Étude de la sécurité de certaines clés compactes pour le schéma de McEliece utilisant des codes géométriques

Elise Barelli

université PARIS-SACLAY

BNNT: 2018SACLX095

Thèse de doctorat

On the security of short McEliece keys from algebraic and algebraic geometry codes with automorphisms

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

École doctorale n° 580 Sciences et technologies de l'information et de la communication (STIC)
Spécialité Informatique

Thèse présentée et soutenue à Palaiseau, le 10 Décembre 2018, par

Élise Barelli

Composition du jury:

Philippe Gaborit Professeur, Université de Limoges	Président
Paulo Barreto *Assistant Professor*, Université de Washington	Rapporteur
Christophe Ritzenthaler Professeur, Institut de recherche mathématique de Rennes	Rapporteur
Magali Bardet Maître de conférence, Université de Rouen	Examinatrice
Françoise Levy-dit-Vehel Maître de conférence HDR, ENSTA, Palaiseau	Examinatrice
Pierre Loidreau Chercheur associé, DGA MI & Université de Rennes 1	Examinateur
Daniel Augot Directeur de recherche, Inria Saclay	Directeur de thèse
Alain Couvreur Chargé de recherche, Inria Saclay	Co-directeur de thèse

Remerciements

Je souhaite tout d'abord remercier chaleureusement mes deux directeurs de thèse, Alain Couvreur et Daniel Augot. Je remercie Alain pour ses nombreuses explications, notamment en géométrie algébrique, qui ont toujours été d'une grande aide pour moi et m'ont permis d'avancer tout au long de ma thèse. Il a su être un directeur de thèse attentif et disponible, tout en m'apprenant à être plus autonome dans mon travail. Ce fût une collaboration fructueuse et très agréable, j'espère que cette première expérience en tant que directeur de thèse lui aura apporté autant qu'à moi. Je remercie Daniel d'avoir accepté d'encadrer également cette thèse. Il a contribué par sa bonne humeur et ses conseils à rendre ces trois années agréables et motivantes, je l'en remercie.

J'exprime tous mes remerciements à l'ensemble des membres qui, en plus de mes directeurs de thèse, composent mon jury. Merci à Magali Bardet, Françoise Levy-dit-Vehel, Philippe Gaborit et Pierre Loidreau d'avoir accepté de prendre part à ce jury. J'adresse mes sincères remerciements à Paulo Barreto et Christophe Ritzenthaler pour avoir accepté de rapporter ma thèse ainsi que pour les corrections qu'ils m'ont suggérées pour ce manuscrit. *I would like to thank a lot Paulo Barreto for accepting to be one of my phD referees and for accepting to make the journey to France for my defense.*

Thanks to Peter Beelen, Johan Rosenkilde, Vincent Neiger and Mrinmoy Datta for the excellent welcome they reserved me during my visit in their team at the Danemark Technical University (DTU) during during the months of February and March 2017.

Je souhaite également remercier les habitués du groupe de travail « Codes et cryptographie » de l'INRIA Paris, pour les discussions toujours intéressantes que j'ai pu avoir avec eux. En particulier, je remercie sincèrement Jean-Pierre Tillich pour l'intérêt qu'il a porté à mes travaux et pour ses précieuses explications.

Merci à tous les membres de l'équipe GRACE qui ont contribué à rendre ces trois années intéressantes tant humainement que scientifiquement. Je pense en premier au cœur de l'équipe, les permanents : Daniel, Françoise, Alain, François et Ben. Il y a également les doctorants, post-doctorants et chercheurs qui ont passé quelques temps dans l'équipe. Ceux qui étaient là au début et que j'ai eu l'occasion de recroiser : Irene, Julia, Cécile, Aurore, Virgile, David et Philippe. Ceux que j'ai côtoyés plus récemment : Nick, Luca et Matthieu. *Nick, I thank you for your patience during our discussions in English. I hope I have improved my English thanks to you.* Et bien sûr les derniers arrivés qui ont égayé la partie la plus pénible d'une thèse, à savoir la rédaction : Isabella, Sarah, Lucas, Christophe et Mathilde.

Je remercie enfin Julien qui a accepté (sans savoir à quoi il s'engageait sûrement) d'être mon cobureau pendant presque 3 ans. Je le remercie pour son amitié et nos discussions tantôt scientifiques tantôt cinéphiles qui ont toujours été très enrichissantes et passionnées. Il a su réactiver ma motivation lorsque j'en ai eu besoin (ce qui fût souvent le cas).

