsymbol{G} = [\boldsymbol{I}_k|\boldsymbol{Q}]$, where $\boldsymbol{H}_0, \boldsymbol{H}_1$ and $\boldsymbol{Q}$ are the $k \times k$ circulant matrices generated by shifting $\boldsymbol{h}_0, \boldsymbol{h}_1$ and $\boldsymbol{q}$ respectively.*

This problem is believed to be hard, but there does not exist any theoretical result to back up this idea. However, it is often the research version of this problem that is stated. There is no proof of equivalence between the research problem and the decisional problem, but in practice there is no known approach to the decisional problem that does not consist in trying to find a solution to the research problem.

Problem 2.13 (Distinguishability of QM-MDPC codes - Research)**.** *Given an instance $\boldsymbol{q} \in \mathbf{F}_2^k$, find vectors $\boldsymbol{h}_0, \boldsymbol{h}_1 \in \mathbf{F}_2^k$, with $|\boldsymbol{h}_0| = |\boldsymbol{h}_1| = w/2$, such that the quasi-cyclic matrix $\boldsymbol{H} = [\boldsymbol{H}_0|\boldsymbol{H}_1]$ is a parity-check matrix of the code generated by the matrix $\boldsymbol{G} = [\boldsymbol{I}_k|\boldsymbol{Q}]$, where $\boldsymbol{H}_0, \boldsymbol{H}_1$ and $\boldsymbol{Q}$ are the $k \times k$ circulant matrices generated by shifting $\boldsymbol{h}_0, \boldsymbol{h}_1$ and $\boldsymbol{q}$ respectively.*

This problem is *ad hoc* to QC-MDPC codes. But it is conjectured in [MTSB13] that solving this problem is not easier than solving the (low-weight) codeword finding problem, defined as follows.

Problem 2.14 (Codeword finding problem - Research)**.** *Given the generator matrix of a code $\mathscr{C}$, and an integer w, find a codeword of weight w in $\mathscr{C}$.*

This problem is equivalent to the syndrome decoding problem, and is therefore NP-hard [MTSB13].

The best algorithm known to solve this problem are again the information set decoding (ISD) algorithms (see Chapter 8). Just like for the message security, the quasi-cyclic structure does not provide any known significant advantage other than the fact that the problem has k solutions (corresponding to the k cyclic shifts), hence the probability to find a solution is k times higher than in the general case. Hence, the complexity exponent is divided by a factor k.

2.4.3 Quantum security

We have seen that both aspects of security reduce to the Syndrome decoding and Low weight codeword problems. For both problems, the most efficient classical algorithms are the ISD algorithms. And in both cases, the only known significant improvement due to quantum computation consists in using Grover's algorithm to search the correct information sets in the ISD algorithms [Ber10]. The complexity exponent remains exponential in the length of the code, and hence the QC-MDPC scheme is considered to be resistant to quantum attacks.

2.4.4 Side-channel attacks and DFR

Finally, we have stated that the QC-MDPC is currently proposed as a key-encapsulation mechanism rather than a public key encryption scheme. This is due to the fact that the decoding algorithm may fail. These failures are rare, but not rare enough to avoid any security issue. Moreover, we have seen that the decoding algorithm is an iterative algorithm. If not implemented carefully, this algorithm can be subject to a type of attacks that is out of scope of the classical theoretical concerns: side-channel attacks. This is the subject of the next chapter.

Chapter 3

Side-channel attacks on the QC-MDPC cryptosystem

In the previous chapter we have seen that moderate density parity-check (MDPC) codes can be used to replace Goppa codes in a McEliece scheme. This yields acceptable public key sizes, good enough for most practical use-cases, especially if one uses quasi-cyclic (QC) MDPC codes. Remember that the large public key size is the main drawback of the original McEliece cryptosystem. Hence, code-based cryptosystems based on QC-MDPC codes are good candidates for post-quantum cryptography. As a result, the BIKE cryptosystem [ABBBB+17] which was submitted to the NIST post-quantum standardisation process, was selected in the third round .

The main issue with MDPC-based cryptosystems it the fact that the decryption phase may fail with very small, though non-negligible probability. This is not an issue from a user's perspective. For instance, a cryptosystem that would fail once in a million time can be acceptable for practical use in many situations. But in a 2016 paper, Guo, Johansson and Stankovski [GJS16] show that this small decryption failure rate can be exploited to extract information about the secret key, and therefore lead to security issues.

In this chapter, we analyse the parameters of the cryptosystem that are correlated with a high decryption failure rate, especially the syndrome weight. This analysis allows us to understand why the attack proposed in [GJS16] works and to propose two new side-channel attacks exploiting these correlations. Finally this analysis provides mitigation and guidelines for a safe implementation of the cryptosystem.

Related publication: Eaton, Lequesne, Parent and Sendrier, *QC-MDPC: A Timing Attack and a CCA2 KEM*, PQCrypto 2018 [ELPS18].

Contents

3.1 Key recovery attack on the QC-MDPC scheme

3.1.1 Side-channel attacks

Despite the cryptographers work to assess the computational hardness of the mathematical problems on which cryptographic schemes rely, there exists a class of attacks that consists in obtaining information directly from the monitoring of a device executing the cryptographic algorithm. For instance, the running time, the power consumption or even the sound of the execution of an algorithm can leak information about the value of some variables of the algorithm, and hence the key. This notion is introduced by Kocher [Koc96] who performed a timing attacks against RSA.

Such attacks are usually out of scope of the security models when one only focuses on the hardness of the mathematical problems. Indeed, mitigations exist to counter such attacks and are usually added at the implementation level. The most generic one is known as “masking” and consists in splitting the crucial information in several shares to prevent information leaks. But this

operation is costly. Therefore, it is important to understand which variables of the algorithm carry significant information that should absolutely be made inaccessible.

3.1.2 The QC-MDPC scheme

3.1.2.1 Parameters

In this chapter, we focus on the QC-MDPC cryptosystem as it was defined in the original paper [MTSB13] and further discussed in [BGGMP+17]. This scheme is detailed in the previous chapter and is at the core of the BIKE candidate for the NIST standardisation process [ABBBB+17]. Results stated in this chapter refer to the security levels corresponding to the parameter sets presented in Table 3.1. These are the parameters proposed in [MTSB13] (with a small modification in the parameters for 128 security bits as proposed in [BGGMP+17]). The parameters used in the latest version of BIKE differ slightly but are in the same range.

Table 3.1: Set of parameters for the QC-MDPC scheme used in this chapter.

Security level (bits)	n	k	w	t	Public key size (kB)
80	9 602	4 801	90	84	0.60
128	20 326	10 163	142	134	1.27
256	65 542	32 771	274	264	4.10

3.1.2.2 Choice of decoder

In the previous chapter, we have seen that the decryption phase of the QC-MDPC cryptosystem involves decoding in the QC-MDPC code, and that there exist many variants of decoding algorithms for such codes. The main principle remains that of Gallager's original bit-flip decoder for LDPC codes [Gal63]. But our goal is to study potential side-channel attacks, and such attacks are *a priori* specific to a choice of decoder (and can even be specific to an implementation). Therefore it makes sense to specify which variant of the decoder we will take into account.

The decoding algorithms studied in this chapter correspond to the state of the art decoders at the time of the first round of the NIST call for post-quantum standardisation, detailed in [CS16b]. There are two main design choices for such algorithms.

1. The thresholds. Some of these bit-flipping decoding algorithms have a threshold which depends only on the iteration count. They are referred to as *fixed threshold* decoders. Others have a threshold function that depends on the iteration count and the syndrome weight. These are referred to as *variable threshold* decoders. See Section 2.2 for details.

2. The implementation. An important implementation detail is whether the computation of the syndrome is made in-place or out-of-place. Again, the two variants have been detailed in Section 2.2.

For our analyses we mainly use the decoder denoted $\mathcal{B}$ in [MOG15] which corresponds to Gallager's original algorithm with an out-of-place implementation and fixed thresholds. But we also conducted some experiments on other decoders, to prove that the observed behaviour is not specific to an implementation, but to a whole family of decoders. The results show that the efficiency of the attacks will differ from one decoder to the other, and this information can help making design choices. The observations involving in-place decoders refer to decoder $\mathcal{D}_1$ fron [MOG15] which is the in-place equivalent of decoder $\mathcal{B}$.

As for the value of the thresholds, until now the thresholds were claimed as experimental results with no generic explanation on the way they were generated. Most of the time the thresholds are only proposed for one fixed set of parameters. Therefore, we proposed in [ELPS18, Appendix B] a generic way to derive fixed and variable thresholds for all parameter sizes. This is the thresholds we use in our experiments. Applying these rules to the parameters of Table 3.1 yields the following values for the fixed thresholds.

Table 3.2: Fixed threshold values used in the QC-MDPC decoder in this chapter. The i-th item of the sequence corresponds to the value of the threshold at the i-th iteration. The dots mean that the last value is repeated as much as necessary.

Security level (bits)	Sequence of fixed threshold values
80	30, 28, 26, 25, 23, ...
128	46, 43, 41, 40, 39, 37, 36, ...
256	83, 80, 77, 74, 72, ...

3.1.3 The GJS reaction attack

A paper from Guo, Johansson and Stankovski [GJS16] makes use of the decoding failure rate (DFR, defined in Section 2.2.2) to propose a reaction attack on

schemes involving the decoding of QC-MDPC codes. We refer to this as the GJS attack. The attack model assumes that an adversary is able to tell when such an error has occurred, for example because a request for resend is sent back.

3.1.3.1 Principle

The idea of the GJS attack involves the notion of a *distance* being present in a binary vector. Given a vector $\boldsymbol{v} \in \mathbf{F}_2^n$, we say that the distance δ is present in $\boldsymbol{v} = (v_1, \ldots, v_n)$ when $\boldsymbol{v}$ admits two non-zero bits distant from δ, *i.e.* if there exists $i \in \mathbf{N}$ such that $v_i = v_{i+\delta} = 1$. The distance is counted cyclically, *i.e.* the indices are considered modulo n.

The main idea of the GJS attack is the following.

Observation 3.1 (GJS, key observation)**.** *When a distance in the error vector used in a QC-MDPC encryption matches a distance in the secret key, a decoding failure is* less *likely to occur.*

Based on this observation, the authors of GJS propose an attack in two steps.

1. Observe a large number of error vectors that result in a decoding failure and deduce from this observation which distances are in the secret key
2. Reconstruct the secret key based on this information.

3.1.3.2 The distance spectrum

The main tool introduced in [GJS16] is the *distance spectrum* of a binary vector.

Definition 3.2 (Distance Spectrum)**.** The distance spectrum of a vector $\boldsymbol{v} \in \mathbf{F}_2^r$, denoted $\Delta(\boldsymbol{v})$, is the binary vector of length $\lfloor \frac{r}{2} \rfloor$ such its δ-th entry $\Delta(\boldsymbol{v})[\delta]$ is equal to 1 if and only if the distance δ is present in $\boldsymbol{v}$, *i.e.* if there exist two non-zero bits of $\boldsymbol{v}$ at distance δ. The distance are counted cyclically.

$$\Delta(\boldsymbol{v})[\delta] = 1 \qquad \Longleftrightarrow \qquad \exists (i,j), \begin{cases} 0 \leqslant i < j < r, \\ v_i = v_j = 1, \\ \min\{j-i, r-(j-i)\} = \delta \end{cases}$$

where v_i denotes the i^{th} entry of the binary vector $\boldsymbol{v}$.

Example 3.3. Let $\boldsymbol{v} = (0,1,0,1,0,0,1,0,1,0,0) \in \mathbf{F}_2^{11}$. Then we have $\Delta(\boldsymbol{v}) = (0,1,1,1,1)$.

Remark 3.4. *Note that any cyclic shift or reversal of a vector will result in the same distance spectrum.*

3.1.3.3 The attack

Consider an instance of the QC-MDPC cryptosystem (defined in Section 2.3) with the secret key $(\boldsymbol{h}_0, \boldsymbol{h}_1)$. The main step of the GJS attack is to compute $\Delta(\boldsymbol{h}_0)$ using Algorithm 7, which is detailed below. From this, one is able to compute the value of $\boldsymbol{h}_0$ and one can finally deduce $\boldsymbol{h}_1$ using elementary linear algebra.

Algorithm 7 is used to obtain the distance spectrum of the first half of the public key $\boldsymbol{h}_0$ by observing the decoding failures. For each distance δ, the value of the ratio $\mathsf{FailedDecoding}[\delta]/\mathsf{ObservedDecoding}[\delta]$ gives an estimate of the decoding failure rate for error vectors containing the distance δ. Using Observation 3.1, we can deduce from this if the distance $delta$ is in the spectrum of $\boldsymbol{h}_0$ or not.

Algorithm 7: The GJS CCA attack to find the distance spectrum

Input: An oracle $\mathcal{O}$ that, given a noisy codeword, returns $\top$ or $\perp$ wether decoding succeeded or failed, , the number of samples N, a threshold value T.

Create three tables ObservedDecoding, FailedDecoding and Spectrum of length $\lfloor k/2 \rfloor$ and initialize their entries to zero
for $i = 1$ *to* N `// Repeat for` N `random ciphertexts`
do
  Let $\boldsymbol{m}$ be a random message and $\boldsymbol{e} \leftarrow [\boldsymbol{e}_0 || \boldsymbol{e}_1]$ an error vector drawn uniformly at random
  Let $\boldsymbol{c} \leftarrow \mathsf{QCMDPC.Enc}(\boldsymbol{m}, \boldsymbol{e})$ and $b \leftarrow \mathcal{O}(\boldsymbol{c})$.
  for $\delta = 1$ *to* $\lfloor \frac{k}{2} \rfloor$ *s.t.* $\Delta(\boldsymbol{e}_0)[\delta] = 1$ **do**
    $\mathsf{ObservedDecoding}[\delta] \leftarrow \mathsf{ObservedDecoding}[\delta] + 1$.
    if $b = \perp$ **then**
      $\mathsf{FailedDecoding}[\delta] \leftarrow \mathsf{FailedDecoding}[\delta] + 1$.
for $\delta = 1$ *to* $\lfloor \frac{k}{2} \rfloor$ **do**
  if $\mathsf{FailedDecoding}[\delta]/\mathsf{ObservedDecoding}[\delta] < T$ **then**
    `// If the DFR is less than some threshold` T
    $\mathsf{Spectrum}[\delta] \leftarrow 1$
return Spectrum

Then, reconstruction $\boldsymbol{h}_0$ from $\Delta(\boldsymbol{h}_0)$ (up to a reversal or cyclic shift) can be done in reasonable time. This operation has been studied [GJS16; FHSZG+17] and tested in practice.

3.1.3.4 Complexity

In a CCA model, the attacker can choose the error pattern. Hence it is possible to choose error vectors where a particular distance δ arrives with an high multiplicity. This choice provides very good data to deduce the value of the spectrum efficiently. The authors of [GJS16] claim that observing $N = 2^{17}$ decoding is sufficient to find the secret key for the parameters claiming 80 security bits.

However this model is not very realistic, since in practice the QC-MDPC scheme is implemented together with a semantically secure transformation (such as the Fujisakiand–Okamoto transform [FO99]). With such a feature, the error pattern is the result of a hash function and hence cannot be imposed by the user. Therefore in the presentation of Algorithm 7 we considered that the error pattern is random. In this model, the authors of [GJS16] found that decoding $N = 2^{29}$ ciphertexts was sufficient to break the 80-bit classical parameter set, using Gallager's decoding algorithm.

In both cases, the complexity of the attack is dominated by the value N. The second phase of the algorithm (reconstructing $\boldsymbol{h}_0$ from its spectrum) has been analysed in [GJS16; FHSZG+17], and shown to be fairly fast and simple as compared to the first step, and is an entirely offline computation, requiring no communication.

Finally, in [GJS16] the author conjecture that using a more sophisticated decoding algorithm, would mean that N would have to be increased by an amount proportional to the difference in the decoding failure rate. They also conjecture that higher parameter sets would not significantly alter the effectiveness of the attack (for the same decoder), as the decoding failure rate does not significantly change.

3.1.3.5 Distance spectrum with multiplicity

An additional tool defined in [GJS16], is the *distance spectrum with multiplicity*. Although this is not directly needed to perform the attack, this object leads to an interesting observation and is useful for further analysis. The idea is to extend the definition of the distance spectrum to take into account the fact that some distances may appear more than once. This yields the following definition.

Definition 3.5 (Distance Spectrum with multiplicity)**.** The distance spectrum with multiplicity of a vector $\boldsymbol{h} \in \mathbf{F}_2^r$, denoted $\Delta^+(\boldsymbol{h})$, is a integer vector of length $\lfloor \frac{r}{2} \rfloor$ such that for every distance $1 \leqslant \delta \leqslant \lfloor \frac{r}{2} \rfloor$, its δ^{th} component $\Delta^+(\boldsymbol{h})[\delta]$ is the number of existing sets of two non-zero bits of $\boldsymbol{h}$ at distance δ. The distance are counted cyclically.

Example 3.6. Let $\boldsymbol{v} = (0,1,0,1,0,0,1,0,1,0,0) \in \mathbf{F}_2^{11}$. Then we have $\Delta^+(\boldsymbol{v}) = (0,2,1,1,2)$.

In general we can see that if a vector $\boldsymbol{v} \in \mathbf{F}_2^r$ has weight w, then the distance spectrum with multiplicity of $\boldsymbol{v}$ will be a vector of size $\lfloor r/2 \rfloor$ such that the sum of the entries of $\Delta^+(\boldsymbol{v})$ is $\binom{w}{2}$. The full knowledge of the spectrum with multiplicity allows to reconstruct the vector more efficiently.

Finding the distance spectrum with multiplicity of the secret key can be achieved by adapting Algorithm 7 to take into account the following observation.

Observation 3.7. *For a fixed key, the decoding failure rate for error vectors with δ in their distance spectrum is inversely proportional to the multiplicity of δ in the distance spectrum of the key.*

For large enough values of N, the decoding failure rate when the distance δ is present in the error clearly separates into bands. These band exactly correspond to the multiplicity of that distance in $\Delta^+(\boldsymbol{h}_0)$. This allows an attacker to recover $\Delta^+(\boldsymbol{h}_0)$, and thus the secret key.

3.2 Analysis

Our goal is to analyse the QC-MDPC scheme and its decoding to understand the causes of observations 3.1 and 3.7, as well as the evolution of the number of observed decoding needed to distinguish the different multiplicities.

We show that these phenomena can be explained by focusing on one variable: the weight of the syndrome. This parameter is somehow a natural parameter to analyze since all the decoding algorithms start by computing the syndrome, and some decoders are even designed to change the threshold value depending on the syndrome weight at each iteration [CS16b]. Let us see that we can explain Observations 3.1 and 3.7 with respect to this variable.

3.2.1 Expected syndrome weight

The QC-MDPC cryptosystem uses a parity-check matrix with two circulant blocks. However, the GJS attack presented in the previous section works by recovering the secret key on one block and then deduce the rest of the key. This suggests that the observed phenomenon is local to each block. Therefore, for the sake of simplicity of the analysis, in this section we will only consider a parity-check matrix made of one single circulant block in $\boldsymbol{H} \in \mathbf{F}_2^{k \times k}$ instead of two. We will see later that the practical results are the same. We denote by

$\boldsymbol{h} \in \mathbf{F}_2^k$ the first row of the matrix $\boldsymbol{H}$. The variable t still represents the weight of the error $\boldsymbol{e}$, so here the numerical value of t should be half its usual value.

We will now compare the expected value of the syndrome weight in two cases. In the first model, we do not make any hypothesis about the spectrum of the error nor the key. In the second model, we suppose that there is a distance that appears both in the spectrum of the key and in the spectrum of the error. We analyse how this affects the expected syndrome weight. For the sake of simplicity, we will study the case of distance 1 (*i.e.* two neighbour bits with non-zero value) but in fact this does not make any difference in the analysis.

3.2.1.1 Model 1. Without any hypothesis.

Let us suppose that we do not have any information on the key. For a random key vector $\boldsymbol{h}$ of size k and weight d and a random error vector $\boldsymbol{e}$ of size k and weight t, denote by $p(k,d,t)$ the probability that the scalar product in $\mathbf{F}_2$ is odd parity.

$$p(k,d,t) \stackrel{\text{def}}{=} \mathbb{P}\left[\langle \boldsymbol{h}, \boldsymbol{e}\rangle = 1 \bmod 2\right] = \sum_{\substack{i=0 \\ i \text{ is odd}}}^{d} \frac{\binom{d}{i}\binom{k-d}{t-i}}{\binom{k}{t}}.$$

If we reason like in the case of a (non quasi-cyclic) MDPC code where each row of the matrix is independent, the average syndrome weight of an error $\boldsymbol{e}$ and the parity-check matrix is k times the probability that a bit is non-zero (see [Cha17, page 91]).

$$\mathbb{E}\left[\mathbf{w}_H(\boldsymbol{H} \cdot \boldsymbol{e}^\intercal)\right] = k \cdot p(k,d,t).$$

Note that the independence hypothesis is not true for QC-MDPC codes, as all the rows of $\boldsymbol{H}$ are obtained by shifting $\boldsymbol{h}$. It is as if each row of the matrix was a random vector of length k and weight d. Still, this provides a good approximation of the syndrome weight.

3.2.1.2 Model 2. Case of a common distance in the spectra.

Now, suppose the key vector $\boldsymbol{h}$ has ℓ times two consecutive non-zero bits, *i.e.* $\Delta^+(\boldsymbol{h})[1] = \ell$. Let us observe the shifts of the vector.

$\mathsf{shift}(\boldsymbol{h}) =$	$1\,1$	$\boldsymbol{u}, \mathbf{w}_H(\boldsymbol{u}) = d-2$	ℓ times
$\mathsf{shift}(\boldsymbol{h}) =$	$1\,0$	$\boldsymbol{u}, \mathbf{w}_H(\boldsymbol{u}) = d-1$	$d-\ell$ times
$\mathsf{shift}(\boldsymbol{h}) =$	$0\,1$	$\boldsymbol{u}, \mathbf{w}_H(\boldsymbol{u}) = d-1$	$d-\ell$ times
$\mathsf{shift}(\boldsymbol{h}) =$	$0\,0$	$\boldsymbol{u}, \mathbf{w}_H(\boldsymbol{u}) = d$	$k-2d+\ell$ times

Consider an error vector $\boldsymbol{e}$ that has two consecutive non-zero bits, *i.e.* such that $\Delta(\boldsymbol{e})[1] = 1$. Up to permutation, we can suppose that these are the first two bits of the vector.

$$\boldsymbol{e} = \boxed{1}\boxed{1}\boxed{u, \mathbf{w}_H(u) = t-2}$$

To compute an estimate of the expected syndrome weight with this assumption on the form of $\boldsymbol{h}$ and $\boldsymbol{e}$, we suppose that the right-most part of the vector (denoted $\boldsymbol{u}$) behaves as if it was chosen uniformly at random among vectors of length $k-2$ and weight $\mathbf{w}_H(\boldsymbol{u})$. This is equivalent to the row-independence hypothesis formulated to compute the estimate in Model 1.

This yields the following estimate of the average syndrome weight of $\boldsymbol{e}$ with respect to the the parity-check matrix $\boldsymbol{H}$ generated by cyclic shifts of $\boldsymbol{h}$.

$$\begin{aligned}\mathbb{E}\left[\mathbf{w}_H(\boldsymbol{H}\cdot\boldsymbol{e}^\intercal)\right] = \quad & \ell \quad & p(k-2,d-2,t-2) \\ + \quad & 2(d-\ell) \quad & (1-p(k-2,d-1,t-2)) \\ + \quad & (k-2d+\ell) \quad & p(k-2,d,t-2)\end{aligned} \tag{3.1}$$

Again, it is important to stress that in both cases, the formulas are approximation. Indeed, in practice, the cyclic structure of the matrix induces a dependence between each row and hence yields to a covariance between the bits of the syndrome. Still, we will see that the approximation is close to the real value and we can neglect the correction term for the rest of the study.

3.2.1.3 Consequences

Link with the previous observations. The results stated above for a distance equal to one can be generalised to all other distances. Hence, the expression 3.1 is similar to Observation 3.7: for a distance δ in the spectrum of the error pattern, the expected value of the syndrome weight behaves linearly with respect to the multiplicity ℓ of the distance δ in the distance spectrum of the key.

Link with decoding failure. Moreover, we can understand how a common value in the distance spectrum of the key and the error is more prone to a decoding failure. Indeed, for each row of the parity-check matrix, the number of error positions involved in this equation is the size of the intersection between the support of a row and the support of the error. Due to the sparsity of the rows of the parity-check matrix and that of the error, the size of this intersection is often very limited. A non-zero bit in the syndrome means that the size of this intersection is odd, but most of the time this intersection is equal to 1, sometimes 3, rarely more. In such a case, there is one (or more) errors,

and the non-zero bit in the syndrome contributes to the correction of this error. On the other hand, when a bit of the syndrome is equal to zero, *i.e.* the size of the intersection is even, this is usually due to an empty intersection (*i.e.* this equation is not involved in any error) but it can also mean that the intersection is of size two (rarely more). This is exactly the tricky case because it corresponds to an error that will not be detected by the decoding algorithm. Indeed, in such a case, the error is not taken into account in the counters, and hence the error positions involved are less likely to be corrected. This increases the possibility of decoding failures. And unsurprisingly, this case corresponds exactly to the fact that a distance is present simultaneously in the spectrum of the error and in the spectrum of the secret key.

3.2.2 Experimental measures

Suppose that we only consider error patterns starting with distance δ in their spectrum. The syndrome weight is expected to be slightly different on average, depending on $\ell = \Delta^+(\boldsymbol{h})[\delta]$. Moreover, the expected value of the syndrome weight varies linearly with ℓ. Therefore, if we observe enough values of the syndrome weight, we can recover the value of ℓ.

Definition 3.8 (Average syndrome weight with multiplicity)**.** Let us denote by $\mathcal{D}_\ell$ the following set:

$$\mathcal{D}_\ell \stackrel{\text{def}}{=} \left\{ (\boldsymbol{h}, \boldsymbol{e}) \in \mathbf{F}_2^k \times \mathbf{F}_2^k, \mathbf{w}_H(\boldsymbol{h}) = d, \mathbf{w}_H(\boldsymbol{e}) = t, \Delta(\boldsymbol{e})[\delta] = 1, \Delta^+(\boldsymbol{h})[\delta] = \ell \right\}.$$

The average syndrome weight with multiplicity $\bar{\sigma}_\ell$ is the expectation of the syndrome weight for a uniform distribution of $(\boldsymbol{h}, \boldsymbol{e})$ over $\mathcal{D}_\ell$:

$$\bar{\sigma}_\ell \stackrel{\text{def}}{=} \mathbb{E}_{(h,e)\sim\mathcal{U}(\mathcal{D}_\ell)} \left[\mathbf{w}_H(\boldsymbol{H} \cdot \boldsymbol{e}^\intercal) \right].$$

From Equation (3.1) we know that we can approximate $\bar{\sigma}_\ell$ by

$$\begin{aligned} \bar{\sigma}_\ell = & \quad \ell & p(k-2, d-2, t-2) \\ & + \quad 2(d-\ell) & (1 - p(k-2, d-1, t-2)) \\ & + \ (k-2d+\ell) & p(k-2, d, t-2). \end{aligned}$$

with $p(k,d,t) \stackrel{\text{def}}{=} \sum_{\substack{i=0 \\ i \text{ is of odd}}}^{d} \frac{\binom{d}{i}\binom{k-d}{t-i}}{\binom{k}{t}}$

Comparison with measured values. The values of $\bar{\sigma}_\ell$ correspond to the different clusters that we can see on the figures. According to the approximation, the value of $\bar{\sigma}_\ell$ is linear in the multiplicity: $\bar{\sigma}_0 - \bar{\sigma}_\ell = \ell \cdot (\bar{\sigma}_0 - \bar{\sigma}_1)$. This is consistent with what we observe..

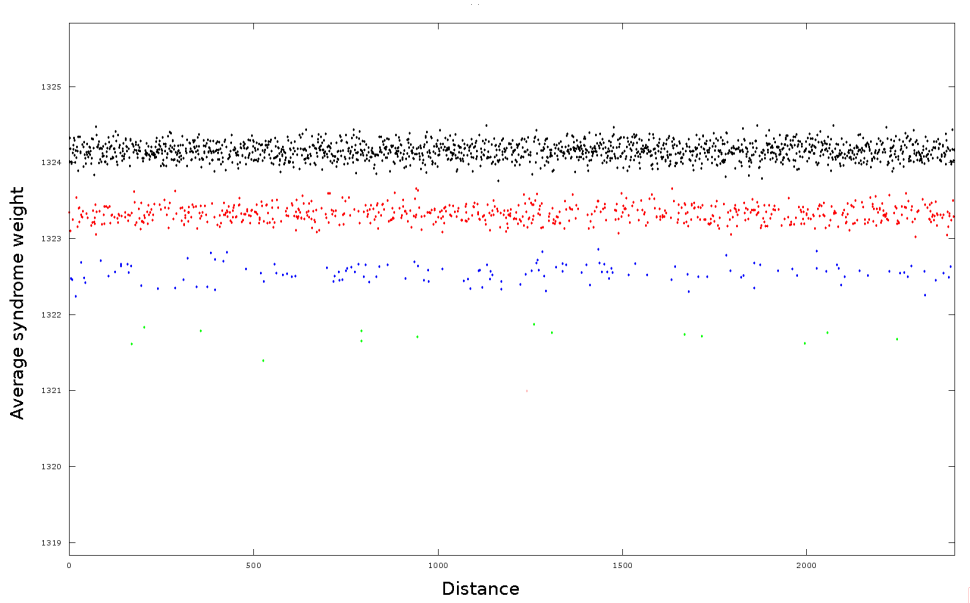


Figure 3.1: Attack on the syndrome weight (1 block) using the parameters for 80-bit security. For a fixed secret key, each data point represents the average syndrome weight for random errors that admit this distance in their spectrum. This attacks uses 10^5 error samples. The color of the distances indicate their multiplicity in the key spectrum (black = 0, red = 1, blue = 2, green = 3) and serve as a control.

With the usual parameters for 80-bit security, (here using $t = 42$ as there is only one block) we obtain $\bar{\sigma}_0 = 1324.23$ and $\bar{\sigma}_1 = 1323.28$.

When comparing these values to those measured on Fig. 3.1, we can see that the measured value of $\bar{\sigma}_0$ corresponds to the computed approximated value. For $\bar{\sigma}_1$, the value given by Equation (3.1) is slightly higher than the measured value. This difference reflects the independence hypothesis made to obtain the formulas. When performing the same experiment on parameters for LDPC codes, where the covariance between the is much smaller, the measures correspond exactly to the computed values.

As a consequence, the real value of the distance $\bar{\sigma}_0 - \bar{\sigma}_1$ is smaller than the one computed using Equation (3.1). Hence, the theoretical analysis gives an interesting bound on the relative distance $\varepsilon \stackrel{\text{def}}{=} \frac{\bar{\sigma}_0 - \bar{\sigma}_1}{k}$.

3.2.3 Required number of samples.

Each syndrome is the result of k scalar products between the error and a parity-check equation. When the error contains a distance present in the spectrum of the key with multiplicity ℓ, the average syndrome weight is $\bar{\sigma}_\ell$, this means that on average $\bar{\sigma}_\ell$ of the k parity-check equations are not verified. Hence, under the independence assumption, we can see each bit of the syndrome as a Bernoulli trial satisfied with probability $\frac{\bar{\sigma}_\ell}{k}$.

Here, our goal is to decide for each distance δ whether or not $\Delta(\boldsymbol{h})[\delta] = 1$. We do not care about the multiplicity. Formally, we want to distinguish $\mathcal{D}_0$ from $\cup_{\ell \geqslant 1} \mathcal{D}_\ell$. Let us denote $\mathcal{D}_{\geqslant 1} \stackrel{\text{def}}{=} \cup_{\ell \geqslant 1} \mathcal{D}_\ell$. We can define $\bar{\sigma}_{\geqslant 1}$ on $\mathcal{D}_{\geqslant 1}$ just like we defined $\bar{\sigma}_\ell$ on $\mathcal{D}_\ell$. The sets are disjoint so we have $\bar{\sigma}_{\geqslant 1} = \frac{\sum_{\ell \geqslant 1} \bar{\sigma}_\ell |\mathcal{D}_\ell|}{\sum_{\ell \geqslant 1} |\mathcal{D}_\ell|}$.

Hence, deciding whether a distance is in the spectrum of the key or not is just like distinguishing a random binary variable with success probability $p_0 \stackrel{\text{def}}{=} \bar{\sigma}_0$ from a random binary variable with success probability $p_1 \stackrel{\text{def}}{=} \bar{\sigma}_{\geqslant 1}$. This is a classic problem of hypothesis testing.

Remark 3.9. *Note that for our parameters, the size of $\mathcal{D}_\ell$ for $\ell \geqslant 2$ is negligible compared to $\mathcal{D}_1$, hence there is no practical need to distinguish $\bar{\sigma}_1$ from $\bar{\sigma}_{\geqslant 1}$.*

There is a lot of literature about hypothesis testing, and in particular a theorem from Chernoff [Che+52] concerning such cases. We reproduce it here as it is stated in [HMRR13, page 195].

Proposition 3.10 (Chernoff's bound)**.** *Let $0 < p < 1$, let $X_1, X_2, \ldots, X_N$ be independent binary random variables, with $\Pr[X_k = 1] = p$ and let $S_N = \frac{\sum_{k=1}^{N} X_k}{N}$. Then for any $t \geqslant 0$,*

$$\mathbb{P}\left[|S_N - p| \geqslant t\right] \leqslant 2\mathrm{e}^{-2Nt^2}.$$

This can be used to understand how the number of samples required to find the key evolves. Here we want to distinguish p_0 from p_1, we will use $\frac{p_0+p_1}{2}$ as the decision threshold. Chernoff's bound states that we should have $N \sim \frac{1}{\varepsilon^2}$ repeated Bernoulli trials for the decision test to be relevant, where $\varepsilon = |p_1 - p_0| = \frac{\bar{\sigma}_0 - \bar{\sigma}_1}{k}$ is the distance between the two outcomes.

To decide whether a particular distance δ is in the spectrum or not, we need to compute the mean of N Bernoulli trials, but each syndrome weight is already the sum of the results of k Bernoulli tests. Hence, we need to observe the weight of $\frac{N}{k}$ syndromes. These syndromes need to be in one of the $\mathcal{D}_\ell$, this means that the distance δ needs to be in the spectrum of the error pattern that generates the syndrome. As the error patterns are generated uniformly, we proceed by rejection sampling to ensure this condition.

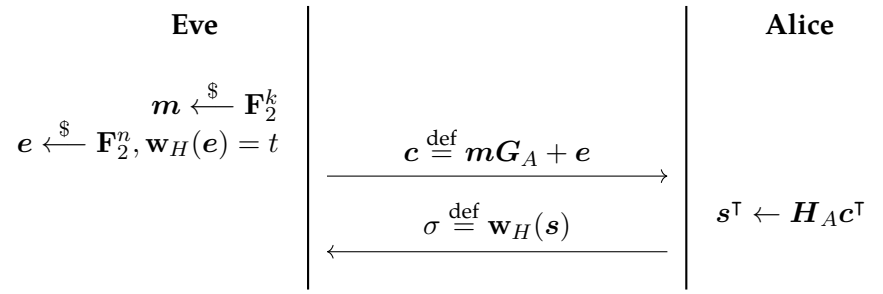


Figure 3.2: Attack on the syndrome weight. $\boldsymbol{G}_A$ denotes Alice's public key and $\boldsymbol{H}_A$ her private key.

For a given distance δ, let $\alpha \stackrel{\text{def}}{=} \mathbb{P}\left[\Delta(e)[\delta] = 1\right]$. We need to estimate the value of α. The number of vectors of length k and weight w that do not contain δ in their spectrum is $\prod_{j=0}^{w-1}(k-3j)$. Hence, neglecting the cases of multiplicity, we obtain a good approximation of α with the following formula.

$$1 - \frac{\prod_{j=0}^{\left\lfloor \frac{t}{2} \right\rfloor - 1}(k-3j)}{\prod_{j=0}^{\left\lfloor \frac{t}{2} \right\rfloor - 1}(k-j)}.$$

Hence, to decide whether or not $\delta \in \Delta(h)$, we need to observe the decoding of $\frac{N}{\alpha k}$ syndromes, with $N \sim \frac{1}{\varepsilon^2}$. As we use the same data to decide for all distances, this is the number of samples needed to recover the whole spectrum.

3.3 Attack on the syndrome weight

Following our analysis, we propose here a new attack.

3.3.1 Attack model

The scenario for our attack is the following. Eve can encrypt random messages using the QC-MDPC scheme described in 3.1 and Alice's public key. She has access to the plaintext but cannot choose the messages. She sends the messages for decryption. Whenever the device decodes a message sent by Eve, she has a way to observe the weight of the syndrome.

The attack we describe here is an abstraction. We do not focus on how, or even if, Eve gets access to the data. It might be possible or not depending on a particular implementation and on the abilities of the attacker. The point is to establish through a simulation that some secret information leaks from the syndrome weight and to compare the cost of that simulation with the theoretical analysis of the previous section.

We suppose that Eve's error patterns are randomly generated. Indeed, in the scheme, semantically secure conversions ensure that the error patterns are random [KI01]. If we allow Eve to choose the error patterns, this will only make the attack easier, as in [GJS16].

Contrary to [GJS16], we collect information from all the error patters, not only those leading to a decoding failure.

3.3.2 The attack

Our goal is to compute the distance spectrum of Alice's private key. For each distance δ between 1 and $\lfloor \frac{k}{2} \rfloor$ we want to decide the value of $\Delta(\boldsymbol{h}_{Alice}[\delta])$. As we have seen in 3.2, for each distance δ such that $\Delta(\boldsymbol{e})[\delta] = 1$, the expected average weight of the syndrome $\sigma = \mathbf{w}_H(\boldsymbol{s})$, where $\boldsymbol{s} = \boldsymbol{H}_{Alice} \cdot \boldsymbol{c}^\intercal = \boldsymbol{H}_{Alice} \cdot \boldsymbol{e}^\intercal$, is expected to be different if $\Delta(\boldsymbol{h}_{Alice})[\delta] = 1$.

Hence, the idea is, for each distance δ, to compute the average value of the syndrome weight σ for error patterns $\boldsymbol{e}$ such that $\Delta(\boldsymbol{e})[\delta] = 1$. The error patterns are generated randomly and each error $\boldsymbol{e}$ can be used to obtain information on all the distances in its spectrum. This leads to Algorithm 8.

Algorithm 8: Computing the distance spectrum using the syndrome weight oracle

Input: An oracle $\mathcal{O}$ that, given a noisy codeword, returns σ its syndrome weight, the number of samples N, a threshold value T.

Create three tables SyndromeCount, OccurenceCount and Spectrum of length $\lfloor k/2 \rfloor$ and initialize their entries to zero
for $i = 1$ *to* N `// Repeat for` N `random ciphertexts`
do
  Let $\boldsymbol{m}$ be a random message and $\boldsymbol{e} \leftarrow [\boldsymbol{e}_0 || \boldsymbol{e}_1]$ an error vector drawn uniformly at random
  Let $\boldsymbol{c} \leftarrow$ QCMDPC.Enc$(\boldsymbol{m}, \boldsymbol{e})$ and $\sigma \leftarrow \mathcal{O}(\boldsymbol{c})$.
  for $\delta = 1$ *to* $\lfloor \frac{k}{2} \rfloor$ *s.t.* $\Delta(\boldsymbol{e})[\delta] = 1$ **do**
    OccurenceCount$[\delta] \leftarrow$ OccurenceCount$[\delta] + 1$.
    SyndromeCount$[\delta] \leftarrow$ SyndromeCount$[\delta] + \sigma$.
for $\delta = 1$ *to* $\lfloor \frac{k}{2} \rfloor$ **do**
  `// If the DFR is less than some threshold` T
  if SyndromeCount$[\delta]$/OccurenceCount$[\delta] < T$ **then**
    Spectrum$[\delta] \leftarrow 1$
return Spectrum

Following the discussion in Section 3.2.3, we will take $\mathsf{threshold} = \frac{\bar{\sigma}_0+\bar{\sigma}_1}{2}$.

3.3.3 Experimental results

The spectrum recovery algorithm was first tried on a simplified version of the scheme using only one block, in order to compare to the expected behaviour. The result is striking. Using the usual parameters for 80-bit security, with one hundred thousand samples, the spectrum appears very clearly and we can even see the multiplicities, that is, distances that appear several times in the key, see Figure 3.1. When pushing to one billion samples, there is no room for confusion.

When performing the same experiment on the real QC-MDPC scheme with two blocks, we obtain similar results. The attack is performed on each block separately, that is for each error pattern, we added the syndrome weight to the counters of all distances present in the first half of the error to recover the spectrum of the first block. Because there is no correlation between the two halves of the error pattern, the presence of the second block acts as a random noise added to the syndrome weight. Hence the only difference is that we need more samples to reduce the variance and distinguish well which distances are in the key spectrum. Note that it is possible to compute the spectrum of both blocks at the same time, so there is no need to double the number of samples to recover the second block.

For 80-bit security parameters, we can see on Figure 3.3 the spectrum appearing more and more distinctively when we increase the number of samples. With 2^{20} samples, we can fully distinguish the spectrum. The same attack requires 2^{23} samples for 128-bit security parameters and 2^{25} for 256-bit security parameters.

This attack was also performed when another error is added to the syndrome, like in the Ouroboros scheme [DGZ17] (with an additional error of weight $3d$). Again, this only adds random noise and we can recover the spectrum with around a few million samples for the 80-bit security parameters.

3.4 Attack on the iteration count

3.4.1 Motivations and attack model

Now that we know that the syndrome weight leaks information, any parameter correlated to this quantity could potentially be used to mount a side channel attack. An interesting parameter that could be easy to measure is the number of iterations of a loop.

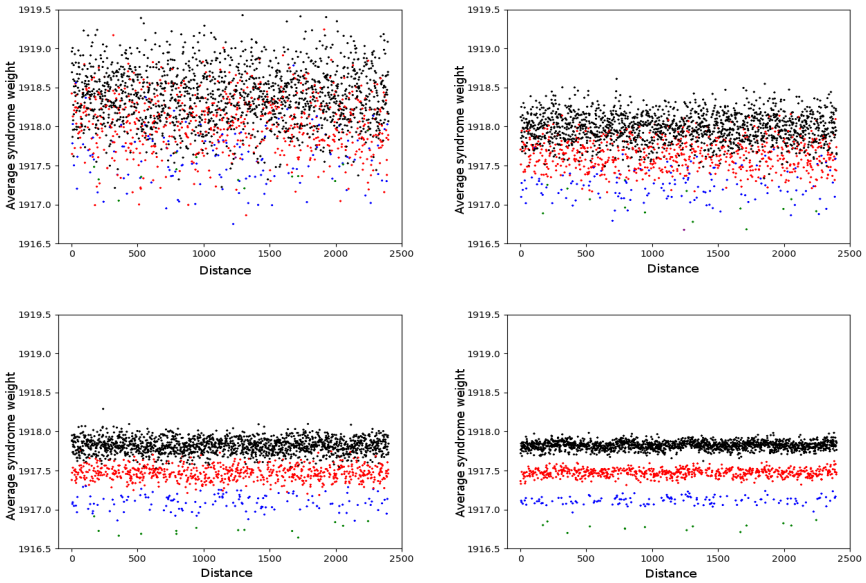


Figure 3.3: Average syndrome weight per distance, (from left to right, from top to bottom) 2^{14}, 2^{16}, 2^{18} and 2^{20} samples, 80-bit security QC-MDPC scheme. The color of the distances indicate their multiplicity in the key spectrum (black $= 0$, red $= 1$, blue $= 2$, green $= 3$, purple $\geqslant 4$)

The decoding algorithm for QC-MDPC codes is an iterative algorithm with no termination proof. The number of rounds needed to correct the errors varies. This has been studied by in [CS16a]. As mentioned in Section 2.2.1.2, the algorithm depends on the way we chose the thresholds. For most instances, using fixed or variable thresholds, the algorithm usually corrects the error in 3 rounds, but some instances need 4, 5 or even more iterations. Usual implementations abort after a certain number of rounds (around 10), this is what was used for the attack in [GJS16].

Experimentally, we observe that the correlations between the spectrum of the error and the spectrum of the key has an impact on the average decryption time. The more distances appear both in spectrum of the error and in the spectrum and the key, the fewer the number of iterations needed to decode on average. This appears clearly on Fig. 3.4. We note that the correlation is slightly more important on Fig. 3.4 when we use variable thresholds than with fixed thresholds (the average value is lower for variable thresholds, but the same scale is used for both figures).

This motivated us to try to perform an attack focusing on the iteration count (Algorithm 9). The scenario is the same as previously, but instead of

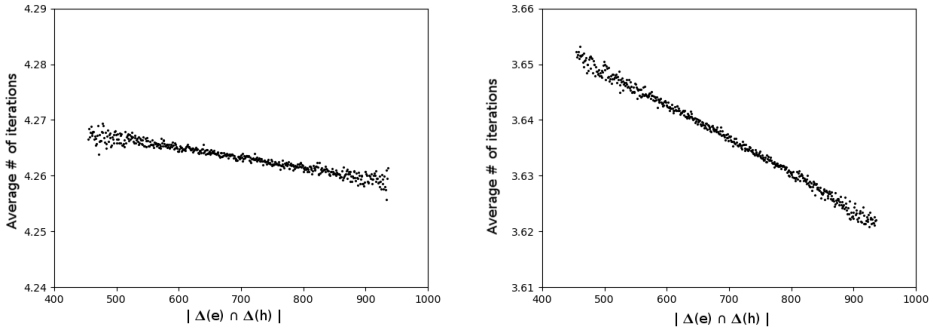


Figure 3.4: Average number of iterations needed for decryption, depending on the size of the intersection of the spectrum of the error and the spectrum of the key. 2^{29} samples, 128-bit security QC-MDPC scheme, decoding with fixed thresholds (left) and variable thresholds (right). Note that use of variable thresholds results in stronger correlation.

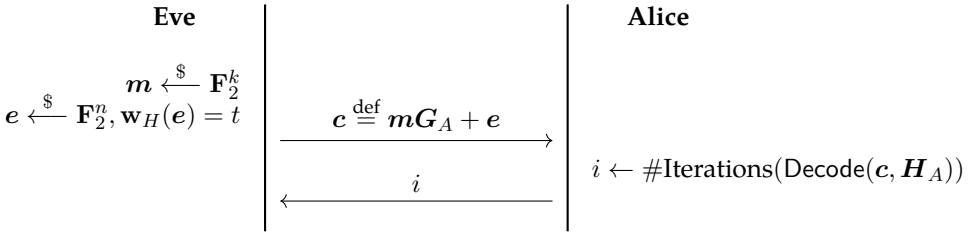


Figure 3.5: Attack on the iteration count. $\boldsymbol{G}_A$ denotes Alice's public key and $\boldsymbol{H}_A$ her private key.

observing the syndrome weight, Eve can measure the number of iterations needed to decode her message. To obtain the spectrum, Eve uses the exact same data collection algorithm: for every distance in the spectrum, she computes the average number of iterations needed to correct an error containing this distance.

If the decoding algorithm is implemented in a textbook manner, the execution time evolves linearly with the number of iterations, and hence the number of iterations can straightforwardly be deduced from the execution time of the decoding. Hence, in such a case, this attack is a timing attack. However, we do not intend to experiment a real timing attack (measuring the execution time), as this result would be specific to an implementation. Instead, our goal here is to demonstrate that an implementation that leaks any information related to the number of iterations is a security threat. Therefore, one should aim at an implementation for which the running time (and any other measurable

parameter) is independent of the number of iterations.

3.4.2 The attack

The previous algorithm can be adapted to this model and yields Algorithm 9.

Algorithm 9: Computing the distance spectrum using the number of iterations

Input: An oracle $\mathcal{O}$ that, given a noisy codeword, returns P the number of iterations needed to decode it using the bitflip algorithm, the number of samples N, a threshold value T.

```
1  Create three tables IterationsCount, OccurenceCount and Spectrum of
     length ⌊k/2⌋ and initialize their entries to zero
2  for i = 1 to N                          // Repeat for N random ciphertexts
3  do
4      Let m be a random message and e ← [e_0||e_1] an error vector
         drawn uniformly at random
5      Let c ← QCMDPC.Enc(m, e) and P ← O(c).
6      for δ = 1 to ⌊k/2⌋ s.t. Δ(e)[δ] = 1 do
7          IteratisCount[δ] ← IterationsCount[δ] + 1.
8          SyndromeCount[δ] ← SyndromeCount[δ] + P.
9  for δ = 1 to ⌊k/2⌋ do
       // If the DFR is less than some threshold T
10     if IterationsCount[δ]/OccurenceCount[δ] < T then
11         Spectrum[δ] ← 1
12 return Spectrum
```

3.4.3 Experimental results

We run Algorithm 9 and plot the value of IterationsCount$[\delta]$/OccurenceCount$[\delta]$ for all distances δ. The resulting plots, in Figure 3.6, look very similar to the plots of the decoding failure rate that result from Algorithm 7. When the bands are completely separated, the distance spectrum (and thus the secret key) can be recovered in the same way it was in the GJS [GJS16] attack. We performed the attack using two types of thresholds (see Section 2.2.1.2).

This attack performs well and it is possible to fully recover the distance spectrum with variable thresholds using 2^{25} samples on 80-bit security QC-MDPC scheme, 2^{25} samples for 128-bit security parameters (see Fig. 3.6) and

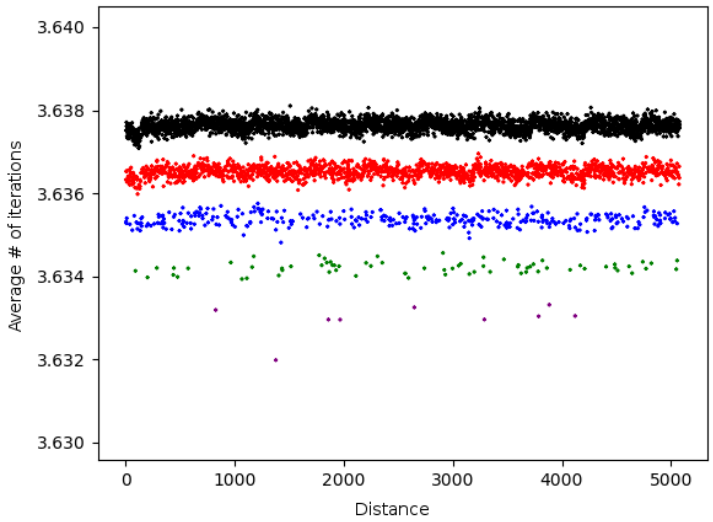


Figure 3.6: Attack using the number of decoding iterations, with 2^{25} samples, against parameters for 128-bit security QC-MDPC scheme with variable threshold decoding. The color of the distances indicate their multiplicity in the key spectrum (black = 0, red = 1, blue = 2, green = 3, purple $\geqslant 4$)

2^{28} samples for 256-bit security parameters. For fixed thresholds, we manage to recover the spectrum for 256-bit security with 2^{28} samples.

Note that these this side-channel attacks are much faster to execute than the GJS reaction attack. An intuitive explanation for the speedup is that the number of iterations varies often, while decoding errors rarely happen. This allows more information about the correlations to be collected per iteration of the attack algorithm.

3.4.4 About spectrum reconstruction

Until here, the number of samples needed for a successful attack has been characterised as the number of decryption to perform in order to obtain a plot where the points corresponding to a distance absent of the spectrum (*i.e.* with multiplicity 0) and the points of multiplicity > 0 are clearly distinct. This means that there exist a threshold value T such that only points with value $< T$ have multiplicity > 0. Indeed, the vector reconstruction algorithm presented in [GJS16] which reconstructs a vector from its spectrum takes as input the full spectrum. More exactly, it only makes use of the positive information ("distance δ is in the spectrum") but does not use the negative information

("distance δ is **not** in the spectrum"). However, such an information is as important, and can be exploited to improve the algorithm.

More importantly, the knowledge of the full distance spectrum usually overdetermines the solution. Hence, one could potentially reconstruct the key from plots where the cluster of points corresponding to different multiplicities are not distinct. Consider for instance the lower left plot of Figure 3.3, corresponding to 2^{20} samples. In such a situation, there are points for which it is clear that their multiplicity is zero, some for which it is clear that the multiplicity is > 0, and some for which it is impossible to decide. One could imagine an algorithm that takes as input only a list of distance that are present in the spectrum for sure (above a certain threshold T_+), and a list of distances absent from the spectrum for sure (bellow a certain threshold T_-). The points between T_+ and T_- do not provide any information. Depending on the number of points provided, this could be enough to yield a unique solution (up to shift and reflection) or at least to drastically reduce searchspace.

Another approach would be to associate to each point a real value corresponding to the level of confidence we have that they belong to one category or the other. Finding a solution then becomes an optimisation problem. This could be qualified as a soft approach to the problem.

The efficiency of these approaches has not been thoroughly studied but the main point of this discussion is that even a partial leak of the distance spectrum, or more exactly, providing the adversary with an access to a parameter that can be exploited to recover even only a portion of the spectrum, is a security issue.

3.5 Possible mitigations

3.5.1 Ephemeral keys

The first and obvious mitigation for such attacks is to prevent the reuse of the same key. This means that the QC-MDPC can be safely used as a key encapsulation mechanism (KEM), but not as a public key encryption (PKE) scheme, where the above scenarios apply. In a key encapsulation mechanism, the goal is to derive a shared secret, so the key will only be used once. This is the reason why the BIKE scheme [ABBBB+17] submitted to the NIST standardisation process is a KEM and not a PKE. However, this is more of a reduction of the use-case than a mitigation. The goal is to be able to make the QC-MDPC scheme CCA-secure to be able to reuse the keys safely.

3.5.2 Parallel encryption

A solution named ParQ and proposed in [ELPS18] is to use parallel encryption. The idea is to perform P different encryption of the message. This constitutes the ciphertexts. The receiver tries to decrypt all the ciphertexts but only needs to successfully decrypt one of them to find the value of the plaintext. Hence, in such a scheme, a decryption failure means that none of the P ciphertexts could be successfully decrypted. Therefore, if the usual QC-MDPC scheme has a decoding failure rate of $2^{-\kappa}$, then the ParQ scheme has a DFR of $2^{-P\kappa}$, as each ciphertext is enciphered with an independent error.

The details of this scheme can be found in [ELPS18]. The factor P can be chosen such that $P\kappa$ is at least equal to the security level of the scheme. In practice, with the QC-MDPC parameters presented in this chapter, the value of P has to be chosen between 3 and 12 depending on the security level. This has a significant impact on the performance. Indeed, the size of the ciphertext, as well as the encryption and decryption time, are multiplied by a factor P. This is not very satisfying, compared to other schemes. For instance this would give an encryption time more than a hundred time slower than the classical McEliece scheme [BCLMM+19] (which on the other hand suffers from its large public key size).

3.5.3 Forcing a full spectrum: monomial codes

Another approach proposed in [SBCC18] is to design the secret key such that all possible distances are in their spectrum. Their proposal relies on monomial codes, which are QC-LDPC codes. Instead of having two circulant block matrices, the idea is to have multiple small circulant block matrices (around one hundred matrices in length and width), and make sure that for all blocks, all distances exist in the spectrum. In such a case, it does not make sense to perform such an attack, because the spectrum does not carry significant information and because the reconstruction algorithm would fail. Indeed, in such a configuration, the reconstruction algorithm as presented in [GJS16] reduces to the problem of finding a clique in a graph, which is hard for such parameters. Hence, this scheme can have a non-negligible DFR while defeating GJS-like attacks.

However, this is not a satisfying solution. Apart from increasing the key size, this scheme requires parameters that are very different from the classic QC-MDPC scheme. The small block matrices induce more structure, which could lead to new structural attacks. Moreover, the security analysis does not take into account the multiplicities in the spectrum. An attacker could take advantage of this. This scheme uses QC-LDPC codes, as in the LEDA scheme [BBCPS19], which have been subject to recent attacks [APRS20]. Finally, this

does not change the fact that the scheme has a significant decoding failure rate. Although this scheme resists attacks based on the distance spectrum, an adversary could still observe the DFR and this may still leak some other information about the key.

3.5.4 Lowering the DFR

The best solution to preserve the same key size while avoiding GJS-like attacks seems to consist in modifying the decoder to obtain a DFR $< 2^{-\kappa}$, where κ is the number of security bits of the scheme (for instance 128 or 192). A recent line of works has made promising progress in this direction.

3.5.4.1 Theoretical tools to compute low DFR

First, one has to find theoretical tools to assess that a decoder would reach such a low DFR. One cannot rely on the classical simulation technique consisting in decoding a large number of random errors and count the number of failures. Indeed, by design such events are too rare to be measured. Two recent results give some insight on how to overcome this pitfall and us understand how the DFR evolves when increasing the length of the code.

- A theoretical model by Tillich [Til18] shows that asymptotically the DFR decreases exponentially in the code length. This model focuses on a classical bit-flip algorithm with two iterations and supposes that the weight of the secret key and of the error vector grow in $\sqrt{n}$ while classical model rather consider increasing the code length for fixed weight of the secret key and error vector. Still, this is the only asymptotical result of this kind and it provides a good insight of the behaviour of the DFR.

- Another result from Sendrier and Vasseur [SV19] studies a simpler step-by-step decoder, where bits are flipped one by one instead of performing iterations on the whole vector. This decoder is modeled by a Markov chain. The theoretical results are compared with simulations and behave rather similarly, which validates this model. Again, the DFR seems to asymptotically decrease exponentially with the code length.

Hence, one can plot the exponent of the DFR relatively to the code size for small values (obtained through simulations), and this almost yields a affine curve. These two results mean that it is reasonable to extend the line and use it to extrapolate the DFR for larger code length. This approach gives a conservative estimate of the required code length to reach a desired DFR.

3.5.4.2 New decoders

In the meantime, new variants of the decoding algorithm have been proposed. Indeed, with threshold decoding, one would be tempted to set a high threshold (*i.e.* making conservative choices), and hence to only flip bits for which we are very confident that they are in the support of the error. But this approach may require to perform more iterations, since less bits are flipped at each step. An iteration is a relatively costly operation, since one has to compute the counters for all positions (tens of thousands of bits). On the other hand, a lower threshold means that we are more likely to flip bits that are already correct, and hence amplify the error.

An idea proposed in the first version of the BIKE submission is to define two threshold, corresponding to two levels of confidence. Once the counters are computed, all bits with counters higher that the high threshold are flipped. These are bits for which we are almost sure that they were incorrect. These are called "black" positions. The positions with counter values higher than the lower threshold correspond to a lower level of confidence. These bits seem to be incorrect but it is less certain. These are called "gray" positions. The idea is to flip the black positions and to recompute the counters only for gray positions. This is much cheaper than a new iteration. This approach, which is inspired by soft decoding, is referred to as gray decoding [DGK20]. Another idea is to check how the counters evolve after flipping some bits and undo some flips if they have not improved the situation. Finally, one can even flip bits with the intention to unflip them a few iterations later. This idea, called backflip [SV20], allows to avoid some undesired error patterns with low counters.

Remark 3.11. *With these new algorithms, the notion of "number of iterations" disappears (or at least becomes less relevant). Indeed, the general spirit is to lesser the number of full iterations (computing the counters for each bit and updating the error). Instead, this operation is applied to a smaller subset of bits. Therefore, the side-channel attacks presented in this chapter looses some of its meaning, because it relies on the hypothesis that the running time of the algorithm reflects on the iterations.*

However, one can adapt the general idea by considering the number of individual counters that are computed instead of the number of iterations. In a non-constant-time implementation, the running time certainly depends on this quantity, and it is reasonable to believe that this quantity is still correlated with the syndrome weight if one is not careful enough in the design of the algorithm. Therefore, all remarks in this chapter should still be taken into account in the design of this new generation of decoding algorithms.

3.5.4.3 Best trade-off

Several combinations of these new ideas have been proposed to improve the efficiency of the decoder. The new theoretical results can be used to find the DFR of these different variants. The underlying hypothesis is that all decoders share the same behaviour: from a certain point, the exponent of the DFR decreases linearly with the code length. Thus, given some code parameters, one can compare different decoders, in order to find the best trade-off between three criteria: the number of steps of the algorithm, the estimated DFR and finally the performance of a constant time implementation of the decoder.

This comparison work is performed in [DGK20] and the authors conclude that the best trade-off is achieved by the "black-gray flip" decoder. This choice of decoder has been included in the latest revision of the BIKE proposal [ABBBB+17] for the final round of the NIST post-quantum standardisation process, with a constant-time implementation.

3.6 Conclusion

The GJS attack [GJS16] is a general idea that can be adapted to several other post-quantum schemes. Indeed, for most cryptographic schemes where the decryption may fail, this failure happens when the error has a specific pattern, correlated to the secret key. Hence, observing errors which yield decoding errors provides information on the secret key. This attack has been adapted to other code-based schemes: for QC-LDPC cryptosystems in [FHSZG+17], for HQC in [GJ20], even for LRPC, the rank-metric equivalent of MDPC codes (see next chapter), in [AG19]. It has also been adapted to lattice-based schemes, such as NTRU in [DGJNV+19] or LAC in [GJY19]. Even a non-careful use of the generic semantically secure Fujisaki-Okamoto transformation in a scheme can lead to such an attack as illustrated in [GJN20].

As for QC-MDPC schemes, the analysis of the GJS attack shows that it does not depend of the details of the decoder. Hence, any decoding algorithm involved in the QC-MDPC scheme should be implemented with constant number of iterations, or at least add some procedure to ensure that the number of iterations is independent of the syndrome weight. If the notion of iteration does not apply (as for instance for gray decoding, where only a small number of counters are involved), this notion can be replaced by the number of counters computed (see Remark 3.11). The DFR should be lower than $2^{-\kappa}$ where κ is the number of security bits, in order to defeat all GJS-like attacks. There is now a general consensus on how to compute the expected DFR, although new theoretical arguments would be welcome. Finally, the hardware implementations should be thoroughly designed, to make sure that no parameter

correlated to the syndrome weight can be observed by a curious attacker having access to the hardware device. This last point requires further investigation. If such conditions are all confidently met, then one could claim CCA security.

The general understanding of the properties of QC-MDPC decoders has evolved quickly in the last years. Although the NIST considers BIKE as "one of the most promising code-based candidates" for standardisation, they stress in their recent report that there remain "serious questions about side-channel protections and CCA security" [AAACD+20]. Our analysis improves the general understanding of these issues and strengthens the general confidence in the security of the BIKE scheme. The NIST announced in 2020 that they decided to select BIKE for the third round of the standardisation process but as an "alternate candidates" rather than a finalist. Indeed, the NIST acknowledges the strong improvements in the decoding and understanding of the failure rate, but estimates that more time is needed to fully address these concerns. Hence, one can expect that within a few years, all conditions will be met for BIKE to become a standardised algorithm.

Chapter 4

Attack on the Edon-K cryptosystem

One promising line of work to overcome the drawback of the large key size in code based cryptography is to replace Hamming metric by another metric in the definition of codes. Especially, the *rank metric* shares a lot of properties with the Hamming metric.

Among the code based cryptosystems proposed as a response to the NIST call for post-quantum standardization, the Edon-K cryptosystem [GG17] can be analysed with respect to the theory of rank metric codes. Indeed, an analysis of the scheme shows that the code used matches the properties of Low Rank Parity Check codes, the rank-metric equivalent of MDPC codes.

Using the tools of rank metric, we propose a polynomial-time key recovery attack against the Edon-K key encapsulation mechanism. This result was published in [LT18] and led to the elimination of the Edon-K cryptosystem from the standardisation process.

Related publication: Lequesne and Tillich, *Attack on the Edon-K Key Encapsulation Mechanism*, ISIT 2018 [LT18].

Contents

4.1 Rank metric and LRPC codes

4.1.1 Introduction to rank metric

The theory of error correcting codes was originally developed in the context of information theory, as a solution to the problem of symbol transmission over a noisy channel. The most classical model of this problem is the binary symmetric channel, and in this context the number of errors in measured by the Hamming distance. Therefore, error correcting codes were defined with respect to this metric.

However, the definition of error correcting codes can be extended to other metrics. Indeed, in a cryptographical context, one is not bound by the direct applications to signal transmission. Hence, studying codes with respect to other metrics can be an interesting source of new hard problems on which one could base new cryptographic primitives. This could typically be a way to overcome the drawback of the large public key size of the classical McEliece cryptosystem.

With this respect, rank metric appears to be one of the most interesting alternative to the Hamming, with which is shares several similarities. This rank metric of a vector $\boldsymbol{x}$ with entries in an extension field $\mathbf{F}_{q^m}$ is defined as the maximal number of entries of $\boldsymbol{x}$ that are linearly independent over $\mathbf{F}_q$.

We can trace back the definition of this metric to a 1951 paper from Hua [Hua51] which introduced the notion of "arithmetic distance" between matrices over a field $\mathbf{F}_q$: the distance between two matrices is defined as the rank of their difference. In 1978, Delsarte [Del78] studies a similar distance (which he calls the "q-distance") over bilinear forms over $\mathbf{F}_q$, which can equivalently be seen as matrices. Delsarte studies the properties of the codes obtained using this metric. He characterises the codes that meet the Singleton bound. These codes are now called maximum rank distance (MRD) codes, which are the rank-metric equivalent of the MDS codes.

In 1985, Gabidulin [Gab85] describes these codes in terms of vectors over $\mathbf{F}_{q^m}$ rather than matrices over $\mathbf{F}_q$. Indeed, given a vector over $\mathbf{F}_{q^m}$, each entry

of the vector can be expanded over a an $\mathbf{F}_q$-basis, and hence represented by its m coefficients. Hence, a vector of length n over $\mathbf{F}_{q^m}$ leads to an $m \times n$ matrix over $\mathbf{F}_q$. As the rank does not depend on the choice of the basis, there is an exact correspondence between these objects. The novelty in Gabidulin's approach is that he studies codes that are linear over $\mathbf{F}_{q^m}$ (and not only $\mathbf{F}_q$). This choice allows for a much more compact representation and yields interesting properties. In fact, this is the main reason why the rank metric based McEliece schemes achieve significantly smaller key sizes. Moreover this vectorial representation allows to view the known families of MRD codes as rank metric analogues of Reed-Solomon codes and to obtain an efficient decoding algorithm for them [Gab85].

Several attempts were made to use these codes in a McEliece cryptosystem [GPT91; FL05] but were subject to attacks [Ove05; Ove08; GOT18; CC19]. A few years later, other families of rank-metric codes with efficient decoding algorithms were proposed, in particular the Low Rank Parity Check codes (LRPC) [GMRZ13], which are the rank-metric analogues of MDPC codes. We will focus on these codes in the rest of this chapter.

4.1.2 Definitions

Definition 4.1 (Rank metric)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{F}_{q^m}^n$ and $(\beta_1, \ldots, \beta_m)$ be a basis of $\mathbf{F}_{q^m}$ viewed as an m-dimensional vector space over $\mathbf{F}_q$. Each coordinate $x_j \in \mathbf{F}_{q^m}$ is associated to a vector of $\mathbf{F}_q^m$ in this basis: $x_j = \sum_{i=1}^{m} m_{i,j}\beta_i$. The $m \times n$ matrix associated to $\boldsymbol{x}$ is given by $\boldsymbol{M}(\boldsymbol{x}) \stackrel{\text{def}}{=} (m_{i,j})_{1 \leqslant i \leqslant m, 1 \leqslant j \leqslant n}$.

The rank weight $\mathbf{w}_R(\boldsymbol{x})$ of $\boldsymbol{x}$ is defined as :

$$\mathbf{w}_R(\boldsymbol{x}) \stackrel{\text{def}}{=} \operatorname{Rank} \boldsymbol{M}(\boldsymbol{x}).$$

The associated distance $d(\boldsymbol{x}, \boldsymbol{y})$ between elements $\boldsymbol{x}$ and $\boldsymbol{y}$ of $\mathbf{F}_{q^m}^n$ is defined by $d(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \mathbf{w}_R(\boldsymbol{x} - \boldsymbol{y})$.

This definition does not depend on the choice of the basis $\mathcal{B}$.

Definition 4.2 (Support of a word)**.** Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbf{F}_{q^m}^n$. The support of $\boldsymbol{x}$, denoted $\mathbf{Support}(\boldsymbol{x})$, is the $\mathbf{F}_q$-subspace of $\mathbf{F}_{q^m}$ generated by the coordinates of $\boldsymbol{x}$:

$$\mathbf{Support}(\boldsymbol{x}) \stackrel{\text{def}}{=} \langle\, x_1, \ldots, x_n \,\rangle_{\mathbf{F}_q}.$$

We have $\dim(\mathbf{Support}(\boldsymbol{x})) = \mathbf{w}_R(\boldsymbol{x})$.

Definition 4.3 (Rank code)**.** An $\mathbf{F}_{q^m}$-linear rank code $\mathscr{C}$ of dimension k and length n is a subspace of dimension k of $\mathbf{F}_{q^m}^n$. $\mathscr{C}$ can be represented in two equivalent ways: by a generator matrix $\boldsymbol{G} \in \mathbf{F}_{q^m}^{k \times n}$ such that $\mathscr{C} = \{\boldsymbol{x}\boldsymbol{G} \text{ s.t. } \boldsymbol{x} \in$

$\mathbf{F}_{q^m}^k\}$ and by a parity-check matrix $\boldsymbol{H} \in \mathbf{F}_{q^m}^{(n-k)\times n}$ such that $\mathscr{C} = \{\boldsymbol{y} \in \mathbf{F}_{q^m}^n \text{ s.t. } \boldsymbol{H}\boldsymbol{y}^\intercal = \mathbf{0}_{n-k}\}$.

4.1.3 Hard problems in rank metric

Similarly to codes in Hamming metric, the security of rank-based cryptosystems can be reduced to a few simple problems that are expected to be hard. The main problem is the generic decoding problem in the rank metric, which can be described as follows.

Problem 4.4 (Generic decoding problem for the rank metric)**.** *Let $\boldsymbol{G}$ be an $k \times n$ matrix over $\mathbf{F}_{q^m}$, with $k \leqslant n$. Denote $\mathscr{C}$ the $\mathbf{F}_{q^m}$-linear code generated by $\boldsymbol{G}$. Given $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c} \in \mathscr{C}$ and $\boldsymbol{e} \in \mathbf{F}_{q^m}^n$ such that $\mathbf{w}_R(\boldsymbol{e}) \leqslant r$, find $\boldsymbol{c}$ and $\boldsymbol{e}$.*

This problem can equivalently be stated in terms of syndrome decoding, which is often more convenient for cryptography.

Problem 4.5 (Rank Syndrome Decoding (RSD))**.** *Let $\boldsymbol{H}$ be an $(n-k)\times n$ matrix over $\mathbf{F}_{q^m}$, with $k \leqslant n$. Given $\boldsymbol{s} \in \mathbf{F}_{q^m}^{n-k}$ and an integer r, find $\boldsymbol{y} \in \mathbf{F}_{q^m}^n$ such that $\boldsymbol{H}\boldsymbol{y}^\intercal = \boldsymbol{s}^\intercal$ and $\mathbf{w}_R(\boldsymbol{y}) \leqslant r$.*

The generic decoding problem in rank metric can also be seen as a particular instance of the more general MinRank problem.

Problem 4.6 (MinRank)**.** *Given matrices $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_k \in \mathbf{F}_q^{m\times n}$, an integer r and a matrix $\boldsymbol{M}_0 \in \mathbf{F}_q^{m\times n}$, find $(\lambda_1, \ldots, \lambda_k) \in \mathbf{F}_q^k$ such that $\mathbf{w}_R(\sum_{i=1}^k \lambda_i \boldsymbol{M}_i - \boldsymbol{M}_0) \leqslant r$.*

Indeed, given an instance of the generic decoding problem and a basis of $\mathbf{F}_{q^m}$, the vector $\boldsymbol{y}$ can be expanded into an $m \times n$ matrix over $\mathbf{F}_q$ (which will be $\boldsymbol{M}_0$) and each row of the generator matrix $\boldsymbol{G}$ can be expanded as an $m \times n$ matrix $\boldsymbol{M}_i$. Hence, the linear combination $\sum_{i=1}^k \lambda_i \boldsymbol{M}_i$ corresponds to a codeword of $\mathscr{C}$ at rank-distance $\leqslant r$ of $\boldsymbol{y}$.

The MinRank problem is known to be NP-complete [Cou01]. However, these two problems are not equivalent, because the $\mathbf{F}_{q^m}$-linearity of the rank codes is not taken into account in the MinRank problem.

Attacks on the RSD problem have first been developed in [OJ02] and then [GRS16]. Two recent articles [BBBGN+20; BBCGP+20] improve these algebraic attacks by using a new way to model the problem as a system of multivariate equations and break several sets of parameters, though the complexity of the problem remains exponential.

4.1.4 LRPC codes

The Low Rank Parity Check (LRPC) codes are introduced in [GMRZ13] and defined as follows.

Definition 4.7 (LRPC code)**.** A Low Rank Parity Check (LRPC) code of rank d, length n and dimension k over $\mathbf{F}_{q^m}$ is a code that admits a parity-check matrix $\boldsymbol{H} = (h_{i,j}) \in \mathbf{F}_{q^m}^{(n-k)\times n}$ such that all its coefficients $h_{i,j}$ lie in an $\mathbf{F}_q$-subspace of $\mathbf{F}_{q^m}$ of dimension at most d. This matrix will be called the low rank parity-check matrix of the code.

Decoding LRPC codes. LRPC codes can be viewed as analogues of LDPC (or MDPC) codes for the rank metric. Indeed, such codes do not have a particular algebraic structure (compared for instance to geometric codes), apart from the fact that there exists (randomly chosen) words of relatively small weight in their dual. Just like LDPC/MDPC codes, LRPC codes enjoy an efficient decoding algorithm based on their low rank parity-check matrix.

The main idea of the decoding algorithm, introduced in [GMRZ13], is that if the entries of the low rank matrix are all in a subspace $\mathcal{B} = \langle \beta_1, \ldots, \beta_d \rangle_{\mathbf{F}_q}$ of dimension d and the entries of $\boldsymbol{e}$ are all in a subspace $\mathcal{E} = \langle \varepsilon_1, \ldots, \varepsilon_r \rangle_{\mathbf{F}_q}$ of dimension r, then the syndrome $\boldsymbol{s} = \boldsymbol{e}\boldsymbol{H}^\intercal$ has all its entries in the product space $\mathcal{P} \stackrel{\text{def}}{=} \mathcal{B} \cdot \mathcal{E} \stackrel{\text{def}}{=} \langle (\beta_i \varepsilon_j)_{i,j} \rangle_{\mathbf{F}_q}$ of dimension $\leqslant rd$. It is reasonable to expect that the entries of $\boldsymbol{s}$ span the whole subspace $\mathcal{P}$ when $n - k > rd$. Hence, knowing $\boldsymbol{y}$ and $\boldsymbol{H}$, one has access to $\mathcal{B}$ and $\mathcal{P}$ and can deduce the value of $\mathcal{E}$. Then it is possible to solve the system expressed in the subspace of small dimension.

Proposition 4.8 ([GMRZ13], Theorem 1)**.** *Let $\boldsymbol{H}$ be an $(n-k) \times n$ parity-check matrix of an LRPC code of rank $d \geqslant 2$, then Algorithm 10 decodes in polynomial time a random error of rank r such that $rd \leqslant n - k$ with failure probability $q^{-(n-k+1-rd)}$.*

Remark 4.9. *Just like in the case of MDPC codes, the decoding of LRPC codes may fail with a small probability. The difference is that in this case there exists a bound on the failure probability. Still, this decoding failure can lead to some reaction attacks similar to the attack on MDPC codes exposed in the previous chapter, as explained in* [*AG19*].

Cryptosystems based on LRPC codes. The paper [GMRZ13] introducing LRPC codes proposes an application to cryptography, in a cryptosystem following the McEliece scheme. The trapdoor is the low-rank parity-check matrix of the code, which is key to decode efficiently. Hence, just like for MDPC codes, the secret key is the low-rank parity-check matrix and the public key is

Algorithm 10: LRPC-Decoding$(\boldsymbol{H}, r, \boldsymbol{y})$ [GMRZ13]

Input: $\boldsymbol{H} \in \mathbf{F}_{q^m}^{(n-k)\times n}$ the low rank parity-check matrix of an LRPC code, an integer r, $\boldsymbol{y} \in \mathbf{F}_{q^m}^n$ such that $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c}$ is a codeword and $\mathbf{w}_R(\boldsymbol{e}) \leqslant r$.

Output: $\boldsymbol{e} \in \mathbf{F}_{q^m}^n$.

1. Compute the syndrome $\boldsymbol{s}$ such that $\boldsymbol{s}^\intercal = \boldsymbol{H}\boldsymbol{y}^\intercal = \boldsymbol{H}\boldsymbol{e}^\intercal$. Let $(s_1, \ldots, s_n)$ denote the entries of $\boldsymbol{s}$ and $\mathcal{S} \stackrel{\text{def}}{=} \langle s_1, \ldots, s_{n-k} \rangle_{\mathbf{F}_q}$ the associated subspace.
2. Let $\mathcal{B}$ be the subspace containing the entries of $\boldsymbol{H}$ and denote $(\beta_1, \ldots, \beta_d)$ be a basis of $\mathcal{B}$.
3. For $i \in [\![1, d]\!]$, denote $\mathcal{S}_i \stackrel{\text{def}}{=} \beta_i^{-1}\mathcal{S}$. Let $\mathcal{E} \stackrel{\text{def}}{=} \bigcap_{i=1}^d \mathcal{S}_i$.
4. Express $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$ as a linear system over $\mathbf{F}_q$ by expanding each coordinate over the product space $\mathcal{E} \cdot \mathcal{B}$. This yields a system with nr unknowns (ev expressed in the basis $\mathcal{E}$) and $(n-k)rd$ equations (the $n-k$ equations expressed in the basis $\mathcal{E} \cdot \mathcal{B}$).
5. Return $\boldsymbol{e}$.

a (random) generator matrix of the code. The encryption consists in encoding the message and adding an error corresponding to the decryption radius of the LRPC code. To decrypt, one applies the decoding algorithm.

The security of such a scheme relies on the hardness of the generic decoding (RSD) problem, as well on another security hypothesis specific to LRPC codes: the indistinguishability of LRPC codes. More specifically, this supposes that given a generator matrix, it is hard to find low rank codewords in the dual. Indeed, collecting such vectors would yield a low rank parity-check matrix and hence allow to use the decoding algorithm.

Problem 4.10 (LRPC problem)**.** *Given a generator matrix $\boldsymbol{G}$ of an LRPC code $\mathscr{C}$ of rank d, find a vector of weight $\leqslant d$ in the dual code of $\mathscr{C}$.*

Two schemes in this line of work have been submitted in response to the NIST call for post-quantum standardization: a key encapsulation mechanism, LAKE, and a public key encryption scheme LOCKER, later merged into a single proposal named ROLLO [ABDGH+19]. However, the recent attacks [BBBGN+20; BBCGP+20] break the parameters proposed by these schemes.

4.2 The Edon-K cryptosystem

Edon-K [GG17] is a key encapsulation mechanism proposed by Danilo Gligoroski and Kristian Gjøsteen for the NIST post-quantum cryptography call. Here we describe the key generation, encapsulation and decapsulation, omitting some details that are not relevant for the attack. We refer to [GG17] for the full description.

4.2.1 Notations

Hash function. The Edon-K scheme makes use of a hash function, denoted $\mathcal{H}(\cdot)$, corresponding to standard SHA2 functions (SHA-256 or SHA-384 depending on the parameters). We denote $\mathcal{H}^i(\cdot) \stackrel{\text{def}}{=} \underbrace{\mathcal{H}(\dots\mathcal{H}(\cdot))}_{i \text{ times}}$.

Quasi-orthogonality. Given two non-zero elements $a, b \in \mathbf{F}_{2^m}$ with $a \neq b$ and a binary matrix $\boldsymbol{P} = (p_{i,j}) \in \mathbf{F}_2^{n\times n}$, let $\boldsymbol{P}_{a,b} = (\tilde{p}_{i,j}) \in \mathbf{F}_{2^m}^{n\times n}$ denote the matrix of the same size with coefficients in $\mathbf{F}_{2^m}$ where $\tilde{p}_{i,j} = a$ if $p_{i,j} = 0$ and $\tilde{p}_{i,j} = b$ if $p_{i,j} = 1$.

Proposition 4.11. *If $\boldsymbol{P}$ is orthogonal (i.e. $\boldsymbol{P}^{-1} = \boldsymbol{P}^\intercal$) and n is even, then*

$$(\boldsymbol{P}_{a,b})^{-1} = \boldsymbol{P}_{c,d}{}^\intercal \tag{4.1}$$

where $c \stackrel{def}{=} \frac{a}{a^2+b^2}$ and $d \stackrel{def}{=} \frac{b}{a^2+b^2}$.

Example 4.12. For $m = 4$, let $\mathbf{F}_{2^m} = \mathbf{F}_2[\alpha]$ where α is a primitive element of minimal polynomial $X^4 + X + 1$. Representing the element $\alpha^i \in \mathbf{F}_{2^m}$ by the integer i, we have for $n = 4, a = 5$ and $b = 7$:

$$\boldsymbol{P} = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix} \quad \boldsymbol{P}_{a,b} = \begin{pmatrix} 7 & 5 & 7 & 7 \\ 7 & 7 & 7 & 5 \\ 5 & 7 & 7 & 7 \\ 7 & 7 & 5 & 7 \end{pmatrix} \quad \boldsymbol{P}_{a,b}^{-1} = (\boldsymbol{P}_{c,d})^\intercal = \begin{pmatrix} 11 & 11 & 9 & 11 \\ 9 & 11 & 11 & 11 \\ 11 & 11 & 11 & 9 \\ 11 & 9 & 11 & 11 \end{pmatrix}$$

Proof. This is a consequence of the characteristic 2 of $\mathbf{F}_{2^m}$. Let us compute the coefficients of $\boldsymbol{M} \stackrel{\text{def}}{=} \boldsymbol{P}_{a,b}\boldsymbol{P}_{c,d}{}^\intercal$ and show that this is the identity matrix.

For $b_i, b_j \in \mathbf{F}_2$, denote $\mathcal{L}_{i,j}(b_i, b_j) \stackrel{\text{def}}{=} |\{k \in [\![0, n-1]\!], (p_{i,k}, p_{j,k}) = (b_i, b_j)\}|$, *i.e.* the number of columns for which the entry of the i-th row is b_i and the value of the j-th row is b_j.

Then for $0 \leqslant i, j < n$, we have

$$\begin{aligned}(\boldsymbol{M})_{i,j} &= \sum_{k=0}^{n-1} (\boldsymbol{P}_{a,b})_{i,k} (\boldsymbol{P}_{c,d})_{j,k} \\ &= ac \cdot \mathcal{L}_{i,j}(0,0) + ad \cdot \mathcal{L}_{i,j}(0,1) + bc \cdot \mathcal{L}_{i,j}(1,0) + bd \cdot \mathcal{L}_{i,j}(1,1).\end{aligned}$$

Note that this operation is in $\mathbf{F}_{2^m}$, hence in characteristic two. Thus, only the parity of the $\mathcal{L}_{i,j}$ coefficients matters.

When $i = j$, we have $\mathcal{L}_{i,j}(0,1) = \mathcal{L}_{i,j}(1,0) = 0$. As P is orthogonal, $\mathcal{L}_{i,j}(1,1)$ is odd. And because n is even, $\mathcal{L}_{i,j}(0,0)$ is odd too. Hence,

$$(\boldsymbol{M})_{i,j} = ac + bd = 1.$$

When $i \neq j$, $\mathcal{L}_{i,j}(1,1)$ is even because $\boldsymbol{P}$ is orthogonal. As the weight of each row of $\boldsymbol{P}$ is odd, $\mathcal{L}_{i,j}(0,1)$ and $\mathcal{L}_{i,j}(1,0)$ have to be odd. And because n is even, $\mathcal{L}_{i,j}(0,0)$ is even too. Hence,

$$(\boldsymbol{M})_{i,j} = ad + bc = 2\frac{ab}{a^2 + b^2} = 0.$$

□

Concatenation. For two vectors (or matrices) $\boldsymbol{x}$ and $\boldsymbol{y}$, we will denote $\boldsymbol{x}||\boldsymbol{y}$ their concatenation.

4.2.2 Key generation

Given the public parameters m, n, k, r, ν, λ such that n is even and $k \leqslant r \leqslant n$, the keys are generated by the following procedure.

1. Let a and b denote two randomly chosen non-zero elements of $\mathbf{F}_{2^m}$ such that $a \neq b$.
2. Let $\boldsymbol{P} \in \mathbf{F}_2^{n \times n}$ be chosen uniformly at random among $n \times n$ orthogonal matrices.
3. Let $\boldsymbol{H} \in \mathbf{F}_2^{r \times n}$ denote a random binary matrix.
4. Define a subspace $\mathcal{V}_g \subseteq \mathbf{F}_{2^m}$ of dimension ν by randomly choosing ν elements $g_0, \ldots, g_{\nu-1} \in \mathbf{F}_{2^m}$. Denote $\mathcal{B}_g \stackrel{\text{def}}{=} (g_0, \ldots, g_{\nu-1})$. $\mathcal{B}_g$ is a basis of $\mathcal{V}_g$.
5. Choose $\boldsymbol{G} \in \mathbf{F}_{2^m}^{k \times n}$ such that all entries of $\boldsymbol{G}$ are in $\mathcal{V}_g$ and $\boldsymbol{G}\boldsymbol{H}^\intercal = \boldsymbol{0}_{k \times r}$.
6. Define $\boldsymbol{G}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{G}\boldsymbol{P}_{c,d}{}^\intercal$, where c and d are defined as in Proposition 4.11.
7. The public key is $\boldsymbol{G}_{\text{pub}}$, the secret key is $(a, b, \boldsymbol{P}, \boldsymbol{H})$.

4.2.3 Encapsulation

Given the public key and the public parameters.

1. Choose a random vector $\boldsymbol{m} \in \mathbf{F}_{2^m}^k$.
2. Define a subspace $\mathcal{V}_e \subseteq \mathbf{F}_{2^m}$ of dimension λ by choosing a basis $\mathcal{B}_e \stackrel{\text{def}}{=} (\tilde{e}_0, \ldots, \tilde{e}_{\lambda-1}) \in \mathbf{F}_{2^m}^{\lambda}$ as follows:
   - choose $(\tilde{e}_0, \tilde{e}_1)$ randomly in $\mathbf{F}_{2^m}$;
   - for $1 \leqslant i \leqslant \lambda/2 - 1$, define $(\tilde{e}_{2i}, \tilde{e}_{2i+1}) \stackrel{\text{def}}{=} \mathcal{H}\left(\tilde{e}_{2i-2} || \tilde{e}_{2i-1}\right)$.

   Denote $\mathcal{V}_e$ the subspace of $\mathbf{F}_{2^m}$ spanned by the elements of $\mathcal{B}_e$.
3. Choose a random vector $\boldsymbol{e} \in \mathbf{F}_{2^m}^n$ such that $\mathbf{Support}(\boldsymbol{e}) \subseteq \mathcal{V}_e$.
4. Let $\boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$.
5. Let $(s_0, s_1) \stackrel{\text{def}}{=} \mathcal{H}\left(\tilde{e}_{L-2} || \tilde{e}_{L-1}\right)$.
6. The shared secret is $\mathcal{H}\left(s_0 || s_1 || \mathcal{H}\left(\boldsymbol{c}\right)\right)$.
7. Let $h \stackrel{\text{def}}{=} \mathcal{H}\left(s_1 || s_0 || \mathcal{H}\left(\boldsymbol{c}\right)\right)$. The ciphertext is $(\boldsymbol{c}, h)$.

4.2.4 Decapsulation

Given the ciphertext, the shared secret, and the public parameters.

1. Recover $\boldsymbol{e}$ by decoding $\boldsymbol{c}$ using the private matrix $\boldsymbol{H}_{\text{sec}} \stackrel{\text{def}}{=} \boldsymbol{H}\boldsymbol{P}_{a,b}{}^{\intercal}$.
2. Deduce $\mathcal{V}_e$ the vector space spanned by the coefficients of the vector $\boldsymbol{e}$.
3. For all $(x, y) \in \mathcal{V}_e \times \mathcal{V}_e$, for $1 \leqslant i \leqslant \lambda/2 - 1$ compute $(s_0', s_1') \stackrel{\text{def}}{=} \mathcal{H}^i\left(x || y || \mathcal{H}\left(\boldsymbol{c}\right)\right)$. If $\mathcal{H}\left(s_1' || s_0' || \boldsymbol{c}\right) = h$, then the shared secret is obtained by computing $\mathcal{H}\left(s_0' || s_1' || \mathcal{H}\left(\boldsymbol{c}\right)\right)$.

Remark 4.13. *Note that step 2 relies on the hypothesis that* $\mathbf{Support}(\boldsymbol{e}) = \mathcal{V}_e$. *This is verified with high probability for* $n >> \lambda$. *More exactly, the probability that* $\mathbf{Support}(\boldsymbol{e})$ *is of dimension* $< \lambda$ *is* $\left(\frac{\lambda-1}{\lambda}\right)^N$. *For the parameters of* edonk128ref *this probability is* 2^{-37}. *One could also explicitly ask for this condition to be fulfilled in step 3 of the encapsulation process.*

Remark 4.14. *The last step of the decapsulation brute-forces all possible couples of elements of* $\mathcal{V}_e$. *In total this operation requires* $\mathcal{O}(\lambda 2^{2\lambda})$ *operations. This requires that the value of* λ *remains small.*

4.2.5 Suggested parameters

The parameters for Edon-K are given in Table 4.1. In this chapter we often refer to the parameters of edonk128ref, the reference version proposed for 128 security-bits.

Table 4.1: Parameters proposed for Edon-K

Name	m	n	k	r	ν	λ
edonk128ref	**128**	**144**	**16**	**40**	**8**	**6**
edonk128K16N80nu8L6	128	80	16	40	8	6
edonk128K08N72nu8L8	128	72	8	40	8	8
edonk128K32N96nu4L4	128	96	32	40	4	4
edonk128K16N80nu4L6	128	80	16	40	4	6
edonk192ref	192	112	16	40	8	8
edonk192K48N144nu4L4	192	144	48	40	4	4
edonk192K32N128nu4L6	192	128	32	40	4	6
edonk192K16N112nu4L8	192	112	16	40	4	8

4.3 Algebraic attack on the Edon-K scheme

4.3.1 Outline of the attack

Our attack is based on three observations

1. The ciphertext is a vector $\boldsymbol{c}$ such that

$$\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}. \tag{4.2}$$

   This error $\boldsymbol{e}$ is of low rank, since its rank is at most λ.

2. This code $\mathscr{C}_{\text{pub}}$ generated by $\boldsymbol{G}_{\text{pub}}$ is a subcode of an LRPC code, namely the code $\mathscr{C}_{\text{sec}}$ with parity-check matrix $\boldsymbol{H}_{\text{sec}} \stackrel{\text{def}}{=} \boldsymbol{H}\boldsymbol{P}_{a,b}{}^{\mathsf{T}}$. This code is indeed an LRPC code of rank 2 since all the entries of $\boldsymbol{H}_{\text{sec}}$ belong to $\langle a, b \rangle_{\mathbf{F}_2}$. We have

$$\mathscr{C}_{\text{pub}} \subseteq \mathscr{C}_{\text{sec}} \tag{4.3}$$

since

$$\begin{aligned}\boldsymbol{G}_{\text{pub}}\boldsymbol{H}_{\text{sec}}^{\intercal} &= \boldsymbol{G}\boldsymbol{P}_{c,d}^{\intercal}(\boldsymbol{H}\boldsymbol{P}_{a,b}^{\intercal})^{\intercal} \\ &= \boldsymbol{G}\boldsymbol{P}_{c,d}^{\intercal}\boldsymbol{P}_{a,b}\boldsymbol{H}^{\intercal} \\ &= \boldsymbol{G}\boldsymbol{H}^{\intercal} \ \text{(from (4.1))} \\ &= \boldsymbol{0}_{k\times r} \ \text{(by definition of } \boldsymbol{G}\text{, see step 5 of key generation).}\end{aligned}$$

This equation also appears as Corollary 1 of [GG17, p.19]. We have given its proof here for the convenience of the reader. Let $k' = n - r$ be the dimension of $\mathscr{C}_{\text{sec}}$.

3. If we recover a parity-check matrix of rank 2 for $\mathscr{C}_{\text{sec}}$ we will be able to recover $\boldsymbol{m}\boldsymbol{G}_{\text{pub}}$ and $\boldsymbol{e}$ from $\boldsymbol{c}$. Indeed, $\boldsymbol{m}\boldsymbol{G}_{\text{pub}} \in \mathscr{C}_{\text{sec}}$ and we can decode in $\mathscr{C}_{\text{sec}}$ using a variation of Algorithm 1 of [GMRZ13] and the knowledge of the parity-check matrix, provided $\mathbf{w}_R(\boldsymbol{e}) \leqslant \lambda < (n-k')/2 = r/2$ is verified, which is the case for the parameters of Edon-K.

Hence the attack proceeds in three steps.

1. constructing and solving a linear system of equations to find a parity-check matrix for the code $\mathscr{C}_{\text{sec}}$;
2. decoding the ciphertext using a variation of Algorithm 1 of [GMRZ13];
3. recovering the secret from the error vector, following the decapsulation procedure.

The first two steps are detailed in the rest of this section.

4.3.2 Reconstructing the parity-check matrix

4.3.2.1 Compressed public key

In order to reduce the public key size, the designers of Edon-K chose to represent the public key in a compressed form. They took advantage of the fact that all the coefficients of $\boldsymbol{G}_{\text{pub}}$ live in the vector space $\mathcal{V}_{\text{pub}}$ defined as $\mathcal{V}_{\text{pub}} \stackrel{\text{def}}{=} \langle cg_1, \ldots, cg_\nu, dg_1, \ldots, dg_\nu \rangle_{\mathbf{F}_2}$ of dimension 2ν. Hence, the compressed public key consists in two parts: first the basis $\mathcal{B}_{\text{pub}} \stackrel{\text{def}}{=} (cg_1, \ldots, cg_\nu, dg_1, \ldots, dg_\nu) \in \mathbf{F}_{2^m}^{2\nu}$ of the vector-space $\mathcal{V}_{\text{pub}}$, then the entries of the matrix $\boldsymbol{G}_{\text{pub}}$ such that each entry is represented by its coefficients in the basis $\mathcal{B}_{\text{pub}}$. For example, if an entry x of $\boldsymbol{G}_{\text{pub}}$ is equal to $c\sum_{i=1}^{\nu}\gamma_i g_i + d\sum_{i=1}^{\nu}\delta_i g_i$ with $\gamma_i, \delta_i \in \mathbf{F}_2$, x will be represented by $(\gamma_1, \ldots, \gamma_\nu, \delta_1, \ldots, \delta_\nu) \in \mathbf{F}_2^{2\nu}$. There is another subtlety in the compression that we do not mention here.

4.3.2.2 Finding a basis

The attacker does not have access to the value of a and b but can deduce the value of $ab^{-1} = cd^{-1} = (cg_1)(dg_1)^{-1}$ from $\mathcal{B}_{\text{pub}}$ as mentioned in paragraph 7.2.2 of the documentation of Edon-K [GG17].

Let us define

$$\alpha \stackrel{\text{def}}{=} ab^{-1}.$$

Notice that $\boldsymbol{H}' \stackrel{\text{def}}{=} b^{-1}\boldsymbol{H}_{\text{sec}}$ is also a parity-check matrix of the LRPC code $\mathscr{C}_{\text{sec}}$. This matrix has all its coefficients in $\langle 1, \alpha \rangle_{\mathbf{F}_2}$. We can use this information to reconstruct such a parity-check matrix of the code $\mathscr{C}_{\text{sec}}$ by solving a linear system, similarly to what is done in [GRS16, Section IV B]. This system is derived from the following facts:

(i) $\boldsymbol{G}_{\text{pub}}\, \boldsymbol{H}'^{\mathsf{T}} = \boldsymbol{0}_{K\times R}$;

(ii) the entries of $\boldsymbol{H}'$ belong to $\langle 1, \alpha \rangle_{\mathbf{F}_2}$, where the value of α is known.

In other words, the possible rows $\boldsymbol{x} = (x_1, \ldots, x_n)$ of $\boldsymbol{H}'$ are solutions of the following system

$$\begin{cases} \boldsymbol{G}_{\text{pub}}\boldsymbol{x}^{\mathsf{T}} = \boldsymbol{0}_k \\ x_i \in \langle 1, \alpha \rangle_{\mathbf{F}_2} \text{ for all } i \in [\![1, n]\!]. \end{cases} \tag{4.4}$$

This system is linear over $\mathbf{F}_2$ and the solution set is an $\mathbf{F}_2$-linear subspace which can be computed in $O(kmn^2)$ operations. A basis of this subspace can then be used as rows for $\boldsymbol{H}'$.

4.3.2.3 The linear system

Let us now present in details how to transform the system defined by (4.4) into a proper linear system over $\mathbf{F}_2$.

Actually in this section we will consider a more general version of (4.4). Given a system

$$\boldsymbol{A}\boldsymbol{x}^{\mathsf{T}} = \boldsymbol{b}^{\mathsf{T}} \tag{4.5}$$

where $\boldsymbol{A} = (a_{ij})_{1\leqslant i\leqslant k, 1\leqslant j\leqslant n}$ is a given matrix in $\mathbf{F}_{2^m}^{k\times n}$ and $\boldsymbol{b}$ is a given vector in $\mathbf{F}_{2^m}^{k}$, and given $\mathcal{V}$ a subspace of dimension t of $\mathbf{F}_{2^m}$ (viewed as vector space over $\mathbf{F}_2$ of dimension m), how to find the affine set of the solutions $\boldsymbol{x} = (x_i)_{1\leqslant i\leqslant n} \in \mathcal{V}^n$ of the system?

We can rewrite the system (4.5) as

$$\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ \qquad\cdots \qquad\qquad\quad = \cdots \\ a_{k1}x_1 + \cdots + a_{kn}x_n = b_k. \end{cases} \tag{4.6}$$

We introduce a basis $(v_1, \ldots, v_t)$ of $\mathcal{V}$ and express each unknown x_j in this basis in terms of t other unknowns $x_{j1}, \ldots, x_{jt} \in \mathbf{F}_2$:

$$x_j = \sum_{i=1}^{t} x_{ji} v_i.$$

In other words, the system (4.5) is equivalent to

$$\begin{cases} \sum_{j=1}^{n} \sum_{i=1}^{t} a_{1j} v_i x_{ji} &= b_1 \\ \qquad \ldots & = \ldots \\ \sum_{j=1}^{n} \sum_{i=1}^{t} a_{kj} v_i x_{ji} &= b_k. \end{cases} \tag{4.7}$$

Let $(\beta_1, \ldots, \beta_m)$ be an $\mathbf{F}_2$-basis of $\mathbf{F}_{2^m}$, we introduce for $1 \leqslant \ell \leqslant m$ the projection π_ℓ from $\mathbf{F}_{2^m}$ to $\mathbf{F}_2$ defined by:

$$\begin{array}{rccc} \pi_\ell : & \mathbf{F}_{2^m} & \longrightarrow & \mathbf{F}_2 \\ & a = \sum_{j=1}^{m} a_j \beta_j & \longmapsto & a_\ell. \end{array} \tag{4.8}$$

The k equations of system (4.7) defined over $\mathbf{F}_{2^m}$ lead to km affine equations over $\mathbf{F}_2$ by applying π_ℓ for $\ell \in [\![1, m]\!]$:

$$\begin{cases} \sum_{j=1}^{n} \sum_{i=1}^{t} \pi_\ell(a_{1j} v_i) x_{ji} &= \pi_\ell(b_1) \\ \qquad \ldots & = \quad \ldots \\ \sum_{j=1}^{n} \sum_{i=1}^{t} \pi_\ell(a_{kj} v_i) x_{ji} &= \pi_\ell(b_k). \end{cases} \tag{4.9}$$

We can solve this affine system in $\mathbf{F}_2$ to recover the solution of (4.5). The system has km binary equations and tn unknowns, hence a complexity of $\mathcal{O}(kmt^2n^2)$. If we apply this technique to (4.4), where $t = 2$, we obtain a basis of the vector space in time $O(kmn^2)$.

4.3.3 The decoding step

The previous step recovers an $k \times n$ matrix $\boldsymbol{H}''$ whose entries all belong to $\langle 1, \alpha \rangle_{\mathbf{F}_2}$. The matrices $\boldsymbol{H}'$ and $\boldsymbol{H}''$ share the property that their rows form a basis of solutions of (4.4). Therefore, there exists an $r \times r$ binary invertible matrix $\boldsymbol{Q}$ such that

$$\boldsymbol{H}'' = \boldsymbol{Q}\boldsymbol{H}'. \tag{4.10}$$

We use $\boldsymbol{H}''$ to decode and recover $\boldsymbol{e}$ from the ciphertext $\boldsymbol{c}$. The vectors are linked by the equation

$$\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}. \tag{4.11}$$

We use here a slight variation of Algorithm 10 to decode. Algorithm 10 would consist in performing the following steps:

1. Compute $\boldsymbol{s}^\intercal \stackrel{\text{def}}{=} \boldsymbol{H}''\boldsymbol{c}^\intercal$ and then $\mathcal{V} \stackrel{\text{def}}{=} \mathbf{Support}(\boldsymbol{s})$. Here we typically have $\mathcal{V} = \mathbf{Support}(\boldsymbol{e}) \cdot \langle 1, \alpha \rangle_{\mathbf{F}_2}$ when $\boldsymbol{H}''$ is a random matrix.

2. Compute $\mathcal{V}' \stackrel{\text{def}}{=} \mathcal{V} \cap \alpha^{-1}\mathcal{V}$. This step typically recovers $\mathbf{Support}(\boldsymbol{e})$ when $\mathcal{V} = \mathbf{Support}(\boldsymbol{e}) \cdot \langle 1, \alpha \rangle_{\mathbf{F}_2}$.

3. Once we have $\mathbf{Support}(\boldsymbol{e})$ we recover $\boldsymbol{e} = (e_1, \ldots, e_n)$ by solving the linear equation $\boldsymbol{H}''\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$ with the additional constraints $e_i \in \mathbf{Support}(\boldsymbol{e})$ for $i \in [\![1, n]\!]$. This is done by using the technique presented in Subsection 4.3.2.3.

However, in our case, $\mathcal{V}$ is not equal to $\mathbf{Support}(\boldsymbol{e}) \cdot \langle 1, \alpha \rangle_{\mathbf{F}_2}$. This is due to the special structure of $\boldsymbol{H}$ which contains only a's and b's . The following result characterises this situation.

Proposition 4.15. *We have for every $\boldsymbol{e} \in \mathbf{F}_{2^m}^n$:*

$$\mathbf{Support}(\boldsymbol{H}''\boldsymbol{e}^\intercal) \subseteq (1+\alpha)\mathbf{Support}(\boldsymbol{e}) + \left\langle \sum_{i=1}^{n} e_i \right\rangle_{\mathbf{F}_2}.$$

Proof. From (4.10), we deduce that

$$\mathbf{Support}(\boldsymbol{H}''\boldsymbol{e}^\intercal) = \mathbf{Support}(\boldsymbol{Q}\boldsymbol{H}'\boldsymbol{e}^\intercal) = \mathbf{Support}(\boldsymbol{H}'\boldsymbol{e}^\intercal).$$

Let $\boldsymbol{s}^\intercal \stackrel{\text{def}}{=} \boldsymbol{H}'\boldsymbol{e}^\intercal$. Denote the i-entry of $\boldsymbol{s}$ by s_i and the entry of $\boldsymbol{H}'$ in row i and column j by h_{ij}. We have:

$$\begin{aligned} s_i &= \sum_{j=1}^{n} h_{ij} e_j \\ &= \sum_{j \text{ s.t. } h_{ij}=1} e_j + \sum_{j \text{ s.t. } h_{ij}=\alpha} \alpha e_j \\ &= \sum_{j=1}^{n} e_j + (1+\alpha) \sum_{j \text{ s.t. } h_{ij}=\alpha} e_j. \end{aligned}$$

This implies the proposition. □

Hence, Proposition 4.15, states that $\mathbf{Support}(\boldsymbol{H}''\boldsymbol{e}^\intercal)$ directly yields a subspace of dimension $\lambda + 1$ that contains $\mathbf{Support}(\boldsymbol{e})$:

$$\mathbf{Support}(\boldsymbol{e}) \subseteq (1+\alpha)^{-1}\mathbf{Support}(\boldsymbol{H}''\boldsymbol{e}^\intercal). \tag{4.12}$$

A slight modification of Algorithm 1 of [GMRZ13] therefore yields $\boldsymbol{e}$:

1. compute the syndrome $\boldsymbol{s}^\intercal \stackrel{\text{def}}{=} \boldsymbol{H}''\boldsymbol{c}^\intercal$ and then $\mathcal{V} \stackrel{\text{def}}{=} (1+\alpha)^{-1}\mathbf{Support}(\boldsymbol{s})$;

2. The space $\mathcal{V}$ contains $\mathbf{Support}(\boldsymbol{e})$, so we can recover $\boldsymbol{e} = (e_1, \ldots, e_n)$ by solving the linear equation $\boldsymbol{H}''\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$ with the additional constraints $e_i \in \mathcal{V}$ for $i \in [\![1, n]\!]$. This is done by using the technique given in Subsection 4.3.2.3.

From there, one deduces the shared secret from the value of $\boldsymbol{e}$ just like in the decapsulation.

Remark 4.16. *Note that we can also skip step 2 and directly look for s_0 and s_1 in the space $\mathcal{V}$ of dimension $\lambda + 1$ instead of decoding exactly the value of $\boldsymbol{e}$. In fact, this is what is specified in the decapsulation of Edon-K* [*GG17*].

4.4 Concluding remarks

4.4.1 Cost of the attack

We can analyse the cost of the three steps of the attack mentioned in Section 4.3.

Step 1 and 2 are polynomial in terms of the parameters of the code. Step 1 only uses linear algebra operations and has a complexity at most $\mathcal{O}(kmn^2)$. The complexity of step 2 is given by Theorem 1 of [GMRZ13] (using $k = n - r$, $r = \lambda$, $d = 2$), hence is equal to $\lambda^2(16m + n^2)$. The complexity of step 3 is $\mathcal{O}(\lambda 2^{2\lambda})$. This is not polynomial in λ but λ is a very small parameter ($4 \leqslant \lambda \leqslant 8$ in the proposal). Moreover this third step is the same as the decapsulation algorithm, so λ needs to stay small, otherwise the decapsulation would become too costly or even impossible (see Remark 4.14). So λ can be considered as a constant $\leqslant 10$ to allow a reasonable decapsulation. Hence, the complexity is given by the most costly operation, which is step 1.

4.4.2 Without compression of the public key

This attack takes advantage of the compressed form of the public key that allows a direct access to the value $\alpha = ab^{-1}$. One could think that this is the origin of the attack, and decide to express the public key in its uncompressed form to fix the attack. As a consequence, the public key would be of size $k \times n \times m$ bits instead of $k \times n \times \nu$ bits in the compressed form. In practice the public key for edonk128ref would be 16 times longer (around 288 kbits). This inflation of the key size could be avoided by sending out a random basis of the space $\mathcal{V}_{\text{pub}}$.

However, this is not enough to mitigate the attack. There is an even more direct way to proceed, without the value of α. Indeed, instead of looking for a matrix $\boldsymbol{H}''$ with entries lying in $\langle 1, \alpha \rangle_{\mathbf{F}_2}$, we can use the following result.

Proposition 4.17. *There exists a full rank* $(r-1) \times n$ *binary matrix* $\boldsymbol{H}_{bin}$ *that satisfies*

$$\boldsymbol{G}_{pub}\boldsymbol{H}_{bin}{}^{\intercal} = \mathbf{0}_{k\times(r-1)}.$$

Proof. Let $\boldsymbol{T}$ be a binary full-rank matrix $(r-1) \times r$ matrix that has rows of even Hamming weight. For instance we can choose

$$\boldsymbol{T} = \begin{pmatrix} 1 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 1 & 0 & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & 1 & 1 \end{pmatrix}.$$

We observe that $\boldsymbol{T}\boldsymbol{H}_{\text{bin}}$ has all its entries in $\{0, a+b\}$. This follows directly from the fact that if we sum an even number of elements in $\{a, b\}$ we either get 0 (if the number of a's is even, and therefore also the number of b's) or $a+b$ (if the number of a's is odd). From this, it follows immediately that

$$\boldsymbol{H}_{\text{bin}} \stackrel{\text{def}}{=} \frac{1}{a+b}\boldsymbol{T}\boldsymbol{H}$$

satisfies the property. □

Obtaining such a matrix $\boldsymbol{H}_{\text{bin}}$ is straightforward. We just have to use the algorithm given in Section 4.3.2 to recover a basis of dimension $r-1$ of binary vectors $\boldsymbol{x}$ satisfying

$$\boldsymbol{G}_{\text{pub}}\boldsymbol{x}^{\intercal} = \mathbf{0}_k.$$

We then use this matrix $\boldsymbol{H}_{\text{bin}}$ to compute the syndrome $\boldsymbol{s}^{\intercal} = \boldsymbol{H}_{\text{bin}}\boldsymbol{c}^{\intercal}$. Since $\boldsymbol{H}_{\text{bin}}\boldsymbol{c}^{\intercal} = \boldsymbol{H}_{\text{bin}}\boldsymbol{e}^{\intercal}$ we directly obtain with very high probability that

$$\mathbf{Support}(\boldsymbol{e}) = \mathbf{Support}(\boldsymbol{s}).$$

This reveals the support of the error and from there we can go directly to the last step of the attack to reconstruct the shared secret.

4.4.3 Conclusion

This attack shows that there is a way to recover the secret of the edonk128ref scheme from a public key without the private key in polynomial time. In practice, the attack implemented with Sage on a personal computer recovers the secret in less than a minute, so the scheme is far from achieving the 128-bits security claimed in [GG17]. Hence this scheme is insecure for the intended use. Moreover, the cost of this attack is polynomial in terms of the parameters, so there is no proper way to increase the parameters to achieve the intended

security level while keeping a reasonably small key size. Following this attack, the Edon-K scheme was removed from the NIST standardization process.

The idea behind Edon-K, consisting in using a secret key with entries in a subspace of small dimension, hidden in a large field, enables compact key sizes and interesting decoding properties. This idea is a the heart of the definition of LRPC codes. However, it seems that this notion was known to the authors of the Edon-K proposal, since the link with rank metric codes is not mentioned in the description of the scheme. Unfortunately, the choice of the dimension of the subspace in (dimension 2 or even 1) in the design of the Edon-K scheme is too small to ensure security.

A more reasonable use of LRPC codes can lead to interesting cryptosystem that are out of reach of such an attack. For instance, the key encapsulation mechanism LAKE and the public key encryption scheme LOCKER (later merged under the name "ROLLO") [ABDGH+19] use LRPC codes. The security of these scheme rely on the RSD problem (Problem 4.5) and the LRPC problem (Problem 4.10). These schemes were selected for the second round of the NIST standardization process. But recent algebraic attacks [BBBGN+20; BBCGP+20] make use of a new model to state the rank decoding problem in terms of multivariate equations and manage to break nearly all proposed parameters of ROLLO.

As a result, these schemes were not selected for the third round of the NIST process. Still, the NIST stated that LRPC codes remain an interesting tool for cryptography. In its latest report, NIST judged that *"the rank metric cryptosystems offer a nice alternative to traditional hamming metric codes with comparable bandwidth"* [AAACD+20] and encouraged further study of this line of work.

Part II

Square-code attacks on GRS-based cryptosystems

Chapters

Chapter 5

GRS codes and public-key cryptography

This chapter introduces the notion of Generalised Reed–Solomon (GRS) codes and their use in code-based cryptography. These codes have an algebraic structure that allows for an efficient and deterministic decoding algorithm, without decryption failures. This, among other properties, make them appealing for cryptographic use. Several cryptosystems similar to McEliece's scheme but relying on GRS codes have been proposed in the last decade. The idea dates back to Niederreiter, who proposed to replace Goppa codes by the use of raw GRS codes in McEliece's scheme. But an attack was found by Sidelnikov and Shestakov. Other proposals use codes derived from GRS codes that seem to resist this approach, but the introduction of a new tool, the *square code distinguisher* showed the weakness of these schemes. We describe this distinguisher, on which we will rely in the next chapters to cryptanalyse two recent GRS-based cryptosystems.

Contents

5.1 Generalised Reed–Solomon codes

5.1.1 Definition and properties

Reed–Solomon codes were formally introduced by Reed and Solomon in 1960 [RS60] as *polynomial* codes. In fact, the same structure had already been proposed eight years earlier by Bush as *orthogonal arrays of index unity* [Bus52] (but not in the context of error correction). Generalised Reed–Solomon codes were formally defined by Delsarte in [Del75] under the denomination *modified Reed–Solomon* codes.

Definition 5.1 (Generalised Reed–Solomon codes)**.** Let $\boldsymbol{x} \in \mathbf{F}_q^n$ be a vector whose entries are pairwise distinct and $\boldsymbol{y} \in \mathbf{F}_q^n$ be a vector whose entries are all nonzero. The *generalised Reed–Solomon* (*GRS*) *code with support* $\boldsymbol{x}$ *and multiplier* $\boldsymbol{y}$ *of dimension* k is defined as

$$\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \{(y_1 f(x_1), \ldots, y_n f(x_n)) \mid f \in \mathbf{F}_q[X]_{<k}\}\,.$$

When $\boldsymbol{y} = (1, \ldots, 1)$, the code is a *Reed–Solomon* (*RS*) code, denoted $\mathbf{RS}_k(\boldsymbol{x})$.

Remark 5.2. *For a given GRS code, the support and multiplier are not unique. For instance, applying an affine transformation to* $\boldsymbol{x}$ *generates the same code, since the set of polynomials of fixed degree is stable under this transformation. See Remark 1.29.*

Reed–Solomon codes and their generalisation have interesting properties. Therefore, they have been widely used in practice, for instance in the encoding of CDs, DVDs and QR codes. Moreover, to describe the code, one only needs to specify the vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ (*i.e.* $2n$ elements of $\mathbf{F}_q$), not the complete generator matrix (which would require to send at least $k(n-k)$ elements of $\mathbf{F}_{q^m}$ if one chooses to represent the generator matrix in systematic form). In cryptographic schemes, the GRS codes usually corresponds to (some part of) the secret key. Therefore this compact way to describe the code gives short secret key, which is one of the main improvements that new code-based cryptographic schemes try to achieve. This explains the interest for GRS-based cryptosystems.

In this section, we detail some of the interesting properties of GRS codes.

5.1.1.1 MDS codes

First of all, GRS codes are maximal distance separable (MDS) codes, *i.e.* they reach the Singleton bound (see Theorem 1.21).

Property 5.3. $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ *is an* $[n, k, n-k+1]$ *code.*

Proof. The code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ is the image of the map

$$\begin{cases} \mathbf{F}_q[X]_{<k} & \longrightarrow & \mathbf{F}_q^n \\ f & \longmapsto & (y_1 f(x_1), \ldots, y_n f(x_n)). \end{cases}$$

This application is injective. Indeed, the only polynomial of degree less than k with n distinct roots is the null polynomial. Hence $\dim \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) = k$.

Let $\boldsymbol{c} = (y_1 f(x_1), \ldots, y_n f(x_n))$ be a non-zero codeword of $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$. The polynomial f is not null so it has at most $k-1$ roots, hence $\boldsymbol{c}$ has at least $n-k+1$ non-zero entries. So the minimal distance of $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ is at least $n-k+1$. The Singleton bound asserts that this distance is at most $n-k+1$, hence we have an equality. □

5.1.1.2 Decoding GRS codes

GRS codes benefit from efficient decoding algorithms. We present here an algorithm due to Berlekamp and Welsch in 1986 [WB86] which can correct up to t errors, where $t \stackrel{\text{def}}{=} \lfloor \frac{n-k}{2} \rfloor$.

The Berlekamp Welsch decoder. Let $\boldsymbol{v} = (v_1, \ldots, v_n)$ be a noisy codeword, such that $\boldsymbol{v} = \boldsymbol{c} + \boldsymbol{e}$ with $\boldsymbol{c} \in \mathbf{RS}_k(\boldsymbol{x})$ and $\mathbf{w}_H(\boldsymbol{e}) \leqslant t$. The value of $\boldsymbol{v}$ is known as well as the value of $\boldsymbol{x}$ generating the code, the goal is to find $\boldsymbol{c}$ and $\boldsymbol{e}$. By definition, there exists a polynomial $f \in \mathbf{F}_q[X]_{<k}$ such that $\boldsymbol{c} = (f(x_1), \ldots, f(x_n))$.

Let us define the polynomial E such that $E(X) \stackrel{\text{def}}{=} \prod_{i \text{ s.t. } e_i \neq 0}(X - x_i)$. The main idea of the Berlekamp Welsch algorithm is that for all $i \in [\![1, n]\!]$, we have

$$v_i E(x_i) = f(x_i) E(x_i).$$

Indeed, either $e_i = 0$ and $v_i = f(x_i)$, or $E(x_i) = 0$. This gives a system of n equations, where the values of v_i and x_i are known, hence the unknowns are the coefficients of E and f. Note that the right-hand side is not linear but can be linearised. This gives a system with $k + 2t + 1$ coefficients. Any non-trivial solution of the system gives a value of f and hence the value of $\boldsymbol{v}$.

The algorithm is presented for RS codes but can be easily generalised for GRS codes. Indeed, to decode a noisy codeword $\boldsymbol{v} = (v_1, \ldots, v_n) = \boldsymbol{c} + \boldsymbol{e}$ where $\boldsymbol{c} \in \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ and $\mathbf{w}_H(\boldsymbol{e}) \leqslant t$ one can equivalently decode $(v_1/y_1, \ldots, v_n/y_n)$ in the code $\mathbf{RS}_k(\boldsymbol{x})$ using this decoder.

This algorithm runs in time $\mathcal{O}(n^3)$ (the resolution of the linear system). In fact, it can be improved to run in $\mathcal{O}(n^2)$ or even $\mathcal{O}(n \log(n))$ using the Euclidian algorithm instead of linear algebra ([MS86] Chapter 12).

Note that these efficient decoding algorithms only works if the decoder has access to the values of $\boldsymbol{x}$ and $\boldsymbol{y}$. This will serve as a trapdoor for cryptosystems.

In the late 1990's, works from Guruswami and Sudan [Sud97; GS98] allowed to decode GRS codes efficiently beyond the decoding radius. In such a case, one must expect more than one nearest codeword. This approach is known as *list decoding*.

These properties makes GRS codes interesting for cryptographic purposes. In cryptographic applications, the decryption phase usually involves decoding. Having an efficient decoding algorithm makes decryption efficient. Besides, the decoding algorithms for GRS codes are deterministic. They have no decoding failure and can easily be implemented to run in constant time, contrary to the decoding of MDPC codes (see Chapter 3). This mitigates the risk of side-channel attacks.

5.1.1.3 Dual of GRS codes

Lemma 5.4 ([MS86] Chapter 10, Theorem 4)**.** *The dual of a GRS code of length n and dimension k is a GRS code of dimension $n-k$. More precisely:*

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

where y'_i depends uniquely of $\boldsymbol{x}$ and $\boldsymbol{y}$.

5.1.2 Relation with other families of codes

Generalised Reed–Solomon codes are a special case of the family of BCH codes. They can also be interpreted as the family of algebraic geometry codes over the projective line [VNT07].

GRS codes are also at the core of the definition of alternant codes. Indeed, alternant codes are subfield subcodes of GRS codes and inherit some properties of the GRS codes.

Definition 5.5 (Alternant codes)**.** Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote a support and a multiplier defined over $\mathbf{F}_{q^m}$. Let r be an integer. The *alternant code* $\mathscr{A}_{r,q}(\boldsymbol{x},\boldsymbol{y})$ is defined as

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

The Goppa codes, used in McEliece's seminal code-based cryptographic scheme, are a particular case of alternant codes, where $\boldsymbol{y}$ is chosen in a particular way to achieve better correction capacity. This scheme remains unbroken today. It is therefore particularly interesting to understand attacks on cryptosystems using GRS codes, especially subspace subcodes of GRS codes which are a first step towards subfield subcodes. We will see in Chapter 7 an attempt

to cryptanalyse a family of codes halfway between GRS codes and alternant codes.

5.2 GRS-based cryptosystems

5.2.1 McEliece with GRS codes

The idea of using Reed–Solomon instead of Goppa codes in McEliece's scheme is proposed by Niederreiter [Nie86]. Such a cryptosystem works as follows.

Key generation. Let q denote a prime power and chose integers k and n such that $k \leqslant n \leqslant q-1$. The values of q, k and n are public.

Pick $\boldsymbol{x}$ and $\boldsymbol{y}$ a support and a multiplier of a GRS code of length n over $\mathbf{F}_q$ (as defined in § 5.1.1), chosen uniformly at random. Denote $\mathscr{C}$ the code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$.

The following Vandermonde matrix is a generator matrix of the code $\mathscr{C}$ which can easily be obtained from $\boldsymbol{x}$ and $\boldsymbol{y}$:

$$\boldsymbol{V}_{k,n}(\boldsymbol{x}, \boldsymbol{y}) \stackrel{\text{def}}{=} \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^k y_1 & \cdots & x_n^k y_n \end{pmatrix}.$$

The goal is to use as the pubic key another generator matrix of $\mathscr{C}$ that does not reveal the values of $\boldsymbol{x}$ and $\boldsymbol{y}$. Indeed, decoding the GRS code knowing $\boldsymbol{x}$ and $\boldsymbol{y}$ is very efficient. Therefore, we create another public key of $\mathscr{C}$. Let $\boldsymbol{S}$ be an invertible $k \times k$ matrix over $\mathbf{F}_q$ chosen uniformly at random. Denote $\boldsymbol{G}_{\text{sec}} \stackrel{\text{def}}{=} \boldsymbol{V}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$ and $\boldsymbol{G}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{S}\boldsymbol{G}_{\text{sec}}$.

Let $t \stackrel{\text{def}}{=} \lfloor \frac{n-k}{2} \rfloor$ denote the error-correction capacity of $\mathscr{C}$.

The public key is $(\boldsymbol{G}_{\text{pub}}, t)$. The private key is $(\boldsymbol{x}, \boldsymbol{y})$.

Remark 5.6. *Contrary to the original proposal, we omit the right-multiplication by a random permutation matrix as this does not make any difference in the distribution.*

Encryption. The set of messages is $\mathbf{F}_q^k$. For a given message $\boldsymbol{m}$, the ciphertext is $\boldsymbol{c} \stackrel{\text{def}}{=} \boldsymbol{m} \cdot \boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$, where $\boldsymbol{e}$ is chosen uniformly at random among the set of elements of $\mathbf{F}_q^n$ such that $\mathbf{w}_H(\boldsymbol{e}) = t$.

Decryption. Given a ciphertext $\boldsymbol{c}$, we have $\boldsymbol{c} = (\boldsymbol{m}\boldsymbol{S}) \cdot \boldsymbol{G}_{\text{sec}} + \boldsymbol{e}$, where $\mathbf{w}_H(\boldsymbol{e}) = t$. Using $\boldsymbol{x}$ and $\boldsymbol{y}$, one can decode using the Berlekamp Welsch decoder to find the value of $\boldsymbol{e}$ and deduce $\boldsymbol{m}$.

5.2.2 Attacking the McEliece GRS cryptosystem

The security of this cryptosystem relies on the following problem.

Problem 5.7. *For a GRS code $\mathscr{C}$ of length n and dimension k over $\mathbf{F}_q$, given any generator matrix of $\mathscr{C}$, find a pair of vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\mathscr{C} = \boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$.*

This computational problem also admits a decisional variant.

Problem 5.8. *Let $\mathcal{D}_1$ denote the distribution of matrices $\boldsymbol{M} \stackrel{def}{=} \boldsymbol{S} \cdot \boldsymbol{V}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are a support and a multiplier of a GRS code of length n over $\mathbf{F}_q$, chosen uniformly at random, and $\boldsymbol{S}$ is chosen uniformly at random among the set of invertible $k \times k$ matrices. Let $\mathcal{D}_2$ denote the uniform distribution over $k \times n$ matrices over $\mathbf{F}_q$ of rank k. Distinguish the distributions $\mathcal{D}_1$ and $\mathcal{D}_2$.*

In 1992, Sidelnikov and Shestakov showed that Problem 5.7 can be solved in polynomial time [SS92]. Their attack relies on the following remark.

Proposition 5.9 ([BL05], Corollary 1)**.** *Let $\boldsymbol{B} = (b_{i,j})$ denote the systematic generator matrix of the code $\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$. Then for all i, j, u, v such that $1 \leqslant i, j \leqslant k$ and $k + 1 \leqslant u, v \leqslant n$, the following relation holds*

$$\frac{b_{i,u} b_{j,v}}{b_{j,u} b_{i,v}} = \frac{(x_j - x_u)(x_i - x_v)}{(x_i - x_u)(x_j - x_v)}.$$

The values $b_{i,j}$ are public since they can be obtained by putting the public generator matrix in systematic form. Moreover, it is always possible to chose arbitrarily three values of x_i's and one value of y_i's. Hence, using this relation, one can solve the system and find vectors $\boldsymbol{x}'$ and $\boldsymbol{y}'$ such that $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{GRS}_k(\boldsymbol{x}', \boldsymbol{y}')$.

This makes Niederreiter's cryptosystem insecure. However, the properties of GRS codes remain an appealing idea for short-key code-based cryptosystems. Therefore, several proposals were made to mitigate the Sidelnikov–Shestakov attack by introducing some randomness to hide the underlying GRS structure.

5.2.3 Other cryptosystems using GRS codes

Berger–Loidreau. In [BL05], Berger and Loidreau proposed using subcodes of GRS codes as the public key. That is, they still define $\boldsymbol{G}_{\text{pub}} = \boldsymbol{S}\boldsymbol{G}_{\text{sec}}$ but instead on a $k \times k$ matrix, $\boldsymbol{S}$ is chosen as a random $\ell \times k$ matrix of rank ℓ, for an integer $\ell < k$.

This cryptosystem was cryptanalysed by Wieschebrink in [Wie06a] and [Wie10]. The first article simply generalises the Sidelnikov–Shestakov attack in the case where ℓ is close to k. The second article introduces a new idea.

It proposes to study the square code and take advantage of a fact that the square of the GRS code is also a GRS code. This breaks the Berger–Loidreau cryptosystem completely.

Wieschebrink. In another paper [Wie06b], Wieschebrink proposed another way to avoid the Sidelnikov–Shestakov attack by adding a few random columns to the GRS matrix. In the key generation phase, r random columns are inserted between the columns of the Vandermonde matrix $\boldsymbol{G}$, at random positions. The matrix $\boldsymbol{S}$ is chosen as a random invertible $(k+r) \times (k+r)$ matrix. The rest of the scheme is unchanged.

Wieschebrink's scheme was broken in [CGGOT14]. The authors reuse the square code idea, but in a quite different manner. Indeed, they do not use the square code to directly recover the GRS code but they use its dimension as a distinguisher to find the positions of the random columns. Once they have found the positions of these columns, they just need to discard them and apply the Sidelnikov–Shestakov attack on the columns corresponding to the GRS code.

BBCRS. Another attempt to modify Niederreiter's scheme was proposed in [BBCRS16]. The difference with the original idea is that the matrix $\boldsymbol{G}$ is right-multiplied by the inverse of a matrix of the form $\boldsymbol{T} + \boldsymbol{R}$, where $\boldsymbol{T}$ is a sparse matrix with a very small average row/column weight denoted $m < 2$ and $\boldsymbol{R}$ is a matrix of small rank (in practice the rank is chosen to be exactly one to keep a small key size).

A first version of the BBCRS cryptosystem [BBCRS11] (where $m = 1$) was attacked in [CGGOT14], and a second version $(1 < m < 2)$ was broken in [COTG15]. In both cases, the authors observe the dimension of products of codes do distinguish the different rows and columns of the public matrix and recover the underlying structure.

New proposals based on GRS codes. In recent years, new attempts were made to use GRS codes in public-key cryptosystems.

- The RLCE cryptosystem [Wan17] was submitted by Wang to the NIST call for post-quantum cryptography. This cryptosystem is a more advanced variant of Wieschebrink's idea of introducing random columns. The difference is that the random columns are mixed with columns from the GRS matrix. The cryptosystem was attacked in [CLT19]. The scheme and the attack are detailed in Chapter 6.

- The XGRS cryptosystem [KRW21] proposed by Khaturia, Rosenthal and Weger, uses a different approach. It relies on the notion of subspace

subcodes, introduced by Solomon and McEliece in [MS94], to propose a variant of Niederreiter's scheme. The cryptosystem was attacked in [CL20]. The notion of subspace subcodes as well as the attack are described in Chapter 7.

- In [BGKR19], Berger, Gueye, Klamti and Ruatta introduce a cryptosystem also based on subspace subcodes of GRS codes. But contrarily to the XGRS scheme, in their proposal the underlying GRS code is not secret. Therefore the security of the scheme does not depend on the secrecy of the GRS structure. This scheme has not been subject to any attack for now.

5.3 Product of codes and square-code distinguisher

As we will see, the idea introduced in [CGGOT14] of using the dimension of the square code as a distinguisher is particularly interesting since it can be generalised to different situations. In the paper [CGGOT14], the authors even explain how to use product of codes to solve Problem 5.7 in polynomial time, with a different approach than Sidelnikov–Shestakov. Their algorithm is a bit more complex but generalises better. In particular, using this approach, they managed to attack q-ary Goppa codes (called *wild* Goppa codes [BLP10]) described in [COT14a]. In this section, we will present the idea of the square-code distinguisher, which will be used in the next chapters to attack the RLCE and XGRS cryptosystems.