Je remercie également les membres de l'équipe administrative de l'INRIA Saclay qui nous facilitent la vie quotidiennement. Merci en particulier à Agustina Ronco et Jessica Gameiro qui ont toujours été disponibles pour répondre à mes demandes et qui ont cela de commun d'être sympathiques et souriantes. Je remercie également la « team Tricot » pour ces moments de détente le mardi midi.

Un grand merci à mes amis qui m'ont soutenue et accompagnée durant ces trois dernières années (et pas seulement). Je pense aux amies de longue date : Julie, Audrey et les Mathilde(s)! Merci d'être toujours là après ces nombreuses années. Merci à la team « CRYPTIS » : Adrien, Pierrick, Anthony et Zoé, Nico et Flo, et Colin pour nos soirées limougeaudes, puis parisiennes, dont je garde de très bons souvenirs.

Mes plus profonds remerciements vont à mes parents et à ma sœur. Ils ont su être présents tout au long de mon cursus et m'ont apporté toute leur aide et leur soutien lorsque j'en ai eu besoin. Mes remerciements vont aussi à l'ensemble de ma famille qui, sans savoir ce qu'est la « cryptographie », n'a jamais douté de moi.

Enfin, merci à Tom d'avoir été présent dans les moments de doute comme dans les moments heureux qui ont constitué mes trois années de thèse. Merci pour sa patience et ses encouragements. Merci d'être toujours là pour moi.

Introduction

Context

Public key cryptography

In the area of cryptography, *public key cryptography*, or *asymmetric cryptography*, concerns schemes using a pair of keys: *the public key* which can be known by everyone and *the private key*, only known by one person which is the recipient of messages. This family of cryptosystems was presented for the first time by Diffie and Hellman in 1976 [DH76]. Before this date, the main family in cryptography was *secret key cryptography*, or *symmetric cryptography*, which concerns schemes using the same key to encrypt and to decrypt a message.

The first scheme in the area of public key cryptography was proposed in 1978 by Rivest, Shamir and Adleman [RSA78], it is the encryption scheme RSA. The security of this scheme is based on the problem of factoring large numbers. Since 40 years, other schemes have been proposed whose security is based on problems coming from number theory too, as the discrete logarithm problem. However, these kinds of problems could be solved in polynomial time using a quantum computer, with the Shor's algorithm [Sho94]. This leads to expand the domain of public key cryptography to scheme relying on other assumptions than number-theory problems and resistant to quantum algorithms. *Post-quantum cryptography* is the domain which concerns these kinds of cryptosystem. In this thesis, we are interested in the case of cryptography using error correcting codes which is based on the problem of decoding a random linear code. For now, no quantum algorithm is known to decode a random linear code in polynomial time, that is why code based cryptography is an interesting family in the area of post-quantum cryptography.

Code-based cryptography

In 1978, McEliece [McE78] introduced a public key encryption scheme based on linear codes. The idea is to use an error correcting code whose structure would be hidden, making it impossible to decode a message for anyone who do not know a specific decoding algorithm for the chosen code. Therefore the encryption consists in encoding a message, thanks to the knowledge of a basis of the code, then introducing errors in the encoded word. The public basis of the chosen code is represented by a matrix referred to as a *generator matrix* of the code. This matrix has to seem random to hide the structure of the code, it is the public key. The private key is an efficient decoding algorithm coming from the knowledge of the structure of the code. So the decryption consists in applying this decoding algorithm to recover the encoded word and then the message.

Someone who wants to decrypt the cyphertext, without the knowledge of the private key, has two solutions. First, one can try to decode the message as a noisy word of a random code, since the public key is a random looking matrix. This method is related to the *(decisional) generic decoding problem* which consists from a random code $\mathscr{C}$, a vector $\mathbf{y}$ in the ambient space and an integer t in deciding if there exists a vector $\mathbf{e}$ of weight t such that $\mathbf{y} - \mathbf{e} \in \mathscr{C}$. In general, this problem is hard to solve and actually in [BMvT78] the authors proved that the problem is NP-complete. The second method consists in recovering the structure of the chosen code from the public generator matrix. From the knowledge of this structure one can build a decoding

algorithm to decode any message. Therefore the security of the McEliece scheme is related both on the hardness of the generic decoding problem and on the difficulty of recovering the structure of the chosen code. In the literature, the attacks corresponding to the first problem are referred to as *message recovery attacks* while attacks corresponding to recovering the structure are known as *key recovery attacks*.