5.3.1 The star-product operation

Notation 5.10. *The component-wise product* (*or* Schur *product*) *of two vectors* $\boldsymbol{a}$ *and* $\boldsymbol{b}$ *in* $\mathbf{F}_2^n$ *is denoted by*

$$\boldsymbol{a} \star \boldsymbol{b} \stackrel{def}{=} (a_0 b_0, \ldots, a_{n-1} b_{n-1}).$$

This definition extends to the product of codes, where the component-wise product *or* $\star$–product *of two* $\mathbb{K}$*-linear codes* $\mathscr{A}$ *and* $\mathscr{B} \subseteq \mathbf{F}_2^n$ *spanned over a field* $\mathbb{K} \subseteq \mathbf{F}_2$ *is defined as*

$$\mathscr{A} \star_{\mathbb{K}} \mathscr{B} \stackrel{def}{=} \langle \boldsymbol{a} \star \boldsymbol{b} \mid \boldsymbol{a} \in \mathscr{A},\ \boldsymbol{b} \in \mathscr{B} \rangle_{\mathbb{K}}.$$

When $\mathscr{A} = \mathscr{B}$*, we denote by* $\mathscr{A}^{\star 2}_{\mathbb{K}} \stackrel{def}{=} \mathscr{A} \star_{\mathbb{K}} \mathscr{A}$ *the* square *code of* $\mathscr{A}$ *spanned over* $\mathbb{K}$*.*

Remark 5.11. *The field $\mathbb{K}$ in Notation 5.10 is almost always equal to $\mathbf{F}_2$ the base field on which the codes are defined. However, it may sometimes be a subfield. For the sake of clarity, we make the value of $\mathbb{K}$ explicit only in the ambiguous cases. The rest of the time we simply write $\mathscr{A}^{\star 2}$ the square product of a code.*

5.3.2 The square-code distinguisher

The quantity which is of particular interest for cryptanalysis is the dimension of this product of codes.

5.3.2.1 Typical dimension of the square code

Proposition 5.12. *Let $\mathscr{A}$ and $\mathscr{B}$ denote two linear codes of equal length n over $\mathbf{F}_q$. Then we have*

$$\dim \mathscr{A} \star \mathscr{B} \leqslant \min\left(n, \dim \mathscr{A} \cdot \dim \mathscr{B} - \binom{\dim \mathscr{A} \cap \mathscr{B}}{2}\right).$$

In particular

$$\dim \mathscr{A}^{\star 2} \leqslant \min\left(n, \binom{\dim \mathscr{A} + 1}{2}\right).$$

In fact, in the typical case, the last inequality is an equality, as show by [CCMZ15]. Indeed, a first result states roughly that for a random code of dimension n and length k, if $n \leqslant \binom{k+1}{2}$ then the square of the code is equal to $\mathbf{F}_q^n$ with probability close to 1. Here is a formal statement.

Proposition 5.13 ([CCMZ15], Theorem 2.5)**.** *There exist constants $c, c' > 0$ (depending only on q) such that if $n : \mathbf{N} \to \mathbf{N}$ satisfies $k \leqslant n(k) \leqslant c \cdot \binom{k+1}{2}$ for all $k \in \mathbf{N}$, then, for all large enough k,*

$$\Pr\left(\dim \mathscr{C}^{\star 2} = n(k)\right) \geqslant 1 - 2^{-c'k},$$

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

Another result states that if $n \geqslant \binom{k+1}{2}$ then the dimension of the square of the code is equal to $\binom{k+1}{2}$ with probability close to 1. Here is a formal statement.

Proposition 5.14 ([CCMZ15], Theorem 2.3)**.** *There exists a constant $c > 0$ such that if $n : \mathbf{N} \to \mathbf{N}$ satisfies $n(k) \geqslant \binom{k+1}{2}$ for all $k \in \mathbf{N}$, then, for all large enough k,*

$$\Pr\left(\dim \mathscr{C}^{\star 2} = \binom{k+1}{2}\right) \geqslant 1 - 2^{-c\left(n(k) - \binom{k+1}{2}\right)},$$

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

All these results can be summarised in the following informal statement about random codes.

Theorem 5.15 (informal)**.** *For a linear code $\mathscr{R}$ chosen at random over $\mathbf{F}_q$ of dimension k and length n, the dimension of $\mathscr{R}^{\star 2}$ is typically $\min(n, \binom{k+1}{2})$.*

Remark 5.16. *It is important to understand that rewriting these results in such an informal manner makes sense in the context of cryptanalysis. Indeed, for codes used in cryptosystems, the probability not to have an equality in Proposition 5.12 is extremely small. Moreover, in cryptanalysis, it does not matter if an attack fails with small probability. The only important thing is that is successes with non-negligible probability. A cryptosystem that could be broken even with small probability is not a secure cryptosystem!*

In the next chapters, we will generalise this statement to other families of codes and describe attacks that use this informal statement (or equivalent results). It is important to keep in mind that this statement comes from a probabilistic result over random codes. We always conduct some experiments to check that the dimension measured in practice matches the result with very high probability. Hence, there is always a small possibility that the derived distinguisher fails for a particular instance but this does not affect the efficiency of the attacks.

5.3.2.2 Square code of a GRS code

Concerning GRS codes, their behaviour is very different.

Proposition 5.17. *Let $n, k_1, k_2, \boldsymbol{x}, \boldsymbol{y}_1$ and $\boldsymbol{y}_2$ be as in Definition 5.1. Then,*

$$\boldsymbol{GRS}_{k_1}(\boldsymbol{x}, \boldsymbol{y}_1) \star \boldsymbol{GRS}_{k_2}(\boldsymbol{x}, \boldsymbol{y}_2) = \boldsymbol{GRS}_{k_1+k_2-1}(\boldsymbol{x}, \boldsymbol{y}_1 \star \boldsymbol{y}_2).$$

In particular,

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

Proof. Let $\boldsymbol{c}_1, \boldsymbol{c}_1$ be codewords of $\mathbf{GRS}_{k_1}(\boldsymbol{x}, \boldsymbol{y}_1)$ and $\mathbf{GRS}_{k_2}(\boldsymbol{x}, \boldsymbol{y}_2)$ respectively. Then there exists polynomials f_1 and f_2, such that $\deg f_i < k_i$ and $\boldsymbol{c}_i = (y_{i,1} f_i(x_1), \ldots, y_{i,n} f_i(x_n))$ for $i = 1, 2$.

Hence, let $g \stackrel{\text{def}}{=} f_1 f_2$. The polynomial g is of degree $< k_1 + k_2$ and we have $\boldsymbol{c}_1 \star \boldsymbol{c}_2 = (y'_1 g(x_1), \ldots, y'_n g(x_n))$, where $\boldsymbol{y}' \stackrel{\text{def}}{=} \boldsymbol{y}_1 \star \boldsymbol{y}_2$.

Hence, $\mathbf{GRS}_{k_1}(\boldsymbol{x}, \boldsymbol{y}_1) \star \mathbf{GRS}_{k_2}(\boldsymbol{x}, \boldsymbol{y}_2) \subseteq \mathbf{GRS}_{k_1+k_2-1}(\boldsymbol{x}, \boldsymbol{y}_1 \star \boldsymbol{y}_2)$.

Conversely, the code $\mathbf{GRS}_{k_1+k_2-1}(\boldsymbol{x}, \boldsymbol{y}_1 \star \boldsymbol{y}_2)$ is spanned by the words

$$\left(y_{1,1} y_{2,1} x_1^i, \ldots, y_{1,n} y_{2,n} x_n^i\right)_{0 \leqslant i < k_1+k_2-1},$$

each of which can be expressed as the star product of a word of $\mathbf{GRS}_{k_1}(\boldsymbol{x}, \boldsymbol{y}_1)$ and $\mathbf{GRS}_{k_2}(\boldsymbol{x}, \boldsymbol{y}_2)$. □

Corollary 5.18.

$$\dim\left(\boldsymbol{GRS}_k(\boldsymbol{x},\boldsymbol{y})\right)^{\star 2} = \min(n, 2k-1).$$

5.3.2.3 The distinguisher

This behaviour of GRS codes is very different from what happens generically. The square of a GRS code has a dimension which is linear in the dimension of the original code, whereas the dimension of the square of a random code grows quadraticaly in that of the code. This provides an efficient way to distinguish GRS codes from random codes.

Proposition 5.19. *For all $k \geqslant 2$, Problem 5.8 can be solved in polynomial time.*

Proof. Let $\boldsymbol{M} \in \mathbf{F}_q^{k\times n}$ be a matrix. If $k \leqslant n/2$, compute the matrix corresponding to the square of the associated code. If the dimension of this matrix is strictly less than $\min(n, \binom{k+1}{2})$, then $\boldsymbol{M}$ defines a GRS code and was generated according to distribution $\mathcal{D}_1$. Else, $\boldsymbol{M}$ is a random code and comes from distribution $\mathcal{D}_2$.

This distinguisher also works in the case $k > n/2$. Indeed, the dual of a GRS code is also a GRS code (see Lemma 5.4) with length n and dimension $(n-k) < n/2$. On the other hand, picking a random $[n,k]$-code and computing its dual yields the same distribution as directly considering a random $[n,n-k]$-code. Hence, one can apply the same criterion to the dual of the code. □

In fact, this operation can also be used to distinguish between random codes and other algebraically structured codes: it has been used for Reed–Muller codes [CB14], polar codes [BCDOT16], high-rate Goppa codes [FGOPT13; COT17] and algebraic geometry codes [CMP17].

5.3.3 Distinguishing shortened codes

The notion of *puncturing* and the dual notion of *shortening* are classical ways to build new codes from existing ones. It happens that the square-code distinguisher generalises nicely to punctured/shortened codes. In the next two chapters, we will see how we can use these operations to adapt the square-code distinguisher to codes where is cannot be applied directly. This will be useful for the next chapters.

Let us first recall the definitions of such operators.

5.3.3.1 Definitions

Here, for a codeword $\boldsymbol{c} \in \mathbf{F}_q^n$, we denote by $(c_1, \ldots, c_n)$ its entries.

Definition 5.20 (Punctured code)**.** Let $\mathscr{C} \subseteq \mathbf{F}_q^n$ and $\mathcal{L} \subseteq [\![0, n-1]\!]$. The *puncturing of $\mathscr{C}$ at $\mathcal{L}$* is defined as the code

$$\mathbf{Punct}_{\mathcal{L}}(\mathscr{C}) \stackrel{\text{def}}{=} \{(c_i)_{i \in [\![0,n-1]\!] \setminus \mathcal{L}} \text{ s.t. } \boldsymbol{c} \in \mathscr{C}\}.$$

Similarly, given a matrix $\boldsymbol{M}$ with n columns, one defines $\mathbf{Punct}_{\mathcal{L}}(\boldsymbol{M})$ as the matrix whose columns with index in $\mathcal{L}$ are removed, so that puncturing a generator matrix of a code yields a generator matrix of the punctured code.

Definition 5.21 (Shortened code)**.** Let $\mathscr{C} \subseteq \mathbf{F}_q^n$ and $\mathcal{L} \subseteq [\![0, n-1]\!]$. The *shortening of $\mathscr{C}$ at $\mathcal{L}$* is defined as the code

$$\mathbf{Short}_{\mathcal{L}}(\mathscr{C}) \stackrel{\text{def}}{=} \mathbf{Punct}_{\mathcal{L}}\left(\{\boldsymbol{c} \in \mathscr{C} \text{ s.t. } \forall i \in \mathcal{L},\ c_i = 0\}\right).$$

Shortening a code is equivalent to puncturing the dual code, as explained by the following proposition.

Proposition 5.22 ([HP03, Theorem 1.5.7])**.** *Let $\mathscr{C}$ be a linear code over $\mathbf{F}_q^n$ and $\mathcal{L} \subseteq [\![0, n-1]\!]$. Then,*

$$\mathbf{Short}_{\mathcal{L}}(\mathbf{Dual}(\mathscr{C})) = \mathbf{Dual}(\mathbf{Punct}_{\mathcal{L}}(\mathscr{C}))$$

and

$$\mathbf{Dual}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})) = \mathbf{Punct}_{\mathcal{L}}(\mathbf{Dual}(\mathscr{C})),$$

where $\mathbf{Dual}(\mathscr{A})$ denotes the dual of the code $\mathscr{A}$.

5.3.3.2 Shortening random codes

Puncturing and shortening random codes gives a random code of lesser length and dimension. Hence, thanks to Proposition 5.12, we have

Corollary 5.23. *Let $\mathscr{C}$ denote a code of length n and dimension k. Let $\mathcal{L} \subseteq [\![1, n]\!]$. Then*

$$\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} \leqslant \min\left(n - |\mathcal{L}|, \binom{\dim \mathbf{Short}_{\mathcal{L}}(\mathscr{C}) + 1}{2}\right).$$

Moreover, if $\mathscr{C}$ is drawn uniformly at random among $[n, k]$-codes, then, with probability close to 1 when k tends to infinity, we have

$$\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} = \min\left(n - |\mathcal{L}|, \binom{k - |\mathcal{L}| + 1}{2}\right).$$

As for puncturing of random codes, with probability close to 1 when k tends to infinity, we have

$$\dim\left(\mathbf{Punct}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} = \min\left(n - |\mathcal{L}|, \binom{k+1}{2}\right).$$

5.3.3.3 Shortening GRS codes

Proposition 5.24. *For a subset $\mathcal{L} \subseteq [\![1, n]\!]$ such that $|\mathcal{L}| \leqslant k$:*

$$\mathbf{Short}_{\mathcal{L}}\left(\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})\right) = \boldsymbol{GRS}_{k-|\mathcal{L}|}(\boldsymbol{x}, \boldsymbol{y}'),$$

where $\boldsymbol{x}' = (x_i)_{i \notin \mathcal{L}}$, $\boldsymbol{y}' = (y_i)_{i \notin \mathcal{L}}$.
Moreover, shortening $|\mathcal{L}| \geqslant k$ columns yields the trivial code of length $n - |\mathcal{L}|$.

Proposition 5.25. *For a subset $\mathcal{L} \subseteq [\![1, n]\!]$ such that $|\mathcal{L}| \leqslant n - k$:*

$$\mathbf{Punct}_{\mathcal{L}}\left(\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})\right) = \boldsymbol{GRS}_k(\boldsymbol{x}', \boldsymbol{y}'),$$

where $\boldsymbol{x}' = (x_i)_{i \notin \mathcal{L}}$, $\boldsymbol{y}' = (y_i)_{i \notin \mathcal{L}}$.
Moreover, puncturing $|\mathcal{L}| \geqslant n - k$ columns yields the full code $\mathbf{F}_q^{n-|\mathcal{L}|}$.

Hence, using Proposition 5.24 and Corollary 5.18, we obtain the following result.

Corollary 5.26. *For a subset $\mathcal{L} \subseteq [\![1, n]\!]$,*

$$\dim\left(\mathbf{Short}_{\mathcal{L}}\left(\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})\right)\right)^{\star 2} = \max(\min(n - |\mathcal{L}|, 2(k - |\mathcal{L}|) - 1), 0),$$

$$\dim\left(\mathbf{Punct}_{\mathcal{L}}\left(\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})\right)\right)^{\star 2} = \min(n - |\mathcal{L}|, 2k - 1).$$

As we can see, the structure of random (*resp.* GRS) codes stays mainly unaffected by the puncturing/shortening operation, and therefore the square-code distinguisher can be applied efficiently on these shorter codes.

5.4 Conclusion

For a given code $\mathscr{C}$, one can compute the dimension of $\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}\right)\right)^{\star 2}$ and that of $\left(\mathbf{Short}_{\mathcal{L}}\left(\mathbf{Dual}(\mathscr{C})\right)\right)^{\star 2}$ for different values of $\mathcal{L}$ and compare them to the expected dimensions if $\mathscr{C}$ were a random code. Any unusual behaviour (for a non-negligible number of samples) means that $\mathscr{C}$ can be distinguished from a random code in polynomial time, and should therefore be avoided in a McEliece-like cryptographic scheme. In [Cou19], Couvreur proposes that any new code-based cryptosystem proposal should be tested for indistinguishability with regards to this operation. Although this only provides a distinguisher, the next two chapters provide concrete examples where such a distinguisher can be turned into a key-recovery attack.

Since the original attack on the GRS encryption scheme [SS92], numerous GRS-based cryptosystem have been propose to keep the appealing properties of GRS codes while defeating these attacks. Sidelnikov and Shestakov's attack

[SS92]relies on the exactness of the equations, and is therefore quite easy to circumvent, by adding some randomness. But the square-code approach [Wie10; CGGOT14] is much more flexible (or robust, from a cryptanalytic point of view). The next two chapters illustrate how this approach can be adapted to new cryptosystems, even when they defeat the direct application of the distinguisher.

Chapter 6

Attack on the RLCE cryptosystem

The *Random Linear Code Encryption* (RLCE) scheme is a code-based cryptosystem introduced by Y. Wang in [Wan16] and submitted for the NIST's call for post-quantum cryptosystems under the name *RLCE-KEM* [Wan17]. This scheme works similarly to McEliece's cryptosystem but Goppa codes are replaced by another family of codes, constructed from GRS codes. As we have seen, using GRS codes as the secret key is tempting, because these codes offer good decoding properties and permit short secret keys, however the raw use of GRS codes is insecure. In [Wie06b], Wieschebrink proposed to add some random columns at random positions of the public-key matrix. This proposal was broken in [CGGOT14], where the authors manage to distinguish the columns of the GRS code and the random columns.

Wang's RLCE scheme can be considered as a way to push Wieschebrink's idea further by mixing each random columns with a column from the GRS code. With this operation, the randomness spreads and each column considered individually seem to share the same characteristics. Therefore, it is not subject to the attack described in [CGGOT14].

However, this is not enough. As we will see, the fact that two columns share the same randomness can actually be used to *derandomize* one of them. Based on this property, we will adapt the square-code distinguisher to distinguish RLCE codes from random codes. We then use this distinguisher to mount a key-recovery attack. Contrary to the GRS case, the distinguisher only works for some parameter ranges. Hence, we will have to reduce the parameters of the code, using puncturing and shortening operations, so that the distinguisher can be applied.

In this chapter, after presenting the RLCE scheme in details, we will explain how to use the square code distinguisher to recover such pairs of columns, and how to use this tool to derive the polynomial-time key-recovery attack on the RLCE scheme. We will also characterise the parameters which resist this attack.

Related publication: Couvreur, Lequesne and Tillich, *Recovering short keys of*

RLCE in polynomial time, PQCrypto 2019 [CLT19].

Contents

6.1 The RLCE scheme

6.1.1 Presentation of the scheme

Key generation . Let q denote a prime power and chose integers n, k and w such that $0 < k, w \leqslant n \leqslant q - 1$. The values of q, n, k and w are public.

1. Pick $\boldsymbol{x}$ and $\boldsymbol{y}$ a support and a multiplier of a GRS code of length n over $\mathbf{F}_q$ (as defined in § 5.1.1), chosen uniformly at random.

2. Denote $\boldsymbol{V}_{k,n}(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{F}_q^{k \times n}$ the Vandermonde matrix generating the generalised Reed–Solomon code $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$. Let $\boldsymbol{S}$ denote a random $k \times k$ invertible matrix. Define $\boldsymbol{G}_0 \stackrel{\text{def}}{=} \boldsymbol{S}\boldsymbol{V}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$. This matrix is a random generator matrix of the code $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$. Denote by $g_1, \ldots, g_n$ the columns of $\boldsymbol{G}_0$.

3. Let $r_1, \ldots, r_w$ be column vectors of length k with entries chosen independently and uniformly at random in $\mathbf{F}_q$. Denote by $\boldsymbol{G}_1$ the matrix obtained by inserting the random columns between GRS columns at the end of $\boldsymbol{G}_0$ as follows:

$$\boldsymbol{G}_1 \stackrel{\text{def}}{=} [g_1, \ldots, g_{n-w}, g_{n-w+1}, r_1, \ldots, g_n, r_w] \in \mathbf{F}_q^{k\times(n+w)}.$$

4. Let $\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_w$ be 2×2 matrices chosen uniformly at random in $\mathbf{GL}_2(\mathbf{F}_q)$. Let $\boldsymbol{Q}$ be the block–diagonal non singular matrix

$$\boldsymbol{Q} \stackrel{\text{def}}{=} \begin{pmatrix} \boldsymbol{I}_{n-w} & & & (0) \\ & \boldsymbol{Q}_1 & & \\ & & \ddots & \\ (0) & & & \boldsymbol{Q}_w \end{pmatrix} \in \mathbf{F}_q^{(n+w)\times(n+w)}.$$

   We denote $\boldsymbol{G}_2 \stackrel{\text{def}}{=} \boldsymbol{G}_1\boldsymbol{Q}$.

5. Let $\pi \in \mathfrak{S}_{n+w}$ be a randomly chosen permutation of $[\![1, n+w]\!]$ and $\boldsymbol{P}$ the corresponding $(n+w)\times(n+w)$ permutation matrix. Denote $\boldsymbol{G}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{G}_2\boldsymbol{P} \in \mathbf{F}_q^{k\times(n+w)}$.

6. The public key is $(\boldsymbol{G}_{\text{pub}}, t)$ where $t \stackrel{\text{def}}{=} \lfloor \frac{n-k}{2} \rfloor$ denotes the error correction capacity of the code $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$. The private key is $(\boldsymbol{x}, \boldsymbol{y}, (\boldsymbol{Q}_i)_{0<i\leqslant w}, \pi)$.

Encryption. For a message $\boldsymbol{m} \in \mathbf{F}_q^k$, the cipher text is $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$ where $\boldsymbol{e} \in \mathbf{F}_q^{n+w}$ is a random error vector of weight t.

Decryption. Let $\boldsymbol{c} = \boldsymbol{m}\boldsymbol{G}_{\text{pub}} + \boldsymbol{e}$ denote an encrypted message. Let $\boldsymbol{c}' \stackrel{\text{def}}{=} \boldsymbol{P}^{-1}\boldsymbol{Q}^{-1}\boldsymbol{c}$. We have $\boldsymbol{c}' = \boldsymbol{m}\boldsymbol{G}_1 + \boldsymbol{e}'$. Then, we remove positions corresponding to random columns. Define $\boldsymbol{c}'' = \mathbf{Punct}_{\mathcal{L}}(\boldsymbol{c}')$ where $\mathcal{L} = \{n - w + 2s \,|\, s \in [\![1, w]\!]\}$. We have $\boldsymbol{c}'' = \boldsymbol{m}\boldsymbol{G}_0 + \boldsymbol{e}''$ with $\mathbf{w}_H(\boldsymbol{e}'') \leqslant t$. Hence, $\boldsymbol{c}''$ can be decoded in $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$ to find $\boldsymbol{m}$.

Remark 6.1. *This presentation of the scheme is not exactly the same as in the original specifications of RLCE [Wan17]. It is however equivalent. Indeed, the scheme described in [Wan17] includes an additional permutation of the columns of the matrix $\boldsymbol{G}_0$. As already mentioned in Remark 5.6, this step is useless and does not change the probability distribution of the public keys.*

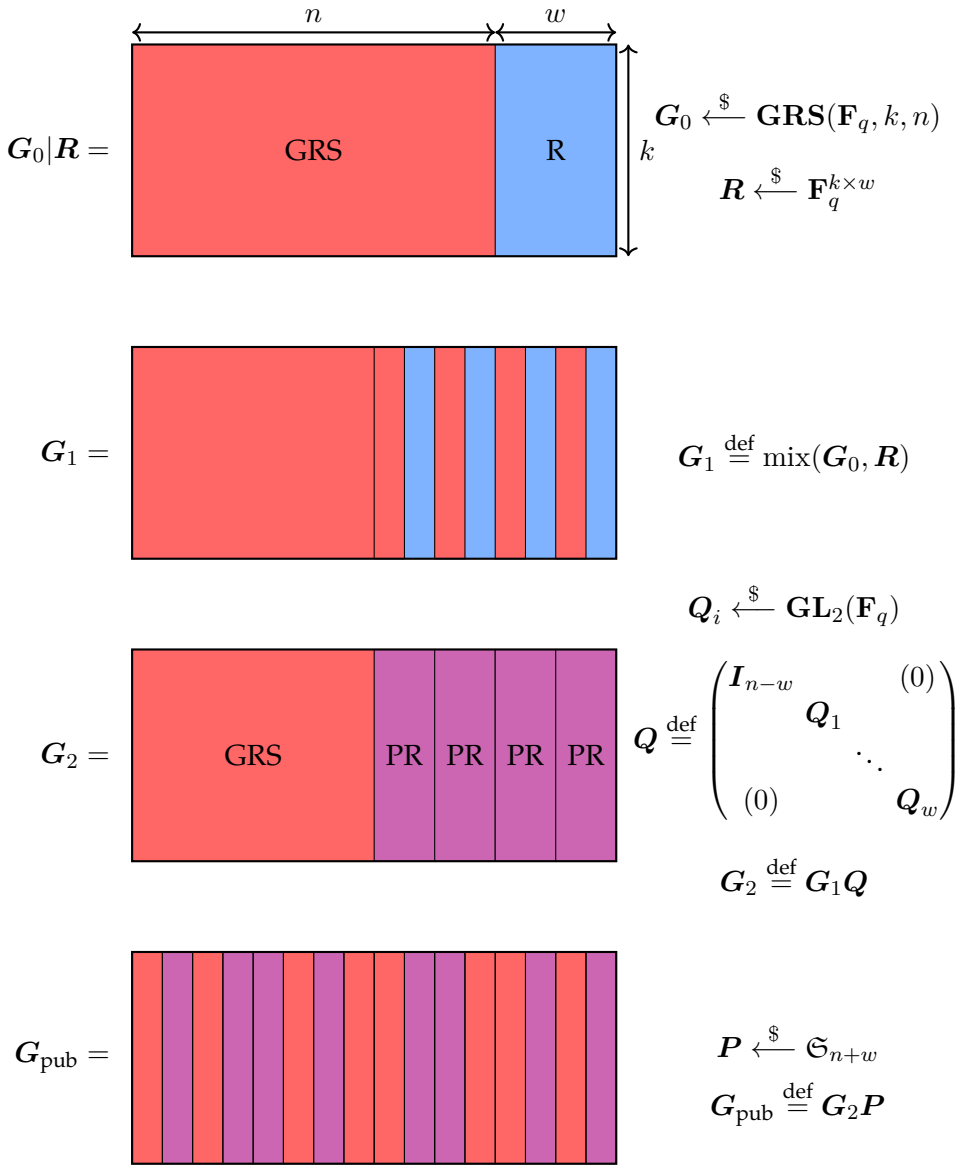


Figure 6.1: The RLCE scheme

6.1.2 Suggested sets of parameters

In [Wan17] the author proposes 2 groups of 3 sets of parameters. The first group (referred to as *odd ID* parameters) corresponds to parameters such that $w \in [0.6(n-k), 0.7(n-k)]$, whereas in the second group (*even ID* parameters) the parameters satisfy $w = n - k$. The parameters of these two groups are listed in Tables 6.1 and 6.2.

The matrix $\boldsymbol{G}_{\text{pub}}$ is a $k \times (n+w)$ matrix over $\mathbf{F}_q$. To transmit the public key, one can perform a Gaussian elimination to write this matrix in systematic form and discard the identity part. Therefore, the size of the public key is $k(n+w-k)\log_2(q)$ bits.

Our attack will recover in polynomial time any secret key when parameters lie in the first group.

Table 6.1: Set of parameters for the first group : $w \in [0.6(n-k), 0.7(n-k)]$.

Security bits	ID [Wan17]	n	k	t	w	q	Public key size (kB)
128	ID 1	532	376	78	96	2^{10}	118
192	ID 3	846	618	114	144	2^{10}	287
256	ID 5	1160	700	230	311	2^{11}	742

Table 6.2: Set of parameters for the second group : $w = n - k$.

Security bits	ID [Wan17]	n	k	t	w	q	Public key size (kB)
128	ID 0	630	470	80	160	2^{10}	188
192	ID 2	1000	764	118	236	2^{10}	450
256	ID 4	1360	800	280	560	2^{11}	1232

6.1.3 Natural questions

In Section 5.3 we have characterised the dimension of the square of a GRS code and the dimensions of the square of a random code (in the generic case). As we have seen, these dimensions are not equal, and this difference can be used to mount some attacks.

The public code of the RLCE cryptosystem (*i.e.* the code generated by the matrix $\boldsymbol{G}_{\text{pub}}$) lies by construction somewhere between GRS codes and random codes. Indeed, it is built from a GRS codes, with random columns added. The attack [CGGOT14] on Wieschebrink's scheme [Wie06b] proves that adding

random columns is not enough to hide the GRS structure. In a way, we can say that the randomness is too *localized*. The simple operation of puncturing one of the random columns cancels the effect of the additional randomness and this is easily noticeable. Hence, the idea behind the design of the RLCE scheme is to *spread* the randomness by mixing each random column with a GRS column: each random column in paired with a columns from the original GRS code and they are replaced by linear combinations of the columns. This is the role of the matrix $\boldsymbol{Q}_i$.

Some questions come naturally from a cryptanalytic point of view.

1. Is there a polynomial algorithm to distinguish public keys of the RLCE scheme from random $k \times (n+w)$ matrices over $\mathbf{F}_q$?
2. What is the dimension of the square of the code generated by $\boldsymbol{G}_{\text{pub}}$? More exactly, we have seen that the square-code distinguisher applies easily to shortened codes. Therefore we would like to characterise the exact dimension of any shortening of the code generated by $\boldsymbol{G}_{\text{pub}}$. Note that if the dimension of the square code is different from that of a random code, this provides an answer to question 1.
3. If such a distinguisher exists, is there a way to use it to reconstruct the private key (or an equivalent private key) starting from the public key?

The next section is dedicated to answering question 2. The consequences of this result regarding questions 1 and 3 will be discussed in the third section.

6.2 Dimension of the square code

Question 2 finds its answer in the following Theorem. This section is dedicated to proving this result.

Theorem 6.2. *Let $\mathscr{C}$ be a code over $\mathbf{F}_q$ of length $n+w$ and dimension k with generator matrix $\boldsymbol{G}_{pub}$ which is the public key of an RLCE scheme that is based on a GRS code of length n and dimension k. Let $\mathcal{L} \subset [\![1, n+w]\!]$. Then,*

$$\dim\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}\right)\right)^{\star 2} \leqslant \min(n+w-|\mathcal{L}|,\ 2(k+w-|\mathcal{L}|)-1).$$

Remark 6.3. *We will see in § 6.2.5 that under some conditions on the parameters n, k, w and $|\mathcal{L}|$ which we can characterise, the inequality established in Theorem 6.2 seems to be an equality with probability close to 1. This observation is based on computer experiments. See Remark 6.27 for further details.*

Remark 6.4. *It is interesting to note that the dimension of $\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}\right)\right)^{\star 2}$ only depends of the cardinality of $\mathcal{L}$ and does not depend of the nature of the columns that*

are shortened. Indeed, in the RLCE schemes, some columns are inherited directly from the GRS code and other are the result of a mixing with random columns. A full characterisation will be given in § 6.2.1. One could have expected that shortening different kinds of column would lead to different dimensions.

For the sake of simplicity, we will make the following assumption in this section. This will especially simplify the notations to prove Theorem 6.2.

Assumption 6.5. *The permutation matrix $\boldsymbol{P}$ is the identity matrix.*

This assumption does not change the general result thanks to the following lemma.

Lemma 6.6. *For any permutation σ of the code positions $[\![1, n+w]\!]$ we have*

$$\dim\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}\right)\right)^{\star 2} = \dim\left(\mathbf{Short}_{\mathcal{L}^\sigma}\left(\mathscr{C}^\sigma\right)\right)^{\star 2},$$

where $\mathscr{C}^\sigma$ is the set of codewords in $\mathscr{C}$ permuted by σ, that is $\mathscr{C}^\sigma = \{\boldsymbol{c}^\sigma : \boldsymbol{c} \in \mathscr{C}\}$ where $\boldsymbol{c}^\sigma \stackrel{def}{=} (c_{\sigma(i)})_{i\in[\![1,n+w]\!]}$ and $\mathcal{L}^\sigma \stackrel{def}{=} \{\sigma(i) : i \in \mathcal{L}\}$.

6.2.1 Analysis of the different kinds of columns

6.2.1.1 Notation and terminology

Before proving the result, let us introduce some notation and terminology. Indeed, the columns of $\boldsymbol{G}_{\text{pub}}$ are of different nature.

- Some columns are directly inherited from the matrix $\boldsymbol{G}_0$, which generates the GRS code $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$. We will call them *GRS columns*.
- Other columns come by pairs, corresponding to a matrix $\boldsymbol{Q}_i$. These pairs of columns share some properties. We will refer to them as *twin* columns.
- These twin columns are obtained by the linear combination of a GRS column and a random column. But they are not independent, hence we will call them *pseudo-random (PR) columns*.
- In some cases, some coefficients of the matrix $\boldsymbol{Q}_i$ may be equal to zero. This is especially problematic if one coefficient corresponding to the random part is zero. Indeed, as we will see, the fact that two twin columns share the common randomness is a key property that makes them not independent. We will refer to these cases as *degenerate cases*. In this special case, we will say that the column with no random part belongs to the category of GRS columns, and its twin column will be called a *random column*.

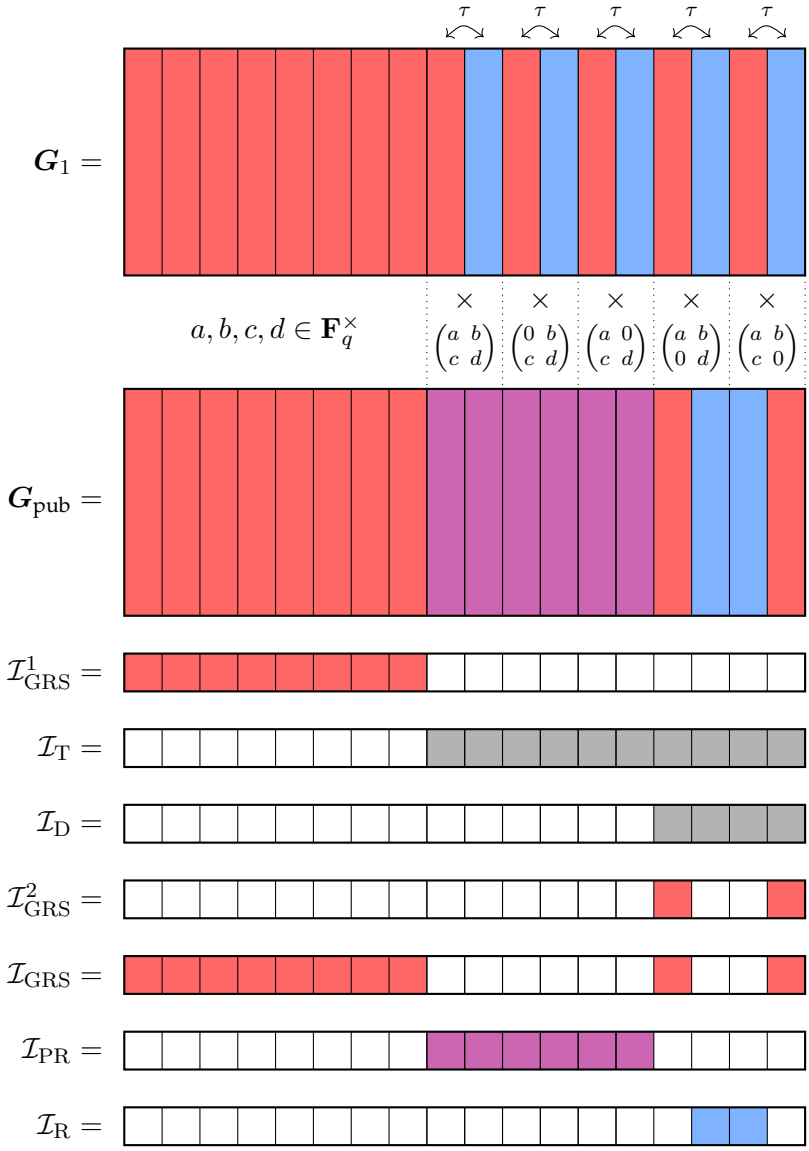


Figure 6.2: Different sets of positions for an example of RLCE scheme for $n = 13, w = 5$, where a, b, c and d denote non-zero elements of $\mathbf{F}_q$.

We will formalise this distinction in the rest of this subsection. Figure 6.2 illustrates the definitions.

6.2.1.2 Twin positions

Definition 6.7. The set of *twin positions*, denoted $\mathcal{I}_{\mathrm{T}}$, corresponds to columns that result in a mix of a random column and a GRS one. This set has cardinality $2w$ and is equal to:

$$\mathcal{I}_{\mathrm{T}} \stackrel{\text{def}}{=} \{i \in [\![1, n+w]\!] \,|\, \pi^{-1}(i) > n-w\}.$$

Under Assumption 6.5, this becomes: $\mathcal{I}_{\mathrm{T}} \stackrel{\text{def}}{=} [\![n-w+1, n+w]\!]$.

Definition 6.8. Each position in $\mathcal{I}_{\mathrm{T}}$ has a unique corresponding *twin* position which is the position of the column with which it was mixed. For all $s \in [\![1, w]\!]$, $\pi(n-w+2s-1)$ and $\pi(n-w+2s)$ are twin positions. Under Assumption 6.5, the positions $n-w+2s-1$ and $n-w+2s$ are twins for all s in $[\![1, w]\!]$.

For convenience, we introduce the following notation.

Notation 6.9. *The twin of a position $i \in \mathcal{I}_{\mathrm{T}}$ is denoted by $\tau(i)$.*

6.2.1.3 Random columns define linear forms

A convenient way to represent the random columns is to think of them as linear forms defined on the set of polynomials. Indeed, each codeword of the GRS code is the evaluation of a polynomial over the points of the support. The matrix $\boldsymbol{G}_0$ is a generator matrix of $\mathbf{GRS}_{k,n}(\boldsymbol{x}, \boldsymbol{y})$, hence for each row of the matrix there exists a polynomial $f \in \mathbf{F}_q[X]_{<k}$ such that the row corresponds to $(y_1 f(x_1), \ldots, y_n f(x_n))$. And because the rows of the matrix form a basis of the GRS code, the set of corresponding polynomials form a basis of $\mathbf{F}_q[X]_{<k}$. Let f_j denote the polynomial corresponding to the j-th row of $\boldsymbol{G}_0$. For each random column r_s added to $\boldsymbol{G}_0$, there exists a unique linear form $\psi_s : \mathbf{F}_q[x]_{<k} \to \mathbf{F}_q$, such that for all $j \in [\![1, k]\!]$, $\psi_s(f_j) = r_s[j]$ (see § 6.1.1, Step 3).

Hence, each random column added to the matrix $\boldsymbol{G}_0$ assigns a random coefficient to each element of the basis of $\mathbf{F}_q[X]_{<k}$, and by linear combination, this defines a linear form on $\mathbf{F}_q[X]_{<k}$.

Therefore, to any random column r_s is associated a unique linear form $\psi_s : \mathbf{F}_q[x]_{<k} \to \mathbf{F}_q$, such that the code generated by $\boldsymbol{G}_1$ is of the form

$$\{(x_1 f(y_1), \ldots, y_{n-w+1} f(x_{n-w+1}), \psi_1(f), \ldots, x_n f(x_n), \psi_w(f)) \,|\, f \in \mathbf{F}_q[X]_{<k}\}.$$

Notation 6.10. *For any $s \in [\![1, w]\!]$, we denote*

$$\begin{pmatrix} a_s & b_s \\ c_s & d_s \end{pmatrix} \stackrel{\text{def}}{=} \boldsymbol{Q}_s. \tag{6.1}$$

Proposition 6.11. *To any twin pair* $\{i, \tau(i)\} = \{\pi(n-w+2s-1), \pi(n-w+2s)\}$ *with* $s \in [\![1, w]\!]$*, for any codeword* $\boldsymbol{v} \in \mathscr{C}$*, we have*

$$\begin{aligned} v_i &= a_s y_j f(x_j) + c_s \psi_s(f) \\ v_{\tau(i)} &= b_s y_j f(x_j) + d_s \psi_s(f), \end{aligned} \tag{6.2}$$

where $j = n - w + s$*.*

Definition 6.12. The set of *degenerate pairs of positions*, is the set of positions where one of the columns has no random component, that is such that c_s or d_s is equal to zero. We will see in Lemma 6.19 why this defines a special case, that will be addressed in § 6.3.5.

$$\mathcal{I}_{\mathrm{D}} \stackrel{\text{def}}{=} \bigcup_{s \in [\![1,w]\!] \text{ s.t. } c_s d_s = 0} \{\pi(n-w+2s-1), \pi(n-w+2s)\}. \tag{6.3}$$

6.2.1.4 GRS positions

Definition 6.13. The set of *GRS positions of the first kind*, denoted $\mathcal{I}^1_{\mathrm{GRS}}$, corresponds to GRS columns which have not been associated to a random column. This set has cardinality $n - w$ and is given by

$$\mathcal{I}^1_{\mathrm{GRS}} \stackrel{\text{def}}{=} \{i \in [\![1, n+w]\!] \,|\, \pi^{-1}(i) \leqslant n - w\}. \tag{6.4}$$

Under Assumption 6.5, this becomes: $\mathcal{I}^1_{\mathrm{GRS}} \stackrel{\text{def}}{=} [\![1, n-w]\!]$.

This set is called this way, because at a position $i \in \mathcal{I}^1_{\mathrm{GRS}}$, any codeword $\boldsymbol{v} \in \mathscr{C}$ has an entry of the form

$$v_i = y_i f(x_i). \tag{6.5}$$

From (6.2), we see that we may obtain more GRS positions: indeed $v_i = a_s y_j f(x_j)$ if $c_s = 0$ or $v_{\tau(i)} = b_s y_j f(x_j)$ if $d_s = 0$. We will call these *GRS positions of the second kind*.

Definition 6.14. The set *GRS positions of the second kind*, denoted $\mathcal{I}^2_{\mathrm{GRS}}$, is defined as

$$\mathcal{I}^2_{\mathrm{GRS}} \stackrel{\text{def}}{=} \{\pi(n-w+2s-1) \,|\, c_s = 0\} \cup \{\pi(n-w+2s) \,|\, d_s = 0\}. \tag{6.6}$$

Under Assumption 6.5, this becomes:

$$\mathcal{I}^2_{\mathrm{GRS}} = \{n-w+2s-1 \,|\, c_s = 0\} \cup \{n-w+2s \,|\, d_s = 0\}. \tag{6.7}$$

We can join these two sets in one set of GRS positions.

Definition 6.15. The set of *GRS positions*, denoted $\mathcal{I}_{\mathrm{GRS}}$, is defined as

$$\mathcal{I}_{\mathrm{GRS}} \stackrel{\text{def}}{=} \mathcal{I}^1_{\mathrm{GRS}} \cup \mathcal{I}^2_{\mathrm{GRS}}. \tag{6.8}$$

6.2.1.5 Pseudo-random positions

For twin columns such that $c_s d_s \neq 0$, the twin pairs are *correlated* in the sense that both columns carry the same randomness. As we will see in Lemma 6.19, if one shortens the code in such a position its twin becomes a GRS position. This property will be useful to distinguish them. We therefore call such positions *pseudo-random* positions.

Definition 6.16. The set of *pseudo-random positions* (PR in short), denoted $\mathcal{I}_{\mathrm{PR}}$, is given by

$$\mathcal{I}_{\mathrm{PR}} \stackrel{\text{def}}{=} \bigcup_{s\in[\![1,w]\!] \text{ s.t. } c_s d_s \neq 0} \{\pi(n-w+2s-1), \pi(n-w+2s)\}. \tag{6.9}$$

Under Assumption 6.5, this becomes:

$$\mathcal{I}_{\mathrm{PR}} = \bigcup_{s\in[\![1,w]\!] \text{ s.t. } c_s d_s \neq 0} \{n-w+2s-1, n-w+2s\}. \tag{6.10}$$

6.2.1.6 Random positions

The random positions are the twin columns of the GRS positions of the second kind. Indeed, even if they are the sum of a GRS column and a random column from the matrix $\boldsymbol{G}_1$, the randomness part is not shared with its twin column. Hence it will not be possible to recover the GRS part (to *derandomise* using Lemma 6.19). Therefore, these columns are completely random. We call them *random positions*.

Definition 6.17. The set of *random positions*, denoted $\mathcal{I}_{\mathrm{R}}$, is defined as

$$\mathcal{I}_{\mathrm{R}} \stackrel{\text{def}}{=} \{\pi(n-w+2s-1) \,|\, d_s = 0\} \cup \{\pi(n-w+2s) \,|\, c_s = 0\}. \tag{6.11}$$

Under Assumption 6.5, this becomes:

$$\mathcal{I}_{\mathrm{R}} = \{n-w+2s-1 \,|\, d_s = 0\} \cup \{n-w+2s \,|\, c_s = 0\}. \tag{6.12}$$

Cardinality. We finish this subsection with a lemma.

Lemma 6.18. *$|\mathcal{I}^2_{\mathrm{GRS}}| = |\mathcal{I}_{\mathrm{R}}|$ and $|\mathcal{I}_{\mathrm{PR}}| = 2(w - |\mathcal{I}_{\mathrm{R}}|)$.*

Proof. Using (6.10), (6.7) and (6.12) we see that, under Assumption 6.5,

$$[\![n-w+1, n+w]\!] = \mathcal{I}_{\mathrm{PR}} \cup \mathcal{I}^2_{\mathrm{GRS}} \cup \mathcal{I}_{\mathrm{R}} \tag{6.13}$$

and the above union is disjoint. Next, there is a one-to-one correspondence relating $\mathcal{I}^2_{\mathrm{GRS}}$ and $\mathcal{I}_{\mathrm{R}}$. Indeed, still under Assumption 6.5, if $c_s = 0$ for some $s \in [\![1, w]\!]$, then $n-w+2s-1 \in \mathcal{I}^2_{\mathrm{GRS}}$ and $n-w+2s \in \mathcal{I}_{\mathrm{R}}$ and conversely if $d_s = 0$. This proves that $|\mathcal{I}^2_{\mathrm{GRS}}| = |\mathcal{I}_{\mathrm{R}}|$, which, together with (6.13) yields the result. □

6.2.2 Intermediate results

Before proceeding to the proof of Theorem 6.2, let us state and prove some intermediate results. We will start by Lemmas 6.19, the *derandomisation lemma*, that proves that twin pairs of pseudo-random positions behave in a very particular way: after shortening one position, its twin becomes a GRS position. Then we prove a short lemma on subcodes of GRS codes, Lemma 6.22. These two results will be useful to prove Proposition 6.23 on the structure of shortened RLCE codes, by induction on the number of shortened positions. This proposition be the core of the proof of Theorem 6.2. Finally, we will prove a general result on modified GRS codes with additional random columns.

6.2.2.1 Derandomisation

This lemma explains that, after shortening a PR position, its twin will behave like a GRS position. This is actually a crucial lemma that explains why PR columns in $\boldsymbol{G}$ do not really behave like random columns after shortening the code at the corresponding position.

Lemma 6.19. *Let i be a PR position and $\mathcal{L}$ a set of positions that neither contains i nor $\tau(i)$. Let $\mathscr{C}' \stackrel{def}{=} \mathbf{Short}_{\mathcal{L}}(\mathscr{C})$. The position $\tau(i)$* behaves like a GRS position *in the code $\mathbf{Short}_{\{i\}}(\mathscr{C}')$. That is, the $\tau(i)$–th column of a generator matrix of $\mathbf{Short}_{\{i\}}(\mathscr{C}')$ has entries of the form*

$$\tilde{y}_j f(x_j)$$

for some j in $[\![n-w+1, n]\!]$ and $\tilde{y}_j$ in $\mathbf{F}_q$.

Proof. Let us assume that $i = n - w + 2s - 1$ for some $s \in \{1, \ldots, w\}$. The case $i = n - w + 2s$ can be proved in a similar way. At position i, for any $\boldsymbol{c} \in \mathscr{C}'$, from (6.2), we have

$$c_i = a y_j f(x_j) + c\psi_s(f),$$

where $j = n - w + s$. By shortening, we restrict our space of polynomials to the subspace of polynomials in $\mathbf{F}_q[x]_{<k}$ satisfying $c_i = 0$. Since i is a PR position, $c \neq 0$ and therefore

$$\psi_s(f) = -c^{-1} a y_j f(x_j).$$

Therefore, at the twin position $\tau(i) = n - w + 2s$ and for any $\boldsymbol{c} \in \mathbf{Short}_{\{i\}}(\mathscr{C}')$, we have

$$\begin{aligned} c_{\tau(i)} &= b y_j f(x_j) + d\psi_j(f) \\ &= y_j (b - dac^{-1}) f(x_j). \end{aligned}$$

□

Remark 6.20. *This lemma does not hold for a random position, since the proof requires that $c \neq 0$. It is precisely because of this that we have to make a distinction between twin pairs, i.e. pairs for which the associated matrix $\boldsymbol{Q}_s$ is such that $c_s d_s \neq 0$ and pairs for which it is not the case. This explains the definition of degenerate positions (see Definition 6.12).*

This lemma allows us to get some insight on the structure of the shortened code $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$. Before giving the relevant statement let us first recall the following result.

6.2.2.2 A lemma on GRS subcodes

Definition 6.21. Let $\mathscr{C} \subseteq \mathbf{F}_q^n$ and $\mathcal{L} \subseteq [\![1, n]\!]$. The *restriction* of $\mathscr{C}$ to $\mathcal{L}$ is defined as a variant of puncturing, keeping only the positions in the subset $\mathcal{L}$.

$$\mathbf{Restr}_{\mathcal{L}}(\mathscr{C}) \stackrel{\text{def}}{=} \mathbf{Punct}_{[\![1,n]\!]\setminus\mathcal{L}}(\mathscr{C}).$$

Lemma 6.22. *Consider a linear code $\mathscr{A}$ over $\mathbf{F}_q$ s.t. $\mathbf{Restr}_{\mathcal{L}}(\mathscr{A})$ is a subcode of a k–dimensional GRS code. Let i be an element of $\mathcal{L}$. Then $\mathbf{Restr}_{\mathcal{L}\setminus\{i\}}\left(\mathbf{Short}_{\{i\}}(\mathscr{A})\right)$ is a subcode of a $(k-1)$–dimensional GRS code.*

Proof. By definition, the restriction of $\mathscr{A}$ to $\mathcal{L}$ is a GRS code so it can be written of the form

$$\mathbf{Restr}_{\mathcal{L}}(\mathscr{A}) = \left\{(y_j f(x_j))_{j\in\mathcal{L}} : f \in L\right\},$$

where the y_j's are nonzero elements of $\mathbf{F}_q$, the x_j's are distinct elements of $\mathbf{F}_q$ and L is a subspace of $\mathbf{F}_q[X]_{<k}$. Clearly the restriction of $\mathbf{Short}_{\{i\}}(\mathscr{A})$ to $\mathcal{L}\setminus\{i\}$ can be written as

$$\mathbf{Restr}_{\mathcal{L}\setminus\{i\}}\left(\mathbf{Short}_{\{i\}}(\mathscr{A})\right) = \left\{(y_j f(x_j))_{j\in\mathcal{L}\setminus\{i\}} : f \in L, f(x_i) = 0\right\}.$$

The polynomials $f(X)$ in L such that $f(x_i) = 0$ can be written as $f(X) = (X - x_i)g(X)$ where $\deg g = \deg f - 1$ and g ranges in this case over a subspace L' of polynomials of degree $< k-1$. We can therefore write

$$\mathbf{Restr}_{\mathcal{L}\setminus\{i\}}\left(\mathbf{Short}_{\{i\}}(\mathscr{A})\right) = \left\{(y_j(x_j - x_i)g(x_j))_{j\in\mathcal{L}\setminus\{i\}} : g \in L'\right\}.$$

This implies our lemma. □

6.2.2.3 Structure of a shortened RLCE code

In order to prove Theorem 6.2, we need to describe exactly what happens when we shorten the code generated by $\boldsymbol{G}_{\text{pub}}$. Especially, to give an upper bound on the dimension of the square code, we want to show that some part

of the shortened code is a subcode of a GRS code. Indeed, we know that the dimension of the square code of a GRS code (or a subcode of a GRS code) is much lower than for a random code. Therefore, in order to obtain a bound as tight as possible, we want to find the largest set of positions such that the shortened code restricted to these positions is a subcode of a GRS code.

Using Lemmas 6.19 and 6.22, we can prove the following result by induction. This result is the key proposition for proving Theorem 6.2.

Proposition 6.23. *Let $\mathcal{L}$ be a subset of $[\![1, n+w]\!]$ and let $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$ be the partition of $\mathcal{L}$ defined as*

- *$\mathcal{L}_0$ the set of GRS positions (see (6.4), (6.6) and (6.8) for a definition) of $\mathcal{L}$:*

$$\mathcal{L}_0 \stackrel{def}{=} \mathcal{L} \cap \mathcal{I}_{\mathrm{GRS}};$$

- *$\mathcal{L}_1$ the set of PR positions (see (6.9)) of $\mathcal{L}$ that do not have their twin in $\mathcal{L}$:*

$$\mathcal{L}_1 \stackrel{def}{=} \{i \in \mathcal{L} \cap \mathcal{I}_{\mathrm{PR}} \,|\, \tau(i) \notin \mathcal{L}\};$$

- *$\mathcal{L}_2$ the set of PR positions of $\mathcal{L}$ whose twin position is also included in $\mathcal{L}$:*

$$\mathcal{L}_2 \stackrel{def}{=} \{i \in \mathcal{L} \cap \mathcal{I}_{\mathrm{PR}} \,|\, \tau(i) \in \mathcal{L}\};$$

- *$\mathcal{L}_3$ the set of random positions of $\mathcal{L}$:*

$$\mathcal{L}_3 \stackrel{def}{=} \mathcal{I}_{\mathrm{R}} \cap \mathcal{L}.$$

Let $\mathscr{C}'$ be the restriction of $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ to $\mathcal{J} \stackrel{def}{=} (\mathcal{I}_{\mathrm{GRS}} \setminus \mathcal{L}_0) \cup \tau(\mathcal{L}_1)$. Then, $\mathscr{C}'$ is a subcode of a GRS code of length $|\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0| + |\mathcal{L}_1|$ and dimension $k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$.

Proof. Let us prove by induction on $\ell = |\mathcal{L}|$ that $\mathscr{C}'$ is a subcode of a GRS code of length $|\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0| + |\mathcal{L}_1|$ and dimension $k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$. Note that the result on the length is straightforward because $|\mathcal{J}| = |\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0| + |\mathcal{L}_1|$. Let us prove the result on the dimension.

This statement is clearly true if $\ell = 0$, *i.e.* if $\mathcal{L}$ is the empty set. Assume that the result is true for all $\mathcal{L}$ up to some size $\ell \geqslant 0$. Consider now a set $\mathcal{L}$ of size $\ell + 1$. We can write $\mathcal{L} = \mathcal{L}' \cup \{i\}$ where $\mathcal{L}'$ is of size ℓ.

Let $\mathcal{L}_0, \mathcal{L}_1, \mathcal{L}_2$ be subsets of $\mathcal{L}$ as defined in the statement and $\mathcal{L}'_0, \mathcal{L}'_1, \mathcal{L}'_2$ be the subsets of $\mathcal{L}'$ obtained by replacing in the statement $\mathcal{L}$ by $\mathcal{L}'$. There are now several cases to consider for i.

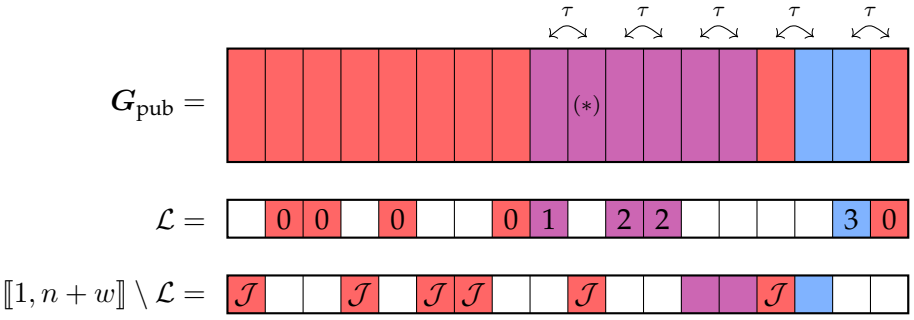


Figure 6.3: Illustration of the partition of $\mathcal{L}$ defined in Proposition 6.23, with the example from Figure 6.2. The positions in $\mathcal{L}$ indexed with a number $i \in \{0, 1, 2, 3\}$ belong to the set $\mathcal{L}_i$. The positions in $[\![1, n+w]\!] \setminus \mathcal{L}$ indexed with $\mathcal{J}$ belong to the set $\mathcal{J}$. Note that the column $(*)$ has been *derandomised* according to Lemma 6.19.

Case 1: $i \in \mathcal{L}_0$. In this case, $\mathcal{L}_0 = \mathcal{L}'_0 \cup \{i\}$, $\mathcal{L}_1 = \mathcal{L}'_1$ and $\mathcal{L}_2 = \mathcal{L}'_2$.

We can apply Lemma 6.22 with $\mathscr{A} = \mathbf{Short}_{\mathcal{L}'}(\mathscr{C})$. By the induction hypothesis, its restriction to $\mathcal{J}'' \stackrel{\text{def}}{=} (\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}'_0) \cup \tau(\mathcal{L}'_1)$ is a subcode of a GRS code of dimension $k - |\mathcal{L}'_0| - \frac{|\mathcal{L}'_2|}{2}$.

Hence, the restriction of the shortened code $\mathbf{Short}_{\mathcal{L}}(\mathscr{C}) = \mathbf{Short}_{\{i\}}(\mathscr{A})$ to $\mathcal{J}'' \setminus \{i\} = \mathcal{J}$ is a subcode of a GRS code of dimension $k - |\mathcal{L}'_0| - \frac{|\mathcal{L}'_2|}{2} - 1 = k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$.

Case 2: $i \in \mathcal{L}_1$. In this case, $\mathcal{L}_0 = \mathcal{L}'_0$, $\mathcal{L}_1 = \mathcal{L}'_1 \cup \{i\}$ and $\mathcal{L}_2 = \mathcal{L}'_2$. This implies that $\mathcal{L}'$ does not contain i nor $\tau(i)$.

We can therefore apply Lemma 6.19 with $\mathscr{C}' = \mathbf{Short}_{\mathcal{L}'}(\mathscr{C})$. Lemma 6.19 states that the position $\tau(i)$ behaves like a GRS position in $\mathbf{Short}_{\{i\}}(\mathscr{C}') = \mathbf{Short}_{\mathcal{L}}(\mathscr{C})$. The column $\tau(i)$ behaves like one more columns of the GRS code, whose dimension stays unchanged. By induction hypothesis, the restriction of the code $\mathscr{C}'$ to $(\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}'_0) \cup \tau(\mathcal{L}'_1)$ is a subcode of a GRS code of dimension $k - |\mathcal{L}'_0| - \frac{|\mathcal{L}'_2|}{2} = k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$.

Therefore the restriction of $\mathbf{Short}_{\{i\}}(\mathscr{C}') = \mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ to $(\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}_0) \cup \tau(\mathcal{L}_1) = (\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}'_0) \cup \tau(\mathcal{L}'_1) \cup \{\tau(i)\}$ is a subcode of a GRS code of dimension $k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$.

Case 3: $i \in \mathcal{L}_2$. In this case, $\mathcal{L}_0 = \mathcal{L}'_0$, $\mathcal{L}_1 = \mathcal{L}'_1 \setminus \{\tau(i)\}$ and $\mathcal{L}_2 = \mathcal{L}'_2 \cup \{i, \tau(i)\}$. In fact, this case can only happen if $\ell \geqslant 1$ and we will rather consider the induction with respect to the set $\mathcal{L}'' = \mathcal{L} \setminus \{i, \tau(i)\}$ of size $\ell - 1$ and the sets $\mathcal{L}''_0, \mathcal{L}''_1, \mathcal{L}''_2$ such that $\mathcal{L}''_0 = \mathcal{L}_0, \mathcal{L}''_1 = \mathcal{L}_1, \mathcal{L}''_2 = \mathcal{L}_2 \setminus \{i, \tau(i)\}$.

By induction hypothesis on $\mathcal{L}''$, the restriction of $\mathscr{C}'' \stackrel{\text{def}}{=} \mathbf{Short}_{\mathcal{L}''}(\mathscr{C})$ to $(\mathcal{I}_{\mathrm{GRS}} \setminus \mathcal{L}_0'') \cup \tau(\mathcal{L}_1'')$ is a subcode of a GRS code of dimension $k - |\mathcal{L}_0''| - \frac{|\mathcal{L}_2''|}{2} = k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2} + 1$.

Following Assumption 6.5, we can write without loss of generality that $i = n - w + 2s - 1$ for some $s \in \{1, \ldots, w\}$. The case $i = n - w + 2s$ can be proved in a similar way.

Denote $\boldsymbol{A}_s = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ the non-singular matrix and $j = n - w + s$. For any $\boldsymbol{c} \in \mathscr{C}'$, at positions i and $\tau(i)$ we have

$$\begin{aligned} c_i &= a y_j f(x_j) + c\psi_s(f), \\ c_{\tau(i)} &= b y_j f(x_j) + d\psi_s(f). \end{aligned}$$

Shortening $\mathscr{C}''$ at $\{i, \tau(i)\}$ has the effect of requiring to consider only the polynomials f for which $f(x_j) = \psi_s(f) = 0$. The dimension of the GRS decreases by one. Therefore the restriction of $\mathbf{Short}_{\{i,\tau(i)\}}(\mathscr{C}'') = \mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ at $(\mathcal{I}_{\mathrm{GRS}} \setminus \mathcal{L}_0'') \cup \tau(\mathcal{L}_1'')$ is a subcode of a GRS code of dimension $k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2} + 1 - 1 = k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$.

Case 4: $i \in \mathcal{L}_3$. In this case $\mathcal{L}_0 = \mathcal{L}_0', \mathcal{L}_1 = \mathcal{L}_1'$ and $\mathcal{L}_2 = \mathcal{L}_2'$. Using the induction hypothesis yields directly that $\mathscr{A} = \mathbf{Short}_{\mathcal{L}'}(\mathscr{C})$ is a subcode of a GRS code of length $|\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0'| + |\mathcal{L}_1'| = |\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0| + |\mathcal{L}_1|$ and dimension $k - |\mathcal{L}_0'| - \frac{|\mathcal{L}_2'|}{2} = k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$. This is also clearly the case for $\mathbf{Short}_{\mathcal{L}}(\mathscr{C}) = \mathbf{Short}_{\{i\}}(\mathscr{A})$.

This proves that the induction hypothesis also holds for $|\mathcal{L}| = \ell + 1$ and finishes the proof of the proposition. □

6.2.2.4 Adding columns to subcodes of GRS codes

We have seen in the previous section that a restriction of the public code (or the shortened public code) is a subcode of a GRS code. Hence, the public code is a concatenation of a subcode of a GRS code and some additional columns. We need a general result to bound the dimension of the square of codes of this form. Such a lemma is already proved in [CGGOT14, Lemma 9]. We repeat its proof below for convenience and in order to provide further details about the equality case, which is of particular interest to us.

Lemma 6.24. *Consider a linear code $\mathscr{A}$ over $\mathbf{F}_q$ with generator matrix of the form $\boldsymbol{G} = \begin{pmatrix} \boldsymbol{G}_{\mathrm{SCGRS}} & \boldsymbol{G}_{\mathrm{rand}} \end{pmatrix} \boldsymbol{P}$ of size $k \times (n + r)$ where $\boldsymbol{G}_{\mathrm{SCGRS}}$ is a $k \times n$ generator matrix of a subcode of a GRS code of dimension k_{GRS} over $\mathbf{F}_q$, $\boldsymbol{G}_{\mathrm{rand}}$ is an arbitrary*

matrix in $\mathbf{F}_q^{k\times r}$ *and* $\boldsymbol{P}$ *is the permutation matrix of an arbitrary permutation* $\sigma \in \mathfrak{S}_{n+r}$*. We have*

$$\dim \mathscr{A}^{\star 2} \leqslant 2k_{\mathrm{GRS}} - 1 + r. \tag{6.14}$$

Moreover, if the equality holds, then for every $i \in [\![1, n+w]\!]$ *we have:*

$$\text{if } i \in [\![n+1, n+r]\!], \quad \dim \mathbf{Punct}_{\{\sigma(i)\}}\left(\mathscr{A}^{\star 2}\right) = \dim \mathscr{A}^{\star 2} - 1,$$

$$\text{else if } i \in [\![1, n]\!] \text{ and } k_{\mathrm{GRS}} > 1, \quad \dim \mathbf{Punct}_{\{\sigma(i)\}}\left(\mathscr{A}^{\star 2}\right) = \dim \mathscr{A}^{\star 2}.$$

Remark 6.25. *According to Proposition 5.12, given a code of dimension* k*, the dimension of its square code is bounded by* $\binom{k+1}{2}$*. Hence, to achieve equality in Equation* (6.14)*, a necessary condition is to have*

$$2k_{\mathrm{GRS}} - 1 + r \leqslant \binom{k+1}{2}. \tag{6.15}$$

Proof. Without loss of generality, we may assume that $\boldsymbol{P}$ is the identity matrix since the dimension of the square code is invariant by permuting the code positions (see Lemma 6.6). Let $\mathscr{B}$ be the code with generator matrix $\left(\boldsymbol{G}_{\mathrm{SCGRS}}\ \mathbf{0}_{k\times r}\right)$, where $\mathbf{0}_{k\times r}$ is the zero matrix of size $k \times r$. We also define the code $\mathscr{B}'$ generated by the generator matrix $\left(\mathbf{0}_{k\times n}\ \boldsymbol{G}_{\mathrm{rand}}\right)$. We obviously have

$$\mathscr{A} \subseteq \mathscr{B} + \mathscr{B}'.$$

Therefore

$$\begin{aligned}(\mathscr{A})^{\star 2} &\subseteq (\mathscr{B} + \mathscr{B}')^{\star 2} \\ &\subseteq \mathscr{B}^{\star 2} + (\mathscr{B}')^{\star 2} + \mathscr{B} \star \mathscr{B}' \\ &\subseteq \mathscr{B}^{\star 2} + (\mathscr{B}')^{\star 2},\end{aligned}$$

where the last inclusion comes from the fact that $\mathscr{B} \star \mathscr{B}'$ is the zero subspace since $\mathscr{B}$ and $\mathscr{B}'$ have disjoint supports. The code $\mathscr{B}^{\star 2}$ has dimension $\leqslant 2k_{\mathrm{GRS}} - 1$ whereas $\dim (\mathscr{B}')^{\star 2} \leqslant r$.

Next, if $\dim \mathscr{A}^{\star 2} = 2k_{\mathrm{GRS}} - 1 + r$, then

$$\mathscr{A}^{\star 2} = \mathscr{B}^{\star 2} \oplus (\mathscr{B}')^{\star 2}, \quad \dim \mathscr{B}^{\star 2} = 2k_{\mathrm{GRS}} - 1 \quad \text{and} \quad \dim(\mathscr{B}')^{\star 2} = r.$$

A necessary condition to have $\dim \mathscr{B}^{\star 2} = 2k_{\mathrm{GRS}} - 1$ is that $n \geqslant 2k_{\mathrm{GRS}} - 1$, hence $n > k_{\mathrm{GRS}}$ provided $k_{\mathrm{GRS}} > 1$. Therefore, the code $\mathscr{B}^{\star 2}$ restricted to its n leftmost positions is a subcode of a GRS code of length n and dimension $2k_{\mathrm{GRS}} - 1$, and such a code admits no codeword of weight 1. Hence, the code

$\mathscr{A}^{\star 2}$ has no codeword of weight 1 with support on the n leftmost positions, so puncturing one of these positions does not decrease the dimension of $\mathscr{A}^{\star 2}$.

Concerning $\mathscr{B}'$, since the size of the support is equal to r, this means that $(\mathscr{B}')^{\star 2} = \{(\mathbf{0}_n, \boldsymbol{c}), \boldsymbol{c} \in \mathbf{F}_q^r\}$ and hence, any word of weight 1 supported by the r rightmost positions is contained in $\mathscr{A}^{\star 2}$. Therefore, puncturing this position decreases the dimension by one. □

6.2.3 Proof of the main theorem

We will now proceed to the proof of Theorem 6.2.

Proof. By using Proposition 6.23, we know that the restriction of $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ to $\mathcal{J} \stackrel{\text{def}}{=} (\mathcal{I}_{\mathrm{GRS}} \setminus \mathcal{L}_0) \cup \tau(\mathcal{L}_1)$ is a subcode of a GRS code of length $|\mathcal{I}_{\mathrm{GRS}}| - |\mathcal{L}_0| + |\mathcal{L}_1| = n - w + |\mathcal{I}_{\mathrm{GRS}}^2| - |\mathcal{L}_0| + |\mathcal{L}_1|$ and dimension $k_{\mathrm{GRS}} \stackrel{\text{def}}{=} k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}$, where:

- $\mathcal{L}_0 \stackrel{\text{def}}{=} \mathcal{I}_{\mathrm{GRS}} \cap \mathcal{L}$;
- $\mathcal{L}_1$ is the set of PR positions of $\mathcal{L}$ that do not have their twin in $\mathcal{L}$;
- $\mathcal{L}_2$ is the union of all twin PR positions that are both included in $\mathcal{L}$;
- $\mathcal{L}_3 \stackrel{\text{def}}{=} \mathcal{I}_{\mathrm{R}} \cap \mathcal{L}$.

We have

$$[\![1, n+w]\!] = \mathcal{L} \sqcup \mathcal{J} \sqcup (\mathcal{I}_{\mathrm{PR}} \setminus (\mathcal{L} \cup \tau(\mathcal{L}_1))) \sqcup (\mathcal{I}_{\mathrm{R}} \setminus \mathcal{L}_3),$$

where $\sqcup$ denotes the disjoint union. We can apply Lemma 6.24 to $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ and derive from it the following upper bound:

$$\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} \leqslant 2k_{\mathrm{GRS}} - 1 + |\mathcal{I}_{\mathrm{PR}} \setminus (\mathcal{L} \cup \tau(\mathcal{L}_1))| + |\mathcal{I}_{\mathrm{R}} \setminus \mathcal{L}_3| \quad (6.16)$$

Finally, we can simplify this expression using Lemma 6.18. We get

$$\begin{aligned}
&\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} \\
&\quad \leqslant 2\left(k - |\mathcal{L}_0| - \frac{|\mathcal{L}_2|}{2}\right) - 1 + 2\left(w - |\mathcal{I}_{\mathrm{R}}|\right) - 2|\mathcal{L}_1| - |\mathcal{L}_2| + |\mathcal{I}_{\mathrm{R}}| - |\mathcal{L}_3| \\
&\quad \leqslant 2\left(k + w - |\mathcal{L}_0| - |\mathcal{L}_1| - |\mathcal{L}_2| - |\mathcal{L}_3|\right) - 1 + \left(|\mathcal{L}_3| - |\mathcal{I}_{\mathrm{R}}|\right) \quad (6.17) \\
&\quad \leqslant 2\left(k + w - |\mathcal{L}|\right) - 1. \quad (6.18)
\end{aligned}$$

The other upper bound on $\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2}$ which is $\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} \leqslant n + w - |\mathcal{L}|$ follows from the fact that the dimension of this code is bounded by its length. Putting both bounds together yields the theorem. □

6.2.4 When is the inequality an equality?

Let us analyse the proof of Theorem 6.2, to see under which condition the upper bound can be met. There are actually two steps in the reasoning where the inequality is not necessarily sharp.

1. To go from Equation (6.17) to Equation (6.18), one supposes that $|\mathcal{L}_3| - |\mathcal{I}_{\mathrm{R}}| = 0$ which means that $\mathcal{I}_{\mathrm{R}} \subseteq \mathcal{L}$. Because we want our result to hold for any choice of $\mathcal{L}$ and depend only of the cardinality of the set $\mathcal{L}$, this implies that $\mathcal{I}_{\mathrm{R}} = \emptyset$. This corresponds exactly to the fact of having no degenerate pairs of positions.

2. The proof uses Lemma 6.24 to obtain Equation (6.16). As we have seen in Remark 6.25, a necessary condition to have an equality in Equation (6.16) is given by Equation (6.15). Assuming there are no degenerate pairs of positions, this condition becomes

$$2(k+w-|\mathcal{L}|)-1 \leqslant \binom{\dim \mathbf{Short}_{\mathcal{L}}(\mathscr{C})+1}{2},$$

   And because $\dim \mathbf{Short}_{\mathcal{L}}(\mathscr{C}) \leqslant k - |\mathcal{L}|$, we obtain

$$2(k+w-|\mathcal{L}|)-1 \leqslant \binom{k-|\mathcal{L}|+1}{2}. \tag{6.19}$$

We can now formulate the following conjecture.

Conjecture 6.26. *Under these two conditions, the inequality of Theorem 6.2 is an equality with high probability.*

$$\mathbb{P}\left[\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2} = \min(n+w-|\mathcal{L}|, 2(k+w-|\mathcal{L}|)-1)\right] \underset{n,k\to\infty}{\longrightarrow} 1$$

Remark 6.27. *To check if this conjecture holds for the parameters used in the cryptosystem, we ran the following simulations using ID 1 parameters (see Table 6.1): for three hundred random independent public keys, we computed* $\dim\left(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})\right)^{\star 2}$ *for* $|\mathcal{L}|$ *ranging over* $[\![\ell_{\min}, \ell_{\max}]\!]$*, as defined in* (6.25)*. For more than* 99% *of the cases, inequality* (6.16) *is an equality. In particular, this means that the inequality of Theorem 6.2 is almost always an equality whenever* $\mathcal{I}_{\mathrm{R}}$ *is the empty set, i.e. when there are no degenerate pairs. In § 6.3.5, we explain how to deal with the rather rare issue of degenerate positions by transforming them into the generic case.*

6.2.5 A distinguisher

We can now address the first question asked in Section 6.1.3: is there a polynomial algorithm to distinguish public keys of the RLCE scheme from random $k \times (n+w)$ matrices over $\mathbf{F}_q$?

Theorem 6.2 shows that the public keys of the RLCE cryptosystem have a behaviour which is different from random $k \times (n+w)$ matrices over $\mathbf{F}_q$ regarding the dimension of their square code.

Indeed, let $\mathscr{C}_{\text{RLCE}}$ be an $[n+w, k]$ code over $\mathbf{F}_q$ whose generator matrix $\boldsymbol{G}_{\text{pub}}$ is the public key of an RLCE scheme. Let $\mathcal{L}$ denote a subset of $[\![1, n+w]\!]$. We suppose that the code contains no degenerate positions and that it fulfills the condition (6.19). Then, according to Theorem 6.2 and Conjecture 6.26, with high probability,

$$\dim\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}_{\text{RLCE}}\right)\right)^{\star 2} = \min\left(n+w-|\mathcal{L}|,\ 2(k+w-|\mathcal{L}|)-1\right).$$

On the other hand, according to Corollary 5.23, if $\mathscr{C}_{\text{Rand}}$ is drawn uniformly at random among $[n+w, k]$-codes, then, with high probability,

$$\dim\left(\mathbf{Short}_{\mathcal{L}}\left(\mathscr{C}_{\text{Rand}}\right)\right)^{\star 2} = \min\left(n+w-|\mathcal{L}|,\ \binom{k-|\mathcal{L}|+1}{2}\right).$$

Computing a square code is a polynomial time operation. Hence, for given parameters (k, n, w), if there exists an integer $\ell \in [\![0, k]\!]$ such that these two formulas give different results, then by choosing a subset $\mathcal{L}$ of size ℓ, we obtain a polynomial time distinguisher.

So we want to find the valued of (n, k, w) such that there exists an ℓ such that

$$\min\left(n+w-\ell,\ 2(k+w-\ell)-1\right) < \min\left(n+w-\ell,\ \binom{k-\ell+1}{2}\right). \tag{6.20}$$

Condition (6.19), which is necessary due to Conjecture 6.26, cannot be an equality, and hence becomes

$$2(k+w-\ell)-1 < \binom{k-\ell+1}{2}. \tag{6.21}$$

The other necessary condition to obtain (6.20) is that

$$2(k+w-\ell)-1 < n+w-\ell. \tag{6.22}$$

On the other hand, equations (6.21) and (6.22) are sufficient to obtain (6.20).

Now, for a fixed value of n, k, w, let us find the values ℓ for which the inequalities (6.21) and (6.22) are satisfied.

First inequality. In order to determine when the first inequality (6.21) is verified, let us denote

$$k' \stackrel{\text{def}}{=} k - \ell.$$

Inequality (6.21) becomes $4k' - 2 + 4w < k'^2 + k'$, or equivalently $k'^2 - 3k' - 4w + 2 > 0$, which after a resolution leads to $k' > \frac{3+\sqrt{16w+1}}{2}$.

Hence, we have:

$$\ell < k - \frac{3 + \sqrt{16w+1}}{2}. \tag{6.23}$$

Second inequality. The second inequality (6.22) is equivalent to

$$\ell \geqslant w + 2k - n. \tag{6.24}$$

Conditions to verify both inequalities. Putting inequalities (6.23) and (6.24) together gives that ℓ should satisfy

$$w + 2k - n \leqslant \ell < k - \frac{3 + \sqrt{16w+1}}{2}.$$

We can therefore find an appropriate $\mathcal{L}$ if and only if

$$w + 2k - n < k - \frac{3 + \sqrt{16w+1}}{2},$$

which is equivalent to

$$n - k > w + \frac{3 + \sqrt{16w+1}}{2} = w + O(\sqrt{w}).$$

In other words, the distinguisher works up to values of w that are close to the second choice $n - k = w$. From now on, we set

$$\ell_{\min} \stackrel{\text{def}}{=} w + 2k - n \qquad \text{and} \qquad \ell_{\max} \stackrel{\text{def}}{=} \left\lceil k - \frac{3 + \sqrt{16w+1}}{2} - 1 \right\rceil. \tag{6.25}$$

Practical results. We have run experiments using Magma [BCP97] and Sage. For the parameters of Table 6.1, here are the intervals of possible values of ℓ so that the code $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})^{\star 2}$ has a non generic dimension:

- ID 1: $n = 532, k = 376, w = 96$, $\ell \in [\![316, 354]\!]$;
- ID 3: $n = 846, k = 618, w = 144$, $\ell \in [\![534, 592]\!]$;
- ID 5: $n = 1160, k = 700, w = 311$, $\ell \in [\![551, 663]\!]$.

The interval always coincides with the theoretical interval $[\![\ell_{\min}, \ell_{\max}]\!]$.

On the contrary, the parameters of the second group (listed in Table 6.2) are chosen such that $w = n - k$. For these parameters, there exists no value of ℓ verifying inequalities (6.23) and (6.24). Thus, the distinguisher cannot be applied for keys generated with these parameters.

We have seen that for some parameters (and especially for the parameters of the first group) there is a way to distinguish RLCE matrices from random matrices. Still, this does not provide a proper way to attack the scheme and recover the key. As we will see in the next section, a smart use of this distinguisher can be turned into an attack.

6.3 The attack

In this section, given a public key $\boldsymbol{G}_{\text{pub}}$ of an instance of the RLCE cryptosystem, we will show how to find an equivalent private key $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{Q}, \boldsymbol{P})$ defining the same code. This allows to decode and recover the original message like a legitimate user.

Remark 6.28. *In the present section where we the goal is to recover the permutation, we no longer work under Assumption 6.5.*

6.3.1 An algorithm to find a set of twin positions

The idea to distinguish different columns is to rely on the result of Lemma 6.24, especially in the equality case. Indeed, we see that the dimension evolves differently if one punctures a column in $\boldsymbol{G}_{\text{SCGRS}}$ than if one punctures a column in $\boldsymbol{G}_{\text{rand}}$. We obtain the following result.

Lemma 6.29. *Let $\mathscr{C}$ denote the public key of the public code of an instance of the RLCE scheme. Let $\mathcal{L}$ denote a subset of $[\![1, n+w]\!]$. Let $\mathcal{J}$ denote the set $(\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}_0) \cup \tau(\mathcal{L}_1)$ (following the notations from § 6.2.3). Suppose that $\mathcal{L}$ is such that such that Lemma 6.24 applied to $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ (where the columns of index in $\mathcal{J}$ correspond to $\boldsymbol{G}_{\text{SCGRS}}$) gives an equality.*

Then Algorithm 8 returns in polynomial time the set

$$\mathcal{T}_{\mathcal{L}} \stackrel{\text{def}}{=} \bigcup_{\{i,\tau(i)\} \subseteq [\![1,n+w]\!] \setminus \mathcal{L}} \{i, \tau(i)\}.$$

Proof. Let $\mathscr{C}$ denote the public key of the public code of an instance of the RLCE scheme. We have seen in the proof of Theorem 6.2 that for a subset $\mathcal{L} \subseteq [\![1, n+w]\!]$, we can apply Lemma 6.24 to $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$, where the columns

Algorithm 11: TwinSet$(\mathscr{C}, \mathcal{L})$

Input: The public RLCE code $\mathscr{C}$, a set $\mathcal{L} \subseteq [\![1, n+w]\!]$
Output: The set $\mathcal{T}_{\mathcal{L}}$

$\mathcal{T}_{\mathcal{L}} \leftarrow \emptyset$
$\mathscr{C}' \leftarrow \mathbf{Short}_{\mathcal{L}}(\mathscr{C})$
$d \leftarrow (\mathscr{C}'^{\star 2})$
for $i \in [\![1, n+w]\!] \setminus \mathcal{L}$ **do**
  $\mathscr{C}'' \leftarrow \mathbf{Punct}_{\{i\}}(\mathscr{C}')$
  if $\dim(\mathscr{C}''^{\star 2}) \neq d$ **then**
    $\mathcal{T}_{\mathcal{L}} \leftarrow \mathcal{T}_{\mathcal{L}} \cup \{i\}$
return $\mathcal{T}_{\mathcal{L}}$

corresponding to the subcode of a GRS code are the columns of the set $\mathcal{J} \stackrel{\text{def}}{=} (\mathcal{I}_{\text{GRS}} \setminus \mathcal{L}_0) \cup \tau(\mathcal{L}_1)$ (following the notations from § 6.2.3).

Suppose that the subset $\mathcal{L}$ is chosen such that Equation (6.14) of Lemma 6.24 is an equality. For all positions i in $[\![1, n+w]\!] \setminus \mathcal{L}$, let us compare the dimension of $(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))^{\star 2}$ with the dimension of $\left(\mathbf{Punct}_{\{i\}}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))\right)^{\star 2}$.

- If $i \in \mathcal{I}_{\text{GRS}}$, then $i \in \mathcal{J}$ so puncturing does not affect the dimension of the square code:

$$\dim(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))^{\star 2} = \dim\left(\mathbf{Punct}_{\{i\}}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))\right)^{\star 2}.$$

- If $i \in \mathcal{I}_{\text{PR}}$ and $\tau(i) \in \mathcal{L}$, then $i \in \tau(\mathcal{L}_1) \subseteq \mathcal{J}$. Indeed, according to Lemma 6.19, the position i is "derandomised" in $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$ and hence behaves like a GRS position in the shortened code. Therefore, very similarly to the previous case, the dimension does not change.

- If $i \in \mathcal{I}_{\text{PR}}$ and $\tau(i) \notin \mathcal{L}$, in $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})$, then $i \notin \mathcal{J}$. Indeed, in this case, the two corresponding columns behave like random ones. Hence, puncturing $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})^{\star 2}$ at i (resp. $\tau(i)$) reduces its dimension. Therefore,

$$\dim\left(\mathbf{Punct}_{\{i\}}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))\right)^{\star 2} = \dim(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))^{\star 2} - 1,$$

$$\dim\left(\mathbf{Punct}_{\{\tau(i)\}}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))\right)^{\star 2} = \dim(\mathbf{Short}_{\mathcal{L}}(\mathscr{C}))^{\star 2} - 1.$$

This provides a way to identify any position in $[\![1, n+w]\!] \setminus \mathcal{L}$ having a twin which also lies in $[\![1, n+w]\!] \setminus \mathcal{L}$: by searching zero columns in a parity–check

matrix of $\mathbf{Short}_{\mathcal{L}}(\mathscr{C})^{\star 2}$, we obtain the set $\mathcal{T}_{\mathcal{L}} \subset [\![1, n+w]\!] \setminus \mathcal{L}$ of even cardinality of all the positions having their twin in $[\![1, n+w]\!] \setminus \mathcal{L}$:

$$\mathcal{T}_{\mathcal{L}} \stackrel{\text{def}}{=} \bigcup_{\{i,\tau(i)\} \subseteq [\![1,n+w]\!] \setminus \mathcal{L}} \{i, \tau(i)\}.$$

□

6.3.2 Identifying pairs of twin positions

Once these positions are identified, we can associate each such position to its twin. This can be achieved through Algorithm 6.

Lemma 6.30. *Under the assumptions of Lemma 6.29, let $\mathcal{P}_{\mathcal{L}}$ denote the result of Algorithm 6, then*

$$\mathcal{P}_{\mathcal{L}} = \{ \{i, \tau(i)\}, \ i \in \mathcal{T}_{\mathcal{L}}\}.$$

Algorithm 12: FindTwins($\mathscr{C}, \mathcal{L}$)

Input: The public RLCE code $\mathscr{C}$, a set $\mathcal{L} \subseteq [\![1, n+w]\!]$
Output: The set $\mathcal{P}_{\mathcal{L}}$

```
P_L ← ∅
T_L ← TwinSet(C, L)
for i ∈ T_L do
    T_L^(i) ← TwinSet(C, L ∪ {i})
    P_L ← P_L ∪ {T_L \ T_L^(i)}
return P_L
```

Proof. For $i \in \mathcal{T}_{\mathcal{L}}$, compute $\mathcal{T}_{\mathcal{L}\cup\{i\}}$. The column corresponding to the twin position $\tau(i)$ has been derandomised and hence will not give a zero column in a parity–check matrix of $\left(\mathbf{Short}_{\mathcal{L}\cup\{i\}}(\mathscr{C})\right)^{\star 2}$, so puncturing the corresponding column will not affect the dimension. Hence, $\mathcal{T}_{\mathcal{L}\cup\{i\}} = \mathcal{T}_{\mathcal{L}} \setminus \{i, \tau(i)\}$. □

This process can be iterated by using various shortening sets $\mathcal{L}$ until obtaining w pairs of twin positions. It is readily seen that considering $O(1)$ such sets is enough to recover all pairs with very large probability.

6.3.3 Description of the attack

In summary, the attack works as follows.

1. Compute the interval $[\![\ell_{\min}, \ell_{\max}]\!]$ as defined in § 6.2.5 and choose ℓ in the middle of the interval. Ensure $\ell < \ell_{\max}$.
2. Apply Algorithm 6 for several sets of indices $\mathcal{L} \subseteq [\![1, n+w]\!]$ such that $|\mathcal{L}| = \ell$. Repeat this process until identifying all pairs of twin positions, as detailed in § 6.3.2.
3. Puncture the twin positions in order to get a GRS code and recover its structure using the Sidelnikov Shestakov attack [SS92].
4. For each pair of twin positions, recover the corresponding 2×2 non-singular matrix A_i, as explained in Section 6.3.4.
5. Finish to recover the structure of the underlying GRS code.

6.3.4 Retrieving the secret key

We explain here the steps 3 to 5 of the attack in order to obtain a key equivalent to the secret key.

Recovering the remainder of the code. As soon as all the pairs of twin positions are identified, consider the code $\mathbf{Punct}_{\mathcal{I}_{\mathrm{PR}}}(\mathscr{C})$ punctured at $\mathcal{I}_{\mathrm{PR}}$. Since the randomised positions have been punctured this code is nothing but a GRS code and, applying the Sidelnikov Shestakov attack [SS92], we recover a pair $\boldsymbol{a}, \boldsymbol{b}$ such that $\mathbf{Punct}_{\mathcal{I}_{\mathrm{PR}}}(\mathscr{C}) = \mathbf{GRS}_k(\boldsymbol{a}, \boldsymbol{b})$.

Joining a pair of twin positions. To recover the remaining part of the code we will consider iteratively the pairs of twin positions. We recall that $\mathcal{I}_{\mathrm{PR}}$ corresponds to the set of positions having a twin. Let $\{i, \tau(i)\}$ be a pair of twin positions and consider the code

$$\mathscr{C}^{(i)} \stackrel{\text{def}}{=} \mathbf{Punct}_{[\![1,n]\!] \setminus (\mathcal{I}_{\mathrm{GRS}} \cup \{i, \tau(i)\})}(\mathscr{C}) .$$

In this code, any position is GRS but positions i and $\tau(i)$. Hence, for any codeword $\boldsymbol{c} \in \mathscr{C}^{(i)}$ we have:

$$\begin{aligned} c_i &= a y_j f(x_j) + c\psi_j(f) \\ c_{\tau(i)} &= b y_j f(x_j) + d\psi_j(f) \end{aligned} \tag{6.26}$$

for some integer $j \in [\![n-w+1, n]\!]$, where ψ_j and $\boldsymbol{Q} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ are defined as in (6.2) and (6.1).

Note that we do not need to recover exactly $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{Q}, \boldsymbol{P})$. We need to recover a 4–tuple $(\boldsymbol{x}', \boldsymbol{y}', \boldsymbol{Q}', \boldsymbol{P}')$ which describes the same code (see Remark 1.29).

Thus, without loss of generality, after possibly replacing a by ay_j and b by by_j, one can suppose that $y_j = 1$. Moreover, after possibly replacing ψ_j by $d\psi_j$, one can suppose that $d = 1$. Recall that in this section we suppose that $cd \neq 0$.

Thanks to these simplifying choices, (6.26) becomes

$$\begin{aligned} c_i &= af(x_j) + c\psi_j(f) \\ c_{\tau(i)} &= bf(x_j) + \psi_j(f). \end{aligned}$$

Shortening $\mathscr{C}^{(i)}$ to recover x_j. If we shorten $\mathscr{C}^{(i)}$ at the $\tau(i)$-th position, according to Lemma 6.19, it will derandomise the i-th position (it implies $\psi_j(f) = -bf(x_j)$) and any $\boldsymbol{c} \in \mathbf{Short}_{\{\tau(i)\}}\left(\mathscr{C}^{(i)}\right)$ verifies

$$c_i = (a - bc)f(x_j).$$

Since the support x_j and multiplier y_j are known at all the positions of $\mathscr{C}^{(i)}$ but the two PR ones, for any codeword $\boldsymbol{c} \in \mathbf{Short}_{\{\tau(i)\}}\left(\mathscr{C}^{(i)}\right)$, one can find the polynomial $f \in \mathbf{F}_q[x]_{<k}$ whose evaluation provides $\boldsymbol{c}$. Therefore, by collecting a basis of codewords in $\mathbf{Short}_{\{\tau(i)\}}\left(\mathscr{C}^{(i)}\right)$ and the corresponding polynomials, we can recover the values of x_j and $a - bc$.

Recovering the entries of the $\boldsymbol{Q}$ matrix. Once we have x_j we need to recover the matrix

$$\boldsymbol{Q} = \begin{pmatrix} a & b \\ c & 1 \end{pmatrix}.$$

Note that, its determinant $\det \boldsymbol{Q} = a - bc$ has already been obtained in the previous section. First, one can guess b as follows. Let $\boldsymbol{G}^{(i)}$ be a generator matrix of $\mathscr{C}^{(i)}$. As in the previous section, by interpolation, one can compute the polynomials $f_1, \ldots, f_k$ whose evaluations provide the rows of $\boldsymbol{G}^{(i)}$. Consider the column vector

$$\boldsymbol{v} \stackrel{\text{def}}{=} \begin{pmatrix} f_1(x_j) \\ \vdots \\ f_k(x_j) \end{pmatrix}$$

and denote by $\boldsymbol{v}_i$ and $\boldsymbol{v}_{\tau(i)}$ the columns of $\boldsymbol{G}^{(i)}$ corresponding to positions c_i and $c_{\tau(i)}$:

$$\boldsymbol{v}_i = \begin{pmatrix} af_1(x_j) + c\psi_j(f_1) \\ \vdots \\ af_k(x_j) + c\psi_j(f_k) \end{pmatrix} \qquad \text{and} \qquad \boldsymbol{v}_{\tau(i)} = \begin{pmatrix} bf_1(x_j) + \psi_j(f_1) \\ \vdots \\ bf_k(x_j) + \psi_j(f_k) \end{pmatrix}.$$

Next, search $\lambda \in \mathbf{F}_q$ such that $\boldsymbol{v}_i - \lambda \boldsymbol{v}_{\tau(i)}$ is collinear to $\boldsymbol{v}$. This relation of collinearity can be expressed in terms of cancellation of some 2×2 determinants which are polynomials of degree 1 in λ. Their common root is nothing but c.

Finally, we can find the pair (a, b) by searching the pairs (λ, μ) such that

libel=(i) $\lambda - c\mu = \det \boldsymbol{Q}$;

liibel=(ii) $\boldsymbol{v}_i - \lambda \boldsymbol{v}$ and $\boldsymbol{v}_{\tau(i)} - \mu \boldsymbol{v}$ are collinear.

Here the relation of collinearity will be expressed as the cancellation of 2×2 determinants which are linear combinations of λ, μ and $\lambda\mu$ and elementary elimination process provides us with the value of the pair (a, b).

6.3.5 The case of degenerate twin positions

Recall that a pair of twin positions $i, \tau(i)$ is such that any codeword $\boldsymbol{c} \in \mathscr{C}$ has i–th and $\tau(i)$–th entries of the form:

$$\boldsymbol{c}_i = a y_j f(x_j) + b\psi_j(f) \qquad \boldsymbol{c}_{\tau(i)} = c y_j f(x_j) + d\psi_j(f).$$

This pair is said to be *degenerate* if either b or d is zero. In such a situation, some of the steps of the attack cannot be applied. In what follows, we explain how this rather rare issue can be addressed.

If either b or d is zero, then one of the positions is actually a pure GRS position while the other one is PR so Algorithm 6 does not manage to associate the two twin columns.

Suppose without loss of generality that $b = 0$. When applying Algorithm 8, the position $\tau(i)$ will be identified as PR but Algorithm 6 will not find its twin sister *a priori*. To find its twin sister, we can proceed as follows. For any GRS position j replace the j–th column $\boldsymbol{v}_j$ of a generator matrix $\boldsymbol{G}$ of $\mathscr{C}$ by an arbitrary linear combination of $\boldsymbol{v}_j$ and the $\tau(i)$–th column, this will "pseudo–randomise" this column and if the j–th column is the twin of the $\tau(i)$–th one, this will be detected by the process of shortening, squaring and searching zero columns in the parity check matrix.

6.3.6 Complexity of the attack

The most expensive part of the attack is the step consisting in identifying pairs of twin positions. Recall that, from [CGGOT14], the computation of the square of a code of length n and dimension k costs $O(k^2 n^2)$ operations in $\mathbf{F}_q$. We need to compute the square of a code $O(w)$ times, because there are w pairs of twin positions. Hence this step has a total complexity of $O(wn^2k^2)$ operations

in $\mathbf{F}_q$. Note that the actual dimension of the shortened codes is significantly less than k and hence the previous estimate is overestimated.

The cost of the Sidelnikov Shestakov attack is that of a Gaussian elimination, namely $O(nk^2)$ operations in $\mathbf{F}_q$ which is negligible compared to the previous step. The cost of the final part is also negligible compared to the computation of the squares of shortened codes. This provides an overall complexity in $O(wn^2k^2)$ operations in $\mathbf{F}_q$.

6.4 Conclusion

In this chapter, we have seen how the square-code distinguisher can be adapted to distinguish in polynomial time another public key encryption scheme involving a GRS structure, and how to use this new distinguisher to mount an attack on the cryptosystem. In this situation, the distinguisher does not work for all possible parameters. We therefore have a polynomial time attack that breaks all the so-called *odd ID* parameters suggested in [Wan17], but the *even ID* parameters, remain out of the reach. Namely, the distinguisher works when the number w of random columns is strictly less than $n - k$, and our analysis suggests that, for this kind of distinguisher by squaring shortenings of the code, the case $w = n - k$ is the critical one. After the publication of this attack in [CLT19], the RLCE cryptosystem was withdrawn from the NIST post-quantum standardization process.

Chapter 7

Subspace subcodes of Reed-Solomon codes

We have seen in Chapter 5 that the instantiation of McEliece's scheme with generalised Reed–Solomon codes is insecure, and that most similar proposals involving variants of GRS codes (among which the RLCE scheme studied in Chapter 6) are subject to attacks. Conversely, as presented in Chapter 1, McEliece's original proposal to instantiate his scheme using binary Goppa codes is still considered secure after forty years of cryptanalysis attempts.

But Goppa codes have a strong connection to GRS codes. Indeed, Goppa codes are a special family of alternant codes, which are subfield subcodes of GRS codes. In this chapter, we consider the spectrum with (full) GRS codes on one end and their subfield subcodes (*i.e.* alternant codes) on the other. The intermediary case is that of *subspace subcodes* of Reed–Solomon (SSRS) codes. This notion was originally introduced without any cryptographic motivation by Solomon, McEliece and Hattori.

This chapter is dedicated to discussing the security of McEliece's encryption scheme instantiated with subspace subcodes of Reed–Solomon codes. We introduce such a cryptosystem and show that it generalises the XGRS cryptosystem from Khaturia, Rosenthal and Weger. Then, we adapt the square-code distinguisher over this new family of codes, by introducing a new tool called the *twisted product*. Finally we show that this distinguisher can be used to build an efficient attack on this scheme when the dimension of the subspace is large enough. In particular, this attack breaks some parameters of the XGRS cryptosystem.

Related publication: Couvreur and Lequesne, *On the security of subspace subcodes of Reed–Solomon codes for public-key encryption* (preprint) [CL20].

Contents

7.1 Subspace subcodes

7.1.1 Motivations

7.1.1.1 Subspace subcodes in cryptography

The significant size of the public key in McEliece's original scheme using binary Goppa codes has encouraged cryptographers to propose the use of other families of codes to instantiate the scheme. As we have seen in the previous chapters, GRS codes have appealing properties that make them a tempting candidate to use in a McEliece scheme. But over the years, almost all attempts of cryptosystem involving variations on GRS codes has proved to be insecure.

In summary, forty years of research on the use of algebraic codes for public key encryption boil down to the following observations.

(1) On one hand, the raw use of GRS codes as well as most of the variants using these codes lead to insecure schemes.

(2) On the other hand, Goppa codes or more generally alternant codes remain robust decades after they were initially proposed by McEliece.

Here, it is important to recall that alternant codes are constructed from GRS codes. Indeed, alternant codes are *subfield subcodes* of GRS codes (see Definition 5.5). Therefore, in order to better understand the hardness of distinguishing Goppa codes, it is interesting to consider GRS codes and alternant codes as the two ends of a continuous spectrum, where the intermediary case is that of *subspace subcodes* of Reed–Solomon codes.

A subspace subcode of a Reed–Solomon code (SSRS) is a subset of a parent Reed-Solomon code over $\mathbf{F}_{q^m}$ consisting of the codewords whose components all lie in a fixed λ-dimensional $\mathbf{F}_q$-vector subspace of $\mathbf{F}_{q^m}$, for some $\lambda \leqslant m$.

Definition 7.1 ([HMS98])**.** Given a linear code $\mathscr{C}$ defined over a field $\mathbf{F}_{q^m}$, and a λ-dimensional subspace $\mathcal{S}$ of $\mathbf{F}_{q^m}$ $(0 \leqslant \lambda \leqslant m)$, the subspace subcode $\mathscr{C}_{|\mathcal{S}}$ is defined to be the set of codewords of $\mathscr{C}$ whose components all lie in $\mathcal{S}$.

$$\mathscr{C}_{|\mathcal{S}} \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathscr{C} \mid \forall i \in [\![0, n-1]\!],\ c_i \in \mathcal{S}\} \subseteq \mathbf{F}_{q^m}^n.$$

As we can see from the definition, when the parent code $\mathscr{C}$ is a GRS code, the case $\lambda = 1$ corresponds to alternant codes (see Definition 5.5), as any subspace of dimension 1 is a subfield. On the other hand, the case $\lambda = m$ corresponds to the usual case of GRS codes, studied in Chapter 5. The notion of subspace subcodes permits a modulation of the parameter λ.

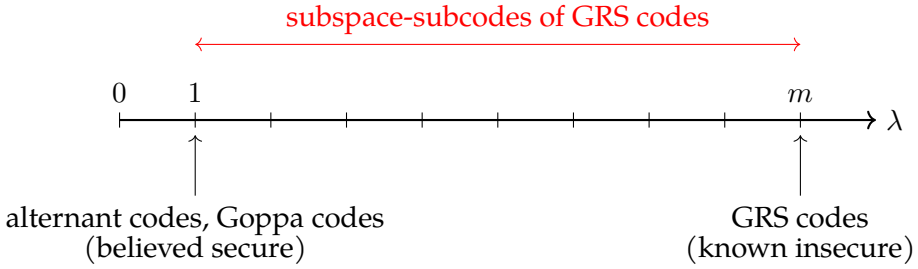


Figure 7.1: Spectrum representing subspace-subcodes of GRS codes of increasing dimension.

We remark that the use of subspace subcodes for cryptography first appears in the context of rank metric. In [GL05; GL08], Gabidulin and Loidreau who propose to use subspace subcodes of Gabidulin codes for a rank-metric based cryptosystem. This can be seen as a rank-metric equivalent of our motivations.

Concerning Hamming-metric public key cryptography, two recent works exploring different approaches appeared in the recent years. First, Berger, Gueye, Klamti and Ruatta are the first to propose a McEliece scheme based on subspace subcodes of Reed–Solomon codes in [BGKR19]. As they intend to reduce the key size, their work focuses more specifically on low-dimensional

subcodes of quasi–cyclic SSRS codes. In another line of work, in the article *"Encryption Scheme Based on Expanded Reed-Solomon Codes"* [KRW21], Khathuria, Rosenthal and Weger propose an encryption scheme using expanded subspace subcodes of GRS codes instead of Goppa codes. Throughout the document, we will refer to this scheme as the *XGRS* scheme (where the *X* stands for *expanded*).

Consequently, the study of SSRS codes is of particular interest for two reasons.

(1) Subspace subcodes may provide interesting codes for encryption with $\lambda > 1$, providing shorter keys than the original McEliece scheme.

(2) Their security analysis encompasses that of Goppa and alternant codes and may help to better understand the security of McEliece encryption scheme. Such a security analysis is of crucial interest since *Classic McEliece* lies among the very few candidates selected by the NIST for the last round of the post-quantum standardisation process.

7.1.1.2 Subspace subcodes in information theory

It is important to recall that, as most tools in code-based cryptography, the notion of subspace subcodes finds its origin in information theory and was first studied as a way to construct codes with good transmission and error-correction properties. The interest from cryptographic perspective is very recent.

Therefore, we explain here the origin of this notion and the motivation behind this line of research.

Trace-shortened codes. The idea behind subspace subcodes, which consists in keeping only the subset of codewords that are defined over a subspace of the field, first appears in a paper by Solomon [Sol93]. In a joint work with McEliece [MS94], they define the notion of *trace-shortened* codes, which is a special case of subspace subcodes where $\lambda = m - 1$ and where the considered subspace $\mathcal{S}$ is the kernel of the trace map.

SSRS codes. In his thesis [Hat95] and in [HMS98], Hattori studies the dimension of subspace subcodes of Reed–Solomon codes. Some of his conjectures are later proved by Spence in [Spe04].

This initial work on subspace subcodes is motivated by the fact that the SSRS construction provides long codes with good parameters over alphabets of moderate size, in the spirit of alternant codes [MS86, Chapter 12]. This makes these codes interesting from an information-theoretic point of view.

An example. The following example comes directly from [HMS98] and explains the interest on these codes from an information-theoretic point of view.

Consider $\mathscr{C}$ the Reed–Solomon code over $\mathbf{F}_{2^4}$ of length 15 and dimension 9. This code has minimum distance 7. Any element of $\mathbf{F}_{2^4}$ can be decomposed over the $\mathbf{F}_2$-basis $(1, \alpha, \alpha^2, \alpha^3)$, where α is a root of the irreducible polynomial $X^4 + X + 1$. Let $\mathcal{S}$ be the subspace spanned by $(1, \alpha, \alpha^2)$. The code $\mathscr{C}_{|\mathcal{S}}$ is the subset of codewords of $\mathscr{C}$ that have no component in α^3. Hence, if one uses this code for communication, there is no need to send the α^3 component, since it is always zero.

So this subspace subcode can be seen as an $\mathbf{F}_2$-linear code of length 15 over the set of binary 3-tuples. But the code is not a linear code over $\mathbf{F}_{2^3}$. The minimum distance of $\mathscr{C}_{|\mathcal{S}}$ is at least 7, because it cannot be less than the minimum distance of the parent code. The number of codewords in $\mathscr{C}_{|\mathcal{S}}$ is 2^{22}. As a comparison, one other way to create a code of length 15 over binary 3-tuples is by shortening the generalised BCH code $[63, 52, 7]$ over $\mathbf{F}_{2^3}$. This gives a $[15, 4, \geqslant 7]$ code over $\mathbf{F}_{2^3}$ which has 2^{12} codewords.

SSRS vs SSGRS. Hattori's work and the later articles focus uniquely on subspace subcodes of Reed–Solomon codes, not on generalised Reed–Solomon. But this point of view turns out to be the most general one since a subspace subcode of a GRS code can always be regarded as a subspace subcode of an RS code by changing the subspaces as we will see in Corollary 7.8. Therefore, we will only talk about subspace-subcodes of RS codes but it is important to keep in mind that this notion encompasses that of GRS codes.

Generalisation. The notion of subspace subcodes is generalised to any kind of subspace and any code by Jensen in [Jen95] under the name *subgroup subcodes*. Later, in [Wu11] Wu proposes a more constructive approach of these codes using the equivalent of the expansion operator that we will introduce in § 7.1.3.

7.1.2 Definition and first properties

We refer the reader to Chapter 1 for general definitions about codes, as well as Chapter 5 for definitions concerning GRS codes and the star-product operation. We emphasize that in this Chapter we will sometimes make use of the star-product spanned over a subfield, as defined in Notation 5.10 and Remark 5.11.

The definition of a *subspace subcode* has been stated in Definition 7.1. It is important to note that the code $\mathscr{C}_{|\mathcal{S}}$ is an $\mathbf{F}_q$–linear subspace of $\mathbf{F}_{q^m}^n$ which is generally neither $\mathbf{F}_{q^m}$–linear nor linear over some intermediary extension.

Since each entry of a codeword can be represented as λ elements of $\mathbf{F}_q$, the code could be converted into a code over the alphabet $\mathbf{F}_q^\lambda$. Such a code would form an additive subgroup over $\left(\mathbf{F}_q^\lambda\right)^n$ (hence the name *subgroup subcode* given by Jensen in [Jen95]). In a context of message transmission, this natural way to represent such a subspace subcode is detailed further in § 7.1.3.

7.1.2.1 Dimension of subspace subcodes

Proposition 7.2. *Let $\mathscr{C}$ be a linear code of length n and dimension k over $\mathbf{F}_{q^m}$ and $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ be a subspace of dimension $\lambda \leqslant m$. Then*

$$\dim_{\mathbf{F}_q} \mathscr{C}_{|\mathcal{S}} \geqslant km - n(m-\lambda). \tag{7.1}$$

This result derives naturally from the representation of subspace subcodes expanded over a basis of $\mathcal{S}$ as we will see in § 7.1.3. Therefore its prove will be given at the end of the section.

The inequality (7.1) is typically an equality (and can therefore be considered as such for cryptanalysis, see Remark 5.16).

Proposition 7.3. *Let $\mathscr{R}$ be a uniformly random code among the codes of length n and dimension k over $\mathbf{F}_{q^m}$. Let $\mathcal{S}_0, \ldots, \mathcal{S}_{n-1}$ be $\mathbf{F}_q$–subspaces of $\mathbf{F}_{q^m}$ of dimension λ. Suppose that $km > n(m-\lambda)$. Then, for any integer ℓ, we have*

$$\mathbb{P}\left[\dim_{\mathbf{F}_q} \mathscr{R}_{|\mathcal{S}} \geqslant km - n(m-\lambda) + \ell\right] \leqslant q^{-\ell}\left(\frac{1}{1-q^{-mn}} + \frac{1}{q^{km-n(m-\lambda)}}\right).$$

In particular, for fixed values of q, m and λ, this probability is in $O(q^{-\ell})$ when $n \to \infty$.

Proof. Let $\boldsymbol{G}_{\mathrm{rand}}$ be a uniformly random variable among the full rank matrices in $\mathbf{F}_{q^m}^{k\times n}$ and

$$\mathscr{R} \stackrel{\text{def}}{=} \{\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}} \mid \boldsymbol{m} \in \mathbf{F}_{q^m}^k\}.$$

The code $\mathscr{R}$ is uniformly random among the set of $[n,k]$ codes over $\mathbf{F}_{q^m}$ ([Cou20, Lemma 3.12]). Let Φ be the $\mathbf{F}_q$–linear canonical projection

$$\Phi : \mathbf{F}_{q^m}^n \longrightarrow \prod_{i=0}^{n-1} \mathbf{F}_{q^m}/\mathcal{S}.$$

Then, $\mathscr{R}_{|\mathcal{S}}$ is the kernel of the restriction of Φ to $\mathscr{R}$ and hence,

$$\begin{aligned}\mathbb{E}\left[|\mathscr{R}_{|\mathcal{S}}|\right] &= \mathbb{E}\left[\sum_{\boldsymbol{m}\in\mathbf{F}_{q^m}^k} \mathbb{1}_{\Phi(\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}})=0}\right] \\ &= \sum_{\boldsymbol{m}\in\mathbf{F}_{q^m}^k} \mathbb{P}\left[\Phi(\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}}) = 0\right] \\ &= 1 + \sum_{\boldsymbol{m}\in\mathbf{F}_{q^m}^k\setminus\{0\}} \mathbb{P}\left[\Phi(\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}}) = 0\right]. \end{aligned} \tag{7.2}$$

Since $\boldsymbol{G}_{\mathrm{rand}}$ is uniformly random among the full–rank matrices, then for any $\boldsymbol{m} \in \mathbf{F}_{q^m}^k \setminus \{0\}$, the vector $\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}}$ is uniformly random in $\mathbf{F}_{q^m} \setminus \{0\}$ ([Cou20, Lemma 3.13]) and hence

$$\begin{aligned}\forall \boldsymbol{m} \in \mathbf{F}_{q^m}^k \setminus \{0\}, \quad \mathbb{P}\left[\Phi(\boldsymbol{m}\boldsymbol{G}_{\mathrm{rand}} = 0)\right] &= \frac{|\ker\Phi \setminus \{0\}|}{\left|\mathbf{F}_{q^m}^n \setminus \{0\}\right|} \\ &= \frac{\left|\prod_i \mathcal{S}_i\right| - 1}{q^{mn} - 1} \\ &= \frac{q^{\lambda n} - 1}{q^{mn} - 1} \leqslant q^{-n(m-\lambda)} \cdot \frac{1}{1 - q^{-mn}}.\end{aligned}$$

Thus, applied to (7.2),

$$\begin{aligned}\mathbb{E}\left[|\mathscr{R}_{|\mathcal{S}}|\right] &\leqslant 1 + |\mathbf{F}_{q^m}^k \setminus \{0\}| \cdot q^{-n(m-\lambda)} \cdot \frac{1}{1 - q^{-mn}} \\ &\leqslant 1 + q^{km - n(m-\lambda)} \cdot \frac{1}{1 - q^{-mn}}.\end{aligned}$$

Finally, using Markov inequality, we get

$$\begin{aligned}\mathbb{P}\left[\dim_{\mathbf{F}_q}(\mathscr{R}_{|\mathcal{S}}) \geqslant km - n(m-\lambda) + \ell\right] &= \mathbb{P}\left[|\mathscr{R}_{|\mathcal{S}}| \geqslant q^{km-n(m-\lambda)+\ell}\right] \\ &\leqslant \frac{\mathbb{E}\left[|\mathscr{R}_{|\mathcal{S}}|\right]}{q^{km-n(m-\lambda)+\ell}} \\ &\leqslant q^{-\ell}\left(\frac{1}{1-q^{-mn}} + \frac{1}{q^{km-n(m-\lambda)}}\right).\end{aligned}$$

□

7.1.2.2 Subspace subcode with different subspaces

We can generalise the definition of subspace subcodes with different subspaces for each entry. This idea is first mentioned in [DT99].

Definition 7.4. Given a linear code $\mathscr{C}$ of length n over a field $\mathbf{F}_{q^m}$, and the λ-dimensional subspaces $(\mathcal{S}_0, \dots, \mathcal{S}_{n-1})$ of $\mathbf{F}_{q^m}$ $(0 \leqslant \lambda \leqslant m)$, the subspace subcode $\mathscr{C}_{|(\mathcal{S}_0,\dots,\mathcal{S}_{n-1})}$ is defined to be the set of codewords of $\mathscr{C}$ such that the i-th components lies in $\mathcal{S}_i$.

$$\mathscr{C}_{|(\mathcal{S}_0,\dots,\mathcal{S}_{n-1})} \stackrel{\text{def}}{=} \{\boldsymbol{c} \in \mathscr{C} \mid \forall i \in [\![0, n-1]\!],\ c_i \in \mathcal{S}_i\}.$$

Remark 7.5. *When $\mathcal{S}_0 = \cdots = \mathcal{S}_{n-1} = \mathbf{F}_q$, then we find the usual definition of* subfield subcode*.*

Remark 7.6. *It is even possible to give a more general definition where the $\mathcal{S}_i$'s do not have the same dimension λ. However, such a broader definition would be useless for our study.*

The previous results concerning the dimension of subspace subcodes remain exactly the same for subcodes with different subspaces. In fact, it will be the case for most results that we will see in this chapter. Therefore, in order to simplify the notations, we will often write the proofs considering a that all the subspaces $\mathcal{S}_i$ are equal to the same subspace $\mathcal{S}$.

The following proposition explains how the different subspaces behave when multiplying each entry of the code by a scalar. As a corollary, we see that with this definition of subspace subcodes with different subspaces, the subspace subcodes of generalised Reed–Solomon codes can be rewritten as subspace subcodes of Reed-Solomon codes. Hence, the notion of subspace subcodes of RS codes encompasses that of GRS codes.

Proposition 7.7. *Let $\mathscr{C} \subseteq \mathbf{F}_{q^m}^n$, $\mathcal{S}_0, \dots, \mathcal{S}_{n-1} \subseteq \mathbf{F}_{q^m}$ be $\mathbf{F}_q$ subspaces and let $\boldsymbol{a} \in (\mathbf{F}_{q^m}^\times)^n$. Then,*

$$(\mathscr{C} \star \boldsymbol{a})_{|(\mathcal{S}_0,\dots,\mathcal{S}_{n-1})} = \mathscr{C}_{|(a_0^{-1}\mathcal{S}_0,\dots,a_{n-1}^{-1}\mathcal{S}_{n-1})} \star \boldsymbol{a}.$$

Corollary 7.8. *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{F}_{q^m}^n$ be a support and a multiplier and $\mathcal{S}_0, \dots, \mathcal{S}_{n-1} \subseteq \mathbf{F}_{q^m}$, then*

$$\boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})_{|(\mathcal{S}_0,\dots,\mathcal{S}_{n-1})} = \boldsymbol{RS}_k(\boldsymbol{x})_{|(y_0^{-1}\mathcal{S}_0,\dots,y_{n-1}^{-1}\mathcal{S}_{n-1})} \star \boldsymbol{y}.$$

7.1.3 Expansion operator and representation

With our definition, subspace subcodes are not linear codes over $\mathbf{F}_{q^m}$. This is not very convenient, especially if we want to implement computations involving such codes. Therefore we will represent them as linear codes over the subfield $\mathbf{F}_q$ with a higher length. This representation requires to choose a specific basis of the subspace $\mathcal{S}$, on which the code will be expanded. Hence,

the same subspace subcode can be represented in different ways depending on the choice of the basis.

For this sake we introduce the *expansion operator* and give some of its properties. The main result, Lemma 7.30, will state that subspace subcodes are equivalent to shortened expanded codes and provide a way to construct their parity-check matrix.

Note that expanding codes, in particular Reed–Solomon codes, over the base field has been studied since the 1980's. For instance, in [KL85; KL88], Kasami and Lin investigate the weight distribution of expanded binary Reed–Solomon codes. Sakakibara, Tokiwa and Kasahara extend their work to q-ary Reed–Solomon codes [STK89].

7.1.3.1 Bases and trace map

First, let us introduce the trace map. The trace is a linear form over $\mathbf{F}_{q^m}$ and is therefore a natural tool to study subfield subcodes.

Definition 7.9 (Trace map)**.** Let q be a prime power and m an integer. The trace map is defined as

$$\mathbf{Tr}: \quad \begin{cases} \mathbf{F}_{q^m} & \longrightarrow & \mathbf{F}_q \\ x & \longmapsto & \sum_{i=0}^{m-1} x^{q^i}. \end{cases}$$

Definition 7.10 ([LN97, Definition 2.30])**.** Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be an $\mathbf{F}_q$-basis of $\mathbf{F}_{q^m}$. There exists a unique basis $\mathcal{B}^* = (b_0^*, \ldots, b_{m-1}^*)$, such that :

$$\forall 0 \leqslant i, j \leqslant m-1, \qquad \mathbf{Tr}(b_i b_j^*) = \begin{cases} 1 \text{ if } i = j, \\ 0 \text{ otherwise.} \end{cases}$$

This basis will be referred to as the *dual* basis of $\mathcal{B}$ and denoted $\mathcal{B}^*$.

Given an $\mathbf{F}_q$–basis $\mathcal{B} = (b_0, \ldots, b_{m-1})$ of $\mathbf{F}_{q^m}$ and x an element of $\mathbf{F}_{q^m}$. Then the expression of x as an $\mathbf{F}_q$–linear combination of the elements of $\mathcal{B}$ writes as

$$x = \mathbf{Tr}(b_0^* x) b_0 + \cdots + \mathbf{Tr}(b_{n-1}^* x) b_{n-1} \tag{7.3}$$

where $\mathcal{B}^* = (b_0^*, \ldots, b_{m-1}^*)$ denotes the dual basis of $\mathcal{B}$.

7.1.3.2 The expansion operator

Definition 7.11 (Expansion of a vector)**.** For a basis $\mathcal{B}$ of $\mathbf{F}_{q^m}$, let $\mathbf{ExpVec}_{\mathcal{B}}$ denote the expansion of a vector over the basis $\mathcal{B}$ defined by

$$\begin{cases} \mathbf{F}_{q^m}^{\ell} & \longrightarrow & \mathbf{F}_q^{m\ell} \\ (x_0, \ldots, x_{\ell-1}) & \longmapsto & (\mathbf{Tr}(b_0^* x_0), \ldots, \mathbf{Tr}(b_{m-1}^* x_0), \ \ldots \ , \\ & & \mathbf{Tr}(b_0^* x_{\ell-1}) \ldots, \mathbf{Tr}(b_{m-1}^* x_{\ell-1})), \end{cases}$$

where $\mathcal{B}^* = (b_0^*, \ldots, b_{m-1}^*)$ denotes the dual basis of $\mathcal{B}$. Note that we will apply this operator to vectors of different lengths ℓ.

As seen in (7.3), regarding an element $x \in \mathbf{F}_{q^m}$ as the vector (x) of length 1, let $(x_0, \ldots, x_{m-1}) \stackrel{\text{def}}{=} \mathbf{ExpVec}_{\mathcal{B}}((x)) \in \mathbf{F}_q^m$, then $x = \sum_{i=0}^{m-1} x_i b_i$.

Definition 7.12 (Expansion of a code)**.** For a linear code $\mathscr{C}$ of length n over $\mathbf{F}_{q^m}$ and a basis $\mathcal{B}$ of $\mathbf{F}_{q^m}$, denote $\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})$ the linear code over $\mathbf{F}_q$ defined by

$$\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C}) \stackrel{\text{def}}{=} \{\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathscr{C}\}.$$

We can also define the expansion operator over matrices.

Definition 7.13 (Expansion of a matrix)**.** Given $\mathcal{B} = (b_0, \ldots, b_{m-1})$ an $\mathbf{F}_q$-basis of $\mathbf{F}_{q^m}$. Let $\mathbf{ExpMat}_{\mathcal{B}}$ denote the following operation.

$$\left\{ \begin{array}{ccc} \mathbf{F}_{q^m}^{k\times n} & \longrightarrow & \mathbf{F}_q^{mk\times mn} \\ \begin{pmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,n-1} \\ \vdots & \vdots & & \vdots \\ m_{k-1,0} & m_{k-1,1} & \cdots & m_{k-1,n-1} \end{pmatrix} & \longmapsto & \begin{pmatrix} \boldsymbol{M}_{0,0} & \boldsymbol{M}_{0,1} & \cdots & \boldsymbol{M}_{0,n-1} \\ \boldsymbol{M}_{1,0} & \boldsymbol{M}_{1,1} & \cdots & \boldsymbol{M}_{1,n-1} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{M}_{k-1,0} & \boldsymbol{M}_{k-1,1} & \cdots & \boldsymbol{M}_{k-1,n-1} \end{pmatrix} \end{array} \right.$$

where

$$\boldsymbol{M}_{i,j} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{Tr}(b_0 b_0^* m_{i,j}) & \mathbf{Tr}(b_0 b_1^* m_{i,j}) & \ldots & \mathbf{Tr}(b_0 b_{m-1}^* m_{i,j}) \\ \mathbf{Tr}(b_1 b_0^* m_{i,j}) & \mathbf{Tr}(b_1 b_1^* m_{i,j}) & \ldots & \mathbf{Tr}(b_1 b_{m-1}^* m_{i,j}) \\ \vdots & \vdots & & \vdots \\ \mathbf{Tr}(b_{m-1} b_0^* m_{i,j}) & \mathbf{Tr}(b_{m-1} b_1^* m_{i,j}) & \ldots & \mathbf{Tr}(b_{m-1} b_{m-1}^* m_{i,j}) \end{pmatrix} \in \mathbf{F}_q^{m\times m},$$

and $\mathcal{B}^* = (b_0^*, \ldots, b_{m-1}^*)$ denotes the dual basis of $\mathcal{B}$ (Definition 7.10).

Remark 7.14. *Caution, applying $\mathbf{ExpMat}_{\mathcal{B}}$ to an $1 \times n$ matrix returns an $m \times nm$ matrix. It is **not** equivalent to applying $\mathbf{ExpVec}_{\mathcal{B}}$ to the vector corresponding to this row.*

Remark 7.15. $\mathbf{ExpMat}_{\mathcal{B}^*}(\boldsymbol{M}) = (\mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{M}^\intercal))^\intercal$.

Proposition 7.16 ([KRW21, Proposition 1])**.** *Let $\mathscr{C}$ be a linear code of dimension k and length n over $\mathbf{F}_{q^m}$. Let $\boldsymbol{G}$ denote a generator matrix of $\mathscr{C}$ and $\boldsymbol{H}$ denote a parity-check matrix of $\mathscr{C}$. Then, for any fixed $\mathbf{F}_q$-basis $\mathcal{B}$ of $\mathbf{F}_{q^m}$, the following hold.*

(i) For all $\boldsymbol{x} \in \mathbf{F}_{q^m}^k$, we have $\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x} \cdot \boldsymbol{G}) = \mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x}) \cdot \mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{G})$.

(*ii*) *For all* $\boldsymbol{y} \in \mathbf{F}_{q^m}^n$, *we have* $\mathbf{ExpVec}_{\mathcal{B}}((\boldsymbol{H} \cdot \boldsymbol{y}^\intercal)^\intercal)^\intercal = \mathbf{ExpMat}_{\mathcal{B}^*}(\boldsymbol{H}) \cdot \mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{y})^\intercal$.

Proof. We will prove the first statement. Denote $(x_0, \ldots, x_{k-1})$ the entries of $\boldsymbol{x}$ and $(g_{i,j})$ the entries of $\boldsymbol{G}$. Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be the basis and $\mathcal{B}^* = (b_0^*, \ldots, b_{m-1}^*)$ its dual basis. For $i \in [\![0, n-1]\!]$ and $j \in [\![0, m-1]\!]$, we show that the entry in position $im + j$ (*i.e.* j-th entry of the i-th block) of the two expressions are equal. Denote $[\cdot]_\ell$ the ℓ-th entry of a vector.

$$
\begin{aligned}
&[\mathbf{ExpVec}_{\mathcal{B}}\ (\boldsymbol{x} \cdot \boldsymbol{G})]_{im+j} \\
&\qquad = \quad \mathbf{Tr}(b_j^* \, [\boldsymbol{x} \cdot \boldsymbol{G}]_i) \\
&\qquad = \quad \mathbf{Tr}\left(b_j^* \sum_{\ell_1=1}^{k-1} x_{\ell_1} g_{\ell_1,i} \right) \\
&\qquad = \quad \mathbf{Tr}\left(b_j^* \sum_{\ell_1=1}^{k-1} \left(\sum_{\ell_2=1}^{m-1} \mathbf{Tr}(b_{\ell_2}^* x_{\ell_1}) b_{\ell_2} \right) g_{\ell_1,i} \right) \\
&\qquad = \quad \sum_{\ell_1=1}^{k-1} \sum_{\ell_2=1}^{m-1} \mathbf{Tr}(b_{\ell_2}^* x_{\ell_1}) \mathbf{Tr}(b_{\ell_2} b_j^* g_{\ell_1,i}) \\
&\qquad = \quad \sum_{\ell_1=1}^{k-1} \sum_{\ell_2=1}^{m-1} [\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x})]_{\ell_1 m+\ell_2} \, [\mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{G})]_{\ell_1 m+\ell_2, im+j} \\
&\qquad = \quad [\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x}) \cdot \mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{G})]_{im+j}
\end{aligned}
$$

□

Corollary 7.17. *Let* $\boldsymbol{G}$ *and* $\boldsymbol{H}$ *be a generator and a parity-check matrix of* $\mathscr{C}$ *respectively. Let* $\mathcal{B}$ *denote an* $\mathbf{F}_q$*-basis of* $\mathbf{F}_{q^m}$. *Then* $\mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{G})$ *and* $\mathbf{ExpMat}_{\mathcal{B}^*}(\boldsymbol{H})$ *are respectively a generator matrix and a parity-check matrix of* $\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})$.

Definition 7.18 (Block)**.** Given a vector $\boldsymbol{v} \in \mathbf{F}_{q^m}^n$, an $\mathbf{F}_q$–basis $\mathcal{B}$ of $\mathbf{F}_{q^m}$ and a non negative integer $i < n$, the i–th *block* of the expanded vector $\mathbf{ExpVec}_{\mathcal{B}}(v) \in \mathbf{F}_q^{mn}$ is the length m vector composed by the entries of index $mi, mi+1, \ldots, mi+m-1$ of $\mathbf{ExpVec}_{\mathcal{B}}(v)$. It corresponds to the decomposition over $\mathcal{B}$ of the i-th entry of $\boldsymbol{v}$. We extend this definition to matrices, where the i-th block of an expanded matrix means the $mk \times m$ matrix whose rows correspond to the i-th block of each row of the expanded matrix.

In particular, the expansion in a basis $\mathcal{B}$ of some $\boldsymbol{x} \in \mathbf{F}_{q^m}^n$ is the concatenation of n blocks of length m.

7.1.3.3 Expansion over various bases

We have seen in Definition 7.4 that we could define a subspace subcode with different subspaces for each entry. Similarly, we can define an expansion with regard to a different basis for each entry. All the previous definitions extend naturally as follows.

Definition 7.19. Given ℓ bases $(\mathcal{B}_0, \dots, \mathcal{B}_{\ell-1})$ of $\mathbf{F}_{q^m}$, let $\mathbf{ExpVec}_{(\mathcal{B}_i)_i}$ denote the expansion of a vector of length ℓ, such that the i^{th} column is expanded over the basis $\mathcal{B}_i$:

$$\begin{aligned}&\mathbf{ExpVec}_{(\mathcal{B}_i)_i}(x_0, \dots, x_{\ell-1}) = \\ &(\mathbf{Tr}(b^*_{0,0}x_0), \dots, \mathbf{Tr}(b^*_{0,m-1}x_0), \ \dots \ , \mathbf{Tr}(b^*_{\ell-1,0}x_{\ell-1}) \dots, \mathbf{Tr}(b^*_{\ell-1,m-1}x_{\ell-1})),\end{aligned}$$

where $\mathcal{B}_i = (b_{i,0}, \dots, b_{i,m-1})$.

Definition 7.20. For a linear code $\mathscr{C}$ of length n over $\mathbf{F}_{q^m}$ and n bases $(\mathcal{B}_i)_i$ of $\mathbf{F}_{q^m}$, denote $\mathbf{ExpCode}_{(\mathcal{B}_i)_i}(\mathscr{C})$ the linear code over $\mathbf{F}_q$ defined by:

$$\mathbf{ExpCode}_{(\mathcal{B}_i)_i}(\mathscr{C}) \stackrel{\text{def}}{=} \{\mathbf{ExpVec}_{(\mathcal{B}_i)_i}(\boldsymbol{c}) \,|\, \boldsymbol{c} \in \mathscr{C}\}.$$

Definition 7.21. Given $n+1$ bases $(\mathcal{B}_0, \dots, \mathcal{B}_{n-1}, \bar{\mathcal{B}})$ of $\mathbf{F}_{q^m}$, let $\mathbf{ExpMat}^{\bar{\mathcal{B}}}_{(\mathcal{B}_j)_j}$ denote the expansion of a matrix

$$\left\{ \begin{array}{ccc} \mathbf{F}_{q^m}^{k\times n} & \longrightarrow & \mathbf{F}_q^{mk\times mn} \\ \begin{pmatrix} m_{0,0} & m_{0,1} & \cdots & m_{0,n-1} \\ m_{1,0} & m_{1,1} & \cdots & m_{1,n-1} \\ \vdots & \vdots & & \vdots \\ m_{k-1,0} & m_{k-1,1} & \cdots & m_{k-1,n-1} \end{pmatrix} & \longmapsto & \begin{pmatrix} \boldsymbol{M}_{0,0} & \boldsymbol{M}_{0,1} & \cdots & \boldsymbol{M}_{0,n-1} \\ \boldsymbol{M}_{1,0} & \boldsymbol{M}_{1,1} & \cdots & \boldsymbol{M}_{1,n-1} \\ \vdots & \vdots & & \vdots \\ \boldsymbol{M}_{k-1,0} & \boldsymbol{M}_{k-1,1} & \cdots & \boldsymbol{M}_{k-1,n-1} \end{pmatrix} \end{array} \right.$$

where

$$\boldsymbol{M}_{i,j} \stackrel{\text{def}}{=} \begin{pmatrix} \mathbf{Tr}(\bar{b}_0 b^*_{j,0} m_{i,j}) & \mathbf{Tr}(\bar{b}_0 b^*_{j,1} m_{i,j}) & \dots & \mathbf{Tr}(\bar{b}_0 b^*_{j,m-1} m_{i,j}) \\ \mathbf{Tr}(\bar{b}_1 b^*_{j,0} m_{i,j}) & \mathbf{Tr}(\bar{b}_1 b^*_{j,1} m_{i,j}) & \dots & \mathbf{Tr}(\bar{b}_1 b^*_{j,m-1} m_{i,j}) \\ \vdots & \vdots & & \vdots \\ \mathbf{Tr}(\bar{b}_{m-1} b^*_{j,0} m_{i,j}) & \mathbf{Tr}(\bar{b}_{m-1} b^*_{j,1} m_{i,j}) & \dots & \mathbf{Tr}(\bar{b}_{m-1} b^*_{j,m-1} m_{i,j}) \end{pmatrix}.$$

With this definition, the properties of Proposition 7.16 still hold for various bases.