The original proposal of McEliece [McE78] suggests to use classical binary Goppa codes which belong to the family of alternant codes. Up to now, all attacks on the scheme using classical Goppa codes have exponential complexity and one considers that this scheme remains secure. Since 1978, several proposals based on other families of algebraic codes appeared in the literature. In 1986, Niederreiter [Nie86] proposed to use Generalized Reed Solomon (GRS) codes but this family is subject to a polynomial attack presented by Sidelnikov and Shestakov few years later in [SS92]. In [Sid94], Sidelnikov proposed a McEliece-like scheme using Reed Muller codes and this family was also attacked [MS07]. In 1996, Janwa and Moreno [JM96] proposed to use algebraic geometry (AG) codes, concatenation or subfield subcode of these codes. For the version with concatenation of codes, Sendrier has discovered an effective attack in [Sen94]. For AG codes on curves with genus ≤ 2, Faure and Minder proposed an attack [FM08, Min07, Fau09], and in 2014 the scheme with AG codes have been completely broken by Couvreur, Marquez-Corbella and Pellikaan [CMCP14] who proposed an attack against AG codes for any genus. These attacks lead us to consider the last proposition, that is schemes using subfield subcode of AG codes (SSAG in short). For these codes no other proposition has been made until now.

Reducing key size by using quasi-cyclic codes

The McEliece scheme has some advantages, encryption and decryption are very fast and it is a good candidate for public-key cryptography in the context of quantum computer. The main constraint is that the public key is too large compared to other actual public-key cryptosystems. Many proposals have been made in order to reduce the key size. A manner to do this is to use quasi-cyclic (QC) or quasi-dyadic (QD) codes. The idea is to use codes with a structure which permits to describe a given generator matrix with only few rows. The case of QC codes is a solution presented in [Gab05] where quasi-cyclic subcodes of BCH codes have been proposed. However, this proposal cannot be used since this family has not enough possible keys. This first paper was followed by proposals using alternant and classical Goppa codes with different automorphism groups like (QC) alternant codes [BCGO09] or quasi-dyadic Goppa codes [MB09, BLM11]. Actually, these codes are nothing else but subfield subcodes of algebraic geometry codes on the projective line.

Since 2010, in the category of key-recovery attacks, new methods appeared, known as *algebraic attacks*. This method consists in recovering the secret element of an alternant code by solving a system of polynomial equations. In [FOPT10], the authors improved this new method to attack QC and QD alternant codes and broke all the parameters proposed in [BCGO09]. Such attacks use the specific structure of QC/QD codes in order to build an algebraic system with much fewer unknowns compared to the generic case. A new approach has been used in [FOP$^+$16a, FOP$^+$16b] to explain that the reduction of the number of unknowns in the algebraic system comes from a smaller code easily computable from the public generator matrix. This smaller code can be obtained by summing up the codewords which belong to the same orbit under the action of the permutation group and is referred to as the *folded* code. This time the authors broke some parameters of proposals [MB09] and [BLM11].

Contributions

Analysis of McEliece scheme using quasi-cyclic alternant codes

In Chapter 3, we study the security of the key of compact McEliece schemes based on alternant/Goppa codes with a non-trivial permutation group, in particular quasi-cyclic alternant codes. We improve the technique of Faugère, Otmani, Perret, Portzamparc and Tillich in [FOP+16a] which uses the folded code to analyse the security. We show that it is possible to reduce the key-recovery problem on the original quasi-cyclic code to the same problem on a smaller code derived from the public key: the *invariant* code. We use a simpler approach with a unified view on quasi-cyclic alternant codes and we treat the case of automorphisms arising from a non affine homography. This last case was not treated in [FOP+16a]. In addition, we provide an efficient algorithm to recover the full structure of the alternant code from the structure of the invariant code.

The invariant code. In [FOP+16a], the authors attack only codes with an automorphism induced by an affine transformation acting on the support and the multiplier, we call them *affine induced* automorphisms. Another kind of quasi-cyclic alternant codes can be built from the action of the projective linear group on the support and multiplier. In order to analyse any QC alternant code built from the action of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ on support and multiplier, we use the invariant code, introduced by Loidreau in [Loi01], and which is the subcode whose elements are fixed by a given permutation. This invariant code can be built easily from the public generator matrix of the alternant code $\mathscr{C}$ since it is the kernel of the linear map: $\mathbf{c} \in \mathscr{C} \mapsto \mathbf{c} - \sigma(\mathbf{c})$, where σ is a permutation of $\mathscr{C}$.