Remark 7.22. *Note that contrary to the expansion of codes, the expansion of a matrix depends on the choice of a basis $\bar{\mathcal{B}}$ for the vertical expansion. When considering the code spanned by an expansion matrix, different choices of $\bar{\mathcal{B}}$ yield the same code, so we will omit the vertical expansion base in the expansion matrix operator.*

7.1.3.4 Squeezing: the inverse of expansion

We can define an operation that performs the "inverse" of the expansion operator: given a basis, it reduces a block of m entries over $\mathbf{F}_q$ to one entry over $\mathbf{F}_{q^m}$. We call this operation *squeezing* as it reduces the length of the code.

Definition 7.23 (Squeezing Operator)**.** Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be a basis of $\mathbf{F}_{q^m}$. Let $\boldsymbol{x} = (x_{0,0}, \ldots, x_{0,m-1}, \ldots, x_{n-1,0}, \ldots, x_{n-1,m-1}) \in \mathbf{F}_q^{mn}$. We define the *squeezed vector* of $\boldsymbol{x}$ with respect to the basis $\mathcal{B}$ as

$$\mathbf{SqueezeVec}_{\mathcal{B}}(\boldsymbol{x}) \stackrel{\text{def}}{=} \left(\sum_{j=0}^{m-1} x_{0,j} b_j, \ldots, \sum_{j=0}^{m-1} x_{n-1,j} b_j \right) \in \mathbf{F}_{q^m}^n.$$

Let $\mathscr{C}$ be an $[m \times n, k]$–code over $\mathbf{F}_q$. We define the *squeezed code* of $\mathscr{C}$ with respect to the basis $\mathcal{B}$ as

$$\mathbf{SqueezeCode}_{\mathcal{B}}(\mathscr{C}) \stackrel{\text{def}}{=} \{\mathbf{SqueezeVec}_{\mathcal{B}}(\boldsymbol{c}) \,|\, \boldsymbol{c} \in \mathscr{C}\}.$$

Proposition 7.24. *Let $\mathscr{C}$ be an $[n, k]$ code over $\mathbf{F}_{q^m}$. Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be a basis of $\mathbf{F}_{q^m}$. Then the following equality holds.*

$$\mathbf{SqueezeCode}_{\mathcal{B}}(\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})) = \mathscr{C}.$$

Finally we can define squeezing over a matrix.

Definition 7.25 (Squeezing matrices)**.** Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be a basis of $\mathbf{F}_{q^m}$. Let $\boldsymbol{M} \in \mathbf{F}_q^{mk \times mn}$ denote an $mk \times mn$ matrix. Then $\mathbf{SqueezeMat}_{\mathcal{B}}(\boldsymbol{M}) \in \mathbf{F}_q^{mk \times n}$ denotes the matrix whose rows are obtained by squeezing each row of the matrix $\boldsymbol{M}$ over $\mathcal{B}$.

Remark 7.26. *Note that this matrix does not necessarily have full rank. In particular, if $\boldsymbol{M}$ is obtained by expanding a matrix of rank r over the basis $\mathcal{B}$, then $\mathbf{SqueezeMat}_{\mathcal{B}}(\boldsymbol{M})$ will be of rank r. It is also worth noting that for a matrix $\boldsymbol{M} \in \mathbf{F}_{q^m}^{k \times n}$, then $\mathbf{SqueezeMat}_{\mathcal{B}}(\mathbf{ExpMat}_{\mathcal{B}}(\boldsymbol{M}))$ is a $km \times n$ matrix and hence is **not** equal to $\boldsymbol{M}$ but generates the same code.*

Remark 7.27. *Similarly to the expansion operators, we can define the squeezing operators with a different basis for each block.*

7.1.3.5 Representation of subspace subcodes

Let $\mathscr{C}$ be a code of length n and dimension k over the field $\mathbf{F}_{q^m}$ and $\mathcal{S}$ denote an $\mathbf{F}_q$-subspace of $\mathbf{F}_{q^m}$ of dimension $\lambda \leqslant m$. Let $\mathcal{B}_{\mathcal{S}} = (b_0, \ldots, b_{\lambda-1}) \in \mathbf{F}_{q^m}^{\lambda}$ be

an $\mathbf{F}_q$-basis of $\mathcal{S}$. Then any vector $\boldsymbol{c} = (c_0, \ldots, c_{n-1}) \in \mathcal{S}^n$, *i.e.* whose entries are all in $\mathcal{S}$ can be expanded as

$$\mathbf{ExpVec}_{\mathcal{B}_\mathcal{S}}(\boldsymbol{c}) \stackrel{\text{def}}{=} (c_{0,0}, \ldots, c_{0,\lambda-1}, \ldots, c_{n-1,0}, \ldots, c_{n-1,\lambda-1}),$$

where the $c_{i,j}{}'s$ are the coefficients of the decomposition of c_i in the $\mathcal{B}_\mathcal{S}$.

Remark 7.28. *Note that the previous definition makes sense only for vectors in $\mathcal{S}^n$.*

Next, the subspace subcode $\mathscr{C}_{|\mathcal{S}}$ can be represented as

$$\mathbf{ExpCode}_{\mathcal{B}_\mathcal{S}}(\mathscr{C}_{|\mathcal{S}}) \stackrel{\text{def}}{=} \{\mathbf{ExpVec}_{\mathcal{B}_\mathcal{S}}(\boldsymbol{c}) \mid \boldsymbol{c} \in \mathscr{C}_{|\mathcal{S}}\}.$$

Here again, as noticed in Remark 7.28, the notion is well–defined only for codes with entries in $\mathcal{S}$.

It is important to stress that $\mathbf{ExpCode}_{\mathcal{B}_\mathcal{S}}(\mathscr{C}_{|\mathcal{S}})$ is exactly the subspace subcode represented as an $\mathbf{F}_q$-linear code since we can reconstruct $\mathscr{C}_{|\mathcal{S}}$ from it by applying the **SqueezeCode** operator (Proposition 7.24 adapted for an incomplete basis).

Similarly to Definition 7.18, a *block* refers to a set of the form $[\![i\lambda, (i+1)\lambda-1]\!]$. That is to say, a set of $\lambda = \dim \mathcal{S}$ consecutive indexes of the expanded code, corresponding to the decomposition of a single entry in $\mathcal{S}$ in the basis $\mathcal{B}_\mathcal{S}$.

7.1.4 An instantiation of McEliece with SSRS codes

Let us first present a generic encryption scheme based on subspace subcodes of GRS codes. This cryptosystem will be referred to as the *Subspace Subcode of Reed–Solomon* (SSRS) scheme. We will later prove that the cryptosystem of [KRW21] is a sub-instance of the SSRS scheme.

In Section 7.1.3 we explained that the same subspace subcode could be represented as an expanded code in different ways depending on the choice of the expansion basis. For all results present in the previous section, the choice of the expansion basis had no influence. In particular, we could easily write the descriptions in the special case where all subspaces (and all expansion basis) are the same, and this would generalise straightforwardly to the case of different subspaces (and different basis).

The idea of the SSRS cryptosystem is to use as public key the expanded code of a subspace subcode, while the subspaces at stake (and the expansion basis chosen for the representation) remains secret. The security of these cryptosystems relies on the fact that the expansion basis are hidden. In particular, different subspaces are used to expand each entry of the code. Therefore, in what follows it is important to use the notations with different subspaces and different basis.

Parameters. The cryptosystem is publicly parametrised by:

- q a prime power;
- m an integer;
- λ such that $0 < \lambda < m$;
- n, k such that $0 \leqslant k < n \leqslant q^m$ and $km > (m-\lambda)n$.

Key generation.

- Generate a uniformly random vector $\boldsymbol{x} \in \mathbf{F}_{q^m}^n$ with distinct entries.
- Choose n uniformly random λ–dimensional vector subspaces denoted $\mathcal{S}_0, \ldots, \mathcal{S}_{n-1} \subseteq \mathbf{F}_{q^m}$ with respective bases $\mathcal{B}_{\mathcal{S}_0}, \ldots, \mathcal{B}_{\mathcal{S}_{n-1}}$.
- Let $\boldsymbol{G}_{\text{pub}} \in \mathbf{F}_q^{(km-n(m-\lambda))\times \lambda n}$ denote a generator matrix of the code

$$\mathscr{C}_{\text{pub}} \stackrel{\text{def}}{=} \mathbf{ExpCode}_{(\mathcal{B}_{\mathcal{S}_0},\ldots,\mathcal{B}_{\mathcal{S}_{n-1}})}\left(\mathbf{RS}_k(\boldsymbol{x})_{|(\mathcal{S}_0,\ldots,\mathcal{S}_{n-1})}\right).$$

  If $\boldsymbol{G}_{\text{pub}}$ is not full-rank, abort and restart the process. See Section 7.2.2 for the practical computation on $\boldsymbol{G}_{\text{pub}}$.
- The public key is $\boldsymbol{G}_{\text{pub}}$ and the secret key is $(\boldsymbol{x}, \mathcal{B}_{\mathcal{S}_0}, \ldots, \mathcal{B}_{\mathcal{S}_{n-1}})$.

Lemma 7.29 (public key size)**.** *The public key is a matrix of size $m(n-k) \times \lambda n$ over $\mathbf{F}_q$. Only the systematic part is transmitted. Hence the public key size in bits is*

$$m(n-k)(\lambda n - m(n-k)) \log_2(q).$$

Encryption. Let $\boldsymbol{m} \in \mathbf{F}_q^{mk-(m-\lambda)n}$ be the plaintext. Denote

$$t \stackrel{\text{def}}{=} \lfloor \frac{n-k}{2} \rfloor.$$

Choose $\boldsymbol{e} \subseteq \mathbf{F}_q^{(m-\lambda)n}$ uniformly at random among vectors of $\mathbf{F}_q^{(m-\lambda)n}$ with exactly t non-zero blocks (see Definition 7.18).

Decryption. From $\boldsymbol{y} \in \mathbf{F}_q^{\lambda n}$, construct a vector $\boldsymbol{y}' \in \mathbf{F}_q^{mn}$ by completing each block of size λ with $m-\lambda$ entries set to zero. Denote

$$\boldsymbol{y}'' = \mathbf{SqueezeVec}_{(\mathcal{B}_i)_i}(\boldsymbol{y}').$$

According to the definition of $\boldsymbol{e}$, the vector $\boldsymbol{y}'' \in \mathbf{F}_{q^m}^n$ is at distance t of the code $\mathbf{RS}_k(\boldsymbol{x})$. Hence, by decoding, one computes the unique $\boldsymbol{c} \in \mathbf{RS}_k(\boldsymbol{x})$ at distance $\leqslant t$ from $\boldsymbol{y}''$ and the expansion of $\boldsymbol{c}$ yields $\boldsymbol{m}\boldsymbol{G}_{\text{pub}}$.

7.1.5 Further properties of the expansion operator

Now that we have seen that the subspace subcodes can be seen as shortened expanded codes, it is useful to study more thoroughly the properties of the expansion operator. More precisely, we study how this operator behaves with respect to other operations (especially those used in the key generation): puncturing/shortening, computing the dual, changing the expansion basis. In the next section, we will use these results to show that the *expanded generalised Reed-Solomon* codes presented in the XGRS cryptosystem [KRW21] are in fact a special instantiation of subspace subcodes of Reed–Solomon codes. We also consider the relation with the square product operation, as this is a natural distinguisher for GRS-based codes.

In this section, for the sake of clarity, all properties will be defined considering the same basis for each entry, but everything works exactly the same way if one considered expansion with a different basis for each entry, as different columns of $\mathbf{F}_{q^m}$ (or blocks of columns of $\mathbf{F}_q$ corresponding to the expansion of same column of $\mathbf{F}_{q^m}$) do not interact.

7.1.5.1 Subspace subcodes as shortening of expanded codes

This lemma explains how to construct the parity-check matrix of a subspace subcode from the parity-check matrix of the parent code. This result is important to perform computations over the subspace subcodes.

Lemma 7.30. *For integers n and $\lambda < m$, denote $\mathcal{J}(\lambda, m)$ the subset of $[\![0, mn-1]\!]$ consisting of the last $m-\lambda$ entries of each block of length m*

$$\mathcal{J}(\lambda, m) \stackrel{def}{=} \{im + j,\ i \in [\![0, n-1]\!],\ j \in [\![\lambda, m-1]\!]\}\,. \tag{7.4}$$

Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be an $\mathbf{F}_q$-basis of $\mathbf{F}_{q^m}$ such that $\mathcal{B}_{\mathcal{S}} = (b_0, \ldots, b_{\lambda-1})$ is a basis of $\mathcal{S}$. Then,

$$\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}}) = \mathbf{Short}_{\mathcal{J}(\lambda,m)}\left(\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})\right).$$

Moreover, let $\boldsymbol{H} \in \mathbf{F}_{q^m}^{k\times n}$ denote a parity–check matrix of $\mathscr{C}$. Complete the basis $\mathcal{B}_{\mathcal{S}} = (b_0, \ldots, b_{\lambda-1})$ with $m-\lambda$ additional elements $(b_\lambda, \ldots, b_{m-1}) \in \mathbf{F}_{q^m}^{m-\lambda}$ such that $\mathcal{B} = (b_0, \ldots, b_{m-1})$ forms an $\mathbf{F}_q$–basis of $\mathbf{F}_{q^m}$. Then, the code $\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})$ admits as parity-check matrix the matrix $\mathbf{Punct}_{\mathcal{J}(\lambda,m)}\left(\mathbf{ExpMat}_{\mathcal{B}^*}(\boldsymbol{H})\right)$.

Remark 7.31. *Proposition 7.2 follows directly from this result.*

Remark 7.32. *Of course, what precedes extends straightforwardly to various subspaces and bases.*

Thanks to this result, we can now write the commutative diagram of Figure 7.2.

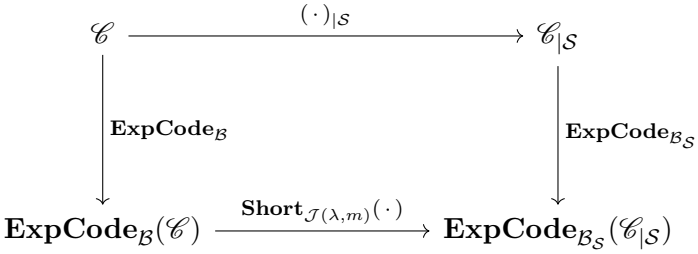


Figure 7.2: Expanding the subspace subcode is equivalent the shortening the expansion of the parent code (Lemma 7.30).

7.1.5.2 Puncturing and shortening

Lemma 7.33. *Let $\mathscr{C}$ be an $[n, k]$ code over $\mathbf{F}_{q^m}$. Let $\mathcal{L}$ denote a subset of $[\![0, n-1]\!]$. Then the following equalities hold.*

$$\mathbf{ExpCode}_{\mathcal{B}}(\mathbf{Punct}_{\mathcal{L}}(\mathscr{C})) = \mathbf{Punct}_{\mathcal{L}'}(\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})),$$

$$\mathbf{ExpCode}_{\mathcal{B}}(\mathbf{Short}_{\mathcal{L}}(\mathscr{C})) = \mathbf{Short}_{\mathcal{L}'}(\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})),$$

where $\mathcal{L}'$ denotes the set of all columns generated from expanding columns in $\mathcal{L}$, that is

$$\mathcal{L}' \stackrel{def}{=} \bigcup_{i \in \mathcal{L}} \{i + j, 0 \leqslant j < m\}.$$

Proof. The result is straightforward for puncturing. The expansion operation is independent for each column, hence puncturing a column before expanding is equivalent to puncturing the corresponding block of m columns. As for shortening, the shortening operation is the dual of puncturing operation, hence the result is a consequence of the next lemma. □

7.1.5.3 Dual code

Lemma 7.34 ([Wu11], Lemma 1)**.** *Let $\mathcal{B}$ be a basis and $\mathcal{B}^*$ denote the dual basis. For all $\boldsymbol{x}, \boldsymbol{y} \in \mathbf{F}_{q^m}^n$, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal, i.e. $\boldsymbol{x} \cdot \boldsymbol{y}^\intercal = 0$, then $\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x})$ and $\mathbf{ExpVec}_{\mathcal{B}^*}(\boldsymbol{y})$ are orthogonal*

$$\mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x}) \cdot (\mathbf{ExpVec}_{\mathcal{B}^*}(\boldsymbol{y}))^\intercal = 0.$$

Proof. Denote $\boldsymbol{x} = (x_0, \ldots, x_{n-1})$, $\boldsymbol{y} = (y_0, \ldots, y_{n-1})$, and for $i \in [\![0, n-1]\!]$, $x_i = \sum_{j=0}^{m-1} x_{i,j} b_j$ and $y_i = \sum_{j=0}^{m-1} y_{i,j} b_j^*$, where $\mathcal{B} = (b_j)$ and $\mathcal{B}^* = (b_j^*)$.

$$\begin{aligned}
0 &= \boldsymbol{x} \cdot \boldsymbol{y}^{\mathsf{T}} \\
&= \sum_{i=0}^{n-1} x_i y_i \\
&= \sum_{i=0}^{n-1} \left(\sum_{j=0}^{m-1} x_{i,j} b_j \right) \left(\sum_{j=0}^{m-1} y_{i,j} b_j^* \right) \\
&= \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_{i,j} y_{i,j} \\
&= \mathbf{ExpVec}_{\mathcal{B}}(\boldsymbol{x}) \cdot (\mathbf{ExpVec}_{\mathcal{B}^*}(\boldsymbol{y}))^{\mathsf{T}}.
\end{aligned}$$

□

Corollary 7.35. *Let $\mathscr{C}$ be an $[n,k]$ code over $\mathbf{F}_{q^m}$. Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be a basis of $\mathbf{F}_{q^m}$. Then the following equality holds.*

$$\mathbf{Dual}(\mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C})) = \mathbf{ExpCode}_{\mathcal{B}^*}(\mathbf{Dual}(\mathscr{C})),$$

where $\mathcal{B}^*$ denotes the dual basis of $\mathcal{B}$.

7.1.5.4 Changing the expansion basis

Lemma 7.36. *Let $\mathscr{C}$ be an $[n,k]$ code over $\mathbf{F}_{q^m}$. Let $\mathcal{B} = (b_0, \ldots, b_{m-1})$ be an $\mathbf{F}_q$–basis of $\mathbf{F}_{q^m}$. Let $\boldsymbol{Q} \in \mathbf{F}_q^{m \times m}$ denote an invertible $m \times m$ matrix. The following equality holds.*

$$\mathbf{ExpCode}_{\mathcal{B}\cdot(\boldsymbol{Q}^{-1})^{\mathsf{T}}}(\mathscr{C}) = \mathbf{ExpCode}_{\mathcal{B}}(\mathscr{C}) \cdot \begin{pmatrix} \boldsymbol{Q} & & \\ & \ddots & \\ & & \boldsymbol{Q} \end{pmatrix}.$$

Proof. Let $\boldsymbol{c}$ be a codeword of $\mathscr{C}$. We only focus on the first entry of $\boldsymbol{c}$. Denote $x \in \mathbf{F}_{q^m}$ this entry and $(x_0, \ldots, x_{m-1}) = \mathbf{ExpVec}_{\mathcal{B}}((x)) \in \mathbf{F}_q^m$. By definition, $x = \sum_{i=0}^{m-1} x_i b_i$. Let $\mathcal{D} = (d_0, \ldots, d_{m-1})$ be the basis $\mathcal{B} \cdot \left(\boldsymbol{Q}^{-1}\right)^{\mathsf{T}}$. For all $i \in [\![0, m-1]\!]$, we have $b_i = \sum_{j=0}^{m} d_j q_{i,j}$ where $\boldsymbol{Q} = (q_{i,j})_{0 \leqslant i,j < m}$. Replacing the b_i's by this formula, we obtain

$$x = \sum_i x_i \left(\sum_j d_j q_{i,j} \right) = \sum_j \left(\sum_i x_i q_{i,j} \right) d_j.$$

Therefore,

$$\mathbf{ExpVec}_{\mathcal{B}}(x) \cdot \boldsymbol{Q} = \mathbf{ExpVec}_{\mathcal{D}}(x).$$

This holds for any entry of any codeword $\boldsymbol{c} \in \mathscr{C}$. □

Lemma 7.37. *Let $\mathscr{C}$ be an $[n,k]$ code over $\mathbf{F}_{q^m}$. Let $(\mathcal{B}_i)_i$ be n bases of $\mathbf{F}_{q^m}$. Let $(\boldsymbol{Q}_i) \in (\mathbf{F}_q^{m\times m})^n$ denote n invertible $m \times m$ matrices. The following equality holds.*

$$\mathbf{ExpCode}_{(\mathcal{B}_i\cdot(\boldsymbol{Q}_i^{-1})^\intercal)_i}(\mathscr{C}) = \mathbf{ExpCode}_{\mathcal{B}_i}(\mathscr{C}) \cdot \begin{pmatrix} \boldsymbol{Q}_0 & & \\ & \ddots & \\ & & \boldsymbol{Q}_n \end{pmatrix}.$$

7.1.5.5 Scalar multiplication in $\mathbf{F}_{q^m}$

Lemma 7.38. *Let $\mathscr{C}$ be an $[n,k]$ code over $\mathbf{F}_{q^m}$. Let $(\mathcal{B}_i)_i$ be n basis of $\mathbf{F}_{q^m}$. Let $\boldsymbol{a} = (a_0,\dots,a_{n-1}) \in \mathbf{F}_{q^m}^n$ denote a vector of length n over $\mathbf{F}_{q^m}$. The following equality holds.*

$$\begin{aligned}\mathbf{ExpCode}_{(\mathcal{B}_i)_i}(\mathscr{C}) &= \mathbf{ExpCode}_{(a_i\mathcal{B}_i)_i}(\{(\boldsymbol{c}\star\boldsymbol{a}) \mid \boldsymbol{c}\in\mathscr{C}\}) \\ &= \mathbf{ExpCode}_{(a_i\mathcal{B}_i)_i}(\mathscr{C}\star\boldsymbol{a}).\end{aligned}$$

7.2 The XGRS cryptosystem

In this section, we define the *eXpanded Generalised Reed–Solomon* (XGRS) cryptosystem introduced in [KRW21] and explain why it is a sub–instance of the SSRS cryptosystem.

7.2.1 The cryptosystem

Remark 7.39. *An initial version of the XGRS cryptosystem, was submitted on ArXiv [KRW19]. In this version, the classical square-code distinguisher could be applied to the cryptosystem and lead to an attack. The authors changed the cryptosystem in order to avoid such attacks. Here we present the latest version of the XGRS proposal.*

Parameters. The cryptosystem is publicly parametrised by:

- q a prime power;
- m an integer;
- λ such that $2 \leqslant \lambda < m$;
- n, k such that $0 \leqslant k < n \leqslant q^m$ and $km > (m-\lambda)n$.

Remark 7.40. *As suggested by the parameters in Table 7.1, m is a small integer. The preprint version of the paper [KRW19] proposed to use $m = 2$ with a slightly modified key generation. The proposed parameters are now $m = 3$ and $m = 4$.*

q	m	λ	n	k	Public Key Size (kB)
13	3	2	1258	1031	579
7	4	2	1872	1666	844

Table 7.1: Parameters proposed for the XGRS scheme [KRW21]

Key Generation

- Generate uniformly random vectors $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbf{F}_{q^m}^n \times (\mathbf{F}_{q^m}^\times)^n$ such that $\boldsymbol{x}$ has distinct entries. Denote $\mathscr{C} = \mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ and let $\boldsymbol{H}_{\text{sec}}$ be a parity-check matrix of $\mathscr{C}$.

- Choose γ, a primitive element of $\mathbf{F}_{q^m}/\mathbf{F}_q$, *i.e.* a generator of the field extension. We consider the basis $\mathcal{B}_\gamma = (1, \gamma, \ldots, \gamma^{m-1})$ of $\mathbf{F}_{q^m}$.

- Set $\boldsymbol{H} \stackrel{\text{def}}{=} \mathbf{ExpMat}_{\mathcal{B}_\gamma^*}(\boldsymbol{H}_{\text{sec}}) \in \mathbf{F}_q^{m(n-k)\times mn}$ which is a parity-check matrix $\mathbf{ExpCode}_{\mathcal{B}_\gamma}(\mathscr{C})$.

- For any $i \in [\![0, n-1]\!]$, choose $\mathcal{L}_i$ a random subset of $[\![(i-1)m, im-1]\!]$ of size $|\mathcal{L}_i| = m - \lambda$. Set $\mathcal{L} = \cup_i \mathcal{L}_i$.

- Set $\boldsymbol{H}_{\mathcal{L}} \stackrel{\text{def}}{=} \mathbf{Punct}_{\mathcal{L}}(\boldsymbol{H}) \in \mathbf{F}_q^{m(n-k)\times \lambda n}$.

- For any $i \in [\![0, n-1]\!]$, choose $\boldsymbol{Q}_i$ a random $\lambda \times \lambda$ invertible matrix. Denote by $\boldsymbol{Q}$ the block-diagonal matrix having $\boldsymbol{Q}_0, \ldots, \boldsymbol{Q}_{n-1}$ as diagonal blocks.

- Denote by $\boldsymbol{S}$ the invertible matrix of $\mathbf{F}_q$ such that $\boldsymbol{S} \cdot \boldsymbol{H}_{\mathcal{L}} \cdot \boldsymbol{Q}$ is in systematic form.

- Set $\boldsymbol{H}_{\text{pub}} \stackrel{\text{def}}{=} \boldsymbol{S} \cdot \boldsymbol{H}_{\mathcal{L}} \cdot \boldsymbol{Q}$.

- The public key is $\boldsymbol{H}_{\text{pub}}$, the private key is $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{Q}, \mathcal{L}, \gamma)$.

Remark 7.41. *Compared to the cryptosystem presented in [KRW21], we omitted the block permutation. Indeed, applying a block permutation after expanding is equivalent to applying the permutation before the expansion and then expanding. As we start with a GRS code chosen uniformly at random, applying a permutation on the columns does not change the probability distribution of the public keys.*

Encryption. Recall that $t \stackrel{\text{def}}{=} \lfloor \frac{n-k}{2} \rfloor$ the error–correcting capacity of a GRS code of length n and dimension k. The message is encoded as a vector $\boldsymbol{y} \in \mathbf{F}_q^{\lambda n}$

whose support is included in t blocks of length λ, *i.e.* there exist positions $i_0, \ldots, i_{t-1} \in [\![0, n-1]\!]$, such that

$$\mathbf{Support}(\boldsymbol{y}) \subseteq \bigcup_{0 \leqslant \ell \leqslant t-1} [\![\lambda(i_\ell - 1), \lambda i_\ell - 1]\!].$$

The ciphertext is then defined as $\boldsymbol{c}^\intercal = \boldsymbol{H}_{\text{pub}} \cdot \boldsymbol{y}^\intercal$.

Decryption. In order to decrypt the ciphertext, a user knowing the private key should:

- generate $\boldsymbol{H}_{\text{sec}}$ from $\boldsymbol{x}$ and $\boldsymbol{y}$.
- compute $\boldsymbol{c}' = \boldsymbol{c} \cdot \boldsymbol{S}^{-1\intercal}$;
- compute $\boldsymbol{c}'' = \mathbf{SqueezeVec}_{\mathcal{B}_\gamma}(\boldsymbol{c}')$;
- find $\boldsymbol{y}'' \in \mathbf{F}_{q^m}^n$ of weight $|\boldsymbol{y}''| \leqslant t$ such that $\boldsymbol{c}''^\intercal = \boldsymbol{H}_{\text{sec}} \boldsymbol{y}''^\intercal$ (*i.e.* decode in $\mathscr{C}$);
- compute $\boldsymbol{y}' = \mathbf{Punct}_{\mathcal{L}} \left(\mathbf{ExpVec}_{\mathcal{B}_\gamma}(\boldsymbol{y}'') \right)$;
- finally recover $\boldsymbol{y} = \boldsymbol{y}' \cdot \left(\boldsymbol{Q}^{-1} \right)^\intercal$.

7.2.2 XGRS is a instance of SSRS

Now we will prove that the XGRS cryptosystem is in fact a sub-instance of the SSRS cryptosystem presented in Section 7.1.4. Although at this point it may seem obvious from the results in the previous section, we recall that the original formulation of the XGRS cryptosystem [KRW21] does not mention the concept of subspace subcodes, only expanded codes. Moreover, the basis used in the XGRS encryption is the same for each entry, whereas different subspace and basis are used in the SSRS scheme. The following statement explains exactly how XGRS relates to SSRS.

Proposition 7.42. *The XGRS scheme with secret key $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{Q}, \mathcal{L}, \gamma)$ is equivalent to the SSRS scheme with secret key $(\boldsymbol{x}, \mathcal{S}_0, \ldots, \mathcal{S}_{n-1})$ where the subspaces $\mathcal{S}_i$ are defined as follows.*

- *Let $\mathcal{B}_i^{(0)} \stackrel{def}{=} \mathbf{Punct}_{\mathcal{L}_i}(\mathcal{B}_\gamma) \in \mathbf{F}_{q^m}^\lambda$ where $\mathcal{L}_i \stackrel{def}{=} \{j - mi, \forall j \in \mathcal{L} \cap [\![im, (i+1)m - 1]\!]\}$.*
- *Set $\mathcal{B}_i^{(1)} \stackrel{def}{=} y_i^{-1} \mathcal{B}_i^{(0)} \cdot (\boldsymbol{Q}_i^{-1})^\intercal$.*
- *$\mathcal{S}_i$ is the subspace of $\mathbf{F}_{q^m}$ spanned by the elements of $\mathcal{B}_i^{(1)}$.*

Proof. Let $\mathscr{C}_{\text{pub}}$ denote the public code of an instance of the XGRS scheme with private key $(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{Q}, \mathcal{L}, \gamma)$, *i.e.* $\mathscr{C}_{\text{pub}}$ is the code over $\mathbf{F}_q$ that admits the public key $\boldsymbol{H}_{\text{pub}}$ as parity-check matrix. We have

$$\begin{aligned}\mathscr{C}_{\text{pub}} &= \mathbf{Dual}\left(\left\langle \boldsymbol{H}_{\text{pub}} \right\rangle_{\mathbf{F}_q}\right) \\ &= \mathbf{Dual}\left(\langle \boldsymbol{H}_{\mathcal{L}} \cdot \boldsymbol{Q} \rangle_{\mathbf{F}_q}\right).\end{aligned}$$

Let us define $\boldsymbol{Q}^{(1)} \stackrel{\text{def}}{=} \left(\boldsymbol{Q}^{-1}\right)^{\mathsf{T}}$. This is still a block-diagonal matrix composed of n blocks of size $\lambda \times \lambda$. We can rewrite this

$$\mathscr{C}_{\text{pub}} = \mathbf{Dual}\left(\langle \boldsymbol{H}_{\mathcal{L}} \rangle_{\mathbf{F}_q}\right) \cdot \boldsymbol{Q}^{(1)}.$$

We can replace $\boldsymbol{H}_{\mathcal{L}}$ by its definition: $\mathbf{Punct}_{\mathcal{L}}\left(\mathbf{ExpMat}_{\mathcal{B}_\gamma^*}(\boldsymbol{H}_{\text{sec}})\right)$. Next, we can swap the **Dual** and **Punct** operators according to Proposition 5.22:

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{L}}\left(\mathbf{Dual}\left(\mathbf{ExpCode}_{\mathcal{B}_\gamma^*}\left(\langle \boldsymbol{H}_{\text{sec}} \rangle_{\mathbf{F}_q}\right)\right)\right) \cdot \boldsymbol{Q}^{(1)}.$$

We can then swap the **Dual** and **ExpCode** operators according to Corollary 7.35.

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{L}}\left(\mathbf{ExpCode}_{\mathcal{B}_\gamma}\left(\mathbf{Dual}\left(\langle \boldsymbol{H}_{\text{sec}} \rangle_{\mathbf{F}_q}\right)\right)\right) \cdot \boldsymbol{Q}^{(1)}.$$

Let $\boldsymbol{G}_{\text{sec}}$ be a generator matrix of the secret code $\mathbf{Dual}\left(\langle \boldsymbol{H}_{\text{sec}} \rangle_{\mathbf{F}_q}\right)$, *i.e.* a generator matrix of the code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$. We have

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{L}}\left(\mathbf{ExpCode}_{\mathcal{B}_\gamma}\left(\langle \boldsymbol{G}_{\text{sec}} \rangle_{\mathbf{F}_q}\right)\right) \cdot \boldsymbol{Q}^{(1)}.$$

Let us denote $\boldsymbol{Q}^{(2)}$ the block-diagonal matrix obtained by replacing each $\lambda \times \lambda$ matrix of $\boldsymbol{Q}^{(1)}$ by the $m \times m$ matrix obtained by inserting "an identity row/column" at the positions corresponding to $\mathcal{L}$. For instance, if $m = 3, \lambda = 2$ and the first element of $\mathcal{L}$ equals 1, which means that the column 1 is shortened, we add a column and a row in the middle of $\boldsymbol{Q}_0^{(1)}$, *i.e.*

$$\text{if} \qquad \boldsymbol{Q}_0^{(1)} = \begin{pmatrix} q_{00} & q_{01} \\ q_{10} & q_{11} \end{pmatrix}, \qquad \text{then} \qquad \boldsymbol{Q}_0^{(2)} = \begin{pmatrix} q_{00} & 0 & q_{01} \\ 0 & 1 & 0 \\ q_{10} & 0 & q_{11} \end{pmatrix}.$$

Hence, we can write

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{L}}\left(\left\langle \mathbf{ExpMat}_{\mathcal{B}_\gamma}(\boldsymbol{G}_{\text{sec}}) \cdot \boldsymbol{Q}^{(2)} \right\rangle_{\mathbf{F}_q}\right).$$

We define $\boldsymbol{Q}_i^{(3)}$ as the matrix obtained from $\boldsymbol{Q}_i^{(2)}$ by permuting the columns so that the inserted columns are the $m-\lambda$ rightmost ones. For instance in the previous example, we would have

$$\boldsymbol{Q}_0^{(3)} = \begin{pmatrix} q_{00} & q_{01} & 0 \\ 0 & 0 & 1 \\ q_{10} & q_{11} & 0 \end{pmatrix}.$$

Therefore, $\boldsymbol{Q}^{(3)} = \boldsymbol{Q}^{(2)}\boldsymbol{P}$ where $\boldsymbol{P}$ is a block–diagonal matrix whose diagonal blocks are $m \times m$ permutations matrices. Then, we replace $\mathcal{L}$ by the set $\mathcal{J}(\lambda, m) = \{mi + j \mid 0 \leqslant i < n,\ \lambda \leqslant j < m\}$. Hence, we get

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{J}(\lambda,m)}\left(\left\langle \mathbf{ExpMat}_{\mathcal{B}_\gamma}(\boldsymbol{G}_{\text{sec}}) \cdot \boldsymbol{Q}^{(3)} \right\rangle_{\mathbf{F}_q}\right).$$

We can apply the basis change explained in Lemma 7.36.

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{J}(\lambda,m)}\left(\left\langle \mathbf{ExpMat}_{(\mathcal{B}'_i)_i}(\boldsymbol{G}_{\text{sec}}) \right\rangle_{\mathbf{F}_q}\right),$$

where $\mathcal{B}'_i \stackrel{\text{def}}{=} \mathcal{B}_\gamma \cdot \left((\boldsymbol{Q}_i^{(3)})^{-1}\right)^{\intercal}$ for all $i \in [\![0, n-1]\!]$. Finally, we apply Corollary 7.8 and Lemma 7.38 to replace the code $\mathbf{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ by $\mathbf{RS}_k(\boldsymbol{x})$. Hence,

$$\mathscr{C}_{\text{pub}} = \mathbf{Short}_{\mathcal{J}(\lambda,m)}\left(\left\langle \mathbf{ExpMat}_{(\mathcal{B}_i)_i}(\boldsymbol{G}'_{\text{sec}}) \right\rangle_{\mathbf{F}_q}\right),$$

where $\boldsymbol{G}'_{\text{sec}}$ is a generator matrix of $\mathbf{RS}_k(\boldsymbol{x})$ and $\mathcal{B}_i \stackrel{\text{def}}{=} y_i^{-1}\mathcal{B}'_i$ for all $i \in [\![0, n-1]\!]$. In other words, $\mathscr{C}_{\text{pub}} = \mathbf{RS}_k(\boldsymbol{x})_{|(\mathcal{S}_0,\ldots,\mathcal{S}_{n-1})}$, where $\mathcal{S}_i$ is the subspace spanned by the λ first elements of $\mathcal{B}_i$. This is indeed an instance of the SSRS cryptosystem. □

7.3 Twisted-square code and distinguisher

In this section, our goal is to adapt results such as the square code distinguisher presented in Chapter 5 to the case of subspace subcodes. Intuitively, we would like to see if the dimension of the square code of a subspace subcode sometimes differs from the dimension of the square code of the subspace subcode of a random code.

But because subspace subcodes (defined as codes over $\mathbf{F}_{q^m}$) are not linear codes, the only codes that we can manipulate are the expanded subspace subcode which are linear code over $\mathbf{F}_q$. Therefore, we need to define an operation to be performed over the expanded code that mimics the effects of the star-product operation applied to the parent code. We call this operation the *twisted*

star-product and we use it to define the *twisted square code* of an expanded subspace subcode. Note that this notion appears implicitly in Randriambololona's work on asymptotically good square codes [Ran13].

We point out that in the case of subspace subcodes, it is particularly interesting to compute the product of codes. Indeed, given a code whose components are restricted to a subspace $\mathcal{S}$ of dimension λ, the components of the products of two codewords lie in the subspace $\mathcal{S}^2$ which is typically of dimension $\binom{\lambda+1}{2}$. Hence we increased the size of the subspace. When λ is large enough, $\mathcal{S}^2$ can span the whole space $\mathbf{F}_{q^m}$. This can be particularly useful in a cryptanalysis because we somehow gain access to dimensions of the parent code that are not part of the subspace subcode. The smallest such example is for $m = 3, \lambda = 2$. We will start by explaining our definitions in this special setting and will provide a general definition of twisted-square codes later.

In the second part of the section, we will compute the typical dimension of the twisted square code of a subspace subcode of a Reed–Solomon code (as in the SSRS scheme) and compare it to the typical dimension of the twisted square code of a subspace subcode of a random code with the same parameters. We will see that this provides us with a distinguisher and how applying the shortening operation can extend the range of this distinguisher.

For the sake of simplicity, all the results of this section are stated using the same subspace and expansion basis for all blocks but they can be straightforwardly generalised to the case of various subspaces and expansion bases.

7.3.1 The twisted square product

The first thing we would like to do is to extend the commutative diagram of Figure 7.2 to include the squaring operation and its equivalent on the expanded code. We would like to obtain something that would look like Figure 7.3.

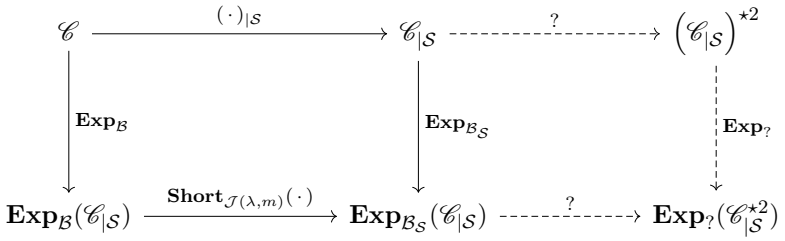


Figure 7.3: Expanding the squaring operation to subspace subcodes. Here, **Exp** stands for **ExpCode**.

Several things are to be defined in order to obtain such a result. First, we should specify the meaning of the squaring operation applied on the subspace subcode. Indeed, in the definition of the squaring operation (see Notation 5.10 and Remark 5.11), the product of two codes is defined as the space generated by all the component-wise products of codewords, spanned over the basefield of the code. It is important to recall that the subspace subcode $\mathscr{C}_{|\mathcal{S}}$ is not an $\mathbf{F}_{q^m}$-linear code like $\mathscr{C}$. Therefore, $\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}$ is the span over $\mathbf{F}_q$ of the star-products of codewords of $\mathscr{C}_{|\mathcal{S}}$. We will denote it $\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q}$.

This clarifies the meaning of the top-right arrow. Now, we have find the proper operation for the bottom-right arrow and the proper expansion basis for the right arrow to obtain a proper commutative diagram.

7.3.1.1 The case $m = 3, \lambda = 2$

Let us first focus on the smallest non-trivial case $m = 3, \lambda = 2$. We will explain *a posteriori* why this special case carries a lot of interesting properties that simplify the reasoning and the notations. Note that this case is not just a simple toy example as it corresponds to the parameters of the XGRS cryptosystem.

Let us introduce a definition that is needed in the sequel.

Definition 7.43. Let $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ be an $\mathbf{F}_q$–vector space, we define the square subspace

$$\mathcal{S}^2 \stackrel{\text{def}}{=} \langle\, ab \mid a, b \in \mathcal{S} \,\rangle_{\mathbf{F}_q}.$$

Lemma 7.44. *Let $\mathcal{S}$ be a subspace of $\mathbf{F}_{q^m}$ of dimension 2. Let $\mathcal{B}_{\mathcal{S}} = (\gamma_0, \gamma_1)$ be a basis of $\mathcal{S}$. Let $a, b \in \mathcal{S}$ such that*

$$\mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}((a)) = (a_0, a_1) \quad \text{and} \quad \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}((b)) = (b_0, b_1).$$

Then,

$$\mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}^2}}((ab)) = (a_0b_0, a_0b_1 + a_1b_0, a_1b_1),$$

where $\mathcal{B}_{\mathcal{S}^2} = (\gamma_0^2, \gamma_0\gamma_1, \gamma_1^2)$.

Note that $\mathcal{B}_{\mathcal{S}^2}$ is a basis of $\mathbf{F}_{q^m}$. Indeed, we have the following property.

Proposition 7.45. *For $m = 3$ and $\lambda = 2$, for any subspace $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ of dimension λ, we have $\mathcal{S}^2 = \mathbf{F}_{q^m}$.*

Proof. Let (γ_0, γ_1) be a basis of $\mathcal{S}$, if $\gamma_0^2, \gamma_0\gamma_1$ and γ_1^2 were not $\mathbf{F}_q$-independent, denoting $\zeta \stackrel{\text{def}}{=} \gamma_1/\gamma_0$, then $1, \zeta$ and ζ^2 would not be $\mathbf{F}_q$-independent either. Hence ζ would have degree $\leqslant 2$ over $\mathbf{F}_q$. But by definition $\zeta \notin \mathbf{F}_q$. □

This motivates the following definition.

Definition 7.46 (Twisted product)**.** Let $\boldsymbol{a}$ and $\boldsymbol{b}$ in $\mathbf{F}_q^{2n}$ whose components are denoted

$$\boldsymbol{a} = (a_0^{(0)}, a_0^{(1)}, a_1^{(0)}, a_1^{(1)}, \ldots, a_{n-1}^{(0)}, a_{n-1}^{(1)});$$
$$\boldsymbol{b} = (b_0^{(0)}, b_0^{(1)}, b_1^{(0)}, b_1^{(1)}, \ldots, b_{n-1}^{(0)}, b_{n-1}^{(1)}).$$

We define the *twisted product* of $\boldsymbol{a}$ and $\boldsymbol{b}$ as

$$\boldsymbol{a} \tilde{\star} \boldsymbol{b} \stackrel{\text{def}}{=} (a_i^{(0)} b_i^{(0)}, a_i^{(0)} b_i^{(1)} + a_i^{(1)} b_i^{(0)}, a_i^{(1)} b_i^{(1)})_{0 \leqslant i \leqslant n-1} \in \mathbf{F}_q^{3n}.$$

This definition extends to the product of codes, where the *twisted product* of two codes $\mathscr{A}$ and $\mathscr{B} \subseteq \mathbf{F}_q^{2n}$ is defined as

$$\mathscr{A} \tilde{\star} \mathscr{B} \stackrel{\text{def}}{=} \langle\, \boldsymbol{a} \tilde{\star} \boldsymbol{b} \mid \boldsymbol{a} \in \mathscr{A},\ \boldsymbol{b} \in \mathscr{B} \,\rangle_{\mathbf{F}_q}.$$

In particular, $\mathscr{A}^{\tilde{\star}2}$ denotes the *twisted square code* of a code $\mathscr{A}$: $\mathscr{A}^{\tilde{\star}2} \stackrel{\text{def}}{=} \mathscr{A} \tilde{\star} \mathscr{A}$.

With this definition, we can rewrite Lemma 7.44 for vectors in the following way.

Lemma 7.47. *Let $\mathcal{S}$ be a subspace of $\mathbf{F}_{q^m}$ of dimension 2. Let $\mathcal{B}_{\mathcal{S}} = (\gamma_0, \gamma_1)$ be a basis of $\mathcal{S}$. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbf{F}_{q^m}^n$ such that all their entries lie in $\mathcal{S}$. Then,*

$$\mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}(\boldsymbol{a}) \tilde{\star} \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}(\boldsymbol{b}) = \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}^2}}(\boldsymbol{a} \star \boldsymbol{b}), \tag{7.5}$$

where $\mathcal{B}_{\mathcal{S}^2} = (\gamma_0^2, \gamma_0\gamma_1, \gamma_1^2)$.

This results shows that the definition of the twisted star-product reaches our goal stated at the beginning of this section: it reflects on the expanded code the effects of the star-product operation applied to the parent code.

We can extend this result to codes and obtain the following theorem (still for $m = 3, \lambda = 2$).

Theorem 7.48. *Let $\mathscr{C}$ be an $[n, k]$ code over $\mathbf{F}_{q^m}$ and $\mathcal{S}$ a subspace of $\mathbf{F}_{q^m}$ of dimension λ. Let $\mathcal{B}_{\mathcal{S}}$ be an $\mathbf{F}_q$–basis of $\mathcal{S}$ such that $\mathcal{B}_{\mathcal{S}} = (\gamma_0, \gamma_1)$. Then,*

$$\left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})\right)^{\tilde{\star}2} = \mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}^2}}\left(\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q}\right), \tag{7.6}$$

where $\mathcal{B}_{\mathcal{S}^2} = (\gamma_0^2, \gamma_0\gamma_1, \gamma_1^2)$. This results generalises straightforwardly to an expansion over various subspaces.

Proof. This is a direct consequence of Lemma 7.47 by definition of the **ExpCode** operator and by $\mathbf{F}_q$-linearity of **ExpVec**. □

There are two important remarks concerning this results.

1. Because $\mathcal{S}^2 = \mathbf{F}_{q^m}$ for $m = 3, \lambda = 2$, on the right-hand side of equation (7.6), $\mathcal{B}_{\mathcal{S}^2}$ is a full basis of $\mathbf{F}_{q^m}$.

2. Given a generator matrix of the expanded subspace subcode, one can compute the generator matrix of the expanded square code by applying the twisted-square operation. This operation is independent of the choice of expansion basis, hence one can perform the computation even if one does not know which basis were used for the expansion. This will be important for the cryptanalysis, we will come back to this point later.

We now have a well-defined commutative diagram, as illustrated in Figure 7.4.

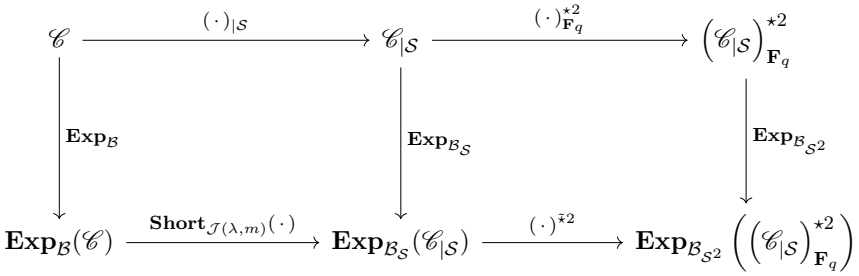


Figure 7.4: Expanding the squaring operation to subspace subcodes (case $m = 3, \lambda = 2$). Here, **Exp** stands for **ExpCode**.

7.3.1.2 General definition of the twisted square code

In what follows we will be interested in generalising this result. Indeed, the case $m = 3, \lambda = 2$ has two special properties.

1. We have seen in Proposition 7.45 that in this special case, we always have $\mathcal{S}^2 = \mathbf{F}_{q^m}$. The consequence of this is that we obtain a code expanded over a full basis of $\mathbf{F}_{q^m}$. But in general, $\mathcal{S}^2 \subseteq \mathbf{F}_{q^m}$ but the inclusion is *a priori* strict. **In what follows, we will restrict our analysis to the case $\mathcal{S}^2 = \mathbf{F}_{q^m}$.** Note that this restriction excludes the case of subfield-subcodes.

2. When $\mathcal{S}$ is a subspaces of $\mathbf{F}_{q^m}$ of dimension λ, the subspace $\mathcal{S}^2$ is of dimension $\min(\binom{\lambda+1}{2}, m)$. The property $\mathcal{S}^2 = \mathbf{F}_{q^m}$ implies that $\binom{\lambda+1}{2} \geqslant m$. In the previous case ($m = 3, \lambda = 2$) we were in the special case where

$\binom{\lambda+1}{2} = m$. Hence, $\mathcal{B}_{\mathcal{S}^2}$ (all the combinations of two elements of $\mathcal{B}_{\mathcal{S}}$) provides a basis of $\mathcal{S}^2 = \mathbf{F}_{q^m}$. For other parameters (m, λ) such that $\binom{\lambda+1}{2} = m$, Theorem 7.48 generalises straightforwardly. But for other values of (m, λ), we may have $\binom{\lambda+1}{2} > m$. In such a case, the set $\mathcal{B}_{\mathcal{S}^2}$ will not be a basis of $\mathcal{S}$, only a generating set. Therefore we need to keep only m elements of $\mathcal{B}_{\mathcal{S}^2}$. This situation leads to some cumbersome notations to generalise the definitions and results properly, but the general spirit is exactly the same.

For arbitrary integer $m \geqslant \lambda \geqslant 2$, let us have the following definition.

Definition 7.49 (Twisted square product, general case)**.** Let $\boldsymbol{a}$ and $\boldsymbol{b}$ in $\mathbf{F}_q^{\lambda n}$ whose components are denoted

$$\boldsymbol{a} = (a_{0,0}, \ldots, a_{0,\lambda-1}, a_{1,0}, \ldots, a_{1,\lambda-1}, \ldots, a_{n-1,0}, \ldots, a_{n-1,\lambda-1}),$$
$$\boldsymbol{b} = (b_{0,0}, \ldots, b_{0,\lambda-1}, b_{1,0}, \ldots, b_{1,\lambda-1}, \ldots, b_{n-1,0}, \ldots, b_{n-1,\lambda-1}).$$

We define the *twisted product* $\boldsymbol{a} \tilde{\star} \boldsymbol{b} \in \mathbf{F}_q^{\binom{\lambda+1}{2} n}$ of $\boldsymbol{a}$ and $\boldsymbol{b}$ such that for any $i \in [\![0, n-1]\!]$ and for r, s such that $0 \leqslant r \leqslant s \leqslant \lambda - 1$,

$$(\boldsymbol{a} \tilde{\star} \boldsymbol{b})_{i\binom{\lambda+1}{2}+\binom{s+1}{2}+r} \stackrel{\text{def}}{=} \begin{cases} a_{i,r}b_{i,s} + a_{i,s}b_{i,r} & \text{if } r < s \\ a_{i,r}b_{i,r} & \text{if } r = s. \end{cases}$$

This definition extends to the product of codes, where the *twisted product* of two codes $\mathscr{A}$ and $\mathscr{B} \subseteq \mathbf{F}_q^{\lambda n}$ is defined as

$$\mathscr{A} \tilde{\star} \mathscr{B} \stackrel{\text{def}}{=} \langle\, \boldsymbol{a} \tilde{\star} \boldsymbol{b} \mid \boldsymbol{a} \in \mathscr{A},\ \boldsymbol{b} \in \mathscr{B} \,\rangle_{\mathbf{F}_q}.$$

In particular, $\mathscr{A}^{\tilde{\star}2}$ denotes the *twisted square code* of a code $\mathscr{A}$: $\mathscr{A}^{\tilde{\star}2} \stackrel{\text{def}}{=} \mathscr{A} \tilde{\star} \mathscr{A}$.

Now, in order to generalise Theorem 7.48, we need to deal with the case $\binom{\lambda+1}{2} > m$ by shortening the redundant component. This yields the following statement.

Lemma 7.50. *Let $\mathcal{S}$ be a subspace of $\mathbf{F}_{q^m}$ of dimension λ such that $\mathcal{S}^2 = \mathbf{F}_{q^m}$. Let $\mathcal{B}_{\mathcal{S}} = (\gamma_0, \ldots, \gamma_{\lambda-1})$ be an $\mathbf{F}_q$-basis of $\mathcal{S}$. Let $\mathcal{B}_{\mathcal{S}^2}$ denote the first m elements of $(\gamma_0^2, \gamma_0\gamma_1, \ldots, \gamma_0\gamma_i, \gamma_1\gamma_i, \ldots, \gamma_i^2, \ldots, \gamma_{\lambda-1}^2)$. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbf{F}_{q^m}^n$ whose entries all lie in $\mathcal{S}$. Denote $\boldsymbol{c}$ the vector of length $\binom{\lambda+1}{2}n$ over $\mathbf{F}_q$ defined as*

$$\boldsymbol{c} \stackrel{def}{=} \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}(\boldsymbol{a}) \tilde{\star} \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}}}(\boldsymbol{b}).$$

Let $\mathcal{K}(\lambda, m)$ denote the set $\mathcal{J}\left(m, \binom{\lambda+1}{2}\right)$, i.e.

$$\mathcal{K}(\lambda, m) \stackrel{def}{=} \left\{ \binom{\lambda+1}{2} i + j, i \in [\![0, n-1]\!], j \in [\![m, \binom{\lambda+1}{2} - 1]\!] \right\}. \tag{7.7}$$

If

(*i*) *for any* $i \in \mathcal{K}(\lambda, m)$*, the* i*-th entry of* $\boldsymbol{c}$ *is zero,*

(*ii*) $\mathcal{B}_{\mathcal{S}^2}$ *is a basis of* $\mathbf{F}_{q^m}$*;*

then,

$$\mathbf{Punct}_{\mathcal{K}(\lambda,m)}(\boldsymbol{c}) = \mathbf{ExpVec}_{\mathcal{B}_{\mathcal{S}^2}}(\boldsymbol{a} \star \boldsymbol{b}). \tag{7.8}$$

Proof. Let $\boldsymbol{c}$ be defined as in the statement. We want to prove that

$$\mathbf{SqueezeVec}_{\mathcal{B}_{\mathcal{S}^2}}(\mathbf{Punct}_{\mathcal{K}(\lambda,m)}(\boldsymbol{c})) = \boldsymbol{a} \star \boldsymbol{b}.$$

This is equivalent to Equation (7.8) because $\mathcal{B}_{\mathcal{S}^2}$ is a basis of $\mathbf{F}_{q^m}$. Without loss of generality, we only need to focus on the block corresponding to the first entry in $\mathbf{F}_{q^m}$.

Let $(a_0, \dots, a_{\lambda-1})$ and $(b_0, \dots, b_{\lambda-1})$ denote the decomposition of the first entries of $\boldsymbol{a}$ (*resp.* $\boldsymbol{b}$) over $\mathcal{B}_{\mathcal{S}}$. The first entry of $\boldsymbol{a} \star \boldsymbol{b}$ is

$$\left(\sum_i a_i\gamma_i\right)\left(\sum_j a_j\gamma_j\right) = \sum_{0 \leqslant i \leqslant j < \lambda} c_{i,j}\gamma_i\gamma_j,$$

where the coefficients $c_{i,j}$ match exactly the definition of the twisted square product, hence correspond to the entries of $\boldsymbol{c}$.

Let $\mathcal{B}_{\text{full}}$ denote the family $(\gamma_0^2, \gamma_0\gamma_1, \dots, \gamma_0\gamma_i, \gamma_1\gamma_i, \dots, \gamma_i^2, \dots, \gamma_{\lambda-1}^2)$. The last entries of each block of $\boldsymbol{c}$ are equal to zero. This corresponds exactly to the elements of $\mathcal{B}_{\text{full}}$ that are not in $\mathcal{B}_{\mathcal{S}^2}$. We therefore have

$$\mathbf{SqueezeVec}_{\mathcal{B}_{\mathcal{S}^2}}(\mathbf{Punct}_{\mathcal{K}(\lambda,m)}(\boldsymbol{c})) = \mathbf{SqueezeVec}_{\mathcal{B}_{\text{full}}}(\boldsymbol{c}) = \boldsymbol{a} \star \boldsymbol{b}.$$

□

Remark 7.51. *Note that it is an arbitrary choice to define* $\mathcal{B}_{\mathcal{S}^2}$ *as the first* m *elements of* $\mathcal{B}_{\text{full}}$ *(which is of size* $\binom{\lambda+1}{2}$*). We could choose any subset of size* m*. Especially, if the set* $\mathcal{B}_{\mathcal{S}^2}$ *obtained is not a basis of* $\mathbf{F}_{q^m}$*, one could try with a different subset until a basis is found. The definition of* $\mathcal{K}(\lambda, m)$ *should be adapted accordingly. Because* $\mathcal{B}_{\text{full}}$ *is a generating set of* $\mathbf{F}_{q^m}$*, there always exists a subset of size* m *that is a basis. Hence, condition* (ii) *can always be matched with a good choice of subset.*

This leads to the following main statement, which is the generalisation of Theorem 7.48.

Theorem 7.52. *Let* $\mathscr{C}$ *be an* $[n, k]$ *code over* $\mathbf{F}_{q^m}$ *and* $\mathcal{S}$ *a subspace of* $\mathbf{F}_{q^m}$ *of dimension* λ *such that* $\mathcal{S}^2 = \mathbf{F}_{q^m}$*. Let* $\mathcal{B}_{\mathcal{S}}$ *be an* $\mathbf{F}_q$*–basis of* $\mathcal{S}$ *such that* $\mathcal{B}_{\mathcal{S}} = (\gamma_0, \dots, \gamma_{\lambda-1})$*. Then,*

$$\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})\right)^{\bar{\star}2}\right) \subseteq \mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}^2}}\left(\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q}\right), \tag{7.9}$$

where $\mathcal{B}_{\mathcal{S}^2}$ and $\mathcal{K}(\lambda, m)$ are defined as in Lemma 7.50, provided $\mathcal{B}_{\mathcal{S}^2}$ is a basis of $\mathbf{F}_{q^m}$. This result generalises straightforwardly to an expansion over various subspaces and bases.

Proof. We intend to apply Lemma 7.50. This lemma has two conditions.

The first condition is met by shortening the left-hand term. Indeed, the effect of shortening is that we keep only the words whose entries indexed by $\mathcal{K}(\lambda, m)$ are all equal to zero.

The second condition ($\mathcal{B}_{\mathcal{S}^2}$ being a basis) is a hypothesis. Again, Remark 7.51 applies.

Compared to Lemma 7.50 and its proof, one should be careful that in general $\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{A}^{\tilde{\star}2}\right)$ (where $\mathscr{A}$ denotes $\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})$) is not spanned by words of the form $\mathbf{Punct}_{\mathcal{K}(\lambda,m)}\left(\boldsymbol{a}\tilde{\star}\boldsymbol{b}\right)$ with $\boldsymbol{a}, \boldsymbol{b} \in \mathbf{ExpCode}_{\mathcal{B}}(\mathscr{A})$ but by linear combinations, *i.e.* words of the form

$$\mathbf{Punct}_{\mathcal{K}(\lambda,m)}\left(\boldsymbol{a}_0\tilde{\star}\boldsymbol{b}_0 + \cdots + \boldsymbol{a}_s\tilde{\star}\boldsymbol{b}_s\right), \quad \text{for} \quad \boldsymbol{a}_0, \ldots, \boldsymbol{a}_s, \boldsymbol{b}_0, \ldots, \boldsymbol{b}_s \in \mathscr{A}.$$

Therefore, one needs to apply the very same reasoning as that of the proof of Lemma 7.50 replacing $\boldsymbol{a}\tilde{\star}\boldsymbol{b}$ by a sum of such vectors. This is not a problem and the proof generalises straightforwardly, since all the involved operators are linear.

Finally, because of the shortening operation, we only obtain an inclusion and not an equality. □

Remark 7.53. *In the special case $\binom{\lambda+1}{2} = m$, $\mathcal{K}(\lambda, m) = \emptyset$, therefore the shortening is useless and the inclusion in* (7.9) *is an equality.*

Remark 7.54. *In the sequel, we will see that under a reasonable conjecture and some condition, the inclusion in* (7.9) *is an equality. In such a case, we obtain the diagram of Figure 7.5.*

7.3.2 Dimension of the twisted square of subspace subcodes

Now that we have defined the twisted star-product, we intend to see if the twisted square code of an expanded subspace subcode of a Reed–Solomon code (like the public key of the SSRS cryptosystem) behaves like a random code. More exactly, we aim at distinguishing the parent code (a Reed–Solomon code in the case of SSRS) from a random code. Therefore we apply the same construction (taking a subspace subcode, expanding and computing the twisted square code) to an RS code and a random code and see if the dimensions differ.

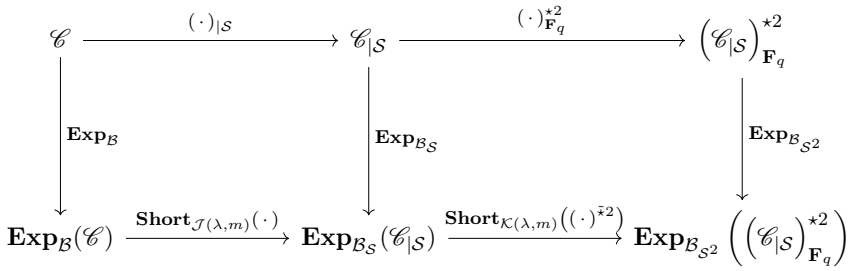


Figure 7.5: Expanding the squaring operation to subspace subcodes (general case, when $\mathcal{S}^2 = \mathbf{F}_{q^m}$). Here, **Exp** stands for **ExpCode**.

This section is therefore an equivalent of 5.3.2 but for subspace subcodes instead of full GRS codes. We will first compute the typical dimension of the twisted square code for a random code and obtain a tight upper bound. We do the same for the twisted square of SSRS codes and obtain a different upper bound. This provides us with a distinguisher.

7.3.2.1 Typical dimension of the twisted square of a random subspace subcode

Similarly to Theorem 5.15 on squares of random codes, we expect that twisted squares of random codes typically have the largest possible dimension. For this reason, we state the following conjecture which is confirmed by our experimental observations using the computer algebra software *Sage*.

Conjecture 7.55. *For any positive integer k such that $2k \leqslant n$, any $\mathbf{F}_q$–subspace $\mathcal{S} \subseteq \mathbf{F}_{q^m}^n$ of dimension $\lambda \geqslant 2$ such that $\mathcal{S}^2 = \mathbf{F}_{q^m}$ and any $\mathbf{F}_q$–basis $\mathcal{B}_{\mathcal{S}}$ of $\mathcal{S}$, let $\mathscr{R}$ denote an $[n,k]$ code chosen uniformly at random, then*

$$\mathbb{P}\left[\dim_{\mathbf{F}_q}\left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{R}_{|\mathcal{S}})\right)^{\tilde{\star}2} = \min\left\{\begin{array}{c}\binom{\lambda+1}{2}n,\\ \binom{km-n\frac{(m-\lambda)+1}{2}}{2}\end{array}\right\}\right] \underset{k\to\infty}{\longrightarrow} 1.$$

It is worth noting that in general $\binom{\lambda+1}{2} > m$. In such a case, as already mentioned before stating Lemma 7.50, the code $\left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}\mathscr{R}_{|\mathcal{S}}\right)^{\tilde{\star}2}$ represents something which is not an expansion of a code with respect to a basis of $\mathbf{F}_{q^m}$ but rather a kind of expansion with respect to a family of generators of the set $\mathcal{S}^2$. This family is denoted $\mathcal{B}_{\text{full}}$ in the proof of Lemma 7.50. This set spans $\mathcal{S}^2$ but is not linearly independent in general. Hence, given a vector with entries in $\mathbf{F}_{q^m}^n$, the decomposition with respect to this family of generators is

not unique. For this reason, it is difficult to identify the twisted square code with the expansion of another code.

To ensure the unique decomposition, the key idea is to proceed as in the statement of Theorem 7.52 and to shorten the twisted square code on the positions of the set $\mathcal{K}(\lambda, m)$ introduced in (7.7), *i.e.* shortening the last $\binom{\lambda+1}{2} - m$ positions of each block of length $\binom{\lambda+1}{2}$. According to Theorem 7.52, a codeword in $\mathbf{Short}_{\mathcal{K}(\lambda,m)} \left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{R}_{|\mathcal{S}})^{\tilde{\star}2} \right)$ is the expansion of a codeword of $\mathscr{R}^{\star 2}$ in a given basis of $\mathbf{F}_{q^m}$. The latter property is in general not satisfied by codewords of $\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{R}_{|\mathcal{S}})^{\tilde{\star}2}$. Therefore, this shortened code turns out to be a more relevant object of study and its dimension is of particular interest in the sequel. This dimension is the purpose of the following statement.

Corollary 7.56. *Let $\mathscr{R}$ be a uniformly random $[n, k]$ code over $\mathbf{F}_{q^m}$ and $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ be a subspace such that $\mathcal{S}^2 = \mathbf{F}_{q^m}$. Denote by $\mathcal{K}(\lambda, m)$ the set introduced in (7.7). Then, under Conjecture 7.55, we typically have*

$$\dim_{\mathbf{F}_q} \mathbf{Short}_{\mathcal{K}(\lambda,m)} \left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{R}_{|\mathcal{S}})^{\tilde{\star}2} \right) \geqslant \min \left\{ mn, \binom{km - n(m-\lambda) + 1}{2} - n \left(\binom{\lambda+1}{2} - m \right) \right\}. \tag{7.10}$$

7.3.2.2 Typical dimension of the twisted square of a subspace subcode of a RS code

On the other hand, subspace subcodes of Reed–Solomon codes have a different behaviour. Indeed, Theorem 7.52 yields the following result.

Corollary 7.57. *Given a GRS code $\mathscr{C} = \boldsymbol{GRS}_k(\boldsymbol{x}, \boldsymbol{y})$ and an $\mathbf{F}_q$–subspace $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ of dimension $\lambda < m$ such that $\mathcal{S}^2 = \mathbf{F}_{q^m}$. Denote by $\mathcal{K}(\lambda, m)$ the set introduced in (7.7). Then,*

$$\dim_{\mathbf{F}_q} \left(\mathbf{Short}_{\mathcal{K}(\lambda,m)} \left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}}) \right)^{\tilde{\star}2} \right) \leqslant \min\{mn, m(2k-1)\}. \tag{7.11}$$

7.3.2.3 The distinguisher

Putting the previous statements together, the twisted product provides a distinguisher between expanded subspace subcodes of GRS codes and expanded subspace subcodes of random codes.

Theorem 7.58. *Let k be a positive integer, $\mathcal{S} \subseteq \mathbf{F}_{q^m}^n$ of dimension $\lambda \geqslant 2$ an $\mathbf{F}_q$–subspace such that $\mathcal{S}^2 = \mathbf{F}_{q^m}$, $\mathcal{B}_{\mathcal{S}}$ an $\mathbf{F}_q$–basis. Let $\mathscr{D}$ be defined as $\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})$,*

where $\mathscr{C}$ is either a random $[n,k]$ code over $\mathbf{F}_{q^m}$ or an $[n,k]$ GRS code over $\mathbf{F}_{q^m}$. Suppose also that

$$m(2k-1) < \min\left\{mn, \binom{km-n(m-\lambda)+1}{2} - n\left(\binom{\lambda+1}{2} - m\right)\right\}. \tag{7.12}$$

Then, assuming Conjecture 7.55, the computation of $\dim_{\mathbf{F}_q} \mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{D}^{\tilde{\star}2}\right)$ provides a polynomial-time algorithm which decides whether $\mathscr{C}$ is an RS code or a random code and succeeds with high probability. This extends straightforwardly to the case of multiple spaces and bases.

Remark 7.59. *Condition (7.12) entails in particular $2k \leqslant n$, which is a necessary condition for the distinguisher to succeed. Indeed, if $2k > n$, the square code of the GRS code spans the whole space $\mathbf{F}_{q^m}^n$. Hence it cannot be distinguished from a random code. When this condition is not met, it is sometimes possible to shorten the code so that the shortened code meets this condition. This is addressed in Section 7.3.2.5.*

7.3.2.4 Experimental results

Using the computer algebra software *Sage*, we tested the behaviour of the dimension of the twisted square (shortened at $\mathcal{K}(\lambda,m)$) of subspace subcodes either of random codes or of RS codes. For each parameter set (see Table 7.2), we ran more than 100 tests and none of them yielded dimensions of the twisted square that was different from the bounds given either by Conjecture 7.55 or by Corollary 7.57.

In particular, these experiments show that bounds (7.10) and (7.11) are typically equalities. Note that this observation is not necessary to distinguish the codes but it will be useful for the attack presented in Section 7.4.

Remark 7.60. *Here again, we discussed the case of a single subspace $\mathcal{S}$ with a unique basis $\mathcal{B}$ for the sake of simplicity, but the distinguisher straightforwardly extends to the case of multiple spaces of dimension λ whose squares fill in $\mathbf{F}_{q^m}$ together with multiple bases.*

7.3.2.5 Broadening the range of the distinguisher by shortening

Similarly to what we presented concerning GRS codes in Section 5.3.3 (and for RLCE in Section 6.2.5), it is tempting to attempt to broaden the range of the distinguisher by shortening the public code. This can hopefully make the distinguisher work in some cases when $2k \geqslant n$. The main idea is to shorten some blocks of length λ (corresponding to a given position of the original code in $\mathbf{F}_{q^m}^n$). For each shortened block the degree k is decreased by 1. Indeed,

Parent code	q	m	λ	n	k	Bounds on the dimension of $\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{C}^{\tilde{\star}2}\right)$	Actual Dimension of $\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{C}^{\tilde{\star}2}\right)$
Random	7	3	2	120	55	$\geqslant 360$	360
RS	7	3	2	120	55	$\leqslant 327$	327
Random	7	5	3	160	75	$\geqslant 800$	800
RS	7	5	3	160	75	$\leqslant 745$	745

Table 7.2: Parameter sets for the tests. The code $\mathscr{C}$ is the shortening at $m-\lambda$ positions per block of the expansion of a parent code. The parent code is either random or a Reed–Solomon code, as indicated in the first column of the table. The penultimate column gives the bounds on the dimension of the twisted square code shortened at $\mathcal{K}(\lambda, m)$: a lower bound for random codes (Corollary 7.56) and an upper bound for SSRS codes (Corollary 7.57). The last column gives the actual dimension of the twisted square code computed using *Sage*. For each set of parameters, at least 100 tests were run and the actual dimension never differed from the bounds. We observe in particular that the bounds stated in Corollaries 7.56 and 7.57 are typically equalities.

from Lemma 7.33 shortening a whole block corresponds to shortening the corresponding position of the parent code over $\mathbf{F}_{q^m}$.

Let us investigate the condition for this to work. Let s_0 be the least positive integer such that $2(k-s_0)-1 < n-s_0$, *i.e.*

$$s_0 \stackrel{\text{def}}{=} 2k-n.$$

If one shortens the public code at $s \geqslant s_0$ blocks, which corresponds to $s(m-\lambda)$ positions, we can apply Theorem 7.58 on the shortened code. The condition of the theorem becomes

$$m(2(k-s)-1) < \min\left\{ \begin{array}{c} m(n-s), \\ \binom{m(k-s)-(n-s)(m-\lambda)+1}{2} - (n-s)\left(\binom{\lambda+1}{2} - m\right) \end{array} \right\}. \tag{7.13}$$

Example 7.61. Consider the parameters of XGRS in the first row of Table 7.1. Suppose we shorten $s = 820$ blocks of the public key (*i.e.* 1260 positions of the parent GRS code). It corresponds to reduce to $n' = n - s = 438$ and $k' = k - s = 211$. The shortened public key will have dimension 195.

Thus, the twisted square of the shortened public key will have typical dimension 1263 while the twisted square of an expanded subspace subcode of a random code would have full length, *i.e.* $3(n-s) = 1314$.

7.3.2.6 Limits of the distinguisher: the "$m/2$ barrier".

When trying to apply the distinguisher, we observe that it is ineffective when $\lambda \leqslant m/2$. This bound is stronger than the $\binom{\lambda}{2} \leqslant m$ bound that we had before. We explain here why this new bounds is unavoidable with this distinguisher.

Suppose that $\lambda \leqslant \frac{m}{2}$ and let $\mathscr{C}$ be a GRS code of dimension k and $\mathcal{S}$ a subspace of dimension λ such that the SSRS code reaches the typical dimension (see Propositions 7.2 and 7.3), *i.e.* $\dim_{\mathbf{F}_q} \mathscr{C}_{|\mathcal{S}} = km - n(m-\lambda)$.

For this dimension to be positive, we must have

$$k > n\left(1 - \frac{\lambda}{m}\right) > \frac{n}{2}.$$

This is incompatible with the necessary condition $2k < n$ (see Remark 7.59) and cannot be overcome by shortening blocks as described in Section 7.3.2.5. Hence, **whenever $\lambda \leqslant m/2$, the distinguisher is ineffective.**

Remark 7.62. *In [COT14b; COT17] a distinguisher on so–called wild Goppa codes over quadratic extensions is established using the square code operation after a suitable shortening. This corresponds precisely to the case $\lambda = 1$ and $m = 2$ which, according to the previous discussion, should be out of reach of the distinguisher. The reason why this distinguisher is efficient for these parameters is precisely because the dimension of such codes significantly exceeds the lower bound of Proposition 7.2 (see [SKHN76; COT14a]).*

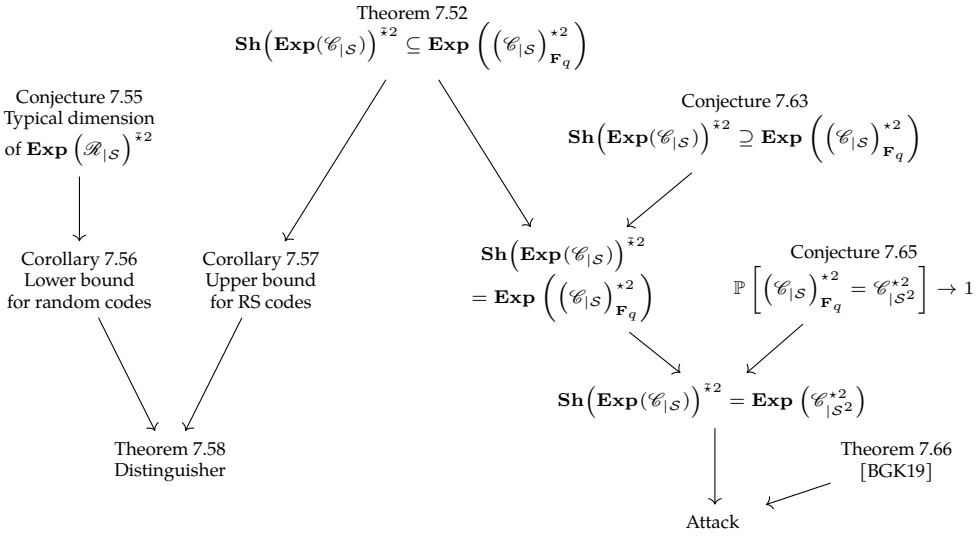


Figure 7.6: Informal summary of the statements. Any statement is the consequence of the statements pointing to it.

7.4 Attacking the SSRS scheme

In this section, we describe how to use these tools to attack the SSRS scheme. Just like in Section 7.3.1, for the sake of convenience, we first focus on the parameters with $m = 3$, $\lambda = 2$ and then discuss the general case.

7.4.1 Further conjectures for the attack

As explained in Section 7.3.2.4 our experiments show that Inequalities (7.10) and (7.11) are typically equalities. This encourages us to state the following two conjectures.

Conjecture 7.63. *Let $\mathcal{S}, \mathcal{B}_{\mathcal{S}}, \mathcal{B}_{\mathcal{S}^2}, \mathcal{K}(\lambda, m)$ be as in Theorem 7.52 and suppose that Equation (7.12) is satisfied. If $\mathscr{C}$ is an $[n, k]$ GRS code, then, with high probability, the inclusion of Equation (7.9) is an equality, i.e.*

$$\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\left(\mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}}}(\mathscr{C}_{|\mathcal{S}})\right)^{\tilde{\star}2}\right) = \mathbf{ExpCode}_{\mathcal{B}_{\mathcal{S}^2}}\left(\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q}\right).$$

In addition, the right–hand term of the last equality satisfies the following inclusion.

Lemma 7.64. *Let $\mathscr{C} \subseteq \mathbf{F}_{q^m}^n$ and $\mathcal{S} \subseteq \mathbf{F}_{q^m}$ be an $\mathbf{F}_q$–vector space. Then*

$$\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q} \subseteq \left(\mathscr{C}^{\star 2}\right)_{|\mathcal{S}^2}. \tag{7.14}$$

Proof. It suffices to observe that the result holds on $\mathbf{F}_q$–generators. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathscr{C}_{|\mathcal{S}}$. Then, $\boldsymbol{a} \star \boldsymbol{b} \in \mathscr{C}^{\star 2}$. In addition, for any $i \in \{0, \ldots, n-1\}$, we have $(\boldsymbol{a} \star \boldsymbol{b})_i \in \mathcal{S}^2$. Thus, $\boldsymbol{a} \star \boldsymbol{b} \in (\mathscr{C}^{\star 2})_{|\mathcal{S}^2}$. □

Moreover, Inclusion (7.14) turns out to be typically an equality in the case of GRS codes as suggested by the following conjecture.

Conjecture 7.65. *Let $\mathcal{S}, \mathcal{B}_{\mathcal{S}}, \mathcal{B}_{\mathcal{S}^2}, \mathcal{K}(\lambda, m)$ be as in Theorem 7.52 and suppose that Equation (7.12) is satisfied. If $\mathscr{C}$ is an $[n, k]$ GRS code, then, with high probability, the inclusion of Equation (7.14) is an equality, i.e.*

$$\left(\mathscr{C}_{|\mathcal{S}}\right)^{\star 2}_{\mathbf{F}_q} = \left(\mathscr{C}^{\star 2}\right)_{|\mathcal{S}^2}.$$

7.4.2 The case $m = 3$ and $\lambda = 2$

7.4.2.1 Constructing the square code

Let $\mathscr{C}_{\text{pub}}$ be the public code of an instance of the SSRS scheme. This code is described by a generator matrix $\boldsymbol{G}_{\text{pub}}$ which is the only data we have access

to. We know that there exist unknown spaces $\mathcal{S}_0, \dots, \mathcal{S}_{n-1}$ with bases $\mathcal{B}_{\mathcal{S}_i} = (b_{i,0}, b_{i,1})$ and an RS code over $\mathbf{F}_{q^m}$ such that

$$\mathscr{C}_{\text{pub}} = \mathbf{ExpCode}_{\left(\mathcal{B}_{\mathcal{S}_i}\right)_i}\left(\mathbf{RS}_k(\boldsymbol{x})_{|(\mathcal{S}_i)_i}\right).$$

We can compute the generator matrix of the twisted square code $\mathscr{C}_{\text{pub}}^{\tilde{\star}2}$, which according to Theorem 7.48 is equal to

$$\mathbf{ExpCode}_{\left(\mathcal{B}_{\mathcal{S}_i^2}\right)_i}\left(\left(\mathbf{RS}_k(\boldsymbol{x})_{|(\mathcal{S}_i)_i}\right)^{\star 2}_{\mathbf{F}_q}\right),$$

where $\mathcal{B}_{\mathcal{S}_i^2} \stackrel{\text{def}}{=} (b_{i,0}^2, b_{i,0}b_{i,1}, b_{i,1}^2)$. Moreover, assuming Conjecture 7.65, this code is likely to be equal to

$$\mathbf{ExpCode}_{\left(\mathcal{B}_{\mathcal{S}_i^2}\right)_i}\left(\mathbf{RS}_{2k-1}(\boldsymbol{x})\right).$$

It is important to stress that, at this stage, we do not know the value of $\boldsymbol{x}$ nor the $\mathcal{B}_{\mathcal{S}_i}$ or the $\mathcal{B}_{\mathcal{S}_i^2}$.

7.4.2.2 Finding the value of $\boldsymbol{x}$

We now have access to a fully expanded RS code (and not a subspace subcode) and want to use this to find the value of $\boldsymbol{x}$. In fact, the authors of [BGK19] propose an algorithm to solve this problem, by using a generalisation of the algorithm of Sidelnikov and Shestakov [SS92] to recover the structure of GRS codes.

Theorem 7.66. *[BGK19, § IV.B] Let $\boldsymbol{x} = (x_0, \dots, x_{n-1}) \in \mathbf{F}_{q^m}^n$ be a vector with distinct entries and $\mathcal{B}_0, \dots, \mathcal{B}_{n-1}$ be an n–tuple of $\mathbf{F}_q$–bases of $\mathbf{F}_{q^m}$. Let*

$$\mathscr{C} = \mathbf{ExpCode}_{(\mathcal{B}_i)_i}(\boldsymbol{RS}_k(\boldsymbol{x})).$$

There exists a polynomial time algorithm which

> ***takes as inputs*** *$\mathscr{C}$, three distinct elements $x'_0, x'_1, x'_2 \in \mathbf{F}_{q^m}$ and an $\mathbf{F}_q$–basis $\mathcal{B}'_0$ of $\mathbf{F}_{q^m}$;*
>
> ***and returns*** *$x'_3, \dots, x'_{n-1} \in \mathbf{F}_{q^m}^n$ and $\mathbf{F}_q$–bases $(\mathcal{B}'_1, \dots, \mathcal{B}'_{n-1})$ of $\mathbf{F}_{q^m}$ such that*
>
> $$\mathscr{C} = \mathbf{ExpCode}_{(\mathcal{B}'_0, \dots, \mathcal{B}'_{n-1})}(\boldsymbol{RS}_k((x'_0, \dots, x'_{n-1}))).$$

The principle of the algorithm is very similar to that of Sidelnikov Shestakov. Starting from a systematic generator matrix of an expanded Reed–Solomon code, the hidden structure of the RS code is deduced from relations satisfied by the $m \times m$ blocks of the right hand side of this systematic generator matrix (as in Proposition 5.9).

Remark 7.67. *Theorem 7.66 asserts in particular that the choice of three values of the support together with one basis uniquely determines a pair $(\boldsymbol{x}, (\mathcal{B}_i)_i)$ describing a code $\mathbf{ExpCode}_{(\mathcal{B}_i)_i}(\boldsymbol{x})$.*

Using this Theorem 7.66, we obtain a vector $\boldsymbol{x}'$ and $\mathbf{F}_q$–bases $\mathcal{B}'_i$ of $\mathbf{F}_{q^m}$ such that

$$\mathscr{C}_{\text{pub}}^{\star 2} = \mathbf{ExpCode}_{(\mathcal{B}'_i)_i}(\mathbf{RS}_{2k-1}(\boldsymbol{x}')).$$

Remark 7.68. *Note that the value of $\boldsymbol{x}'$ is not necessarily the same as the one contained in the secret key but we are looking for an equivalent secret key, i.e. we only need a code description which allows us to decode. See Remark 1.29.*

7.4.2.3 Recovering a secret key

Once $\boldsymbol{x}'$ is found, there remains to find bases $\mathcal{B}_{\mathcal{S}'_0}, \ldots, \mathcal{B}_{\mathcal{S}'_{n-1}}$ of 2–dimensional subspaces $\mathcal{S}'_0, \ldots, \mathcal{S}'_{n-1} \subseteq \mathbf{F}_{q^m}$ such that

$$\mathscr{C}_{\text{pub}} = \mathbf{ExpCode}_{\left(\mathcal{B}_{\mathcal{S}'_i}\right)_i}(\mathbf{RS}_k(\boldsymbol{x}')).$$

These bases can be obtained by solving a linear system. They are the pairs

$$\mathcal{B}_{\mathcal{S}'_0} = (b_0^{(0)}, b_0^{(1)}), \ldots, \mathcal{B}_{\mathcal{S}'_{n-1}} = (b_{n-1}^{(0)}, b_{n-1}^{(1)})$$

such that

$$\mathbf{SqueezeCode}_{\left(\mathcal{B}_{\mathcal{S}'_i}\right)_i}(\mathscr{C}_{\text{pub}}) \subseteq \mathbf{RS}_k(\boldsymbol{x}'),$$

which can be equated as follows. Let $\boldsymbol{H}$ be a parity–check matrix of $\mathbf{RS}_k(\boldsymbol{x})$ and $\boldsymbol{G}_{\text{pub}}$ a generator matrix of $\mathscr{C}_{\text{pub}}$. Let

$$\boldsymbol{B} = \begin{pmatrix} b_0^{(0)} & & & (0) \\ b_0^{(1)} & & & \\ & b_1^{(0)} & & \\ & b_1^{(1)} & & \\ & & \ddots & \\ & & & b_{n-1}^{(0)} \\ (0) & & & b_{n-1}^{(1)} \end{pmatrix} \in \mathbf{F}_{q^3}^{2n \times n}.$$

The unknown entries of $\boldsymbol{B}$ are the solutions of the linear system

$$\boldsymbol{G}_{\text{pub}} \boldsymbol{B} \boldsymbol{H}^{\intercal} = 0. \tag{7.15}$$

There are

- $2n$ unknowns in $\mathbf{F}_{q^3}$ which yields $6n$ unknowns in $\mathbf{F}_q$;
- for $(3k-n)(n-k) = O(n^2)$ equations.

Thus, the matrix $\boldsymbol{B}$ is very likely to be the unique solution up to a scalar multiple. From this, we obtain a complete equivalent secret key, which allows to decrypt any ciphertext.

Remark 7.69. *After presenting a polynomial time recovery of the structure of expanded GRS codes in* [*BGK19, § IV.B*]*, the extension to expanded SSRS codes is discussed* [*BGK19, § VI.C*]*. The suggested approach consists in performing a brute–force search on the expansion bases $\mathcal{B}_0, \dots, \mathcal{B}_{n-1}$. But the cost of such an approach is exponential in n and λ. Our use of the twisted square code permits to address the same problem in polynomial time.*

7.4.2.4 Extending the reach of the attack by shortening blocks

As explained in Section 7.3.2.5, it may happen that $\mathscr{C}_{\text{pub}}^{\tilde{\star}2} = \mathbf{F}_q^{3n}$, *i.e.* the twisted square of the public code equals the whole ambient space. In such a situation, the distinguisher fails and so does the attack. To overcome this issue, it is sometimes possible to shorten a fixed number s of blocks of $\mathscr{C}_{\text{pub}}$ and apply the previous attack to this block–shortened code.

More precisely, let $\mathcal{I} \subseteq [\![0, 2n-1]\!]$ be a set of indices corresponding to a union of blocks, *i.e.* of the form $\mathcal{I} = \{2i_0, 2i_0+1, \dots, 2i_s, 2i_s+1\}$. We apply the previous algorithm to the code $\mathbf{Short}_{\mathcal{I}}\left(\mathscr{C}_{\text{pub}}\right)$ which returns $((x'_i)_{i\notin\mathcal{I}}, (\mathcal{B}'_i)_{i\notin\mathcal{I}})$ such that

$$\mathbf{Short}_{\mathcal{I}}\left(\mathscr{C}_{\text{pub}}\right)^{\tilde{\star}2} = \mathbf{ExpCode}_{(\mathcal{B}'_i)_{i\notin\mathcal{I}}}((x'_i)_{i\notin\mathcal{I}}).$$

Then, one can re-apply the same process with another set of blocks $\mathcal{I}_1$ such that there are at least 3 blocks that are neither in $\mathcal{I}_0$ nor in $\mathcal{I}_1$, *i.e.* $|([\![1,n]\!] \setminus \mathcal{I}_0) \cap ([\![1,n]\!] \setminus \mathcal{I}_1)| \geqslant 3$. Up to permutation of the blocks, we use these positions in common to play the role of x'_0, x'_1, x'_2 in Theorem 7.66. Recall that the choice of three of the x'_i's and one of the $\mathcal{B}_i$'s entirely determines the other ones. Hence, this allows to deduce new values for x'_i's for $i \in \mathcal{I} \setminus \mathcal{I}_1$ that are consistent with the previously computed values of $\boldsymbol{x}'$. We repeat this operation until $\boldsymbol{x}'$ is entirely computed. Then, we proceed as in Section 7.4.2.3 to recover the rest of the secret key.

7.4.2.5 Application: attacking some parameters of the XGRS system

The proposed attack permits to break efficiently any parameters of Type I proposed in [KRW21] (*i.e.* with $\lambda = 2$ and $m = 3$). Using a *Sage* implementation, the calculation of $\mathscr{C}_{\text{pub}}^{\tilde{\star}2}$ takes a few minutes. Next, we obtained a full

key recovery using the "guess and squeeze" approach described further in Section 7.4.6 followed by a usual Sidelnikov Shestakov attack. The overall attack runs in less than one hour for keys corresponding to a claimed security level of 256 bits. The previously described approach consisting in applying directly the algorithm of [BGK19, § VI.B] on $\mathscr{C}_{\mathrm{pub}}^{\tilde{\star}2}$ has not been implemented but is probably even more efficient.

7.4.3 The general case

The attack presented in Section 7.4.2 generalises straightforwardly (up to the following details) when the conditions of Conjecture 7.63 are met.

- According to Theorem 7.52, we should no longer work with $\mathscr{C}_{\mathrm{pub}}^{\tilde{\star}2}$ but with $\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{C}_{\mathrm{pub}}^{\tilde{\star}2}\right)$, where $\mathcal{K}(\lambda, m)$ is defined in Lemma 7.50 (7.7). Assuming Conjectures 7.63 and 7.65, we deduce that this code is the expansion of a GRS code. Hence, the algorithm of [BGK19, § VI.B] can be applied to it.
- The recovery of the subspaces and bases described in Section 7.4.2.3 involves a matrix $\boldsymbol{B} \in \mathbf{F}_q^{\lambda n \times n}$ with λn nonzero entries, which will be the unknowns of the system (7.15). Hence, this system has λn unknowns in $\mathbf{F}_{q^m}$, *i.e.* λmn unknowns in $\mathbf{F}_q$ for $(mk - n(m-\lambda))(n-k) = O(n^2)$ equations. As the value of m (and hence λ) remain very small compared to n, there is still in general a unique solution up to a scalar multiple.

7.4.4 Summary of the attack

The attack can be summarised by Algorithms 13 and 14, depending on the values of k and n.

Algorithm 13: The attack when $2k \leqslant n$

1. Compute $\mathbf{Short}_{\mathcal{K}(\lambda,m)}\left(\mathscr{C}_{\mathrm{pub}}^{\tilde{\star}2}\right)$, where $\mathcal{K}(\lambda, m)$ is the the union of the last $\binom{\lambda+1}{2} - m$ positions of each block (see Lemma 7.30 (7.4));
2. Apply the algorithm of [BGK19, § VI.B] to recover a support $\boldsymbol{x}$ of the parent Reed–Solomon code;
3. Apply the calculations of Section 7.4.2.3 to recover the bases $\mathcal{B}_i$.

Remark 7.70. *In the case $\lambda = 2$ and $m = 3$, $\binom{\lambda+1}{2} = m$ and hence $\mathcal{K}(\lambda, m) = \emptyset$.*

Algorithm 14: Attack when $2k > n$

1. Choose a number s of blocks to shorten satisfying condition (7.13) so that the distinguisher succeeds.
2. Pick a union of s blocks $\mathcal{I}$ and
   (a) Compute $\mathbf{Short}_{\mathcal{K}(\lambda,m)'}\left(\mathbf{Short}_{\mathcal{I}}\left(\mathscr{C}_{\text{pub}}\right)^{\star 2}\right)$, where $\mathcal{K}(\lambda,m)'$ is the union of the last $\binom{\lambda+1}{2} - m$ positions of each block;
   (b) Apply the algorithm of [BGK19, § VI.B] to recover a partial support $(x_i)_{i\notin\mathcal{I}}$;
   (c) Repeat this process with another $\mathcal{I}$ until you got the whole support $\boldsymbol{x}$.
3. Apply the calculations of Section 7.4.2.3 to recover the bases $\mathcal{B}_i$.

7.4.5 Complexity

For the complexity analysis and according to the parameters proposed in [KRW21], we suppose that $m = O(1),\ \lambda = O(1)$ and $k = \Theta(n)$.

7.4.5.1 Step 1, the twisted square computation

First let us evaluate the cost of the computation of the twisted square of the code $\mathscr{C}_{\text{pub}} \subseteq \mathbf{F}_q^{\lambda n}$ of dimension $k_0 \stackrel{\text{def}}{=} (mk - n(m-\lambda)))$.

1. Starting from a $k_0 \times \lambda n$ generator matrix of $\mathscr{C}_{\text{pub}}$, any non ordered pair of rows provides a generator of the twisted square. Hence there are $\binom{k_0+1}{2} = O(n^2)$ generators to compute, each computation costing $n\binom{\lambda+1}{2}$ operations. This is an overall cost of $O(n^3)$ operations in $\mathbf{F}_q$.
2. Then, deducing a row echelon generator matrix of this twisted square from these $O(n^2)$ generators has the cost of the computation of the row echelon form of a $O(n^2) \times O(n)$ matrix, which requires $O(n^{\omega+1})$ operations in $\mathbf{F}_q$ (see [BCGLL+17, Théorème 8.6]), where $\omega \leqslant 3$ is the complexity exponent of operations of linear algebra.

Thus, the overall cost of the computation of this twisted square code is $O(n^{\omega+1})$. In addition, in the situation where $2k - 1 > n$, we need to iterate the calculation on a constant number of shortenings of the public code, which has no influence on the complexity exponent.

7.4.5.2 Step 2, recovering $\boldsymbol{x}$

The second step of the attack, *i.e.* performing the algorithm of [BGK19, § VI.B] to recover $\boldsymbol{x}$ is not that expensive. A quick analysis of this algorithm permits to observe that the most time consuming step is the calculation of the systematic form of the generator matrix, which has actually been performed in the previous step. Therefore, this second step can be neglected in the complexity analysis.

7.4.5.3 Step 3, recovering the bases

Finally, the last step of the attack, consisting in recovering the bases $\mathcal{B}_i$, boils down to the resolution of a linear system of $O(n^2)$ equations and $O(n)$ unknowns, which costs $O(n^{\omega+1})$ operations.

Summary. The overall cost of the attack is of $O(n^{\omega+1})$ operations in $\mathbf{F}_q$.

7.4.6 The guess-and-squeeze approach

To conclude this section, we present an alternative approach to detect the hidden structure of expanded codes and recover the expansion bases. This method applies to the expansion of any code. It can in particular apply to the twisted square of SSRS codes. As explained in Section 7.4.2.5, this is the approach we implemented. The interest of this approach is that it may apply to expansions of codes which are not RS codes and hence may be an interesting tool for other cryptanalyses.

Given a code $\mathscr{C} \subseteq \mathbf{F}_{q^m}^n$ and bases $\mathcal{B}_0, \dots, \mathcal{B}_{n-1}$ of $\mathbf{F}_{q^m}$, suppose you only know a generator matrix of

$$\mathscr{C}_{\mathrm{exp}} \stackrel{\mathrm{def}}{=} \mathbf{ExpCode}_{(\mathcal{B}_i)}(\mathscr{C}).$$

The goal is to guess the $\mathcal{B}_i$'s iteratively instead of brute forcing any n–tuple of bases, which would be prohibitive.

1. Shorten $\mathscr{C}_{\mathrm{exp}}$ at $k-1$ blocks (which corresponds to $m(k-1)$ positions). This yields a code whose dimension most of the times equals m. According to Lemma 7.33, this is the expansion of a code of dimension 1 obtained by shortening $\mathscr{C}$ at $k-1$ positions.

2. Puncture this shortened code in order to keep only two blocks. We get a $[2m, m]$ code which we call $\mathscr{C}_{\mathrm{exp,tiny}} \subseteq \mathbf{F}_q^{2m}$. This code is the expansion of a $[2, 1]$ code called $\mathscr{C}_{\mathrm{tiny}} \subseteq \mathbf{F}_{q^m}^2$ obtained from $\mathscr{C}$ by shortening $k-1$ positions and puncturing the remaining code in order to keep only 2 positions.

3. Now, for any pair of bases $(\mathcal{B}_0, \mathcal{B}_1)$ of $\mathbf{F}_{q^m}$, compute

$$\mathbf{SqueezeCode}_{(\mathcal{B}_0,\mathcal{B}_1)}(\mathscr{C}_{\mathrm{exp,tiny}}).$$

The point is that, for a wrong choice of bases, we get a generator matrix with m rows and 2 columns which is very likely to be full rank. Hence a wrong choice provides the trivial code $\mathbf{F}_{q^m}^2$. On the other hand, a good choice of bases provides the code $\mathscr{C}_{\mathrm{tiny}}$ which has dimension 1. This property permits to guess the bases.

Actually, according to Lemma 7.38, if one guesses the bases $a_0\mathcal{B}_0, a_1\mathcal{B}_1$ for some $a_0, a_1 \in \mathbf{F}_{q^m}^\times$, the squeezing will provide $\mathscr{C}_{\mathrm{tiny}} \star (a_0, a_1)$ which also has dimension 1. Therefore, it is possible to first guess the bases up to a scalar multiple in $\mathbf{F}_{q^m}^\times$. Therefore, the cost of computing these two bases is in $O(q^{2m(m-1)})$ operations.

Once the first two bases are known, one can restart the process by with another pair of blocks involving one of the two blocks for which the basis is already known, which requires $O(q^{m(m-1)})$ operations. This yields an overall complexity of $O(q^{2m(m-1)} + nq^{m(m-1)})$ operations in $\mathbf{F}_q$ for this *guess and squeeze* algorithm.

Remark 7.71. *Note that in the attack of XGRS scheme, the bases to guess are known to be of the form $(1, \gamma, \gamma^2, \ldots, \gamma^{m-1})$ for some generator $\gamma \in \mathbf{F}_{q^m}$. This additional information permits to significantly improve this search and reduce the cost of the calculation of the n bases to $O(q^{2m} + nq^m)$ operations.*

Remark 7.72. *Proceeding this way, only permits to get back the code $\mathscr{C} \star \boldsymbol{a} \subseteq \mathbf{F}_{q^m}^n$ for an unknown vector $\boldsymbol{a} \in (\mathbf{F}_{q^m}^\times)^n$. However, this is an important first step. For instance, if $\mathscr{C}$ is a Reed–Solomon, we obtain a generalised Reed–Solomon code whose structure is computable using the classical Sidelnikov and Shestakov attack. It is then possible to decode.*

7.5 Conclusion

In this chapter, we extended the line of work on square-code distinguishers to the case of subspace subcodes. For this, we had to adapt the square-product operation to expanded codes, with a tool that we call the twisted square-product. This yields a polynomial-time distinguisher on subspace subcodes of Reed–Solomon codes, under some conditions on the parameters. We are hence able to distinguish SSRS codes from random ones as soon as the dimension λ of the subspaces exceeds $\frac{m}{2}$. From this distinguisher, we derived an attack breaking in particular the parameter set $\lambda = 2$ and $m = 3$ of the XGRS system [KRW21].

These results contribute to better understanding the McEliece encryption scheme instantiated using algebraic codes. On one hand, we have generalised Reed–Solomon codes, which are known to be insecure since the early 90's. On the other hand, alternant codes seem to resist to any attack except some Goppa codes with an extension degree $m = 2$ [COT17; FPP14]. The present work provides an analysis of a family of codes including these two cases as the two extremities of a spectrum. Concerning the subspace subcodes lying in between, we show an inherent weakness of SSRS codes when $\lambda > m/2$ (see Figure 7.7). The case $\lambda = m/2$ is in general out of reach of our distinguisher, but remains border line as testified by some attacks on the cases $\lambda = 1, m = 2$ in the literature [COT17; FPP14].

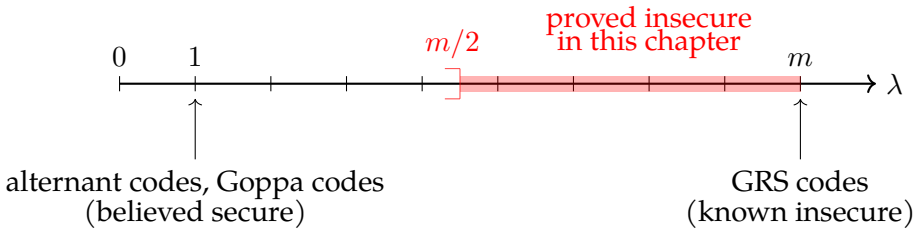


Figure 7.7: The twisted square-code distinguisher attack presented in this chapter covers the cases where $m/2 < \lambda \leqslant m$.

A question which remains open is the actual security of the cases $1 < \lambda < m/2$ which are out of reach of the twisted square code distinguisher. These codes, which include alternant codes, deserve to have a careful security analysis in the near future. Indeed, if they turn out to be resistant to distinguishing attack, they could provide an alternative to *Classic McEliece* [BCLMM+19], presumably with shorter key sizes. On the other hand, if one finds a distinguisher for such codes, this could impact the security of *Classic McEliece* which is a crucial question in the near future.

Part III

Generic decoding

Chapters

Chapter 8

Binary syndrome decoding

The syndrome decoding problem is the central problem upon which code-based cryptography is built. This problem is equivalent to decoding in a random code, and the security of most code-based primitives rely on the hypothesis that this problem is hard. Although this problem is proven to be NP-hard, there is no result concerning its average complexity. Still, a forty years long line of research has tried to come up with the best algorithms, and solving it remains exponentially hard. The best known algorithm use of the *information set decoding* technique. This chapter is a succinct introduction to the state-of-the-art results in this direction. As most of these studies concern binary linear codes, we will restrict our survey to this case. We will see in the next chapter the difference induced by the use of a larger basefield.

Contents

8.1 The syndrome decoding problem

8.1.1 The problem

The syndrome decoding problem has been introduced in Chapter 1. It is one of the oldest problems in coding theory and cryptography [McE78]. We recall here its definition.

Problem 8.1 (Syndrome Decoding - $\mathrm{SD}(q, R, W)$)**.**
Instance: $\boldsymbol{H} \in \mathbf{F}_q^{(n-k)\times n}$ *of full rank,*
$\boldsymbol{s} \in \mathbf{F}_q^{n-k}$ (*usually called the* syndrome).
Output: $\boldsymbol{e} \in \mathbf{F}_q^n$ *such that* $\mathbf{w}_H(\boldsymbol{e}) = w$ *and* $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$,
where $k \stackrel{def}{=} \lceil Rn \rceil$ *and* $w \stackrel{def}{=} \lceil Wn \rceil$.

We have seen in Chapter 1 that this problem is equivalent to the general decoding problem.

Moreover, it is known to be NP-complete [BMT78] and conjectured to be hard on average [Ale11]. Finally, this problem is believed to remain hard in the presence of a quantum adversary, which makes code-based cryptography a serious solution for post-quantum cryptography.

The problem $\mathrm{SD}(q, R, W)$ is parametrised by three parameters.

1. the field size q. In this chapter we will restrict our description to the case $q = 2$ and hence denote $\mathrm{SD}(R, W) \stackrel{\text{def}}{=} SD(2, R, W)$.
2. The rate $R \in [0, 1]$ of the code.
3. The relative weight $W \in [0, 1]$.

Remark 8.2. *In the binary case,* $\mathrm{SD}(R, W)$ *can be reduced to* $\mathrm{SD}(R, 1-W)$*. Indeed, given an instance* $(\boldsymbol{H}, \boldsymbol{s})$ *of* $\mathrm{SD}(R, W)$*, we can solve* $(\boldsymbol{H}, \boldsymbol{s} + \mathbf{1}\boldsymbol{H}^\intercal)$ *where* $\mathbf{1}$ *denotes the vector with all its components equal to* 1*. This is an instance of* $\mathrm{SD}(R, 1-W)$ *which gives the same solution. Hence the problem is symmetric with respect to* W*, and we can therefore restrict our study to the case* $W \in [0, 1/2]$*. This is specific to the binary case. The general case with* $W > 1/2$ *will be discussed in Chapter 9.*

Remark 8.3. *The matrix length n is not considered as a parameter of the problem since we are only interested in the asymptotic complexity (see below). Still, when talking about a particular instance, the length of the code matters and we will be talking about an instance of* $\mathrm{SD}(R, W)$ *of length* n.

8.1.2 Workfactor and asymptotic formulas

In this chapter, as in most of the literature on the subject, we will only consider the asymptotic complexity of the syndrome decoding problem. Indeed, the algorithms solving this problem have exponential complexity, and the main issue in cryptography is to see how this complexity scales with the parameters. This approach requires some appropriate definitions.

Hence, our interest will be focused on the asymptotic exponent of the complexity of the algorithms solving the problem $\mathrm{SD}(q, R, W)$. This will be referred to as the *workfactor* of the algorithm, denoted $\mathcal{F}(R, W)$.

Definition 8.4 (Workfactor [CG90])**.** For an algorithm $\mathcal{A}$ solving the syndrome decoding problem $\mathrm{SD}(q, R, W)$, let $\mathcal{T}_{\mathcal{A}}(q, n, k, w)$ denote the average time complexity for algorithm $\mathcal{A}$ to solve an instance of $\mathrm{SD}(q, R, W)$ of length n, with $k = \lceil Rn \rceil$ and $w = \lceil Wn \rceil$, the workfactor $\mathcal{F}_{\mathcal{A}}(q, R, W)$ is defined as

$$\mathcal{F}_{\mathcal{A}}(q, R, W) \stackrel{\text{def}}{=} \lim_{n \to \infty} \frac{1}{n} \log_q \mathcal{T}_{\mathcal{A}}(q, n, \lceil Rn \rceil, \lceil Wn \rceil) .$$

This definition fulfills our need to focus on the asymptotic exponent, since we can write

$$\mathcal{T}_{\mathcal{A}}(q, n, k, w) = \tilde{\mathcal{O}}\left(q^{n(\mathcal{F}_{\mathcal{A}}(q,R,W)+o(1))}\right),$$

where the coefficient $\mathcal{F}$ only depends on R and W.

For the complexity analysis, we will often make use of the entropy function h_q, introduced in Definition 1.20. The binary entropy function h_2 will often simply be denoted by h. Note that the entropy function is continuous and strictly increasing between $h_q(0) = 0$ and $h_q(1 - 1/q) = 1$. Hence we will sometimes use the notation h_q^{-1} to denote the inverse of the entropy function on this interval. The function h_q^{-1} is defined over $[0, 1]$.

Finally, let us recall the classic consequence of Stirling's formula which is particularly useful for asymptotic complexity analysis:

$$\binom{\alpha n}{\beta n} = \sqrt{\frac{\alpha}{2\pi\beta(\alpha - \beta)}} 2^{\alpha h(\beta/\alpha)n - o(n)} = \tilde{\Theta}\left(2^{\alpha h(\beta/\alpha)n}\right).$$

8.1.3 Number of solutions

For a fixed value R, the number of solutions to $\mathrm{SD}(R, W)$ greatly varies with the value of W. Informally, we can see that when W is close to zero, the number of solutions is at most one, and probably zero. On the other hand, for W close to $1/2$, the number of solutions is on average $\binom{n}{w}/2^{n-k}$ which is exponentially large. Moreover, because we study this problem in the context of cryptanalysis, we always suppose that the instance is defined such that a solution exists. Hence, the number of solutions of an instance of $\mathrm{SD}(R, W)$ of length n is given by

$$\mathcal{S}(n, k, w) = \max\left\{1, \frac{\binom{n}{w}}{2^{n-k}}\right\}.$$

The value of w bellow which the number of solutions is (at most) one is easy to characterise. This is known as the Gilbert-Varshamov bound (see Theorem 1.27).

Definition 8.5 (Gilbert-Varshamov distance)**.** For $k \leqslant n$, the Gilbert-Varshamov distance for $[n, k]$-codes is defined as the largest integer w such that

$$\sum_{d=0}^{w-1} \binom{n}{d} \leqslant 2^{n-k}.$$

We denote $w_{GV}(n, k)$ this distance.

Asymptotically, we obtain the following result.

Definition 8.6. For $R \in [0, 1]$, let the relative Gilbert-Varshamov distance for codes of rate R be defined as

$$\lim_{n\to\infty} w_{GV}(n, \lceil nR \rceil)/n.$$

Stirling's formula yields

$$W_{GV}(R) \stackrel{\text{def}}{=} h^{-1}(1 - R),$$

where h_2 denotes the binary entropy function.

Hence, when $W < W_{GV}(R)$, we expect to have (at most) one solution, whereas there are exponentially many solutions in the case $W > W_{GV}(R)$. We will see that for a fixed value of R, the regime with W close to W_{GV} corresponds to the hardest case of the problem.

8.2 Combinatorial approach

Let us first state the combinatorial approaches to solve the $\mathrm{SD}(R, W)$ problem.

8.2.1 Exhaustive search

The naive algorithm to solve the decoding problem is to exhaustively try all error patterns of weight w. For syndrome decoding, this means trying all sums of w columns of the matrix $\boldsymbol{H}$ and see if one matches the syndrome $\boldsymbol{s}$.

The complexity of this approach is dominated by the term $\binom{n}{w}$, which leads to a workfactor of

$$\mathcal{F}_{Exhaustive} = h(W).$$

With this approach, we directly obtain all solutions to the problem.

8.2.2 Birthday decoding

A classical algorithmic improvement for this kind of problems is to use the birthday paradox to look for collisions. Namely, instead of reaching for a single target $\boldsymbol{s}$, one can construct two lists and look for a collision between the two lists.

Let us split the matrix $\boldsymbol{H}$ in two halves $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$. Enumerate the sets

$$\mathcal{L}_1 = \{\boldsymbol{e}_1 {\boldsymbol{H}_1}^\intercal \,|\, \boldsymbol{e}_1 \in \mathbf{F}_2^{n/2}, \mathbf{w}_H(e_1) = w/2\},$$

$$\mathcal{L}_2 = \{\boldsymbol{e}_2 {\boldsymbol{H}_2}^\intercal + \boldsymbol{s} \,|\, \boldsymbol{e}_2 \in \mathbf{F}_2^{n/2}, \mathbf{w}_H(e_2) = w/2\}.$$

If $\mathcal{L}_1 \cap \mathcal{L}_2 \neq \emptyset$, let $\boldsymbol{v}$ be in the intersection, then $\boldsymbol{v} = \boldsymbol{e}_1 {\boldsymbol{H}_1}^\intercal = \boldsymbol{e}_2 {\boldsymbol{H}_2}^\intercal + \boldsymbol{s}$. Hence $(\boldsymbol{e}_1, \boldsymbol{e}_2)\boldsymbol{H}^\intercal = \boldsymbol{s}$ and the vector $(\boldsymbol{e}_1, \boldsymbol{e}_2)$ is a solution to the problem.

The time complexity of such an algorithm (using hashtables) is $2|\mathcal{L}| + |\mathcal{L}|^2/2^{n-k}$ where $|L| = \binom{n/2}{n/2}$ is the size of each list.

But this algorithm only succeeds if there exists a solution which splits evenly in two halves. In such a case, all such solutions are found. On the other hand, in there is no such solution, one can rerun the algorithm after permuting the columns of $\boldsymbol{H}$.

The probability of success is $\binom{n/2}{w/2}^2 / \binom{n}{w}$. Asymptotically, this probability is close to one, so on average there should be no much need to run the algorithm more than once. Reruning the algorithm only adds a polynomial factor.

Hence, the workfactor of this algorithm is

$$\mathcal{F}_{Bday} = \max\{h(W)/2, h(W) - (1 - R)\}.$$

Note that when $W \leqslant W_{GV}$, the right hand part is zero, hence the workfactor is

$$\mathcal{F}_{Bday} = h(W)/2,$$

which corresponds to a quadratic gain compared to exhaustive search.

8.2.3 Average complexity to find one solution

It is important to note that in both cases, the complexity computed corresponds to obtaining all solutions to the problem. If we are in the regime where the expected number of solutions is greater than one ($W > W_{GV}$), then we obtain $\binom{n}{w}/2^{n-k}$ solutions with this complexity.

To quantify this, we can compute the average complexity per solution obtained. Indeed, if we have an algorithm that obtains M solution in time T, we say that it finds one solution in amortized time T/M.

Especially, using the birthday decoding in the case $W > W_{GV}$, because the number of solutions is asymptotically equal to the complexity, then this algorithm can be used to find solutions in amortized time $\mathcal{O}(1)$.

8.3 Using linear algebra: Prange's approach

The approach presented until here is purely combinatorial and ignores all the linear algebraic structure of the codes. A different approach making use of this properties was introduced by Prange in 1962 [Pra62].

8.3.1 Information sets

The idea of Prange relies on the concept of *information sets*. For an $[n, k]$-code $\mathscr{C}$, an information set is a subset of positions of $\mathscr{C}$ that uniquely defines each codeword, *i.e.* a set $\mathcal{I} \subseteq [\![1, n]\!]$ such that $|\mathcal{I}| = k$ and

$$\forall \boldsymbol{m} \in \mathbf{F}_2^k, \exists! \boldsymbol{c} \in \mathscr{C}, \boldsymbol{c}_{\mathcal{I}} = \boldsymbol{x},$$

where $\boldsymbol{c}_{\mathcal{I}}$ denotes the restriction to the positions indexed by $\mathcal{I}$.

Because $\mathscr{C}$ is a vector space of $\mathbf{F}_2^n$ of dimension k, there exists such sets. In practice, for a given generator matrix $\boldsymbol{G}$ of the code, it corresponds to sets $\mathcal{I}$ such that the square submatrix $\boldsymbol{G}_{\mathcal{I}}$ formed by the columns indexed by $\mathcal{I}$ is invertible.

For a random binary code (*i.e.* a random matrix $\boldsymbol{G} \in \mathbf{F}_2^{k \times n}$), the probability that a subset of k columns forms an invertible matrix is $\prod_{i=1}^{k}(1 - 2^i)$ which tends towards a constant value $\simeq 0.289$ when k tends to infinity. Hence, a constant proportion of the subsets of positions of a code are information sets.

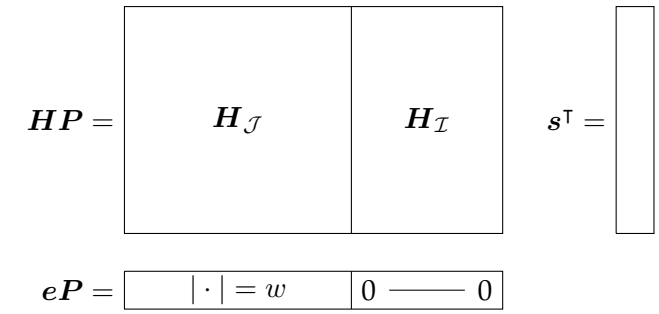
8.3.2 Prange's idea

Prange's idea can be stated very simply. We have a noisy codeword $\boldsymbol{y}$ of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, with $\boldsymbol{c} \in \mathscr{C}$ and $\mathrm{w}_H(\boldsymbol{e}) = w$. To find $\boldsymbol{c}$, choose an information set $\mathcal{I}$ and find the unique codeword $\boldsymbol{c}' \in \mathscr{C}$ such that $\boldsymbol{y}_{\mathcal{I}} = \boldsymbol{c}'_{\mathcal{I}}$.

If we are lucky in the choice of $\mathcal{I}$, none of the positions of $\mathcal{I}$ contain errors, *i.e.* $\boldsymbol{e}_{\mathcal{I}} = \boldsymbol{0}_k$. In this case, $\boldsymbol{c}'_{\mathcal{I}} = \boldsymbol{c}_{\mathcal{I}}$ and by definition of information sets, $\boldsymbol{c}' = \boldsymbol{c}$. This can be checked easily by checking that $\boldsymbol{y} - \boldsymbol{c}'$ is of weight w as expected for $\boldsymbol{e}$. If not, it means that the choice of $\mathcal{I}$ contains an error, and we should retry with another information set $\mathcal{I}$ until a solution is found.

8.3.3 Prange's information set decoding algorithm

The transposition of Prange's idea to the syndrome decoding problem is the following. The columns of the parity-check matrix are permuted such that the columns corresponding to the information set $\mathcal{I}$ correspond to the k right-most columns, and the $n-k$ columns corresponding to $\mathcal{J} \stackrel{\text{def}}{=} [\![1, n]\!] \setminus \mathcal{I}$ are on the left-hand side.

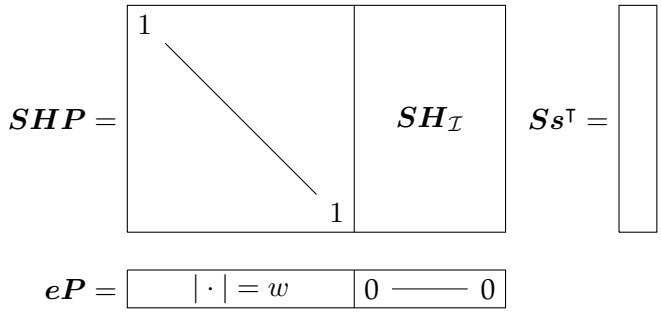


Because we suppose that there if no error on the columns corresponding to $\mathcal{I}$, then we must find a combination of w columns of the square matrix $\boldsymbol{H}_{\mathcal{J}}$ which add up to form the syndrome $\boldsymbol{s}$. An easy way to find these columns, provided they exist, is to perform a Gaussian elimination on this part of the matrix (and to apply the same row transformation to the syndrome).

If the guess is correct, then the syndrome after Gaussian elimination will be of weight w. Otherwise, one should repeat the process with a different information set, *i.e.* a different permutation, until it succeeds.

8.3.4 Complexity of Prange's algorithm

Let us discuss the complexity of this algorithm. This algorithm relies on the fact that at some point we will be successful in picking an information set $\mathcal{I}$ containing no errors. In such a case, the algorithm succeeds.



Algorithm 15: Prange's algorithm

Input: $\boldsymbol{H} \in \mathbf{F}_2^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbf{F}_2^{n-k}$, $w \in \mathbf{N}$

while $true$ **do**
  Choose $\boldsymbol{P}$ a random $(n-k) \times (n-k)$ permutation matrix.
  Let $\boldsymbol{S}$ be the invertible matrix such that $\boldsymbol{SHP}$ is in systematic form.
  if $\mathrm{w}_H(\boldsymbol{sS}^\intercal) = w$ **then**
    return $(\boldsymbol{sS}^\intercal, 0-0)\boldsymbol{P}^{-1} \in \mathbf{F}_2^n$

Let us denote $\mathcal{P}$ this probability. It is the probability, for a random instance $\boldsymbol{H}$ and $\boldsymbol{s} = \boldsymbol{He}^\intercal$ with $\mathrm{w}_H(\boldsymbol{e}) = w$, that a random subset $\boldsymbol{I} \subseteq [\![1, n]\!]$ of size k verifies $\boldsymbol{e}_{\mathcal{I}} = \mathbf{0}_k$.

The probability to find the solution when it is unique is the following.

$$\mathcal{P} = \frac{\binom{n-k}{w}}{\binom{n}{w}}$$

But in the case where there exist many solutions to the problem, *i.e.* when $W > W_{GV}$, there is an exponential number of solutions, namely $\binom{n}{w}/2^{n-k}$, so the probability of success is multiplied by the number of existing solutions to the problem. Hence, we obtain the following formula which encompasses both cases.

$$\mathcal{P} = \frac{\binom{n-k}{w}}{\min\{\binom{n}{w}, 2^{n-k}\}}$$

Each run of the loop has a polynomial cost (performing a Gaussian elimination), which is negligible compared to the success probability. Therefore the total complexity to find a solution using Prange's algorithm is $\tilde{\mathcal{O}}\left(\mathcal{P}^{-1}\right)$. Hence we obtain the following workfactor.

$$\mathcal{F}_{Prange} = \min\{(1-R), h(W)\} - (1-R)h\left(\frac{W}{1-R}\right)$$

It is important to note that when $W = (1 - R)/2$, the inverse of the success probability becomes polynomial, hence the algorithm runs in polynomial time and the workfactor is exactly equal to 1. Indeed, if we force k bits to be equal to zero, because of the randomness assumptions, the remaining part of the vector (subvector of size $n - k$) will typically have half of its bits equal to 1, which gives a relative weight of $(1 - R)/2$.

Moreover, when $w > (n - k)/2$, we can reduce the problem to this case. Indeed, instead of forcing positions of the information set to have a zero error, we can force some positions to contain an error. To do this, we just add the corresponding columns to the targeted syndrome. Hence, when trying to solve the problem with $w > (n - k)/2$, we can apply this to $w - (n - k)/2$ positions and reduce to the case $w = (n - k)/2$ which runs in polynomial tine. Therefore, for $W \in [\frac{1-R}{2}, \frac{1}{2}]$, the algorithm solves the problem in polynomial time.

In summary, we see in Figure 8.1 the complexity of Prange's algorithm depending on the value of W for the case $R = 0.5$, and depending on the value of R for $W = W_{GV}(R)$. Note that other values of R give similar curves.

8.4 Combining both approaches

8.4.1 General idea

Now we have on one hand Prange's algorithm that uses the linear algebraic structure of the problem but succeeds with low probability, on the other hand a combinatorial approach which is more costly in general, but as we have seen, this approach can be efficient in certain special regimes, namely when the rate is large (R close to 1) and when we are looking for a large number of solutions.

In order to improve the complexity of the algorithm, two parameters of Prange's information set decoding algorithm were relaxed.

1. As we have seen, the complexity of Prange's algorithm comes directly from the extremely low probability that a random information set contains no error position. Hence, a possibility is to relax the constraint on the partition of the errors, in order to increase the success probability of Prange's algorithm. Instead of looking for an information set containing none of the error positions, we accept information sets that contain a small, though non-zero, number of errors p.

2. Another possible generalisation is to consider a subset of positions larger than an information set. Hence, instead of having a proper information set, *i.e.* a subset of positions $\mathcal{I}$ of size k that uniquely defines each codeword, we consider a subset $\mathcal{I}$ of $k + \ell$ positions and look for codewords

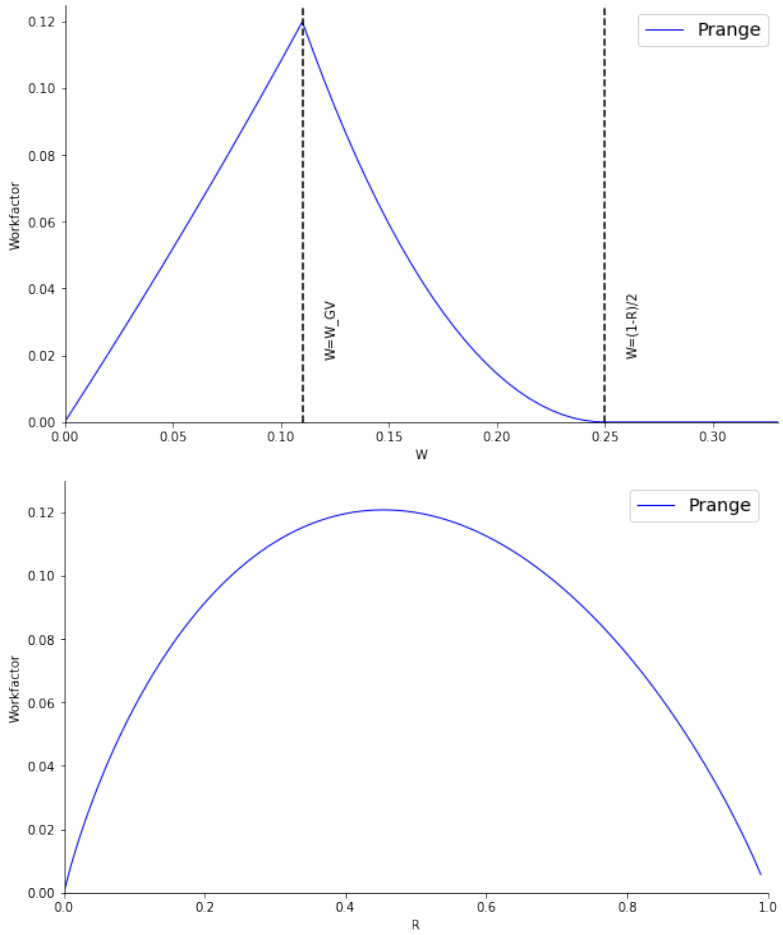


Figure 8.1: Workfactor of Prange's algorithm: for $R = 0.5$ and variable W (above) and for different variable R with $W = W_{GV}(R) = h^{-1}(1-R)$ (below).

at distance p on this subset. This idea alone would only decrease the success probability, but combined with the previously stated generalisation which allows the information set to contain a few errors, this can help improve the complexity by taking advantage of the efficiency of the combinatorial approach for specific decoding regimes.

Hence, given a noisy codeword $\boldsymbol{y}$ of the form $\boldsymbol{y} = \boldsymbol{c} + \boldsymbol{e}$, with $\boldsymbol{c} \in \mathscr{C}$ and $\mathrm{w}_H(\boldsymbol{e}) = w$, to find $\boldsymbol{c}$, we proceed as follows.

1. Pick a random subset $\mathcal{I} \subseteq [\![1, n]\!]$ of size $k + \ell$. Suppose that there are at most p error positions in $\mathcal{I}$, *i.e.* $\mathrm{w}_H(\boldsymbol{e}_{\mathcal{I}}) = p$.

2. Decode in the code restricted to $\mathcal{I}$, *i.e.* find all possible codewords $\boldsymbol{c}' \in \mathscr{C}$ such that $\mathbf{w}_H(\boldsymbol{c}'_{\mathcal{I}} - \boldsymbol{y}_{\mathcal{I}}) = p$.

3. For each of these codewords, check if $\mathbf{w}_H(\boldsymbol{c}' - \boldsymbol{y}) = w$. If none of them fulfills the constraint, start again from step 1 with a different choice of $\mathcal{I}$.

Note that Step 2 is exactly equivalent of decoding at distance p on the subcode of length $k+\ell$ defined as the restriction to the positions of $\mathcal{I}$. The rate of this code is $(k+\ell)/k$ which is close to 1, hence the combinatorial techniques introduced earlier can be applied. Moreover, in this setting we look for all solutions, hence can benefit from the numerous number of solutions if the parameters are chosen properly.

The hope is that the extra cost induced by this step of decoding an error of very low weight in a subcode of high rate will be compensated by the gain in the probability of success.

8.4.2 Generalised information set decoding algorithm

In terms of syndrome decoding, the ideas described here adapts as follows. Instead of a full Gaussian elimination we perform a partial Gaussian elimination (corresponding to the positions not in the information set). This yields Algorithm 16.

Algorithm 16: Generalised information set decoding

Input: $\boldsymbol{H} \in \mathbf{F}_2^{(n-k)\times n}$, $\boldsymbol{s} \in \mathbf{F}_2^{n-k}$, $w \in \mathbf{N}$
while $true$ **do**
  Choose $\boldsymbol{P}$ a random $(n-k)\times(n-k)$ permutation matrix.
  Let $\boldsymbol{S}$ be an invertible matrix such that $\boldsymbol{SHP}$ has an $n-k-\ell$ identity matrix on the top left side.
  Define $\boldsymbol{H}', \boldsymbol{H}'', \boldsymbol{s}', \boldsymbol{s}''$ as on Figure 8.2.
  Compute $\mathcal{S} = \{\boldsymbol{e}'' \in \mathbf{F}_2^{k+\ell} \mid \boldsymbol{H}''\boldsymbol{e}''^{\mathsf{T}} = \boldsymbol{s}''^{\mathsf{T}} \text{ and } \mathbf{w}_H(\boldsymbol{e}'') = p\}$.
  for $\boldsymbol{e}'' \in \mathcal{S}$ **do**
    Let $\boldsymbol{e}' = \boldsymbol{e}''\boldsymbol{H}'^{\mathsf{T}} + \boldsymbol{s}''$.
    if $\mathbf{w}_H(\boldsymbol{e}') = w - p$ **then**
      return $(\boldsymbol{e}', \boldsymbol{e}'')\boldsymbol{P}^{\mathsf{T}} \in \mathbf{F}_2^n$

Correctness. The algorithm returns a solution of the form $\boldsymbol{e} = (\boldsymbol{e}', \boldsymbol{e}'')\boldsymbol{P}^{\mathsf{T}}$. Let us check that this is indeed a solution to the syndrome decoding problem.

The vectors $\boldsymbol{e}', \boldsymbol{e}''$ are chosen such that the following conditions are fulfilled.

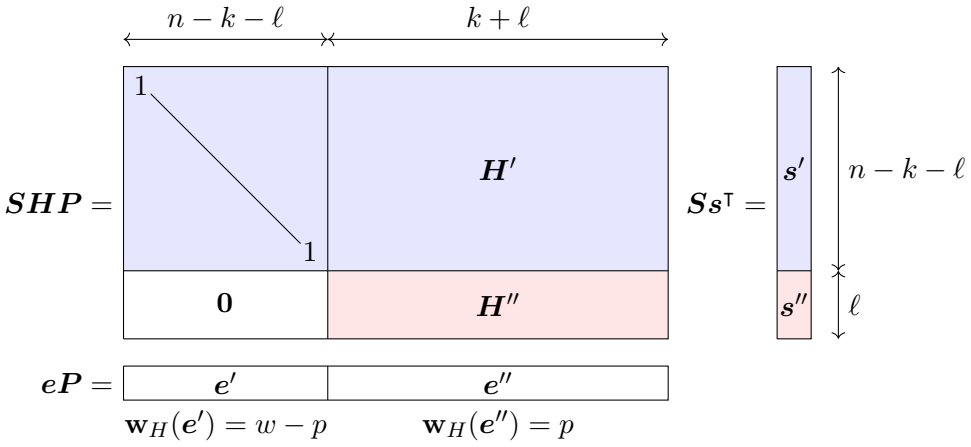


Figure 8.2: Generalised information set decoding algorithm

$$\begin{cases} \boldsymbol{e}'^\intercal = \boldsymbol{s}'^\intercal + \boldsymbol{H}'\boldsymbol{e}''^\intercal \\ \boldsymbol{H}''\boldsymbol{e}''^\intercal = \boldsymbol{s}''^\intercal \end{cases}$$

With this conditions, we have $\boldsymbol{He}^\intercal = \boldsymbol{s}^\intercal$.

$$\begin{cases} \boldsymbol{e}'^\intercal + \boldsymbol{H}'\boldsymbol{e}''^\intercal = \boldsymbol{s}'^\intercal \\ \boldsymbol{H}''\boldsymbol{e}''^\intercal = \boldsymbol{s}''^\intercal \end{cases} \iff \begin{pmatrix} \boldsymbol{I}_{n-k-\ell} & \boldsymbol{H}' \\ \mathbf{0} & \boldsymbol{H}'' \end{pmatrix} \begin{pmatrix} \boldsymbol{e}'^\intercal \\ \boldsymbol{e}''^\intercal \end{pmatrix} = \begin{pmatrix} \boldsymbol{s}'^\intercal \\ \boldsymbol{s}''^\intercal \end{pmatrix}$$

$$\iff \boldsymbol{SHP}(\boldsymbol{e}', \boldsymbol{e}'')^\intercal = \boldsymbol{Ss}^\intercal$$

$$\iff \boldsymbol{He}^\intercal = \boldsymbol{s}^\intercal$$

Concerning the weight constraint, the algorithm yields $\mathbf{w}_H(\boldsymbol{e}') = w - p$ and $\mathbf{w}_H(\boldsymbol{e}'') = p$, hence $\mathbf{w}_H((\boldsymbol{e}', \boldsymbol{e}'')) = w$ and the permutation does not affect the weight. Hence this algorithm provides a correct solution to the syndrome decoding problem.