Our main contribution is to consider more general tools coming from algebraic geometry and use the invariant code instead of folded code. This approach has two advantages. First the geometric point of view simplifies the analysis by giving a unified view of quasi-cyclic alternant codes. It also simplifies some proofs and enables to consider alternant codes as algebraic geometric codes on the projective line. With these two tools, the geometric point of view and the invariant code, we prove the following results.

Theorem 3.7. *Let $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) \subset \mathbb{F}_{q^m}^n$ be a quasi-cyclic GRS code, and $\sigma \in \mathfrak{S}_n$ of order ℓ, such that $\ell | n$, the permutation acting on the code $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$. Then the invariant code $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})^\sigma$ is a GRS code of length n/ℓ and dimension $\lfloor k/\ell \rfloor$.*

Corollary 3.8. *Let $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}) \subseteq \mathbb{F}_q^n$ be a quasi-cyclic alternant code, and $\sigma \in \mathfrak{S}_n$ of order ℓ, such that $\ell | n$, the permutation acting on the code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$. Then the invariant code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})^\sigma$ is an alternant code of length n/ℓ and order r/ℓ.*

Reduction of the key security. The structure of the invariant code can be exploited to recover a part of the secret element of the proposed alternant code. It remains to know if this part is sufficient to recover completely the secret elements. It is the study of Section 3.3. In that section we show that the key recovery of McEliece scheme using QC alternant codes built from induced automorphisms on the support and the multiplier, reduces to the key security of the invariant code. In other words, from the knowledge of the support and multiplier of the invariant code we are able to recover the support and the multiplier of the QC alternant code. However, we can notice that the key recovery on the invariant code can be hard if we chose good parameters for the QC alternant code.

NIST competition for post-quantum cryptography

In the last decade, quantum computing progressed and the existence of a quantum computer would permit to break usual cryptography primitives based on number theoretic problems. This explains the growing interest for post-quantum cryptography and with this for code-based cryptography. This change of interest was proved by the recent call of the National Institute for Standards and Technology (NIST) for post quantum cryptography [1].

Contribution to the scheme BIG QUAKE using QC Goppa codes In this context, I participated to the submission BIG QUAKE which proposed to use QC Goppa codes. More precisely, the proposal is a key-encapsulation scheme based on quasi-cyclic binary Goppa codes, reducing the key size by a moderate factor ℓ in the range [3..19] which is the order of quasi-cyclicity of the code. We provide an analysis of the key security based on the security of the invariant code. We study the cost of algebraic attacks against the invariant code and choose parameters out of the reach of these attacks. Moreover, we choose very carefully the order of quasi-cyclicity ℓ. These parameters are chosen in order to provide no more information than the invariant code itself. That is, we avoid the case where another automorphism group acts on the public code. In that way an attacker cannot compute another invariant code from another automorphism. We do the same for folded code.

An attack on the submission DAGS using quasi-dyadic alternant codes Among the NIST submissions, the proposal DAGS [BBB+17a] is based on using quasi-dyadic (QD) Strivastava codes. Strivastava codes form a subfamily of alternant codes and the term quasi-dyadic means that codes are stable under a permutation group of the form $(\mathbb{Z}/2\mathbb{Z})^s$, with $s \in \mathbb{N}^*$. The study of the proposal DAGS showed a weaknesses coming from the large size of the permutation group and the choice of the authors to take an extension degree 2 for alternant codes. An attack exploiting these weaknesses permits to recover the secret key in $O(n^{3+\frac{2q}{|\mathcal{G}|}})$ operations over $\mathbb{F}_q$, where n is the length, $\mathcal{G}$ is the permutation group and $\mathbb{F}_q$ is the base field of the code. This attack is based on the use of *Schur product* and *conductor* operations. The key step of the attack consists in finding some subcode of the public code referred to as $\mathscr{D}$. From this subcode, using the conductor operation, we can compute a subfield subcode of Reed Solomon (RS) code whose support is same as the public code. This subfield subcode of RS code is very small and it is easy to recover its support. The difficult part of the attack is to recover the code $\mathscr{D}$. For this, the naive approach is to preform a brute force search on subcodes of given dimension. Actually, we can improve this brute force search and the code $\mathscr{D}$ can be recovered in 2^{70}, 2^{80} and 2^{58} operations over $\mathbb{F}_q$ for the proposed schemes claiming a security of 2^{128}, 2^{192} and 2^{256} binary operations. Another method was studied to recover directly an RS code from which the secret key can easily be recovered. This approach is based on solving a polynomial system of degree 2, using Gröbner bases. We are not able to provide a complexity analysis for this second method. However its practical implementation using Magma [BCP97] is impressively efficient on some DAGS parameters. In particular, it permits to break claimed 256 bits security keys in less than one minute!