A general scheme. As we can see, this is a generalisation of Prange's algorithm, which corresponds to the choice of parameters $p = 0, \ell = 0$.

Note that the *search step* corresponding to the 5th line of Algorithm 16 does not specify which method is used to search for these elements. The complexity of the subroutine used at this step impacts the choice of p and ℓ to reach the best complexity trade-off.

Hence, the algorithm presented here is a general scheme and different choices of subroutine yield to different complexities.

8.4.3 Using exhaustive search

Lee and Brickell. The first generalisation of Prange's information set decoding algorithm was introduced by Lee and Brickell [LB88] and corresponds to the special case $\ell = 0$ (*i.e.* only using the first improvement). It uses exhaustive search to perform the search step.

Each call to the search step enumerates all $\binom{k}{p}$ possible values of $\boldsymbol{e}''$. Hence each run of the while look takes time $\mathcal{T} = \mathcal{O}\left(\binom{k}{p}\right)$.

The probability of success of one loop run is the probability that a random error of weight w splits in a subvector of length k and weight p and a subvector of length $n-k$ and weight $w-p$, namely

$$\mathcal{P} = \frac{\binom{k}{p}\binom{n-k}{w-p}}{\binom{n}{w}}.$$

Hence the overall average complexity is $\mathcal{T}/\mathcal{P}$ which yields the following workfactor.

$$\mathcal{F}_{LB} = h(W) - (1-R)h\left(\frac{W-P}{1-R}\right),$$

where $P \stackrel{\text{def}}{=} p/n$ denotes the relative value corresponding to p.

An optimised choice of p improves the running time compared to Prange's original algorithm.

Leon. The first mention of the second generalisation ($\ell > 0$) is due to Leon [Leo88] in the context of small weight codeword finding. His algorithm slightly differs from Algorithm 16 because it supposes that the p errors in the information set are not among the ℓ additional positions. Still, forgetting about this difference, we can make a complexity analysis of the generalises information set decoding algorithm with exhaustive search.

Each call to the search step enumerates all $\binom{k+\ell}{p}$ possible values of $\boldsymbol{e}''$. Hence each run of the while look takes time $\mathcal{T} = \mathcal{O}\left(\binom{k+\ell}{p}\right)$.

The probability of success of one loop run is

$$\mathcal{P} = \frac{\binom{k+\ell}{p}\binom{n-k-\ell}{w-p}}{\binom{n}{w}}.$$

Hence the overall average complexity is $\mathcal{T}/\mathcal{P}$ which yields the following workfactor.

$$\mathcal{F}_{Leon} = h(W) - (1-R-L)h\left(\frac{W-P}{1-R-L}\right),$$

where $P \stackrel{\text{def}}{=} p/n$ and $L \stackrel{\text{def}}{=} \ell/n$ are the relative notations for p and ℓ respectively.

8.4.4 Using birthday decoding

But further improvements comes when the birthday decoding algorithm is used in the search step of the information set decoding. This idea was introduced independently by Stern [Ste88] and Dumer in [Dum89]. As Leon in [Leo88], they were supposing that the ℓ additional positions of the information set did not contain any of the p errors but this does not make a big difference. An intermediary approach known as ball-collision decoding [BLP11] considers a fixed small amount of error among the ℓ additional positions. Finally, the generalised information set decoding as presented in Algorithm 16 using birthday decoding in the search step corresponds to the proposal from Finiasz and Sendrier [FS09]. This more general version is what is today often referred to as the Stern and Dumer's algorithm, since the general idea of using birthday decoding in the search step was theirs.

In this algorithm, each run of the while loop corresponds to the birthday algorithm introduced in Section 8.2.2. The matrix $\boldsymbol{H}''$ is divided in two halves $\boldsymbol{H}''_1$ and $\boldsymbol{H}''_2$, and the following sets are constructed.

$$\mathcal{L}_1 = \{\boldsymbol{e}_1 \boldsymbol{H}''^{\mathsf{T}}_1 \,|\, \boldsymbol{e}_1 \in \mathbf{F}_2^{(k+\ell)/2}, \mathbf{w}_H(e_1) = p/2\},$$

$$\mathcal{L}_2 = \{\boldsymbol{e}_2 \boldsymbol{H}''^{\mathsf{T}}_2 + \boldsymbol{s}'' \,|\, \boldsymbol{e}_2 \in \mathbf{F}_2^{(k+\ell)/2}, \mathbf{w}_H(e_2) = p/2\}.$$

The set $\mathcal{S}$ corresponds to all collisions, *i.e.* $\mathcal{S} = \mathcal{L}_1 \cap \mathcal{L}_2$.

Each run of the loop has time complexity $\mathcal{T} = \mathcal{O}\left(\max(|\mathcal{L}|, |\mathcal{L}|^2/2^\ell)\right)$, where $|\mathcal{L}| = \binom{(k+\ell)/2}{p/2}$ is the size of $\mathcal{L}_1$, $\mathcal{L}_2$. This yields $|L|^2/2^\ell$ solutions to the subproblem. The memory complexity is $\mathcal{O}(\mathcal{L})$. For this approach to be the most efficient, we need to choose p and ℓ such that $|\mathcal{L}| = 2^\ell$ which yields

$$L = \frac{R+L}{2} h\left(\frac{P}{R+L}\right). \tag{8.1}$$

The probability of success is the probability that an error of weight w splits correctly, that is with p errors evenly distributed among the two halves of the information set, and $w-p$ errors in the remaining $n-k-\ell$ positions. This gives

$$\mathcal{P} = \frac{\binom{(k+\ell)/2}{p/2}^2 \binom{n-k-\ell}{w-p}}{\binom{n}{w}}.$$

Hence, we obtain the following workfactor.

$$\mathcal{F}_{Dumer} = h(W) + L - (R+L)h\left(\frac{P}{R+L}\right) - (1-R-L)h\left(\frac{W-P}{1-R-L}\right),$$

with the constraint binding L and P mentioned in Equation (8.1). Note that with this equation, we can directly compute P for a fixed value of L. Hence, for a given R and W, we can try all values of L to optimise the workfactor. In practice, the workfactor is a unimodal function of L hence one can efficiently find the optimal value.

This method approach, with the right optimisation of the parameters P and L, improves the asymptotic coefficient, as we can see on Figure 8.3.

8.5 Further improvements of ISD

Dumer's algorithm is a reference algorithm that generalises Prange's approach and improves its asymptotic complexity. In the last decades, several new proposals were made to further reduce the complexity exponent by adding some changes to Dumer's idea. We present here different possibilities of improvement.

All these improvements concern the search step of the algorithm. Hence, the scheme of the generalised information set decoding scheme remains the same, as presented above. A change of the algorithm used in the search step usually induces a change in the equations binding the parameters, and hence (hopefully) achieves a lower total complexity.

8.5.1 Recursive birthday algorithm

Stern and Dumer's idea to use birthday collision search to solve the search step of the information set decoding can be generalised. The main idea of the birthday decoding is to divide the matrix $\boldsymbol{H}''$ in two halves, construct two lists and find collisions between the two. But to construct the two initial lists, one can recursively apply the birthday algorithm (hence dividing $\boldsymbol{H}''$ in 4, 8, etc.). This idea is first stated by Wagner in [Wag02] and used in [CJ04] to cryptanalyse a code-based signature. This approach will be introduced in more details in Chapter 9.

8.5.2 Using representations

Another possible improvement that directly comes from the study of the knapsack problem. Birthday decoding consists in searching for an error $\boldsymbol{e}'' \in \mathbf{F}_2^{k+\ell}$ of weight p written as the sum of two vectors of weight $p/2$, with disjoint

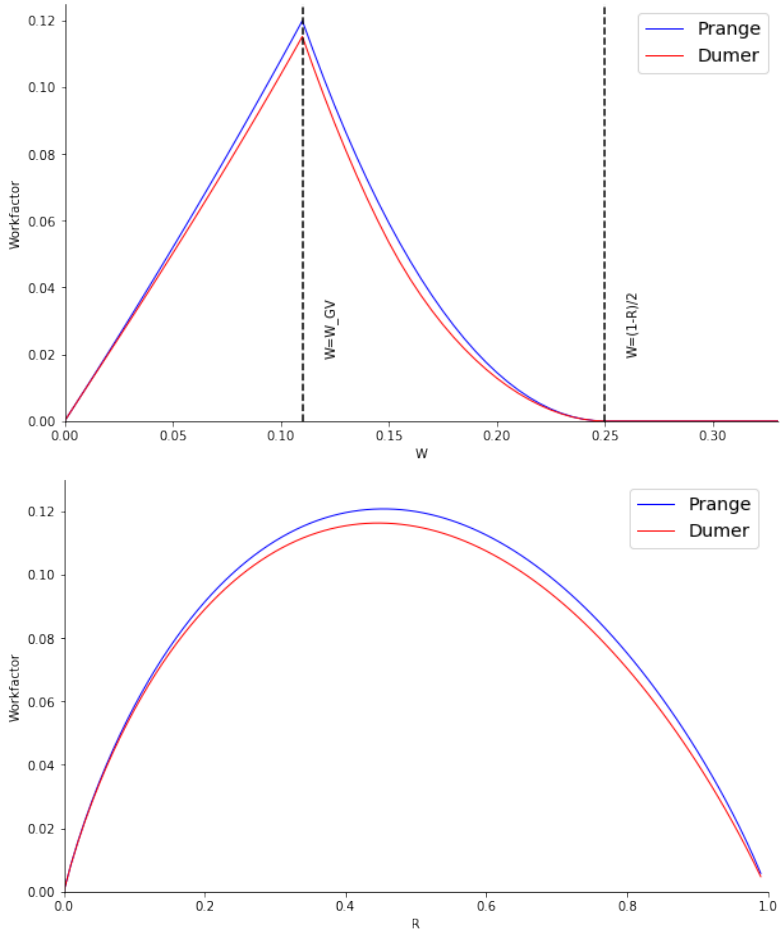


Figure 8.3: Workfactor of Prange and Dumer's algorithms: for $R = 0.5$ and variable W (above) and for different variable R with $W = W_{GV}(R) = h^{-1}(1 - R)$ (below).

supports: one vector has all its non-zero entries on the left half, the other on the right half, as illustrated in Figure 8.4.

In [HJ10], the authors propose to improve birthday decoding by looking the error of weight p as the sum of two vectors of weight $p/2$, but whose support is not restricted to the left or right half. This is illustrated in Figure 8.5.

Intuitively, this operation increases the search space, so it should not lower the complexity. However, a consequence of this choice is that for each vector $\boldsymbol{e}'' \in \mathbf{F}_2^{k+\ell}$ of weight p, there are many combinations of vectors $\boldsymbol{e}_1'', \boldsymbol{e}_2''$ each of weight $p/2$ that verify $\boldsymbol{e}_1'' + \boldsymbol{e}_2'' = \boldsymbol{e}''$. A pair of such vectors $(\boldsymbol{e}_1'', \boldsymbol{e}_2'')$ is called a *representation* of $\boldsymbol{e}''$. We can compute the number of representations of each

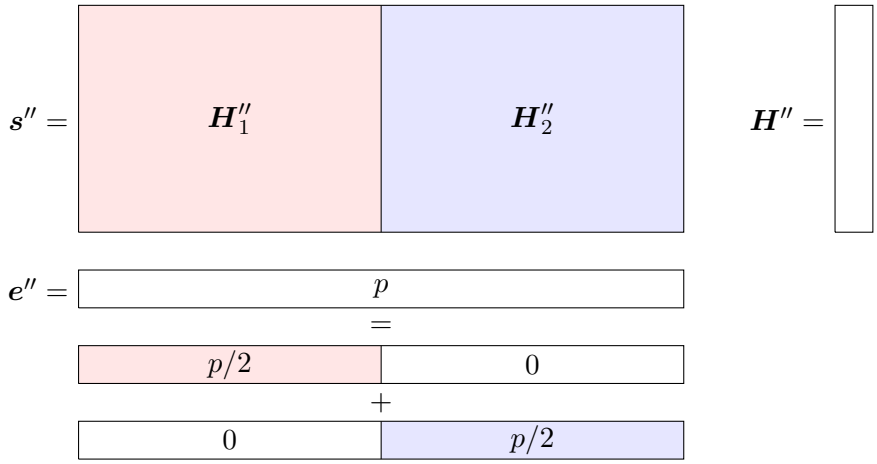


Figure 8.4: Classical birthday decoding with disjoint supports

vector, namely $\binom{p}{p/2}$.

But there is no need to find *all* representation of each vectors, since they would all lead to the same solution. Therefore, we add an additional criterion that the representations should fulfill, and design it such that, on average, we find exactly one representant of each vector $\boldsymbol{e}''$. This is usually referred to as "filtering".

For instance, if the number of representations of each vector is 2^r, we keep only the representations such that the corresponding syndrome fits a predefined target value on its first r bits. Because of the randomness assumption on the code, with this criteria there will only be one representation of $\boldsymbol{e}''$ on average. But with this simple "filtering" criterion, we are already one step closer to the goal (matching the target syndrome) because we already ensure that the syndrome will be matched on r bits. Hence, this changes the balance between the parameters and we gain something. Then, a thorough complexity analysis and a wise choice of parameters permits to make sure that this gain compensates the loss due to the increase of the search space.

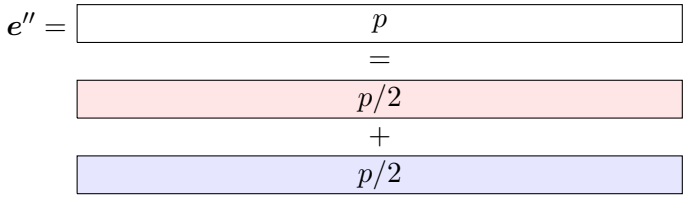


Figure 8.5: Birthday decoding without support restriction [MMT11]

In [BCJ11], the authors remark that on average, the sum of two vectors

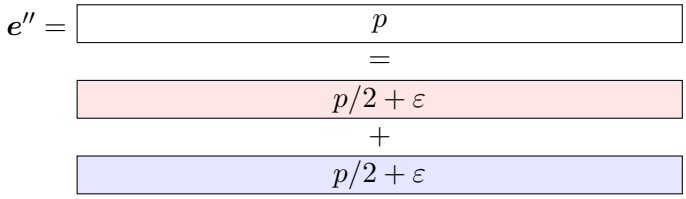


Figure 8.6: Birthday decoding as in [BJMM12]

of weight $p/2$ (with no restriction on the support, as in [MMT11]) is slightly less than p. Indeed, there might be an overlap in the support of $\boldsymbol{e}_1''$ and $\boldsymbol{e}_2''$ and hence the two 1s cancel out. Hence, in order to maximise the chances to obtain a vector of weight p at the end, one should pick $\boldsymbol{e}_1''$ and $\boldsymbol{e}_2''$ as vectors of weight $p/2 + \varepsilon$, with ε a small positive value, as in Figure 8.6.

This "$1 + 1 = 0$" trick is included in the search step of the information step algorithm in [BJMM12], together with a 3-level recursion in the birthday algorithm as suggested above.

This yields an improvement over Dumer's algorithm. We compare the complexity in Figure 8.7. Here the complexity of the BJMM algorithm [BJMM12] is computed using the CaWof software [Can16].

8.5.3 Nearest neighbour search

Finally, a recent line of work [MO15; BM17; BM18] proposes to use nearest neighbour search instead of collision search in the search step of the ISD algorithm. Instead of looking for collisions in two lists, one looks for a couple of elements (one in each list) that are close in terms of Hamming distance. The collision search is just a special case where we require neighbours to be at distance zero.

According to the results of these papers, this yields an improvement in the workfactor, compared to [BJMM12]. However, this approach adds a super-polynomial factor to the complexity, which is not reflected in the asymptotic exponent but could still be very significant in practice. For now, it is not clear if this approach is a gain of complexity to solve instances of cryptographic size.

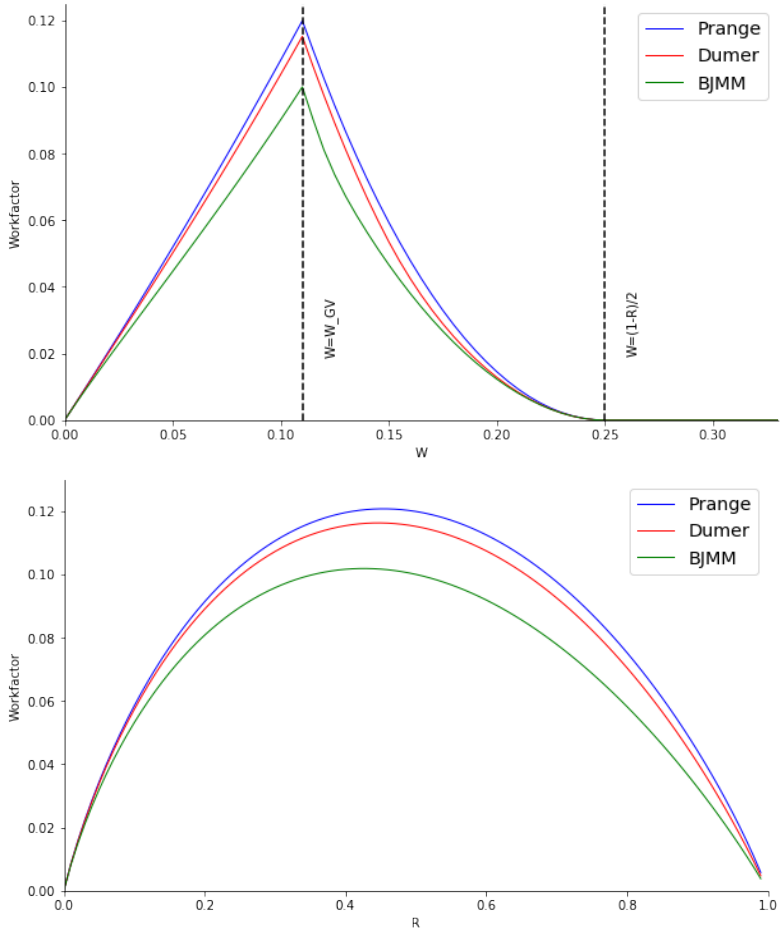


Figure 8.7: Workfactor of Prange and Dumer and BJMM's algorithms: for $R = 0.5$ and variable W (above) and for different variable R with $W = W_{GV}(R) = h^{-1}(1-R)$ (below).

Chapter 9

Ternary syndrome decoding with large weight errors

We have seen that the syndrome decoding problem is at the core of most code-based cryptosystems. In Chapter 8, we have presented the main algorithms to solve this problem over $\mathbf{F}_2$. In this chapter, we study this problem over $\mathbf{F}_q$ for an integer $q \geqslant 3$, and we will especially focus on the ternary case. We will see that there is a fundamental difference with the previous chapter. In the binary case, we have the syndrome decoding problem is symmetric with respect to the weight parameter $W \in [0, 1]$. Hence we can restrict the study to the case $W < 1/2$. This does not apply anymore in the case $q \geqslant 3$.

We will see that the algorithms presented in Chapter 8 generalise to the case $W < 1/2$, but we have to come up with a new proposal of algorithm for the case of *large weight* $W > 1/2$. The problem of ternary syndrome decoding in large weight has been introduced in the Wave signature scheme. In this work, we perform the first algorithmic study of this problem. As a result of our analysis, we will see that the original parameter proposed for the Wave scheme do not reach the claimed security, and we propose new parameters. More importantly, we show that for a fixed key size, the ternary syndrome decoding problem in large weight is harder than its binary counterpart.

Related publication: Bricout, Chailloux, Debris-Alazard and Lequesne, *Ternary Syndrome Decoding with Large Weight*, International Conference on Selected Areas in Cryptography 2019 [BCDL19].

Contents

9.1 Information set decoding for $q \geqslant 3$

9.1.1 Asymmetry of the non-binary case

The syndrome decoding problem has been introduced in Chapter 1 and discussed in Chapter 8. This problem has been thoroughly studied in its binary setting, because most code-based cryptosystem rely on the hardness of binary syndrome decoding as a security hypothesis.

As explained in Remark 8.2, in the case $q = 2$, the problem $\mathrm{SD}(2, R, W)$ with $W > 1/2$ can be reduced to $\mathrm{SD}(2, R, W')$ with $W' = 1 - W < 1/2$. Hence, the problem is symmetric with respect to the variable W. As a consequence, the literature focuses on optimising the algorithms to solve the problem for small values of W and especially $W < (1 - R)/2$ where the problem has exponential complexity. The case $W > 1/2$ is strictly equivalent.

But for the syndrome decoding problem defined over $\mathbf{F}_q$ with $q > 2$ the property that ensures symmetry does not hold, and as we will see, the large weight case behaves differently from the small weight case. This is illustrated in Figure 9.1. The loss of symmetry is easily understandable since the Hamming metric is very poor: it only distinguishes between the zero and non-zero values. In the binary case, this corresponds to 0 and 1. But in the q-ary case, there are $q - 1$ possible non-zero values. Hence, when $q \geqslant 3$, we can see that there is an inherent asymmetry between having small and large weight.

It is worth noting that the case $q \geqslant 3$ has received much less attention than the binary case, and no attention at all was given to the large weight regime. One possible explanation for this is that there were only few cryptographic applications for the general case. Hence, the claims of worst case complexities in the literature only refer to the case of weight $W < 1/2$, but as we can see on Figure 9.1 the worst case complexity is in general achieved for large weight.

However, in 2019, a new signature scheme named Wave was proposed in [DST19], based on the difficulty of syndrome decoding on a ternary alphabet and with large weight. This scheme is the first cryptographic scheme that relies on the hardness of *large* weight syndrome decoding.

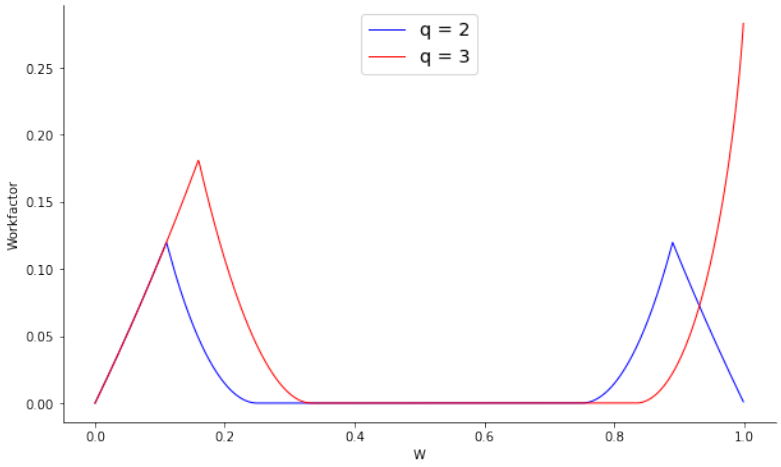


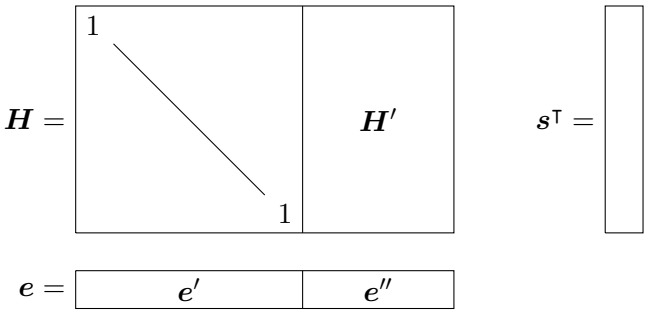
Figure 9.1: Workfactor of Prange's algorithm: for $R = 0.5$ and variable W, for $q = 2$ and $q = 3$.

9.1.2 Adaptation of Prange's algorithm

9.1.2.1 Algorithm

In Section 8.3, we presented Prange's algorithm for binary syndrome decoding. Prange's approach adapts straightforwardly to the q-ary case. This generalisation was originally presented by Coffey and Goodman in [CG90].

Let us recall the main idea behind Prange's algorithm. We perform a Gaussian elimination on the parity-check matrix $\boldsymbol{H}$. Let us write the error vector in two parts, $\boldsymbol{e}'$ corresponding to the identity part and $\boldsymbol{e}''$ corresponding to the remaining part. Here, we omit the matrices corresponding to the permutation ($\boldsymbol{P}$) and Gaussian elimination ($\boldsymbol{S}$) introduced in Section 8.3 and focus on the core of the algorithm.



If we fix a particular value for $\boldsymbol{e}'' \in \mathbf{F}_q^k$, there exist one unique solution $\boldsymbol{e}' \in \mathbf{F}_q^{n-k}$ such that $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$. Indeed, we have $\boldsymbol{e}' = \boldsymbol{s} - \boldsymbol{e}''\boldsymbol{H}''^\intercal$. Because $\boldsymbol{H}$

and $\boldsymbol{s}$ are chosen randomly, for a fixed choice of $\boldsymbol{e}''$, the value of $\boldsymbol{e}'$ follows a uniform distribution over $\mathbf{F}_q^{n-k}$. Hence, the weight distribution of $\boldsymbol{e}'$ is the sum of $n-k$ Bernoulli trials of success probability $(q-1)/q$. On average, we will find a solution $\boldsymbol{e}'$ of weight $\frac{(q-1)}{q}(n-k)$.

Therefore, the weight of $\boldsymbol{e}$ is on average $\mathbf{w}_H(\boldsymbol{e}'') + \frac{(q-1)}{q}(n-k)$. The weight of $\boldsymbol{e}'' \in \mathbf{F}_q^k$ depends entirely of our choice of $\boldsymbol{e}''$ and can be anything between 0 and k. This provides a polynomial time algorithm to find a solution of weight $w \in [\![\frac{q-1}{q}(n-k), \frac{q-1}{q}(n-k)+k]\!]$. One should proceed as follows.

1. Randomly choose a value $\boldsymbol{e}'' \in \mathbf{F}_q^k$ of weight $w - \frac{q-1}{q}(n-k)$;
2. Compute the value $\boldsymbol{e}'$ such that $\boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal$;
3. If this value is of weight $\frac{q-1}{q}(n-k)$, which is often the case, then $\boldsymbol{e} = (\boldsymbol{e}', \boldsymbol{e}'')$ provides a solution, otherwise restart from step 1 after randomly permuting the columns of $\boldsymbol{H}$.

9.1.2.2 Complexity

This algorithm returns a solution $\boldsymbol{e}$ of weight $w \in [\![\frac{q-1}{q}(n-k), \frac{q-1}{q}(n-k)+k]\!]$ in polynomial time.

If we are looking for a solution of weight $w < \frac{q-1}{q}(n-k)$, we can use the exact same algorithm with the choice of $\boldsymbol{e}''$ as the zero vector, but the probability to obtain $\boldsymbol{e}'$ of weight w is exponentially small, hence we have to run the algorithm an exponential number of time. Similarly, if we are looking for a solution of weight $w > \frac{q-1}{q}(n-k)+k$, we have to choose $\boldsymbol{e}''$ as a full-weight vector and run the algorithm an exponential number of time until we luckily obtain $\boldsymbol{e}'$ of weight $w-k$. Both cases yield an exponential time algorithm.

We can compute the success probability in each case. Let w' and w'' denote the weights that we attribute to $\boldsymbol{e}'$ and $\boldsymbol{e}''$ in each of the three cases. Namely:

- if $w < \frac{q-1}{q}(n-k)$, we set $w' \stackrel{\text{def}}{=} w$, $w'' \stackrel{\text{def}}{=} 0$;
- if $w \in [\![\frac{q-1}{q}(n-k), \frac{q-1}{q}(n-k)+k]\!]$, we set $w' \stackrel{\text{def}}{=} \frac{q-1}{q}(n-k)$, $w'' \stackrel{\text{def}}{=} w - w'$;
- if $w > \frac{q-1}{q}(n-k)+k$, we set $w' \stackrel{\text{def}}{=} w-k$, $w'' \stackrel{\text{def}}{=} k$;

In all cases we have $w = w' + w''$. The probability of success of the algorithm is the probability that for a choice of $\boldsymbol{e}''$ of weight w'', the value $\boldsymbol{e}'$ obtained is indeed of weight w'. As we said, because of the randomness assumption

on $\boldsymbol{H}$ and $\boldsymbol{s}$, vector $\boldsymbol{e}'$ follows a uniform distribution over $\mathbf{F}_q^{n-k}$, hence the probability that its weight is equal to w' is

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

Here, the numerator is just the number of vectors of length $n-k$ and weight w', and the numerator q^{n-k} corresponds to enumerating all words of length n. But because we know that the vector $\boldsymbol{e}'$ is a subvector of the solution $\boldsymbol{e}$ of length n and weight w, for small values of w this condition restricts the search space. Hence the minimum in the numerator.

The inverse of this probability yields the complexity of the algorithm. From this, we can deduce the workfactor of Prange's algorithm.

$$\begin{aligned}\mathcal{F}_{Prange}(q,R,W) = &\min\left\{(1-R)\log_2(q), h_2(W) + W\log_2(q-1)\right\} \\ &- (1-R)h_2\left(\frac{W'}{1-R}\right) - W'\log_2(q-1).\end{aligned}$$

Here, $W' = W$ for $W < \frac{q-1}{q}(1-R)$ and $W' = W - R$ for $W > \frac{q-1}{q}(1-R)+R$. For the intermediate weight, the workfactor is zero because the algorithm is polynomial. This is how we obtain the result of Figure 9.1 in the case $R = 1/2$. The workfactor for $q = 3$ for different rates is illustrated on Figure 9.2.

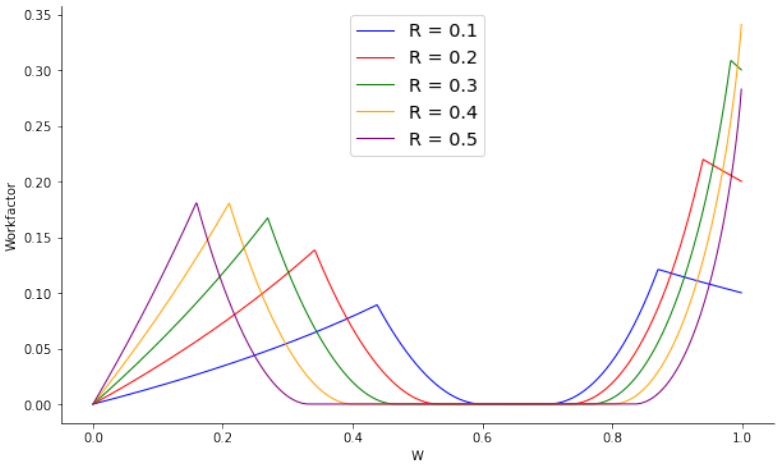


Figure 9.2: Workfactor of Prange's algorithm: for $q = 3$ and variable W, for different values of R.

9.1.2.3 Gilbert-Varshamov bound for large weight

In Chapter 8, we have seen that for $q = 2$ and $W < 1/2$, the complexity reaches a maximum for W such that $1 - R = h_2(W)$. This exactly corresponds to the case where the two terms in the minimum are equal. We have seen that this corresponds to the Gilbert-Varshamov bound, where on average there exists exactly one solution that matches the syndrome.

For $q \geqslant 3$, this extends straightforwardly for small weight: when $W < \frac{(q-1)}{q}(1-R)$, the complexity is exponential and reaches a maximum, when the two terms of the minimum are equal, in the workfactor equation. Again, this corresponds to the settings where there exists on average exactly one solution to the problem, *i.e.* $\binom{n}{w}q^w = q^{n-k}$. We can therefore extend the definition of the relative Gilbert-Varshamov bound for q-ary codes:

$$W_{GV} = h_q^{-1}(1-R),$$

where h_q denotes the q-ary entropy function (Definition refdef:entropy). Here, $h_q^{-1}(y)$ denotes the value $x \in [0, \frac{1}{q}]$ such that $h_q(x) = y$.

Moreover, as we can see on Figure 9.2, the workfactor also reaches a maximum value for high weights. For certain rates, we see a change of scope, due to the exact same reason as for small weight: at some point, due to the combinatorics, when the weight is too high, the search space is reduced (this is reflected by the minimum in the denominator). Hence, we can quantify this as the Gilbert-Varshamov bound for high weight, defined as the $w > \frac{q-1}{q}(n-k)$ such that $\binom{n}{w}q^w = q^{n-k}$.

Hence, we can define of the relative Gilbert-Varshamov bound for high weight for q-ary codes as

$$W_{GV+} = h_q^{-1}(1-R),$$

where $h_q^{-1}(y)$ denotes the value $x \in [\frac{q-1}{q}, 1]$ such that $h_q(x) = y$. For a fixed value of R, the maximum workfactor for high weight is reached when $W = W_{GV+}$, if this value is well-defined.

For some higher rates, the workfactor is strictly increasing with respect to W. In this case, the definition of W_{GV+} does not hold and the maximum complexity is reached when $W = 1$.

9.1.3 Generalised information set decoding algorithms

In Chapter 8, we presented the different improvement of Prange's algorithm. All of them extends to q-ary code for small weight. Stern and Dumer's algorithm was adapted by Peter in [Pet10]. Meurer adapted the BJMM algorithm in his dissertation thesis [Meu17]. Hirose [Hir16] proposed a generalization

of Stern's algorithm with May-Ozerov's approach (using nearest neighbors) and showed that for $q \geqslant 3$ this does not improve the complexity compared to Stern's classical approach. Later, Gueye, Klamti and Hirose [GKH17] extended the BJMM algorithm with May-Ozerov's approach and improved the complexity of the general SD problem. Again, note that these papers focus solely on the SD problem for small weight ($W < 0.5$).

The general scheme of the information set decoding algorithms has been stated in Section 8.4.2, particularly in Algorithm 16 and Figure 8.2. This scheme still holds for q-ary codes, even for high weight decoding.

Again, a key factor for the complexity is the probability that a solution of the sub-problem (*i.e.* an element $\boldsymbol{e}''$ found at step 5 of Algorithm 16) yields a solution $\boldsymbol{e}$ of weight w. Let us analyse this probability for q-ary codes.

Notation 9.1. *On an input $(\boldsymbol{H}, \boldsymbol{s})$ uniformly drawn at random, suppose that we have a vector $\boldsymbol{e}''$ such that $\boldsymbol{H}''\boldsymbol{e}''^{\intercal} = \boldsymbol{s}''^{\intercal}$ and $|\boldsymbol{e}''| = p$. Let $\boldsymbol{e}'^{\intercal} = \boldsymbol{s}'^{\intercal} - \boldsymbol{H}'\boldsymbol{e}''^{\intercal}$. We will denote*

$$\mathcal{P}_{p,\ell} \stackrel{def}{=} \mathbb{P}\left(|\boldsymbol{e}'| = w - p \mid |\boldsymbol{e}''| = p\right).$$

Proposition 9.2. *We have, up to a polynomial factor,*

$$\mathcal{P}_{p,\ell} = \frac{\binom{n-k-\ell}{w-p}(q-1)^{w-p}}{\min\left(q^{n-k-\ell}, \binom{n}{w}(q-1)^w q^{-\ell}\right)}.$$

Proof. This result is similar to what we had in the binary case. The numerator corresponds to the number of vectors $\boldsymbol{e}'$ of weight $w - p$. The denominator reflects the probability that $\boldsymbol{e}'^{\intercal} = \boldsymbol{s}'^{\intercal} - \boldsymbol{H}'\boldsymbol{e}''^{\intercal}$. For a typical random behavior, this is equal to $q^{n-k-\ell}$. But here we know that there is at least one solution. Therefore, we know that the number of vectors of weight $w - p$ is bounded from above by the number of vectors $\boldsymbol{e}$ such that $\boldsymbol{H}''\boldsymbol{e}''^{\intercal} = \boldsymbol{s}''^{\intercal}$. This explains the second term of the minimum. □

Proposition 9.3. *Assume that we have an algorithm that finds in time T a set $\mathcal{S}$ of solutions $\boldsymbol{e}''$ of weight p such that $\boldsymbol{H}''\boldsymbol{e}''^{\intercal} = \boldsymbol{s}''^{\intercal}$. The average running time of the generalised information set decoding algorithm is, up to a polynomial factor,*

$$T \cdot \max\left(1, \frac{1}{|\mathcal{S}| \cdot \mathcal{P}_{p,\ell}}\right).$$

Again, as we can see, all the parameters are entwined. The success probability $\mathcal{P}_{p,\ell}$ depends of p and ℓ, as well as the time T to find the set $\mathcal{S}$ of solutions.

The different improvements of Prange's algorithm all respect this general scheme. They differ by using different sub-algorithms to find the set $\mathcal{S}$ of solutions at step 5. We will see in the next section that we can also use this general framework to find solutions to the high weight problem.

9.1.4 ISD for $q \to \infty$

Finally, let us state a result from Canto-Torres [Can17] about the general decoding of q-ary codes. This result states that all ISD-based algorithms (Prange, Stern-Dumer, MMT, BJMM) converge to the same asymptotic complexity when $q \to \infty$. This means that for large values of q, the complexity improvement due to generalised ISD algorithms becomes negligible and the complexity converges to that of Prange's algorithm. Hence, the case $q = 3$ is the most interesting one, in the sense that this is the case where the difference in complexity obtained by using different ISD algorithms is the most significant.

This result is to be related with the fact that the Hamming metric becomes less meaningful as q grows larger. Indeed, the Hamming weight only counts the number of non-zero elements but not their partition. Hence, the Hamming weight loses a significant amount of information for large values of q. Therefore, $q = 3$ seems to be the best candidate to understand the structure of the non-binary case without losing too much information. Therefore, in the rest of this Chapter, we will focus on the case $q = 3$.

9.2 Large weight ternary syndrome decoding

9.2.1 Reduction to subset sum

In step 5 of Algorithm 16, we have a matrix $\boldsymbol{H}'' \in \mathbf{F}_q^{\ell \times (k+\ell)}$, a vector $\boldsymbol{s}'' \in \mathbf{F}_q^{\ell}$ and we want to compute a set $\mathcal{S} \subseteq \mathbf{F}_q^{k+\ell}$ of solutions $\boldsymbol{e}''$ of $\boldsymbol{H}''\boldsymbol{e}''^{\intercal} = \boldsymbol{s}''^{\intercal}$ such that $|\boldsymbol{e}''| = p$.

At first sight, this looks exactly like a Syndrome Decoding problem with inputs $\boldsymbol{H}''$ and $\boldsymbol{s}''$ so we could just recursively apply the best SD algorithm on this sub-instance. But the main difference is that, in this case, we want to find many solutions to the problem and not just one. We can state this as the Sub-ISD problem.

Problem 9.4 (Sub-ISD problem - Sub-SD(q, m, ℓ, p, L))**.**

Instance: m *vectors* $\boldsymbol{x}_i \in \mathbf{F}_q^{\ell}$ *for* $1 \leqslant i \leqslant m$*, a target vector* $\boldsymbol{s} \in \mathbf{F}_q^{\ell}$*.*

Output: L *solutions* $\boldsymbol{b}^{(j)} = (b_1^{(j)}, \dots, b_m^{(j)}) \in \mathbf{F}_q^m$ *for* $1 \leqslant j \leqslant L$*, such that for all* j*,* $\sum_{i=1}^{m} b_i^{(j)} \boldsymbol{x}_i = \boldsymbol{s}$ *and* $\mathbf{w}_H(\boldsymbol{b}^{(j)}) = p$*.*

We see that the step 5 of Algorithm 16 is exactly Sub-SD$(q, k + \ell, \ell, p, |\mathcal{S}|)$.

9.2.1.1 The subset sum problem

This problem is very close to another well known problem in the literature: the subset sum problem. The subset sum problem is defined as follows.

Problem 9.5 (Subset Sum problem - $\mathrm{SS}(q, m, \ell, L)$)**.**

Instance: m *vectors* $\boldsymbol{x}_i \in \mathbf{F}_q^\ell$ *for* $1 \leqslant i \leqslant m$*, a target vector* $\boldsymbol{s} \in \mathbf{F}_q^\ell$*.*

Output: L *solutions* $\boldsymbol{b}^{(j)} = (b_1^{(j)}, \ldots, b_m^{(j)}) \in \{0,1\}^m$ *for* $1 \leqslant j \leqslant L$*, such that for all* j*,* $\sum_{i=1}^m b_i^{(j)} \boldsymbol{x}_i = \boldsymbol{s}$*.*

We see that this problem differs from the Sub-ISD problem in two ways. First, the coefficients $b_i^{(j)}$ are in $\{0,1\}$ and not in $\mathbf{F}_q$. Secondly, there is no more weight constraint.

There is an extensive literature about the Subset Sum problem for specific parameter ranges. The number of solutions to the subset problem is on average $2^m/q^\ell$. The most studied case is $L = 1, q = 2^m, \ell = 1$ [HJ10; BCJ11]. These parameters correspond to the case where there is on average one solution, which is the hardest case.

The difficulty of the problem (for $\ell = 1$) decreases when the number of solutions becomes larger. For instance, when $q = \mathcal{O}(m)$, the complexity is polynomial [CFG89; GM91]. An intermediary case corresponds to a choice of $q = \mathcal{O}(()\, 2^{m^\varepsilon})$ for $0 < \varepsilon < 1$. In this case, there is an exponential number of solutions. In [Lyu05] an algorithm is proposed to solve this problem in subexponential time.

9.2.2 From large weight ISD to subset sum

Now, recall that our goal is to adapt the general information set decoding algorithm for large weight. We explained in Section 9.1.2.2 that Prange's algorithm can be used in the large weight case by looking for a solution $\boldsymbol{e}'' \in \mathbf{F}_q^k$ of full weight, *i.e.* $\mathrm{w}_H(\boldsymbol{e}'') = k$. For the same reason, in the generalised ISD scheme, we are looking for a set $\mathcal{S}$ of vectors $\boldsymbol{e}'' \in \mathbf{F}_q^{k+\ell}$ of weight $p = k + \ell$, in order to maximise the probability of success.

Hence, the search phase, corresponding to step 5 of Algorithm 16, corresponds exactly to Sub-SD$(q, k+\ell, \ell, k+\ell, |\mathcal{S}|)$.

In the particular case of $q = 3$, looking for vectors $\boldsymbol{e}''$ of full weight means that the entries of $\boldsymbol{e}''$ are to be taken among the two non-zero elements of $\mathbf{F}_3$: $\boldsymbol{e}'' \in \{1,2\}^{k+\ell}$. We see here the parallel with the subset sum problem. We can write the reduction formally.

Lemma 9.6. *If we have an algorithm that solves* $\mathrm{SS}(3, k+\ell, \ell, L$ *then we have an algorithm that solves Sub-SD*$(3, k+\ell, \ell, k+\ell, L)$ *with the same complexity.*

Proof. Let $\mathcal{A}$ be an algorithm that solves $\mathrm{SS}(3, k+\ell, \ell, L)$ and consider an instance $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k+\ell})$, $\boldsymbol{s}$ of Sub-ISD$(3, k+\ell, \ell, k+\ell, L)$. We want to find $b_1, \ldots, b_{k+\ell} \in \{1,2\}$ (where $\mathbf{F}_3 = \{0,1,2\}$) such that $\sum_{i=1}^{k+\ell} b_i \boldsymbol{x}_i = s$. Let $\boldsymbol{s}' = 2\boldsymbol{s} + \sum_i \boldsymbol{x}_i$ and let us run $\mathcal{A}$ on input $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{k+\ell}), \boldsymbol{s}'$. We obtain $b'_1, \ldots, b'_{k+\ell} \in$

$\{0,1\}$ such that $\sum_{i=1}^{k+\ell} b'_i \boldsymbol{x}_i = \boldsymbol{s}'$. Take $b_i = \frac{b'_i - 1}{2}$ for $1 \leqslant i \leqslant k+\ell$, where the division is done in $\mathbf{F}_3$ and return $(b_1, \ldots, b_{k+\ell})$.

Indeed, this gives a valid solution to the problem: the elements b_i belong to $\{1,2\}$ and we have:

$$\sum_{i=1}^{k+\ell} b_i \boldsymbol{x}_i = \sum_{i=1}^{k+\ell} \frac{b'_i - 1}{2} \boldsymbol{x}_i = \frac{\boldsymbol{s}'}{2} - \frac{\sum_{i=1}^{k+\ell} \boldsymbol{x}_i}{2} = \boldsymbol{s}.$$

□

Hence, in order to solve SD in large weight using Algorithm 16, it is sufficient to solve $\mathrm{SS}(3, k+\ell, \ell, |\mathcal{S}|)$.

Recall that the ISD problem is parametrised by two values, p and ℓ. We explained that we choose $p = k + \ell$ to solve the large weight case. This fixes the parameter p. We still need to choose the parameter ℓ. If we take $\ell = o(n)$, asymptotically this will boil down to Prange's algorithm as we presented before. Hence, if we wish to obtain a gain in the asymptotic complexity, we have to choose $\ell = \Theta(n) = \Theta(k)$ (as $R = k/n$ is fixed).

Therefore, we are in a regime where solving the subset sum problem requires exponential complexity. But we want to use the fact that we are in a situation where we are looking for many solutions. The more solutions we have to the Sub-SD problem, the higher the probability that one of them is a solution to the general SD problem. Hence, our approach consists in trying to find as many solutions as possible in the minimum amount of time. This approach is comparable with what is done in Dumer's algorithm in the low weight case. We are looking for L solutions in time $\mathcal{O}(L)$, *i.e.* we are looking for solutions in amortised time $\mathcal{O}(1)$. We present here how to achieve this using Wagner's algorithm.

9.2.3 Wagner's algorithm

We are trying to solve $\mathrm{SS}(3, k+\ell, \ell, L)$. We are interested in the average case, which means that all the vectors $\boldsymbol{x}_i$ are independent and follow a uniform law over $\mathbf{F}_3^\ell$.

9.2.3.1 Presentation of the algorithm

Wagner's algorithm, introduced in [Wag02], consists in recursively applying birthday search. We will denote $a \in \mathbb{N}^*$ the number of recursion levels. The classical birthday search ($a = 1$) consists in dividing the vectors $\boldsymbol{x}_j$ in two sets. In our case we start by dividing them in 2^a sets containing $(k+\ell)/2^a$ vectors each. Each such set consists of the vectors $\{\boldsymbol{x}_j, j \in \mathcal{I}_i\}$, where $\mathcal{I}_i$ is defined as follows.

Notation 9.7. *For $i \in [\![1, 2^a]\!]$, denote by $\mathcal{I}_i$ the sets $\mathcal{I}_i \stackrel{def}{=} [\![1 + \frac{(i-1)(k+l)}{2^a}, \frac{i(k+l)}{2^a}]\!]$. The sets $\mathcal{I}_i$ form a partition of $[\![1, k+\ell]\!]$.*

The first step of Wagner's algorithm is to compute for each $i \in [\![1, 2^a]\!]$ a list of L random linear combinations of elements of $\{\boldsymbol{x}_j, j \in \mathcal{I}_i\}$. We define the 2^a lists $(\mathcal{L}_i)_{1 \leqslant i \leqslant 2^a}$ of size L such that:

$$\forall i \in [\![1, 2^a]\!],\ \mathcal{L}_i \subseteq \left\{ \sum_{j \in \mathcal{I}_i} b_j \boldsymbol{x}_j \ :\ \forall j \in \mathcal{I}_i,\ b_j \in \{0, 1\} \right\} \text{ and } |\mathcal{L}_i| = L. \quad (9.1)$$

Each list $\mathcal{L}_i$ consists of L random elements of the form $\sum_{j \in \mathcal{I}_i} b_j \boldsymbol{x}_j$ where the randomness is on $b_j \in \{0, 1\}$. By construction, we make sure that given $\boldsymbol{y} \in \mathcal{L}_i$ we have access to the coefficients $(b_j)_{j \in \mathcal{I}_i}$ such that $\boldsymbol{y} = \sum_{j \in \mathcal{I}_i} b_j \boldsymbol{x}_j$. The running time to build theses lists is $O(L)$. Note that this constructions yields the constraint

$$L \leqslant 2^{(k+\ell)/2^a}. \quad (9.2)$$

Once we have computed these lists, we merge the lists two by two to obtain 2^{a-1} lists of size L. For every $p \in [\![1, 2^{a-1}]\!]$, create a list $\mathcal{L}_{2p-1,2p}$ from $\mathcal{L}_{2p-1}$ and $\mathcal{L}_{2p}$ such that:

$$\mathcal{L}_{2p-1,2p} \stackrel{\text{def}}{=} \left\{ \boldsymbol{y}_{2p-1} + \boldsymbol{y}_{2p} \ :\ \boldsymbol{y}_i \in \mathcal{L}_i \text{ and the last } t \text{ bits of } \boldsymbol{y}_{2p-1} + \boldsymbol{y}_{2p} \text{ are 0s.} \right\}.$$

If we want $\mathcal{L}_{2p-1,2p}$ to be typically of size L, we need to choose t such that

$$\frac{L^2}{3^t} = L,$$

which yields

$$L = 3^t. \quad (9.3)$$

Note that the construction of the last list $\mathcal{L}_{2^a-1,2^a}$ differs. We ask for the last t bits to be equal to those of $\boldsymbol{s}$.

We now have 2^{a-1} lists of size L. We repeat the merging operation $a-1$ more times, until we obtain one single list of size L. By construction, elements of this list will be equal to $\boldsymbol{s}$ on their last at bits. For these elements to be solutions, we need to choose

$$t = \ell/a. \quad (9.4)$$

Hence,

$$L = 3^{\ell/a}. \quad (9.5)$$

At each step, the running time to build the lists is $\mathcal{O}(L)$. This operation is repeated a times, with a a small integer. We can therefore state the following result.

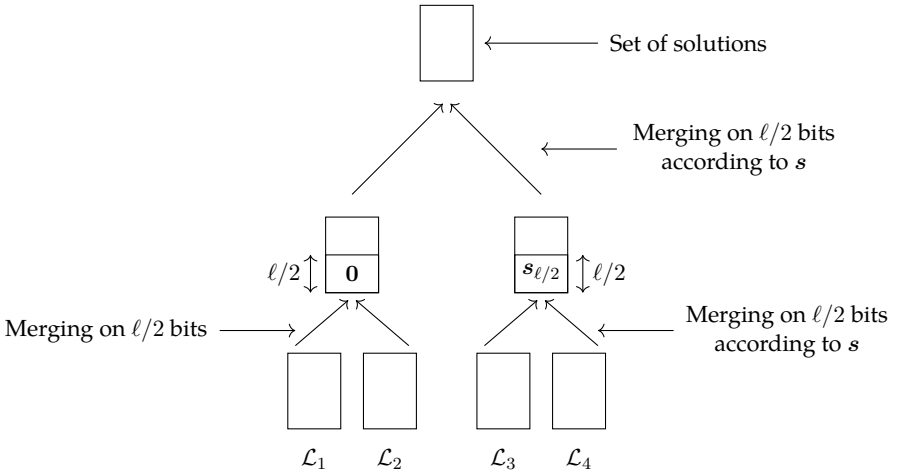


Figure 9.3: Wagner's algorithm with $a = 2$.

Theorem 9.8. *Fix $k, \ell \in \mathbb{N}^*$ and let a be any non zero integer such that*

$$3^{\ell/a} \leqslant 2^{(k+\ell)/2^a}. \tag{9.6}$$

The associated $\mathrm{SS}(3, k+\ell, \ell, 3^{\ell/a})$ problem can be solved in average time and space $O(3^{\ell/a})$.

Note that the constraint $3^{\ell/a} \leqslant 2^{(k+\ell)/2^a}$ comes from Equations (9.2) and (9.5). Since k and ℓ are fixed parameters of the problem, this restrains the choice of a. In practice, one will choose the largest integer a such that Equation (9.6) holds. But as we can see on Figure 9.4, the fact that a has to be an integer induces some discontinuity in the scope of the workfactor.

9.2.3.2 Smoothing of Wagner's Algorithm

We show here a refinement of Theorem 9.8 that reduces the discontinuity.

Proposition 9.9. *Let a be the largest integer such that $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$. If $a \geqslant 3$, the above algorithm can find 2^λ solutions in time $O(2^\lambda)$ with*

$$\lambda = \frac{\ell \log(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

We see that we retrieve the result of Theorem 9.8 when $3^{\ell/a} = 2^{(k+\ell)/2^a}$. Let us prove the proposition.

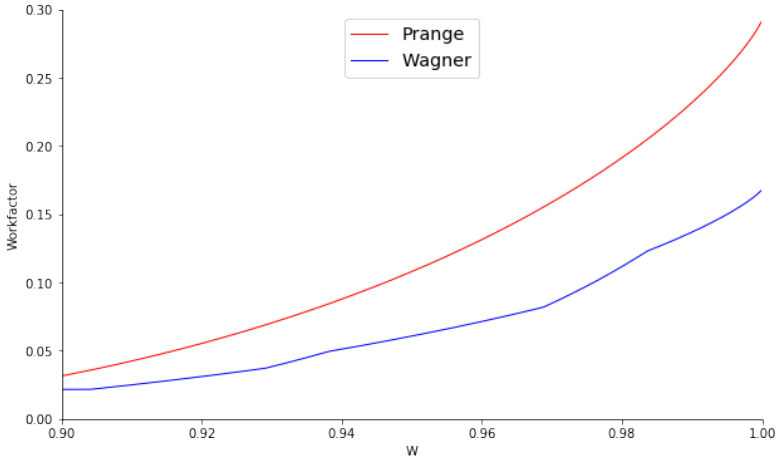


Figure 9.4: Workfactor of Wagner's algorithm: for $q = 3$, $R = 0.5$ and variable W.

Proof. The parameters k and ℓ are fixed. Let a be the largest integer such that $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$. Suppose that $a \geqslant 3$. In the classical version of Wagner's algorithm presented above, each list $\mathcal{L}_i$ at the bottom of the tree is of size L and represents only a subset of the $2^{(k+\ell)/2^a}$ possible combinations of the $(x_j)_{j\in\mathcal{I}_i}$. This explains the gap and the discontinuity.

Therefore, we consider a variant of Wagner's algorithm on a levels but with one difference: the lists at the bottom of the tree are of the maximal possible size: $2^{(k+\ell)/2^a}$. At all other levels of the tree, we want lists of size 2^λ for some value λ to be determined.

Hence, the first step of merging is performed on t bits, such that the merging two lists of size $2^{(k+\ell)/2^a}$ yields a list of size 2^λ. Therefore, we have to choose t such that

$$\frac{\left(2^{(k+\ell)/2^a}\right)^2}{3^t} = 2^\lambda \qquad \textit{i.e.} \qquad \frac{2(k+\ell)}{2^a} - t\log_2(3) = \lambda. \tag{9.7}$$

The other $(a-1)$ merging steps are designed such that merging two lists of size 2^λ gives a new list of size 2^λ, which means that we merge on $\lambda/\log_2(3)$ bits. However, in the final list we want to obtain solutions to the problem, which means that in total we have to put a constraint on all bits. Therefore, λ and t have to verify:

$$t + (a-1)\frac{\lambda}{\log_2(3)} = \ell. \tag{9.8}$$

Combining Equations (9.7) and (9.8) yields

$$\lambda = \frac{\ell \log_2(3)}{a-2} - \frac{k+\ell}{(a-2)2^{a-1}}.$$

It is easy to check that under the conditions $3^{\ell/(a-1)} < 2^{(k+\ell)/2^{a-1}}$ and $a \geqslant 3$, λ and t are positive which concludes the proof. □

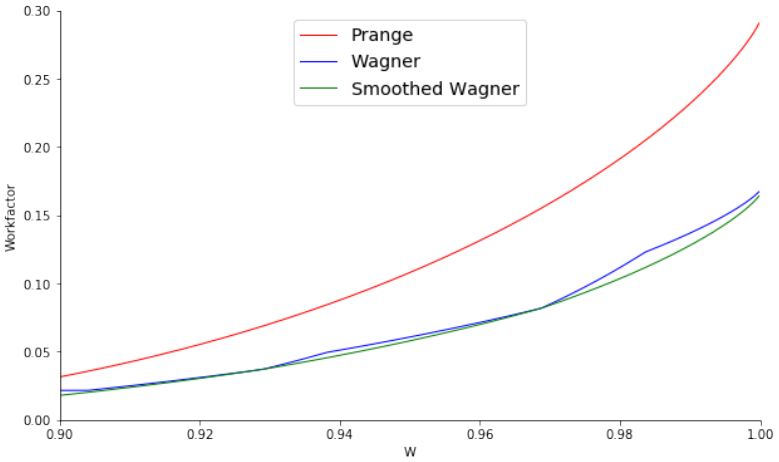


Figure 9.5: Workfactor of the smoothed variant of Wagner's algorithm: for $q = 3$, $R = 0.5$ and variable W.

We can observe on Figure 9.5 the gain induced by smoothing.

9.2.4 Using representations

9.2.4.1 Ternary representations

Just like we have seen in Section 8.5.2 that we can further improve the efficiency of Dumer's algorithm by introducing representations, as explained in [BJMM12], we can improve Wagner's algorithm by using the same idea.

We explained in the binary setting that the idea of representations is to get rid of the constraint that the support should be disjoint. Let us see what this means in the ternary setting and in the context of Wagner's algorithm.

If we look at the list tree of Wagner's algorithm (see Figure 9.3) from top to bottom, we split each list in two, according to what is called the *left-right* procedure. This means that if we want to have a set $S = \{\sum_{j \in [\![A,B]\!]} b_j \boldsymbol{x}_j : |b_j| = p\}$ at some level, we decompose each element of $\boldsymbol{y} \in S$ as $\boldsymbol{y} = \boldsymbol{y}_1 + \boldsymbol{y}_2$

where $\boldsymbol{y}_1 \in S_1$ and $\boldsymbol{y}_2 \in S_2$, where

$$S_1 \stackrel{\text{def}}{=} \left\{ \sum_{j \in [\![A, \lfloor \frac{B+A}{2} \rfloor]\!]} b_j \boldsymbol{x}_j : b_j \in \{0,1\},\ |\boldsymbol{b}| = p/2 \right\}$$
$$S_2 \stackrel{\text{def}}{=} \left\{ \sum_{j \in [\![\lfloor \frac{B+A}{2} \rfloor + 1, B]\!]} b_j \boldsymbol{x}_j : b_j \in \{0,1\},\ |\boldsymbol{b}| = p/2 \right\}.$$

Such a decomposition does not always exist, but it exists with probability at least $\frac{1}{p}$. Indeed, the probability that a vector of weight p can be split this way is

$$\frac{\binom{n/2}{p/2}^2}{\binom{n}{p}} \geqslant \frac{1}{p}.$$

Wagner's algorithm uses this principle. When looking for vectors $\mathbf{b}$ containing the same number of 0's and 1's, it looks for $\mathbf{b}$ in the form $\mathbf{b} = \mathbf{b}_1 + \mathbf{b}_2$, where the second half of $\mathbf{b}_1$ and the first half of $\mathbf{b}_2$ are only zeros. The first half of $\mathbf{b}_1$ and the second half of $\mathbf{b}_2$ are expected to have the same number of 0s and 1s. This ensures that $\mathbf{Support}(\boldsymbol{b}_1) \cap \mathbf{Support}(\boldsymbol{b}_2 = \emptyset$. Hence, for each vector $\boldsymbol{b}$, there is (at most) a unique way to write it as $\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2$ with $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ matching the support constraints.

The idea of representations is to follow Wagner's approach of list merging while allowing more possibilities to write $\boldsymbol{b}$ as the sum of two vectors $\boldsymbol{b} = \boldsymbol{b}_1 + \boldsymbol{b}_2$. We remove the constraint that $\boldsymbol{b}_1$ has zeros on its right half and $\boldsymbol{b}_2$ has zeros on its left half. We replace it by a less restrictive constraint: we fix the number of 0s, 1s and 2s (as elements of $\mathbf{F}_3$) in $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$.

More precisely, we consider the set

$$S(p_1, p_2) = \left\{ \sum_{j \in [\![A,B]\!]} b_j \boldsymbol{x}_j : b_j \in \mathbb{F}_3,\ |\{b_j = 1\}| = p_1 \text{ and } |\{b_j = 2\}| = p_2 \right\} \tag{9.9}$$

for some weights p_1 and p_2 and we want to decompose each $\boldsymbol{y}$ into $\boldsymbol{y}_1 + \boldsymbol{y}_2$ such that $\boldsymbol{y}_1, \boldsymbol{y}_2 \in S(p_1, p_2)$. On the example of Figure 9.6, we have $p = 4$, $p_1 = 3$ and $p_2 = 1$.

Notice that in this definition of $S(p_1, p_2)$, the elements b_j belong to the set $\mathbf{F}_3$ and not $\{0, 1\}$, even though we want to obtain a binary solution. This ternary structure increases the number of representations as shown in Figure 9.6. This approach may seem unusual, but it is actually the high number of possible representations that yields a gain in complexity, as we already

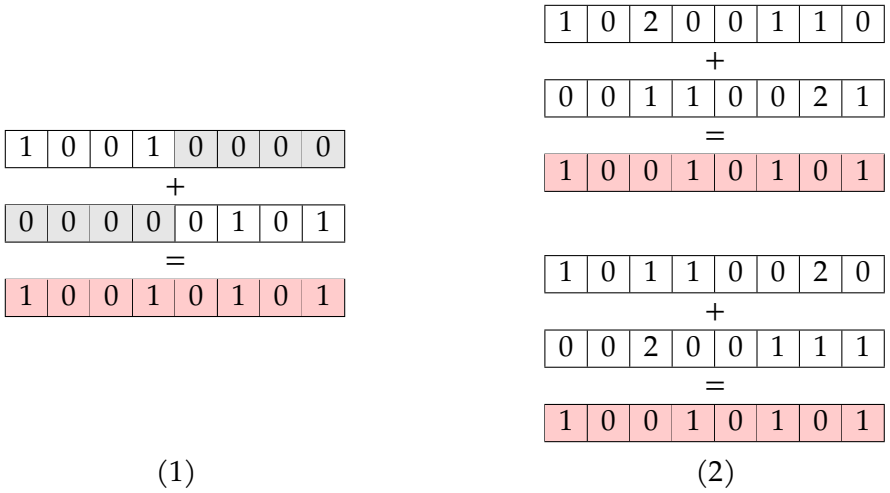


Figure 9.6: Same vector (1) using left-right split and (2) using representations

discussed in the binary setting. Indeed, just like we explained for [BJMM12], the key idea is that each element $\boldsymbol{y} \in S$ accepts many decompositions (the so-called *representations*) $\boldsymbol{y}_1 + \boldsymbol{y}_2$ where $\boldsymbol{y}_1, \boldsymbol{y}_2 \in S(p_1, p_2)$. We make sure that (on average) only one such representation is present by merging on some bits.

Of course, the values p_1 and p_2 should be chosen very carefully so that $\boldsymbol{y}_1 + \boldsymbol{y}_2 \in S$ with a relatively high probability. Moreover, most elements of the form $\boldsymbol{y}_1 + \boldsymbol{y}_2$ will not match the expected weight constraints. These elements are called *badly-formed* elements and must be discarded. To compensate these discarded sums, one can slightly lower the number of agreement bits when merging the lists, in order to obtain on average the desired number of elements in the merged list. The whole point of this approach is that the large number of ways to represent each element can compensate the fact that most sums are badly-formed.

We see that the representation technique introduces a lot of new parameters. One should decide the values p_1, p_2 at each level of the search tree. Contrary to the binary case, our search tree is deeper (often 7 or 8 levels). Moreover, the equations linking the parameters from one level to the other can become be quite cumbersome, and it is hard to correctly optimize all parameters to find the right equilibrium in the general case.

For this reason, we will not introduce representations at each level, only at the bottom of the search tree. Moreover, if we relieve too many constraints and allow too many representations of a solution, it may happen that we end up with multiple copies of the same solution. In order to avoid this situation, we use *partial representations*, which is an intermediate approach between *left-right* splitting and using *representations*, as illustrated in Figure 9.7.