Security analysis of quasi-cyclic algebraic geometry codes

As in the case of classical alternant codes, we show that in the general case of quasi-cyclic AG code, the invariant code is also an AG codes. Actually, we can described this smaller AG code as an AG code on the quotient curve $\mathcal{X}/\langle\sigma\rangle$. More precisely we show the two following results.

[1] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`

Theorem 5.2. *Let $\mathcal{X}$ be an algebraic curve and G be a divisor of $\mathcal{X}$ invariant by an automorphism $\sigma \in Aut(\mathcal{X})$. Let $\mathcal{P}$ be a set of n distinct places of $\mathcal{X}$, of degree 1, such that $\sigma(\mathcal{P}) = \mathcal{P}$. Then the invariant code $\mathscr{C}_L(\mathcal{X}, \mathcal{P}, G)^\sigma$ is the AG code $\mathscr{C}_L(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$, for some $\tilde{\mathcal{P}} \subseteq \mathcal{X}/\langle\sigma\rangle$ and $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$.*

Corollary 5.3. *With the notation of Theorem 5.2, let $SSAG_q(\mathcal{X}, \mathcal{P}, G)$ be a subfield subcode of an AG code and σ acting on it. Then the invariant code $SSAG_q(\mathcal{X}, \mathcal{P}, G)^\sigma$ is the code $SSAG_q(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$ for some $\tilde{\mathcal{P}} \subset \mathcal{X}/\langle\sigma\rangle$ and $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$*

A first example: QC codes from cyclic cover of the projective line An illustrative example to understand the link between the security of the QC AG code and the invariant code is codes over cyclic covers of the projective line. In this thesis, we choose to study curves with an equation on the form $y^\ell = f(x)$. It is easy to provide an automorphism for such a curve, it suffices to consider the automorphism $\sigma : (x, y) \mapsto (x, \xi y)$, where ξ is an ℓ-th primitive root of unity. Moreover, we can compute Riemann-Roch spaces stable under this automorphism σ, then we can construct easily QC SSAG codes on these curves. In this particular case, the invariant code is an SSAG code over the projective line, that is a subfield subcode of GRS code. In this thesis we show that, from the knowledge of secret elements of the invariant code we are able to recover the secret elements of the SSAG code over the cyclic cover of the projective line. This means that the key security of QC SSAG codes over the curve "$y^\ell = f(x)$" reduces to the key security of a subfield subcode of a GRS code.

QC codes from the Hermitian curve To build quasi-cyclic SSAG alternant codes, we need to use curves with non-trivial automorphisms for which we can easily compute Riemann-Roch spaces. In this thesis, we chose to propose QC SSAG codes on the Hermitian curve. The Hermitian curves presents two advantages. First, it is a maximal curve and so we can construct codes with bigger length than the size of the field. This permits us to obtain better parameters than codes on random curves. The second advantage is that its automorphism group is well known and we can use it to construct QC codes. Actually, all quotient curves of the Hermitian curve have been studied in [GSX00] and the authors provide a formula to know the genus of these quotient curves. From that, we are able to construct QC SSAG code on the Hermitian curve such that the invariant code is not a classical alternant code. Then we provide an analysis of the key security based on the security of the invariant code. We know that the invariant code is an SSAG code over the quotient curve and we saw that in the case of cyclic cover of the projective line, the quotient curve is the projective line. Then to avoid this case, the choice of the automorphism on the Hermitian curve is crucial.

Improvement of lower bound on the minimum distance for some AG codes

In March 2017, I had the opportunity to work in a research team from the COMPUTE laboratory at the University of Lyngby in Denmark. This collaboration resulted in an article [BBD+17]. This work is an improvement of the lower bound on the minimum distance of certain geometrical codes "two points" on the generalized Giulietti–Korchmaros (GGK) curves. These codes admit a divisor whose support has only two points and the improvement of the lower bound on the minimum distance is based on the study of the Weierstrass semi-group of these two points. This result is not directly related to code-based cryptography and reduction of key length in this domain, that is why we choose to report this result in Appendix A.

Résumé

Contexte

Cryptographie à clé publique

Dans le domaine de la cryptographie, la *cryptographie à clé publique*, ou *cryptographie asymétrique*, regroupe les schémas utilisant une paire de clés : *la clé publique* qui est connue de tous et permet de réaliser le chiffrement et *la clé privée*, seulement connue du destinataire des messages chiffrés et qui permet le déchiffrement. Cette famille de schémas de chiffrement a été présentée pour la première fois par Diffie et Hellman en 1976 [DH76]. Avant cela, les schémas cryptographiques provenaient de la *cryptographie à clé secrète*, ou *cryptographie symétrique* qui utilise la même clé pour le chiffrement et le déchiffrement.

Le premier schéma en cryptographie asymétrique fût proposé en 1978 par Rivest, Shamir et Adelman [RSA78], c'est le schéma de chiffrement RSA. La sécurité de ce schéma repose sur le problème de factorisation des grands entiers. Depuis 40 ans, d'autres schémas à clé publique ont été proposés dont la sécurité repose principalement sur des problèmes issus de la théorie des nombres, comme par exemple le problème du logarithme discret. Cependant, ce type de problèmes pourrait être résolu en temps polynomial par un ordinateur quantique, en utilisant l'algorithme de Shor [Sho94]. Cela a renforcé l'intérêt pour des schémas de chiffrement à clé publique dont la sécurité repose sur d'autres problèmes qui peuvent résister à l'ordinateur quantique. La *cryptographie post-quantique* est précisément le domaine qui étudie ces schémas. Dans cette thèse, on s'intéresse plus particulièrement au cas de la cryptographie utilisant des codes correcteurs d'erreurs et dont la sécurité repose sur le problème du décodage d'un code linéaire. À ce jour, aucun algorithme quantique n'est connu pour résoudre le problème du décodage d'un code linéaire en temps polynomial. C'est pour cela que la cryptographie basée sur les codes est une famille de schémas intéressante dans le domaine de la cryptographie post-quantique.

Cryptographie basée sur les codes correcteurs d'erreurs

En 1978 McEliece introduit dans [McE78] un schéma de chiffrement à clé publique issu de la théorie des codes correcteurs d'erreurs. L'idée du schéma d de McEliece est d'utiliser un code correcteur dont la structure est masquée, rendant le décodage de ce code difficile pour toute personne ne connaissant pas cette structure. Le schéma de chiffrement est alors défini de la manière suivante : le chiffrement consiste à encoder un message, grâce une matrice génératrice d'un code $\mathscr{C}$, et y introduire une erreur. La matrice génératrice choisie doit sembler aléatoire, c'est la clé publique du système. La clé privée est la connaissance de la structure du code $\mathscr{C}$ permettant d'utiliser un algorithme de décodage efficace; le déchiffrement est précisément l'application de l'algorithme de décodage.

La sécurité du schéma de chiffrement repose à la fois sur la difficulté du décodage d'un code aléatoire et sur la difficulté de retrouver la structure du code à partir de la seule clé publique. Le premier problème consiste, à partir d'une matrice génératrice $\mathbf{G}$ d'un code linéaire, d'un vecteur y de l'espace ambiant et d'un entier positif t, à décider de l'existence d'un vecteur x de poids

de Hamming inférieur à t tel que $x\mathbf{G} = y$. Ce problème spécifique fut prouvé NP-complet en 1978 par Berlekamp, McEliece et Van Tilborg [BMvT78]. Le second problème est la recherche de la clé privée, qui est la structure du code utilisé permettant la construction d'un algorithme de décodage efficace pour ce code. La méthode générique est la recherche exhaustive sur la famille du code choisi. Les méthodes plus spécifiques dépendent de la famille de codes choisie pour le schéma. Cela représente donc deux types d'attaques possibles : le premier type est appelé « attaque sur le message », le second est appelé « attaque structurelle » ou « attaque sur la clé privée ».