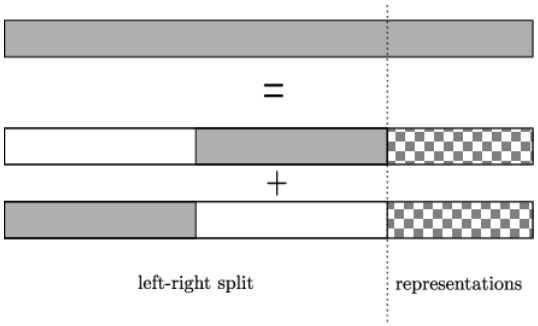


Figure 9.7: Decomposing a vector using partial representations.

9.2.4.2 The algorithm

We presented the general idea of how to use representations to improve Wagner's algorithm. We explained that using representations opens the way for numerous parameter choices, especially the numbers of 1s and 2s in each representation, which are difficult to optimize. In the algorithm we present here, some design choices were made to restrict the number of parameters to optimize. The parameters have been obtained mostly by trial and error, in order to optimize the complexity.

We are still relying on the generalised information set decoding algorithm detailed in Algorithm 16. We take $p = k + \ell$ because we are in the high weight regime. Moreover, our experiments yield $\ell = 0.060835n$ as a convenient choice.

For the search step, we use Wagner's algorithm with 5 to 7 floors. From bottom to top, we have first one floor of left-right splits, then two floors using partial representations, and finally two to four floors using left-right splits again. This is illustrated in Figure 9.8, in the case $a = 7$, where yellow lists correspond to partial representations and blue lists to left-right splits.

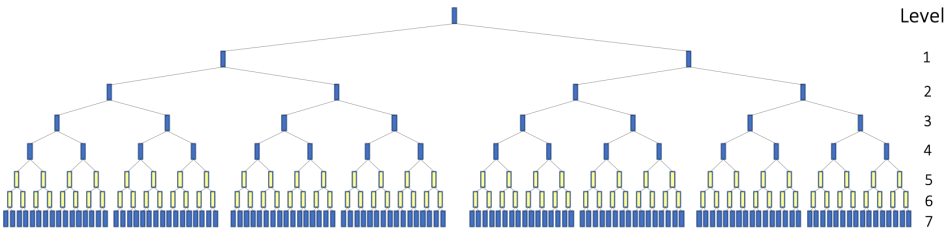


Figure 9.8: Representation of the Wagner tree for $a = 7$

As we already stated, the presence of representations at one floor yields

badly-formed element at the floor above. Hence, having representations at floors 5 and 6 (for the case $a = 7$), we expect to have badly-formed elements at floors 4 and 5. These have to be dealt with by filtering, and the number of merging bits has to be adapted accordingly to preserve the size of the list. Figure 9.9 illustrated the bottom part of the Wagner tree, with badly-formed elements displayed in red.

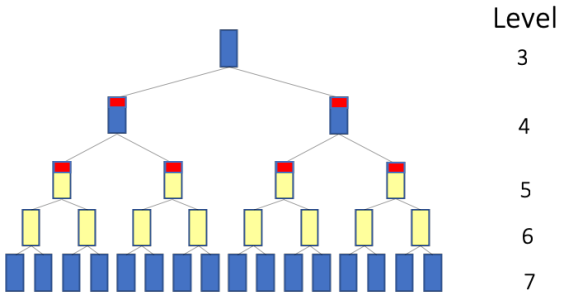


Figure 9.9: Detail of the bottom floors of Figure 9.8

The number of 1s and 2s in each floor, as well as the number of merge bits, depend on the parameters R and W. We will explicit some choices specific to the parameters used in the Wave signature scheme in the next section.

Finally, the size of the leaves can be adapted using the idea of smoothing introduced earlier.

9.3 Applications

In this section, we present two applications of the algorithms previously introduced.

9.3.1 Application to the Wave signature

Wave is a new code-based signature scheme proposed in [DST19]. It uses a *hash-and-sign* approach and follows the paradigm introduced in [GPV08]. To forge a signature, one has to solve the syndrome decoding problem in large weight $\mathrm{SD}(q, R, W)$ with $q = 3$, $R = 0.676$ and $W = 0.948366$. To see the impact of our algorithm on the Wave scheme, we detail our choice of parameters for this specific regime.

9.3.1.1 Parametrisation of the algorithm

We claim that the algorithm presented in Section 9.2.4 can solve $\mathrm{SD}(3, R, W)$ with $R = 0.676$ and $W = 0.948366$ with complexity $\tilde{\mathcal{O}}\left(2^{0.0176}\right)$. Indeed, our algorithm solves $SS(3, k+\ell, \ell, L)$ in amortised time $O(1)$ with $\ell = 0.060835$ and $L = 2^{0.0176n}$. This means that if finds $O(L)$ solutions in time $O(L)$.

The algorithm has $a = 7$ floors. At all levels, we want lists of size L. Only the leaves are smaller (of size $2^{0.0139n}$), because of the smoothing technique. Here is a complete description.

- At the top, we want to find L solutions.
- Levels 1 to 4 consist of left-right splits, like in the classical setting of Wagner's algorithm.
- At levels 5 and 6, we use partial representations. More exactly, we divide each vector in two parts that are treated separately. On one part, we will only use representations for level 5 and then the usual left-right split for level 6. This part represents a proportion $\lambda_1 = 0.7252$ of each vector. For remaining part of the vector, we use representations on both levels.
- In practice, at level 4 we had 16 lists of the form $\mathcal{L}_i$ containing sums of elements $\boldsymbol{x}_j$ with $j \in \mathcal{I}_i \stackrel{\text{def}}{=} [\![1 + \frac{(i-1)(k+\ell)}{16}, \frac{i(k+\ell)}{16}]\!]$, for $i \in [\![1, 16]\!]$. At level 5 and 6, we split the lists $\mathcal{I}_i$ in two, according to Figure 9.10.
- Hence, we have different densities of 0s, 1s and 2s, in the parts corresponding to one layer of representations (in proportion λ_1) and the part where we apply two layers of representations (in proportion λ_2). We denote ρ_1, ρ_2, ρ_3 the different densities, as in Figure 9.10.
  - For ρ_1, we ask for 74.8% of 0s, 25.1% of 1s and 0.1% of 2s
  - For ρ_2, we ask for 74.2% of 0s, 25.4% of 1s and 0.4% of 2s.
  - For ρ_3, we ask for 86.9% of 0s, 13.1% of 1s and 0.0% of 2s.
- The expected number of badly-formed elements can be computed theoretically. These computations are stated in full details in the appendix of [BCDL19]. This is important to adapt the number of bits on which we merge, to maintain lists of size L while discarding badly-formed elements. We obtain the following results.
  - The number of badly-formed elements at level 4 is $2^{0.0116n}$ and we therefore merge on $2^{0.0055n}$ bits.
  - The number of well-formed elements at level 5 is $2^{0.0174n}$. Therefore, we merge on $2^{0.0173n}$ bits to obtain lists of size L at level 4.

- From level 6 to level 5, all lists are of size L with no badly-formed elements, so we merge on $L = 2^{0.0176n}$ bits.
- Finally, because the bottom lists of level 7 are of size $2^{0.0139n}$ due to smoothing, we only merge on $2^{0.0032n}$ bits to create level 6.

Remark 9.10. *Here, we say that we* merge on t bits*, because it is convenient to count in base two, but we should keep in mind that in practice the vectors are in* $\mathbf{F}_3$ *and therefore the real operation amounts to making sure that* $\log_2(3)t$ *symbols (in* $\mathbf{F}_3$*) of the vectors are equal to the desired value.*

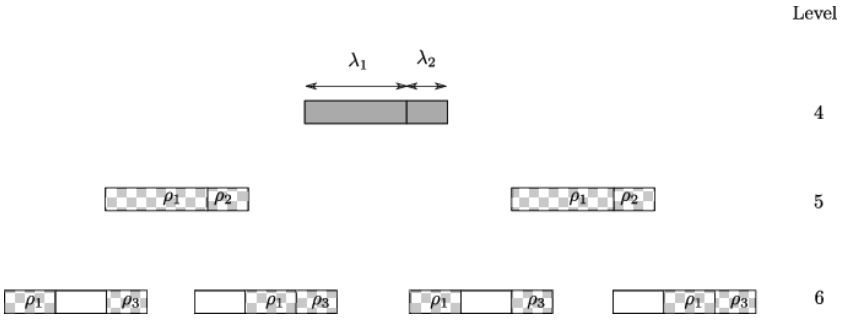


Figure 9.10: Detail of the partial representations

One can check that in total we merge on of $2^{0.0964n}$ bits, which is exactly equal to 3^ℓ. This ensures that the list obtained at the top indeed contains solutions to the subset sum problem.

Moreover, using the results of [BCDL19] on the number of ternary representations, we can also check that in this setting, each solution to the subset sum problem admits $2^{0.4915n}$ representations. Because there are $2^{k+\ell}/3^\ell = 2^{0.6404n}$ solutions to the subset sum problem, this yields $2^{0.6404n} \times 2^{0.4915n} = 2^{1.1319n}$ representations leading to one solution of the problem. The merging constraints filter $2^{1.1143n}$ solution vectors, and hence we obtain as stated $2^{1.1319n}/2^{1.1143n} = 2^{0.0176n}$ solutions at the end.

9.3.1.2 Parameters for the Wave signature

Forging a signature in the Wave scheme amounts to solving the syndrome decoding problem. The public key is a pseudo-random parity-check matrix $\boldsymbol{H}$ of size $(n-k) \times n$ and the signature of a message $\boldsymbol{m}$ is an error $\boldsymbol{e}$ of weight w such that $\boldsymbol{e}\boldsymbol{H}^\intercal = \mathcal{H}(\boldsymbol{m})$ with $\mathcal{H}$ a hash function. However, instead of trying to forge a signature for one message of our choice, a natural idea is to try to forge one message among a selected set of messages. This context leads directly to a slight variation of the classical syndrome decoding problem: instead of

having one syndrome, there is a list of possible syndromes and the goal is to decode one of them. This problem is known as the *Decoding One Out of Many* (DOOM) problem [Sen11a].

The difference induced by DOOM is that it increases the search space. Namely, instead of searching $\boldsymbol{e}$ of weight w in the space $\{\boldsymbol{e} \,|\, \boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}^\intercal\}$, we search in $\{\boldsymbol{e} \,|\, \exists i \in [\![1, M]\!], \boldsymbol{H}\boldsymbol{e}^\intercal = \boldsymbol{s}_i{}^\intercal\}$.

The idea to solve this problem with Wagner's approach is to take $M \geqslant q^{\ell/a}$ and replace the bottom-right list of the tree (the list $\mathcal{L}_{2^a}$) by a list containing all the syndromes. Hence, there are only $2^a - 1$ lists to generate from the search space. Therefore, the constraint of Theorem 9.8 becomes

$$q^{\ell/a} \leqslant 2^{(k+\ell)/(2^a-1)}.$$

In practice, we have $a = 7$ so the change from 2^a to $2^a - 1$ has a negligible impact.

The parameters proposed in the first version of the Wave signature scheme [DST18] are derived from the complexity of a key attack. Our new algorithms introduced in Section 9.2 provide another attack to consider. In Table 9.1, we computed the minimal parameters of a code (supposedly random) with a rate equivalent to the rate of the Wave signature, and such that our best algorithm has a time complexity of at least 2^{128}. The parameters n, k and w denote respectively the length of the code, its dimension and the weight of the signature. These results have been taken into account to propose parameters for the latest version of the Wave signature scheme [DST19].

(n, k, w)	Public key size (in MB)	Signature length (in kB)
(7236,4892,6862)	2.27	1.434

Table 9.1: Parameters for a code in the same regime as the Wave signature scheme and achieving 128 security-bits with regards to our attack.

9.3.2 Hardest instance of ternary large weight decoding

We have tried to optimize the algorithms for the parameters of the Wave signature scheme, as this is the only cryptographic scheme that relies on ternary large weight decoding for now. We have seen that this corresponds to a regime where there is an exponential number of solutions to the decoding problem.

Another interesting question is to understand, with our current knowledge of the decoding algorithms in large weight, for which choice of R and W the

problem is the hardest. Indeed, Figure 9.2 (showing the complexity of Prange's algorithm) shows that for appropriate parameter choice, the ternary decoding problem can be harder with large weight than with small weight.

As we can see on Figure 9.2, there are two cases to consider. For some rates, the complexity is strictly increasing with the weight, and the hardest case corresponds to $W = 1$. This is the case for the examples $R = 0.5$ and $R = 0.676$ that we studied. But for some lower rates, we observe a peak in the complexity. We explained in Section 9.1.2.3 that this corresponds to a large weight equivalent of the Gilbert-Varshamov bound, where there is on average one unique solution.

Hence, the hardest complexity for fixed R is attained in $W = h_{GV+}(R)$ when it exists, and $W = 1$ otherwise. This allows us to compute the hardest complexity only depending on the variable R.

Unsurprisingly, the maximal complexity (over R and W) is reached exactly at the limit between the two cases, when the peak is in $W_{GV+} = 1$. This corresponds to the rate R such that $R = 1 - \log_q(q-1)$. For $q = 3$ this yields $R \simeq 0.36907$.

This allows us to compare the best exponents of the algorithms to solve the hardest instance of the problem, in the binary and ternary cases. This yields Table 9.2. In the binary case, the hardest instance corresponds to the (usual) Gilbert-Varshamov bound for small weight. For the ternary case, we optimised Wagner's algorithm with a two-level tree, including one layer of representations. This yields the exponent 0.247. Using a larger tree did non give any improvement.

Algorithm	$q = 2$, $W = W_{GV}(R)$	$q = 3$ and $W = 1$
Prange	0.121 ($R = 0.454$)	0.369 ($R = 0.369$)
Dumer/Wagner	0.116 ($R = 0.447$)	0.269 ($R = 0.369$)
BJMM/Wagner with repr.	0.102 ($R = 0.427$)	0.247 ($R = 0.369$)

Table 9.2: Best exponents with associated rates.

The ternary syndrome decoding in large weight appears significantly harder than its binary counterpart. But in the context of cryptography, one tries to achieve the best trade-off between the complexity and the key size. This is not reflected in our comparison. For instance, for the ternary problem, the key is a matrix with elements in $\mathbf{F}_3$ rather than $\mathbf{F}_2$ so this naturally increases the information by a factor $\log_2(3)$, for fixed matrix dimensions n and k. It could be that the ternary problem has hardest instances, but the larger public keys size make the trade-off less interesting.

Hence, we have to ask the question differently. Considering the asymp-

totic complexity exponents, what is the smallest input size of the syndrome decoding problem for which the algorithms need at least 2^{128} operations to decode?

The input of syndrome decoding problem is the matrix $\boldsymbol{H} \in \mathbf{F}_q^{(1-R)n \times n}$ (the syndrome $\boldsymbol{s} \in \mathbf{F}_q^{(1-R)n}$ corresponds to one additional column and can be neglected). To lower the input size, this matrix is represented in systematic form. This means that we write $\boldsymbol{H} = (\boldsymbol{I}_{(1-R)n} | \boldsymbol{H}')$. The only relevant part that needs to be specified is $\boldsymbol{H}'$. This requires $R(1-R)n^2 \log_2(q)$ bits.

In Table 9.3, we show that, even in this metric, the ternary syndrome decoding problem is harder, *i.e.* requires 2^{128} operations to decode inputs of smaller sizes.

Algorithm	$q = 2$	$q = 3$ and $W > 0.5$
Prange	275 ($R = 0.384$)	44 ($R = 0.369$)
Dumer/Wagner	295 ($R = 0.369$)	83 ($R = 0.369$)
BJMM/Wagner with representations	374 ($R = 0.326$)	99 ($R = 0.369$)

Table 9.3: Minimum input sizes (in Kbits) for a time complexity of 2^{128}

Note that in the binary case, the best trade-off is obtained for lower rates than the hardest case, because this reduces the key size. In the ternary case, this does not happen because the complexity decreases quickly when R decreases.

9.3.3 Conclusion

In this chapter, we explained a fundamental difference between the cases $q = 2$ and $q \geqslant 3$ of the syndrome decoding problem. Before this work, the syndrome decoding problem for $q \geqslant 3$ had only been considered for small weight and the large weight case had never been addressed. The Wave signature scheme, which relies on this problem for its security, motivated us to conduct a study of the complexity of the syndrome decoding in this regime.

We proposed algorithms to solve the large weight problem with a similar approach to what is achieved by Prange, Dumer and BJMM for small weight. This work is still preliminary, and there could still be room for improvements of these algorithms, especially with the representations that require *ad hoc* design choices.

But the fact that the instances in large weight are harder that in the small weight (even for equivalent key size) is a very promising result. This should encourage cryptographers to propose new code-based cryptosystems, signatures, or other primitives, relying on the large weight syndrome decoding problem. These should yield smaller key sizes for the same security level.

Conclusions and perspectives

Throughout this work, we have come to address most of the aspects of the security of code-based cryptosystems: from the generic decoding algorithms that serve as a reference for the message security, to the possibility to recover the secret key from the algebraic structure of the public key, as well as the exploitation of information leaking from the physical implementation of the decryption algorithm.

While this work was being conducted, the National Institute of Standards and Technologies (NIST) was advancing on its post-quantum standardisation process. The procedure is not over yet, but in July 2020 the NIST announced the remaining candidates selected for the third round [AAACD+20]. The NIST selected seven "finalists" and eight "alternate candidates". Out of the four public-key encryption systems selected as "finalists", we find the code-based Classic McEliece cryptosystem introduced in Chapter 1, as well as three lattice-based proposals: CRISTAL-KYBER, NTRU and SABER. The selection of the McEliece scheme is not a surprise, since it is the oldest post-quantum proposal and it still inspires confidence after forty years of research. However, as we already stated, this cryptosystem has very large public keys which make it unfit for general use (for instance for internet protocols). The three other selected finalists have a better profile for this use but all rely on lattice-based security assumptions. We see here that, in order to obtain some diversity in the solutions, a code-based cryptosystem with competitive parameter size would be appreciated.

The BIKE cryptosystem, presented in Chapter 2, is the most promising candidate to fit these criteria. However, the issue of the decoding failure, which we extensively addressed in Chapter 3, makes its CCA security still uncertain. For this reason, the NIST decided to select BIKE as an alternate candidate in its third round, rather than a finalist. Another code-based candidate, HQC, also based on quasi-cyclic codes, was selected as an alternate candidate, because its performance are not as good as the comparable lattice cryptosystems.

As for signatures, no code-based signature was submitted to the process in the first place. The lack of diversity in the signatures selected as finalists is a current concern of the NIST. The Institute explicitly stated its interest for

new digital signature schemes *"not based on structured lattices"*.

These temporary conclusions of the NIST are very encouraging for code based cryptography, which appears to be the most promising way to obtain post-quantum primitives not based on lattice problems. The recent improvements in the analysis of the decoding failure of MDPC codes, as well as the new code-based signature proposal Wave based on large weight decoding, bring code-based cryptography one step closer to providing secure and practical solutions for post-quantum security.

Prevention of side channel attacks on code-based cryptosystems. If code-based cryptosystem are to be become standards, these will immediately be implemented. We have seen in Chapter 3 that the decoding algorithm of MDPC codes may leak significant information. Hence, one should be particularly careful when implementing the bit-flipping algorithm, and especially make sure that no variable correlated with the syndrome weight is accessible to a malicious user.

McEliece-like cryptosystem based on GRS codes. We have seen two cases of cryptosystem based on GRS codes, which can be attacked up to some threshold: the RLCE cryptosystem (in Chapter 6) is still secure for $w \geqslant \frac{n-k}{2}$ and the XGRS scheme (in Chapter 7) is secure for $\lambda \leqslant \frac{m}{2}$. In both cases, there is either a possibility to attack the scheme in these regimes, or it means that we can build secure GRS-based cryptosystems. Understanding such thresholds is important. This could also yield interesting constructions for multiparty computation and threshold cryptography. The case of XGRS cryptosystem with small value λ is particularly interesting since it is very close to the Goppa McEliece scheme. If it is indeed secure, we can propose a cryptosystem very close to the original McEliece proposal but potentially achieving smaller key size. On the other hand, any progress in the cryptanalysis of these codes would be of interest for the security of the Classic McEliece scheme. Further study in this direction should be conducted.

Ternary decoding with large weight. We proposed in Chapter 9 an analysis of the hardness of a new mathematical problem on which one could build cryptographic primitives: the (well-known) decoding problem in the (new) regime of large-weight errors. This idea was first introduced and used in the Wave signature scheme. Our study shows that for equivalent complexity, the instances of the ternary decoding in large weight have smaller size. This opens the way for new cryptosystems with smaller key size, which is one of the main challenges of code-based cryptography. We hope that new encryption systems

(or other cryptographic primitives) exploiting this idea will appear in the coming years.

Decoding challenge and practical complexity of information set decoding algorithms. Chapters 8 and 9 have been dedicated to the study of information set decoding algorithms. These algorithm solve the generic decoding problem, that is, decoding a noisy codeword, regardless of the structure of the code. Most code-based primitives rely on the hardness of this problem to make security claims. Hence, the proposed parameters (and hence the key size) are often optimised under the constraint that the best information set decoding cannot decode in less than 2^κ operations, where κ is the security parameter. A good understanding of the complexity of these algorithms is therefore crucial.

As we have seen, the study of the complexity of these algorithms mostly focuses on the asymptotic complexity. The algorithms achieving the best asymptotic efficiency (using representations, such as BJMM's algorithm) are the result of a fine optimisation of numerous parameters. However, the latest algorithms are getting more and more complex, to achieve a small gain in the exponent. As a result, there is no implementation of the most recent algorithms. There are several questions about these algorithms for which we do not have an answer. The most natural question is: for a fixed hardware configuration and running time, what is the largest instance of the syndrome decoding that we can solve in practice? But there are other interrogations. For instance, the new algorithms with lower asymptotic complexity may come with larger polynomial factors. Hence they do not perform best on small instances. For which instance size does it become more efficient to use the algorithms that achieve the lowest asymptotic complexity? Another question arises when it comes to optimising the parameters (ℓ, p, ε_i) of information set decoding algorithms. When studying the asymptotic complexity, these are considered as continuous parameters, but in practice, they correspond to integer values (a certain number of rows/columns). How does this restriction in the optimisation affect the performance of the algorithms?

To a larger extent, if ones wants code-based cryptography to be part of the next generation of standards, one should make sure that a large community of scientists are convinced that it relies on solid ground. It is usual to consider that a problem is all the more secure if it has been existing for a long time with no significant improvement in the cost of the resolution, and this is indeed the case of the syndrome decoding problem. But the confidence in the hardness of a problem also depends on the number of persons who attempted to solve it. Therefore, there is a need to broaden the community of people trying to actively solve instances of the syndrome decoding problem. This can only increase the confidence of a larger audience in the security of code-based

cryptography. In the case of code-based cryptography, this should not be too difficult since the syndrome decoding problem is very easy to state.

The best way to do so is to publish a series of challenges, corresponding to instances of the syndrome decoding problem of increasing size, for anyone to solve. Such an approach is usual in cryptography: the RSA factoring challenge [Lab91] has been introduced by RSA Laboratories to encourage the study of the practical difficulty of factoring integers, and hence prove the strength of the RSA cryptosystem. Other branches of post-quantum cryptography have launched such challenges: lattice-based cryptography [LRBN14] and multivariate cryptosystems [YDHTS15].

For these reasons we launched the Decoding challenge [ALL19] in 2019. We hope that this will serve as an incentive for people to implement information set decoding algorithms. This can also serve as a reference to benchmark different implementations and compare design choices. The next step is to come up with a reference implementation of the latest algorithms, in order to answer some of the questions listed above. We are confident that this is an important step to strengthen the confidence of a large community of computer scientists in code-based cryptographic schemes.

Bibliography

[AAACD+20] Gorjan Alagic, Jacob Alperin-Sheriff, Daniel Apon, David Cooper, Quynh Dang, John Kelsey, Yi-Kai Liu, Carl Miller, Dustin Moody, Rene Peralta, Ray Perlner, Angela Robinson, and Daniel Smith-Tone. *Status Report on the Second Round of the NIST Post-Quantum Cryptography Standardization Process*. 2020 (cit. on pp. 88, 105, 241).

[AABBB+17a] C. Aguilar Melchor, N. Aragon, S. Bettaieb, L. Bidoux, O. Blazy, J.-C. Deneuville, P. Gaborit, and G. Zémor. *RQC*. NIST Round 1 submission for Post-Quantum Cryptography. `https://pqc-rqc.org`. Nov. 2017 (cit. on p. 41).

[AABBB+17b] Carlos Aguilar Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, and Gilles Zémor. *HQC*. NIST Round 1 submission for Post-Quantum Cryptography. `https://pqc-hqc.org/`. Nov. 2017 (cit. on pp. 41, 60).

[ABBBB+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. `https://bikesuite.org`. Nov. 2017 (cit. on pp. 40, 56, 59, 63, 65, 83, 87).

[ABDGH+19] Nicolas Aragon, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Adrien Hauteville, Olivier Ruatta, Jean-Pierre Tillich, Gilles Zémor, Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Bardet Magali, and Ayoub Otmani. *ROLLO* (*merger of Rank-Ouroboros, LAKE and LOCKER*). Second round submission to the NIST post-quantum cryptography call. NIST Round 2 submission for Post-Quantum Cryptography. `https://pqc-rollo.org`. Mar. 2019 (cit. on pp. 60, 94, 105).

[ABGHZ19] 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 (cit. on p. 43).

[AG19] Nicolas Aragon and Philippe Gaborit. "A key recovery attack against LRPC using decryption failures". In: *WCC 2019 - Workshop on Coding Theory and Cryptography*. 2019 (cit. on pp. 87, 93).

[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 (cit. on p. 42).

[Ale03] Michael Alekhnovich. "More on average case vs approximation complexity". In: *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.* IEEE. 2003, pp. 298–307 (cit. on pp. 31, 41).

[Ale11] Michael Alekhnovich. "More on Average Case vs Approximation Complexity". In: *Computational Complexity* 20.4 (2011), pp. 755–786 (cit. on pp. 60, 198).

[ALL19] Nicolas Aragon, Julien Lavauzelle, and Matthieu Lequesne. *decodingchallenge.org*. `http://decodingchallenge.org`. 2019 (cit. on p. 244).

[APRS20] 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 (cit. on pp. 49, 60, 84).

[BBBGN+20] 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 (cit. on pp. 92, 94, 105).

[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 (cit. on pp. 49, 51).

[BBCGP+20] 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 (cit. on pp. 92, 94, 105).

[BBCPS19] Marco Baldi, Alessandro Barenghi, Franco Chiaraluce, Gerardo Pelosi, and Paolo Santini. *LEDAcrypt*. Second round submission to the NIST post-quantum cryptography call. `https://www.ledacrypt.org`. Jan. 2019 (cit. on pp. 60, 84).

[BBCRS11] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. *Enhanced public key security for the McEliece cryptosystem*. submitted. 2011 (cit. on p. 115).

[BBCRS16] Marco Baldi, Marco Bianchi, Franco Chiaraluce, Joachim Rosenthal, and Davide Schipani. "Enhanced Public Key Security for the McEliece Cryptosystem". In: *J. Cryptology* 29.1 (2016), pp. 1–27 (cit. on p. 115).

[BBD09] Daniel J. Bernstein, Johannes Buchmann, and Erik Dahmen, eds. *Post-Quantum Cryptography*. Springer-Verlag, 2009 (cit. on p. 24).

[BC07] Marco Baldi and Franco Chiaraluce. "Cryptanalysis of a new instance of McEliece cryptosystem based on QC-LDPC Codes". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. Nice, France, June 2007, pp. 2591–2595 (cit. on pp. 49, 50).

[BCDL19] Rémi Bricout, André Chailloux, Thomas Debris-Alazard, and Matthieu Lequesne. "Ternary Syndrome Decoding with Large Weights". In: *Selected Areas in Cryptography - SAC 2019 - 26th International Conference, Waterloo, ON, Canada, August 12-16, 2019, Revised Selected Papers*. Ed. by Kenneth G. Paterson and Douglas Stebila. Vol. 11959. Lecture Notes in Computer Science. Springer, 2019, pp. 437–466 (cit. on pp. 217, 235, 236).

[BCDOT16] Magali Bardet, Julia Chaulet, Vlad Dragoi, Ayoub Otmani, and Jean-Pierre Tillich. "Cryptanalysis of the McEliece Public Key Cryptosystem Based on Polar Codes". In: *Post-Quantum Cryptography 2016*. LNCS. Fukuoka, Japan, Feb. 2016, pp. 118–143 (cit. on p. 119).

[BCGLL+17] Alin Bostan, Frédéric Chyzak, Marc Giusti, Romain Lebreton, Grégoire Lecerf, Bruno Salvy, and Éric Schost. *Algorithmes Efficaces en Calcul Formel*. French. 686 pages. Imprimé par CreateSpace. Aussi disponible en version électronique. Palaiseau: Frédéric Chyzak (auto-édit.), Sept. 2017 (cit. on p. 191).

[BCGO09] 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 (cit. on p. 50).

[BCJ11] Anja Becker, Jean-Sébastien Coron, and Antoine Joux. "Improved Generic Algorithms for Hard Knapsacks". In: *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*. 2011, pp. 364–385 (cit. on pp. 213, 225).

[BCLMM+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 (cit. on pp. 40, 84, 194).

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust. "The Magma Algebra System I: The User Language". In: *J. Symbolic Comput.* 24.3/4 (1997), pp. 235–265 (cit. on p. 143).

[Ber10] Daniel J. Bernstein. "Grover vs. McEliece". In: *Post-Quantum Cryptography 2010*. Ed. by Nicolas Sendrier. Vol. 6061. LNCS. Springer, 2010, pp. 73–80 (cit. on p. 62).

[BGGMP+17] Paulo S. L. M. Barreto, Shay Gueron, Tim Güneysu, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, and Jean-Pierre Tillich. "CAKE: Code-Based Algorithm for Key Encapsulation". In: *Cryptography and Coding - 16th IMA International Conference, IMACC 2017, Oxford, UK, December 12-14, 2017, Proceedings*. Vol. 10655. LNCS. Springer, 2017, pp. 207–226 (cit. on pp. 56, 59, 65).

[BGK19] Thierry P. Berger, Cheikh Thiécoumba Gueye, and Jean Belo Klamti. "Generalized subspace subcodes with application in cryptology". In: *IEEE Trans. Inform. Theory* 65.8 (2019), pp. 4641–4657 (cit. on pp. 185, 187, 189–192).

[BGKR19] Thierry P. Berger, Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Olivier Ruatta. "Designing a Public Key Cryptosystem Based on Quasi-cyclic Subspace Subcodes of Reed–Solomon Codes". In: *International Conference on Algebra, Codes and Cryptology*. Springer. 2019, pp. 97–113 (cit. on pp. 116, 153).

[BGT93] Claude Berrou, Alain Glavieux, and Punya Thitimajshima. "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1". In: *Proceedings of ICC'93-IEEE International Conference on Communications*. Vol. 2. IEEE. 1993, pp. 1064–1070 (cit. on p. 48).

[BJMM12] 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 (cit. on pp. 214, 230, 232).

[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 (cit. on p. 114).

[BLP08] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. "Attacking and Defending the McEliece Cryptosystem". In: *Post-Quantum Cryptography 2008*. Vol. 5299. LNCS. 2008, pp. 31–46 (cit. on p. 40).

[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 (cit. on pp. 40, 116).

[BLP11] Daniel J. Bernstein, Tanja Lange, and Christiane Peters. "Smaller decoding exponents: ball-collision decoding". In: *Advances in Cryptology - CRYPTO 2011*. Vol. 6841. LNCS. 2011, pp. 743–760 (cit. on p. 210).

[BM17] Leif Both and Alexander May. "Optimizing BJMM with Nearest Neighbors: Full Decoding in $2^{2/21n}$ and McEliece Security". In: *WCC Workshop on Coding and Cryptography*. on line proceedings, see `http://wcc2017.suai.ru/Proceedings_WCC2017.zip`. Sept. 2017 (cit. on p. 214).

[BM18] Leif Both and Alexander May. "Decoding Linear Codes with High Error Rate and Its Impact for LPN Security". In: *Post-Quantum Cryptography 2018*. Ed. by Tanja Lange and Rainer Steinwandt. Vol. 10786. LNCS. Fort Lauderdale, FL, USA: Springer, Apr. 2018, pp. 25–46 (cit. on p. 214).

[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 (cit. on pp. 31, 60, 198).

[Bus52] Kenneth A Bush. "Orthogonal arrays of index unity". In: *The Annals of Mathematical Statistics* (1952), pp. 426–434 (cit. on p. 110).

[Can16] Rodolfo Canto Torres. *CaWoF, C library for computing asymptotic exponents of generic decoding work factors*. `https://gforge.inria.fr/projects/cawof/`. 2016 (cit. on p. 214).

[Can17] Rodolfo Canto Torres. "Asymptotic Analysis of ISD algorithms for the $q-$ary case". In: *Proceedings of the Tenth International Workshop on Coding and Cryptography WCC 2017*. Sept. 2017 (cit. on p. 224).

[CB14] Ivan V. Chizhov and Mikhail A. Borodin. "Effective attack on the McEliece cryptosystem based on Reed-Muller codes". In: *Discrete Math. Appl.* 24.5 (2014), pp. 273–280 (cit. on p. 119).

[CC19] Daniel Coggia and Alain Couvreur. "On the security of a Loidreau's rank metric code based encryption scheme". In: *WCC 2019 - Workshop on Coding Theory and Cryptography*. Saint Jacut de la mer, France, Mar. 2019 (cit. on p. 91).

[CCMZ15] 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 (cit. on p. 117).

[CFG89] Mark Chaimovich, Gregory Freiman, and Zvi Galil. "Solving dense subset-sum problems by using analytical number theory". In: *J. Complexity* 5.3 (1989), pp. 271–282 (cit. on p. 225).

[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 (cit. on pp. 41, 42).

[CG90] John T Coffey and Rodney M Goodman. "The complexity of information set decoding". In: *IEEE Trans. Inform. Theory* 36.5 (1990), pp. 1031–1037 (cit. on pp. 199, 219).

[CGGOT14] 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 (cit. on pp. 115, 116, 122, 123, 127, 138, 149).

[Cha17] Julia Chaulet. "Étude de cryptosystèmes à clé publique basés sur les codes MDPC quasi-cycliques". PhD thesis. University Pierre et Marie Curie, Mar. 2017 (cit. on p. 71).

[Che+52] Herman Chernoff et al. "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations". In: *The Annals of Mathematical Statistics* 23.4 (1952), pp. 493–507 (cit. on p. 75).

[Cho16] Tung Chou. "QcBits: Constant-Time Small-Key Code-Based Cryptography". In: *CHES 2016*. Ed. by Benedikt Gierlichs and Axel Y. Poschmann. Vol. 9813. LNCS. Springer, 2016, pp. 280–300 (cit. on p. 54).

[CJ04] Jean-Sebastien Coron and Antoine Joux. *Cryptanalysis of a provably secure cryptographic hash function*. IACR Cryptology ePrint Archive, Report 2004/013. 2004 (cit. on p. 211).

[CL20] Alain Couvreur and Matthieu Lequesne. *On the security of subspace subcodes of Reed-Solomon codes for public key encryption*. 2020 (cit. on pp. 116, 151).

[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 (cit. on pp. 115, 124, 150).

[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 (cit. on pp. 40, 119).

[COT14a] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich. "New Identities Relating Wild Goppa Codes". In: *Finite Fields Appl.* 29 (2014), pp. 178–197 (cit. on pp. 116, 185).

[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 (cit. on pp. 40, 185).

[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 (cit. on pp. 119, 185, 194).

[COTG15] Alain Couvreur, Ayoub Otmani, Jean–Pierre Tillich, and Valérie Gauthier-Umaña. "A Polynomial-Time Attack on the BBCRS Scheme". In: *Public-Key Cryptography - PKC 2015*. Ed. by J. Katz. Vol. 9020. LNCS. Springer, 2015, pp. 175–193 (cit. on p. 115).

[Cou01] Nicolas Courtois. "Efficient zero-knowledge authentication based on a linear algebra problem MinRank". In: *Advances in Cryptology - ASIACRYPT 2001*. Vol. 2248. LNCS. Gold Coast, Australia: Springer, 2001, pp. 402–421 (cit. on p. 92).

[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 (cit. on pp. 36, 121).

[Cou20] Alain Couvreur. *Introduction to coding theory*. Lecture notes available on `http://www.lix.polytechnique.fr/~alain.couvreur/doc_ens/lecture_notes.pdf`. 2020 (cit. on pp. 33–35, 156, 157).

[CS16a] Rodolfo Canto-Torres and Nicolas Sendrier. "Analysis of Information Set Decoding for a Sub-linear Error Weight". In: *Post-Quantum Cryptography 2016*. LNCS. Fukuoka, Japan, Feb. 2016, pp. 144–161 (cit. on p. 79).

[CS16b] Julia Chaulet and Nicolas Sendrier. "Worst case QC-MDPC decoder for McEliece cryptosystem". In: *IEEE Conference, ISIT 2016*. IEEE Press, 2016, pp. 1366–1370 (cit. on pp. 55, 65, 70).

[Del75] Philippe Delsarte. "On subfield subcodes of modified Reed-Solomon codes". In: *IEEE Trans. Inform. Theory* 21.5 (1975), pp. 575–576 (cit. on p. 110).

[Del78] Philippe Delsarte. "Bilinear Forms over a Finite Field, with Applications to Coding Theory". In: *J. Comb. Theory, Ser. A* 25.3 (1978), pp. 226–241 (cit. on p. 90).

[Dem97] Michel Demazure. *Cours d'algèbre: primalité, divisibilité, codes*. Vol. 1. Cassini Paris, 1997 (cit. on p. 34).

[DGJNV+19] Jan-Pieter D'Anvers, Qian Guo, Thomas Johansson, Alexander Nilsson, Frederik Vercauteren, and Ingrid Verbauwhede. "Decryption failure attacks on IND-CCA secure lattice-based schemes". In: *IACR International Workshop on Public Key Cryptography*. Springer. 2019, pp. 565–598 (cit. on p. 87).

[DGK20] Nir Drucker, Shay Gueron, and Dusan Kostic. "QC-MDPC Decoders with Several Shades of Gray". In: *PQCrypto 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Vol. 12100. LNCS. Springer, 2020, pp. 35–50 (cit. on pp. 86, 87).

[DGZ17] Jean-Christophe Deneuville, Philippe Gaborit, and Gilles Zémor. "Ouroboros: A Simple, Secure and Efficient Key Exchange Protocol Based on Coding Theory". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Vol. 10346. LNCS. Springer, 2017, pp. 18–34 (cit. on pp. 59, 78).

[DH76a] Whitfield Diffie and Martin Hellman. "New directions in cryptography". In: *IEEE transactions on Information Theory* 22.6 (1976), pp. 644–654 (cit. on pp. 20, 23, 36).

[DH76b] Whitfield Diffie and Martin E Hellman. "Multiuser cryptographic techniques". In: *Proceedings of the June 7-10, 1976, national computer conference and exposition*. 1976, pp. 109–112 (cit. on p. 20).

[DST18] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. *Wave: A New Code-Based Signature Scheme*. Cryptology ePrint Archive, Report 2018/996. Oct. 2018 (cit. on p. 237).

[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 (cit. on pp. 42, 218, 234, 237).

[DT18] T. Debris-Alazard and J.-P. Tillich. "Two attacks on rank metric code-based schemes: RankSign and an Identity-Based-Encryption scheme". In: *ASIACRYPT*. 2018 (cit. on p. 42).

[DT99] Marten van Dijk and Ludo Tolhuizen. "Efficient encoding for a class of subspace subcodes". In: *IEEE Trans. Inform. Theory* 45.6 (1999), pp. 2142–2146 (cit. on p. 157).

[Dum89] Il'ya Dumer. "Two decoding algorithms for linear codes". In: *Probl. Inf. Transm.* 25.1 (1989), pp. 17–23 (cit. on p. 210).

[Dur14] Marie-José Durand-Richard. "Du message chiffré au système cryptographique". In: *Cryptologie et mathématiques : une mutation des enjuex*. Ed. by Philippe Guillot Marie-José Durand-Richard. L'Harmattan, 2014 (cit. on p. 17).

[ELPS18] Edward Eaton, Matthieu Lequesne, Alex Parent, and Nicolas Sendrier. "QC-MDPC: A Timing Attack and a CCA2 KEM". In: *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018, Fort Lauderdale, FL, USA, April 9-11, 2018, Proceedings*. 2018, pp. 47–76 (cit. on pp. 63, 66, 84).

[FGOPT11] 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 (cit. on pp. 40, 41).

[FGOPT13] 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 (cit. on p. 119).

[FHSZG+17] Tomás Fabsic, Viliam Hromada, Paul Stankovski, Pavol Zajac, Qian Guo, and Thomas Johansson. "A Reaction Attack on the QC-LDPC McEliece Cryptosystem". In: *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017, Utrecht, The Netherlands, June 26-28, 2017, Proceedings*. Vol. 10346. LNCS. Springer, 2017, pp. 51–68 (cit. on pp. 68, 69, 87).

[FL05] Cédric Faure and Pierre Loidreau. "A New Public-Key Cryptosystem Based on the Problem of Reconstructing p-Polynomials". In: *Coding and Cryptography, International Workshop, WCC 2005, Bergen, Norway, March 14-18, 2005. Revised Selected Papers*. 2005, pp. 304–315 (cit. on p. 91).

[FL08] Pierre-Alain Fouque and Gaëtan Leurent. "Cryptanalysis of a Hash Function Based on Quasi-cyclic Codes". In: *Topics in Cryptology - CT-RSA 2008, The Cryptographers' Track at the RSA Conference 2008, San Francisco, CA, USA, April 8-11, 2008. Proceedings*. Vol. 4964. LNCS. Springer, 2008, pp. 19–35 (cit. on p. 57).

[FM08] Cédric Faure and Lorenz Minder. "Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves". In: *Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory*. Pamporovo, Bulgaria, June 2008, pp. 99–107 (cit. on p. 40).

[FO99] Eiichiro Fujisaki and Tatsuaki Okamoto. "Secure integration of asymmetric and symmetric encryption schemes". In: *Annual International Cryptology Conference*. Springer. 1999, pp. 537–554 (cit. on pp. 59, 69).

[FOPT10] 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 (cit. on p. 50).

[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 (cit. on p. 194).

[FRXKM+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. `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/RaCoSS.zip`. Nov. 2017 (cit. on p. 42).

[FS09] Matthieu Finiasz and Nicolas Sendrier. "Security Bounds for the Design of Code-based Cryptosystems". In: *Advances in Cryptology - ASIACRYPT 2009*. Ed. by M. Matsui. Vol. 5912. LNCS. Springer, 2009, pp. 88–105 (cit. on p. 210).

[Gab05] Philippe Gaborit. "Shorter keys for code based cryptography". In: *Proceedings of the 2005 International Workshop on Coding and Cryptography* (*WCC 2005*). Bergen, Norway, Mar. 2005, pp. 81–91 (cit. on p. 50).

[Gab85] Ernest M. Gabidulin. "Theory of codes with maximum rank distance". In: *Problemy Peredachi Informatsii* 21.1 (1985), pp. 3–16 (cit. on pp. 90, 91).

[Gal63] Robert G. Gallager. *Low Density Parity Check Codes*. Cambridge, Massachusetts: M.I.T. Press, 1963 (cit. on pp. 48, 51, 56, 65).

[GG17] Danilo Gligoroski and Kristian Gjøsteen. *Edon-K*. First round submission to the NIST post-quantum cryptography call. `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/EdonK.zip`. Nov. 2017 (cit. on pp. 89, 95, 99, 100, 103, 104).

[Gil52] Edgar N Gilbert. "A comparison of signalling alphabets". In: *The Bell system technical journal* 31.3 (1952), pp. 504–522 (cit. on p. 34).

[GJ20] Qian Guo and Thomas Johansson. "A New Decryption Failure Attack Against HQC". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 353–382 (cit. on p. 87).

[GJN20] Qian Guo, Thomas Johansson, and Alexander Nilsson. "A key-recovery timing attack on post-quantum primitives using the Fujisaki-Okamoto transformation and its application on Frodo-KEM". In: *Annual International Cryptology Conference*. Springer. 2020, pp. 359–386 (cit. on p. 87).

[GJS16] Qian Guo, Thomas Johansson, and Paul Stankovski. "A Key Recovery Attack on MDPC with CCA Security Using Decoding Errors". In: *Advances in Cryptology - ASIACRYPT 2016*. Ed. by Jung Hee Cheon and Tsuyoshi Takagi. Vol. 10031. LNCS. 2016, pp. 789–815 (cit. on pp. 63, 66–69, 77, 79, 81, 82, 84, 87).

[GJY19] Qian Guo, Thomas Johansson, and Jing Yang. "A novel CCA attack using decryption errors against LAC". In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2019, pp. 82–111 (cit. on p. 87).

[GKH17] Cheikh Thiécoumba Gueye, Jean Belo Klamti, and Shoichi Hirose. "Generalization of BJMM-ISD Using May-Ozerov Nearest Neighbor Algorithm over an Arbitrary Finite Field \mathbb F_q". In: *Codes, Cryptology and Information Security - Second International Conference, C2SI 2017, Rabat, Morocco, April 10-12, 2017, Proceedings - In Honor of Claude Carlet*. 2017, pp. 96–109 (cit. on p. 223).

[GL05] Ernst M Gabidulin and Pierre Loidreau. "On subcodes of codes in rank metric". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. IEEE. 2005, pp. 121–123 (cit. on p. 153).

[GL08] Ernst M. Gabidulin and Pierre Loidreau. "Properties of subspace subcodes of Gabidulin codes". In: *Adv. Math. Commun.* 2.2 (2008), pp. 147–157 (cit. on p. 153).

[GM91] Zvi Galil and Oded Margalit. "An Almost Linear-Time Algorithm for the Dense Subset-Sum Problem". In: *SIAM J. Comput.* 20.6 (1991), pp. 1157–1189 (cit. on p. 225).

[GMRZ13] 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 (cit. on pp. 60, 91, 93, 94, 99, 102, 103).

[GOT18] Philippe Gaborit, Ayoub Otmani, and Hervé Talé-Kalachi. "Polynomial-time key recovery attack on the Faure-Loidreau scheme based on Gabidulin codes". In: *Des. Codes Cryptogr.* 86.7 (2018), pp. 1391–1403 (cit. on p. 91).

[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 (cit. on p. 91).

[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 (cit. on pp. 42, 234).

[GRS16] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. "On the Complexity of the Rank Syndrome Decoding Problem". In: *IEEE Trans. Information Theory* 62.2 (2016), pp. 1006–1019 (cit. on pp. 92, 100).

[GRSZ14] P. Gaborit, O. Ruatta, J. Schrek, and G. Zémor. "RankSign: An Efficient Signature Algorithm Based on the Rank Metric". In: *Post-Quantum Cryptography*. 2014 (cit. on p. 42).

[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 (cit. on p. 112).

[Ham50] Richard W Hamming. "Error detecting and error correcting codes". In: *The Bell system technical journal* 29.2 (1950), pp. 147–160 (cit. on p. 33).

[Hat95] Masayuki Hattori. "Subspace Subcodes of Reed–Solomon Codes". PhD thesis. California Institute of Technology, 1995 (cit. on p. 154).

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. "A modular analysis of the Fujisaki-Okamoto transformation". In: *Theory of Cryptography Conference*. Springer. 2017, pp. 341–371 (cit. on p. 59).

[Hir16] Shoichi Hirose. "May-Ozerov Algorithm for Nearest-Neighbor Problem over $\mathbb{F}_q$ and Its Application to Information Set Decoding". In: *Innovative Security Solutions for Information Technology and Communications - 9th International Conference, SECITC 2016, Bucharest, Romania, June 9-10, 2016, Revised Selected Papers*. 2016, pp. 115–126 (cit. on p. 222).

[HJ10] Nicholas Howgrave-Graham and Antoine Joux. "New generic algorithms for hard knapsacks". In: *Advances in Cryptology - EUROCRYPT 2010*. Ed. by Henri Gilbert. Vol. 6110. LNCS. Sringer, 2010 (cit. on pp. 212, 225).

[HMRR13] Michel Habib, Colin McDiarmid, Jorge Ramirez-Alfonsin, and Bruce Reed. *Probabilistic methods for algorithmic discrete mathematics*. Vol. 16. Springer Science & Business Media, 2013 (cit. on p. 75).

[HMS98] Masayuki Hattori, Robert J. McEliece, and Gustave Solomon. "Subspace subcodes of Reed–Solomon codes". In: *IEEE Trans. Inform. Theory* 44.5 (1998), pp. 1861–1880 (cit. on pp. 153–155).

[HP03] W. Cary Huffman and Vera Pless. *Fundamentals of error-correcting codes*. Cambridge University Press, Cambridge, 2003, pp. xviii+646 (cit. on p. 120).

[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 (cit. on p. 25).

[Hua51] Loo-Keng Hua. "A theorem on matrices over a field and its applications". In: *J. Chinese Math. Soc.* 1.2 (1951), pp. 109–163 (cit. on p. 90).

[JD11] 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 (cit. on p. 25).

[Jen95] Jørn M. Jensen. "Subgroup subcodes". In: *IEEE Trans. Inform. Theory* 41.3 (1995), pp. 781–785 (cit. on pp. 155, 156).

[JM96] Heeralal Janwa and Oscar Moreno. "McEliece Public Key Cryptosystems Using Algebraic-Geometric Codes". In: *Des. Codes Cryptogr.* 8.3 (1996), pp. 293–307 (cit. on p. 40).

[Joz01] Richard Jozsa. "Quantum factoring, discrete logarithms, and the hidden subgroup problem". In: *Computing in science & engineering* 3.2 (2001), pp. 34–43 (cit. on p. 23).

[Kah67] D. Kahn. *The Codebreakers: The Story of Secret Writing*. Macmillan, 1967 (cit. on pp. 16, 17).

[Ker83] August Kerckhoffs. *La cryptographie militaire, ou, Des chiffres usités en temps de guerre: avec un nouveau procédé de déchiffrement applicable aux systèmes à double clef*. Extrait du Journal des sciences militaires. Librairie militaire de L. Baudoin, 1883 (cit. on p. 18).

[KI01] Kazukuni Kobara and Hideki Imai. "Semantically Secure McEliece Public-Key Cryptosystems-Conversions for McEliece PKC". In: *Public-Key Cryptography - PKC 2001*. Ed. by Kwangjo Kim. Vol. 1992. LNCS. Cheju Island, Korea: Springer, Feb. 2001, pp. 19–35 (cit. on p. 77).

[Kin09] Abu Yūsuf Ya'qūb ibn 'Ishāq as-Sabbāh al-Kindī. *Manuscript on Deciphering Cryptographic Messages*. 9th century (cit. on p. 17).

[KL85] Tadao Kasami and Shu Lin. *On the binary weight distribution of some Reed–Solomon codes*. Tech. rep. NASA, 1985 (cit. on p. 159).

[KL88] Tadao Kasami and Shu Lin. "The binary weight distribution of the extended $(2m, 2m-4)$ code of the Reed–Solomon code over $GF(2^m)$ with generator polynomial $(x-\alpha)(x-\alpha^2)(x-\alpha^3)$". In: *Linear Algebra and its Applications* 98 (1988), pp. 291–307 (cit. on p. 159).

[Koc96] Paul C Kocher. "Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems". In: *Annual International Cryptology Conference*. Springer. 1996, pp. 104–113 (cit. on p. 64).

[KRW19] Karan Khathuria, Joachim Rosenthal, and Violetta Weger. *Encryption Scheme Based on Expanded Reed–Solomon Codes*. ArXiv:1906.00745 (Version 2). 2019 (cit. on p. 169).

[KRW21] Karan Khathuria, Joachim Rosenthal, and Violetta Weger. "Encryption scheme based on expanded Reed-Solomon codes". In: *Advances in Mathematics of Communications* 15 (2021), p. 207 (cit. on pp. 40, 115, 154, 160, 164, 166, 169–171, 189, 191, 193).

[Lab91] RSA Laboratories. *RSA Factoring Challenge*. `https://web.archive.org/web/20131110040730/http://www.emc.com/emc-plus/rsa-labs/historical/the-rsa-factoring-challenge.htm`. 1991 (cit. on p. 244).

[Lam79] Leslie Lamport. *Constructing digital signatures from a one way function*. Tech. rep. CSL-98. SRI International, Oct. 1979 (cit. on p. 25).

[LB88] Pil J. Lee and Ernest F. Brickell. "An Observation on the Security of McEliece's Public-Key Cryptosystem". In: *Advances in Cryptology - EUROCRYPT'88*. Vol. 330. LNCS. Springer, 1988, pp. 275–280 (cit. on p. 209).

[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 (cit. on p. 38).

[Leo88] Jeffrey Leon. "A probabilistic algorithm for computing minimum weights of large error-correcting codes". In: *IEEE Trans. Inform. Theory* 34.5 (1988), pp. 1354–1359 (cit. on pp. 209, 210).

[LN97] Rudolf Lidl and Harald Niederreiter. *Finite fields*. Second. Vol. 20. Encyclopedia of Mathematics and its Applications. With a foreword by P. M. Cohn. Cambridge University Press, Cambridge, 1997, pp. xiv+755 (cit. on p. 159).

[LRBN14] R Lindner, M Rückert, P Baumann, and L Nobach. *TU Darmstadt Lattice Challenge*. 2014 (cit. on p. 244).

[LT18] Matthieu Lequesne and Jean-Pierre Tillich. "Attack on the Edon-K Key Encapsulation Mechanism". In: *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. 2018, pp. 981–985 (cit. on p. 89).

[LXY20] Zhe Li, Chaoping Xing, and Sze Ling Yeo. *A New Code Based Signature Scheme without Trapdoors*. Cryptology ePrint Archive, Report 2020/1250. 2020 (cit. on p. 42).

[Lyu05] Vadim Lyubashevsky. "On Random High Density Subset Sums". In: *Electronic Colloquium on Computational Complexity (ECCC)* 1.007 (2005) (cit. on p. 225).

[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 (cit. on p. 42).

[MB09] Rafael Misoczki and Paulo Barreto. "Compact McEliece Keys from Goppa Codes". In: *Selected Areas in Cryptography*. Calgary, Canada, Aug. 2009 (cit. on p. 51).

[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 (cit. on pp. 24, 36, 37, 40, 198).

[Mer87] Ralph C Merkle. "A digital signature based on a conventional encryption function". In: *Conference on the theory and application of cryptographic techniques*. Springer. 1987, pp. 369–378 (cit. on p. 25).

[Meu17] Alexander Meurer. "A Coding-Theoretic Approach to Cryptanalysis". PhD thesis. Ruhr University Bochum, Nov. 2017 (cit. on p. 222).

[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 (cit. on p. 25).

[Mil82] Frank Miller. *Telegraphic Code to Insure Privacy and Secrecy in the Transmission of Telegrams*. C.M. Cornwell, 1882 (cit. on p. 18).

[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 (cit. on pp. 213, 214).

[MN96] David JC MacKay and Radford M Neal. "Near Shannon limit performance of low density parity check codes". In: *Electronics letters* 32.18 (1996), pp. 1645–1646 (cit. on p. 48).

[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 (cit. on p. 214).

[MOG15] Ingo Von Maurich, Tobias Oder, and Tim Güneysu. "Implementing QC-MDPC McEliece Encryption". In: *ACM Trans. Embed. Comput. Syst.* 14.3 (Apr. 2015), 44:1–44:27 (cit. on p. 66).

[Moo19] Dustin Moody. *The 2nd Round of the NIST PQC Standardization Process-Opening Remarks at PQC 2019*. `https://csrc.nist.gov/Presentations/2019/the-2nd-round-of-the-nist-pqc-standardization-proc`. 2019 (cit. on p. 24).

[Moo20] Dustin Moody. *NIST PQC Standardization Update - Round 2 and Beyond*. `https://csrc.nist.gov/Presentations/2020/pqc-update-round-2-and-beyond`. 2020 (cit. on p. 24).

[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 (cit. on p. 48).

[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 (cit. on p. 40).

[MS86] Florence J. MacWilliams and Neil J. A. Sloane. *The Theory of Error-Correcting Codes*. Fifth. Amsterdam: North–Holland, 1986 (cit. on pp. 111, 112, 154).

[MS94] Robert J. McEliece and Gustave Solomon. "Trace-Shortened Reed-Solomon Codes". In: *The Telecommunications and Data Acquisition Progress Report 42-117* (1994), p. 119 (cit. on pp. 116, 154).

[MTSB13] 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 (cit. on pp. 40, 49, 51, 54, 56, 57, 62, 65).

[Nie86] Harald Niederreiter. "Knapsack-type cryptosystems and algebraic coding theory". In: *Problems of Control and Information Theory* 15.2 (1986), pp. 159–166 (cit. on pp. 37, 39, 113).

[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 (cit. on p. 92).

[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 (cit. on pp. 49, 50).

[Ove05] Raphael Overbeck. "A New Structural Attack for GPT and Variants". In: *Mycrypt*. Vol. 3715. LNCS. 2005, pp. 50–63 (cit. on p. 91).

[Ove08] Raphael Overbeck. "Structural Attacks for Public Key Cryptosystems based on Gabidulin Codes". In: *J. Cryptology* 21.2 (2008), pp. 280–301 (cit. on p. 91).

[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 (cit. on p. 25).

[Per12] Edoardo Persichetti. "Improving the efficiency of code-based cryptography". PhD thesis. ResearchSpace@ Auckland, 2012 (cit. on p. 42).

[Per18] Edoardo Persichetti. "Efficient one-time signatures from quasi-cyclic codes: A full treatment". In: *Cryptography* 2.4 (2018), p. 30 (cit. on p. 42).

[Pet10] Christiane Peters. "Information-Set Decoding for Linear Codes over $\mathbf{F}_q$". In: *Post-Quantum Cryptography 2010*. Vol. 6061. LNCS. Springer, 2010, pp. 81–94 (cit. on p. 222).

[Pra62] Eugene Prange. "The use of information sets in decoding cyclic codes". In: *IRE Transactions on Information Theory* 8.5 (1962), pp. 5–9 (cit. on p. 202).

[Ran13] Hugues Randriambololona. “Asymptotically good binary linear codes with asymptotically good self-intersection spans”. In: *IEEE transactions on information theory* 59.5 (2013), pp. 3038–3045 (cit. on p. 174).

[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 (cit. on p. 110).

[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 (cit. on pp. 20, 23).

[SBCC18] Paolo Santini, Marco Baldi, Giovanni Cancellieri, and Franco Chiaraluce. “Hindering reaction attacks by using monomial codes in the McEliece cryptosystem”. In: *2018 IEEE International Symposium on Information Theory (ISIT)*. IEEE. 2018, pp. 951–955 (cit. on p. 84).

[Sen02] Nicolas Sendrier. “Cryptosystèmes à clé publique basés sur les codes correcteurs d’erreurs”. In: *Mémoire d’habilitation à diriger des recherches, Université Paris 6*. 2002 (cit. on p. 37).

[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 (cit. on pp. 51, 60).

[Sen11a] Nicolas Sendrier. “Decoding One Out of Many”. In: *Post-Quantum Cryptography 2011*. Vol. 7071. LNCS. 2011, pp. 51–67 (cit. on pp. 61, 237).

[Sen11b] Nicolas Sendrier. “The tightness of security reductions in code-based cryptography”. In: *Proc. IEEE Inf. Theory Workshop- ITW 2011*. IEEE, 2011, pp. 415–419 (cit. on p. 39).

[Sen94] Nicolas Sendrier. “On the structure of a randomly permuted concatenated code”. In: *EUROCODE’94*. 1994, pp. 169–173 (cit. on p. 40).

[Sen98] Nicolas Sendrier. “On the Concatenated Structure of a Linear Code”. In: *Appl. Algebra Eng. Commun. Comput. (AAECC)* 9.3 (1998), pp. 221–242 (cit. on p. 40).

[Sha48] Claude E Shannon. “A mathematical theory of communication”. In: *The Bell system technical journal* 27.3 (1948), pp. 379–423 (cit. on pp. 18, 25, 35).

[Sha49] Claude Elwood Shannon. “Communication theory of secrecy systems”. In: *The Bell system technical journal* 28.4 (1949). The material in this paper appeared in a confidential report "A Mathematical Theory of Cryptography" dated Sept.1, 1946,which has now been declassified, pp. 656–715 (cit. on pp. 18, 19).

[SHMWW20] 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 (cit. on p. 42).

[Sho94] Peter W. Shor. "Algorithms for quantum computation: Discrete logarithms and factoring". In: *FOCS*. Ed. by S. Goldwasser. 1994, pp. 124–134 (cit. on p. 23).

[Sid94] Vladimir Michilovich Sidelnikov. "A public-key cryptosytem based on Reed-Muller codes". In: *Discrete Math. Appl.* 4.3 (1994), pp. 191–207 (cit. on p. 40).

[Sin00] S. Singh. *The Code Book: The Secret History of Codes and Codebreaking*. Fourth Estate, 2000 (cit. on pp. 17, 20).

[SKHN76] Yasuo Sugiyama, Masao Kasahara, Shigeichi Hirasawa, and Toshihiko Namekawa. "Further results on Goppa codes and their applications to constructing efficient binary codes". In: *it* 22 (1976), pp. 518–526 (cit. on p. 185).

[Sol93] Gustave Solomon. "Non-linear, non-binary cyclic group codes". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT*. IEEE. 1993, pp. 192–192 (cit. on p. 154).

[Spe04] Sarah A. Spence. "Identifying high-dimension subspace subcodes of Reed-Solomon codes". In: *IEEE Transactions on Information Theory* 50.6 (2004), pp. 1280–1282 (cit. on p. 154).

[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 (cit. on pp. 39, 114, 121, 122, 147, 187).

[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 (cit. on p. 210).

[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 (cit. on p. 42).

[STK89] Katsumi Sakakibara, Kin-Ichiroh Tokiwa, and Masao Kasahara. "Notes on q-ary expanded Reed–Solomon codes over $GF(q^m)$". In: *Electronics and Communications in Japan* (*Part III: Fundamental Electronic Science*) 72.2 (1989), pp. 14–23 (cit. on p. 159).

[Sud97] Madhu Sudan. "Decoding of Reed–Solomon Codes beyond the Error–Correction Bound". In: *J. Complexity* 13.1 (1997), pp. 180–193 (cit. on p. 112).

[Sue21] Gaius Suetonius Tranquillus. "Vita divi Iuli". In: *De vita Caesarum*. Vol. 1. 121 (cit. on p. 16).

[SV19] Nicolas Sendrier and Valentin Vasseur. "On the Decoding Failure Rate of QC-MDPC Bit-Flipping Decoders". In: *Post-Quantum Cryptography 2019*. Ed. by Jintai Ding and Rainer Steinwandt. Vol. 11505. LNCS. Chongquing, China: Springer, May 2019, pp. 404–416 (cit. on p. 85).

[SV20] Nicolas Sendrier and Valentin Vasseur. "About Low DFR for QC-MDPC Decoding". In: *Post-Quantum Cryptography 2020*. Ed. by Jintai Ding and Jean-Pierre Tillich. Vol. 12100. LNCS. Springer, 2020 (cit. on pp. 59, 86).

[Til18] Jean-Pierre Tillich. "The Decoding Failure Probability of MDPC Codes". In: *2018 IEEE International Symposium on Information Theory, ISIT 2018, Vail, CO, USA, June 17-22, 2018*. 2018, pp. 941–945 (cit. on p. 85).

[Var57] Rom Rubenovich Varshamov. "Estimate of the number of signals in error correcting codes". In: *Docklady Akad. Nauk, SSSR* 117 (1957), pp. 739–741 (cit. on p. 34).

[Vér96] Pascal Véron. "Improved identification schemes based on error-correcting codes". In: *Appl. Algebra Eng. Commun. Comput.* 8.1 (1996), pp. 57–69 (cit. on p. 42).

[VNT07] Serge Vladut, Dmitry Nogin, and Michael Tsfasman. "Algebraic geometric codes: basic notions". In: (2007) (cit. on p. 112).

[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 (cit. on pp. 211, 226).

[Wan16] Yongge Wang. "Quantum resistant random linear code based public key encryption scheme RLCE". In: *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2016*. Barcelona, Spain: IEEE, July 2016, pp. 2519–2523 (cit. on p. 123).

[Wan17] Yongge Wang. *RLCE–KEM*. `http://quantumca.org`. First round submission to the NIST post-quantum cryptography call. 2017 (cit. on pp. 39, 115, 123, 125, 127, 150).

[WB86] Lloyd R Welch and Elwyn R Berlekamp. *Error correction for algebraic block codes*. US Patent 4,633,470. Dec. 1986 (cit. on p. 111).

[Wie06a] Christian Wieschebrink. "An attack on a modified Niederreiter encryption scheme". In: *Public-Key Cryptography - PKC 2006*. Ed. by Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malk. Vol. 3958. LNCS. Springer, 2006, pp. 14–26 (cit. on p. 114).

[Wie06b] 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 (cit. on pp. 115, 123, 127).

[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 (cit. on pp. 114, 122).

[Wu11] Yingquan Wu. “On expanded cyclic and Reed–Solomon codes”. In: *IEEE Trans. Inform. Theory* 57.2 (2011), pp. 601–620 (cit. on pp. 155, 167).

[YDHTS15] Takanori Yasuda, Xavier Dahan, Yun-Ju Huang, Tsuyoshi Takagi, and Kouichi Sakurai. *Fukuoka MQ Challenge*. `https://www.mqchallenge.org/`. 2015 (cit. on p. 244).