La première famille de codes proposée par McEliece [McE78], est celle des codes de Goppa binaires classiques qui reste actuellement une bonne candidate pour un système résistant aux deux types d'attaques. Depuis 1978, plusieurs propositions utilisant des codes algébriques ont été faites. En 1986, Niederreiter [Nie86] propose un schéma équivalent au schéma de McEliece et donne en exemple l'utilisation des codes de Reed Solomon généralisés (GRS). Cette famille de codes n'est cependant pas résistante aux attaques structurelles puisque en 1992 Sidelnikov et Shestakov propose un algorithme polynomial qui permet de retrouver la structure des codes GRS [SS92]. Dans l'article [Sid94], Sidelnikov propose un schéma utilisant les codes de Reed-Muller mais cette famille a aussi été attaquée [MS07]. En 1996, Janwa et Moreno [JM96] proposent d'utiliser des codes issus de la géométrie algébrique. Ils proposent notamment les codes algébriques géométriques (AG), la concaténation de code AG ou les sous-codes sur un sous- corps des codes AG. Pour la version utilisant des codes concaténés, Sendrier avait déjà proposé une attaque en 1994 [Sid94]. Pour les codes AG sur les courbes de genre ≤ 2, Faure et Minder ont proposé une attaque [FM08, Min07, Fin09] et en 2014 le schéma utilisant les codes AG a été complètement attaqué par Couvreur, Marquez-Corbella et Pellikaan [CMCP14] qui proposent une attaque contre les codes AG sur des courbes de n'importe quel genre. Cette série d'attaque nous pousse à considérer la dernière proposition, à savoir les sous-codes sur un sous-corps de code AG (SSAG). Pour ces codes, aucune proposition de paramètres n'a été faite jusqu'à maintenant.

Réduire la taille des clés publiques en utilisant des codes quasi-cycliques

Le schéma de McEliece présente plusieurs avantages: le chiffrement et le déchiffrement sont assez rapides, de plus ce schéma est un bon candidat dans le contexte de la cryptographie post-quantique. La principale contrainte pour une utilisation pratique de ce schéma est la taille de la clé publique qui est importante en comparaison à d'autres systèmes de chiffrement à clé publique. Pour cette raison, de nombreuses propositions ont été faites avec des codes plus structurés admettant des clés plus faciles à stocker. C'est le cas des codes quasi-cycliques et en particulier les codes de Goppa et les codes alternants quasi-cycliques ont été proposés. L'idée est d'utiliser des codes structurés permettant une description compacte de la matrice génératrice, c'est à dire avec seulement quelques lignes. Le cas des codes quasi-cycliques a été présenté dans l'article [Gab05] où des sous-codes quasi-cycliques de codes BCH ont été proposés. Cependant, cette proposition n'est pas sécurisée puisque la famille choisie ne contient pas suffisamment de codes distincts. Ce premier papier a été suivi par des propositions utilisant des codes alternants et des codes de Goppa quasi-cycliques [BCGO09] ou des codes de Goppa quasi-dyadique [MB09, BLM11]. Ces codes sont simplement des sous-codes sur un sous-corps de codes AG construits sur la droite projective.

Depuis 2010, dans le domaine des attaques structurelles, de nouvelles méthodes sont apparues, elles sont connues sous le nom d'*attaques algébriques*. Cette technique se base sur la construction d'un système algébrique dont la clé secrète est solution. Cette modélisation algébrique ne permet pas en soi de retrouver la clé secrète dans le cas général des codes alternants lorsque les paramètres du code sont trop importants. Cependant dans le cas de codes alternants quasi-cycliques, ce système d'équation peut être simplifié et donne lieu à une attaque. Dans l'article [FOPT10], les auteurs proposent cette méthode pour retrouver la structure de certains codes alternants quasi-cycliques et quasi- dyadiques et attaquent différents paramètres

proposés dans [BCGO09]. Cette attaque utilise la structure spécifique des codes quasi-cycliques ou quasi-dyadiques pour construire un système algébrique avec moins d'inconnues que pour le cas générique. Dans [FOP$^+$16b, FOP$^+$16a] les auteurs expliquent que cette réduction du nombre d'inconnues provient d'un code alternant plus petit que l'on peut construire à partir de la matrice génératrice. Cette fois les auteurs ont pu attaquer les paramètres proposés dans [MB09] et [BLM11].

Contribution

Sécurité du schéma de McEliece utilisant des codes alternants quasi-cycliques

Dans le chapitre 3, nous étudions la sécurité du schéma de McEliece utilisant des codes alternants, ou des codes de Goppa, quasi- cycliques en améliorant le travail de Faugère, Otmani, Perret, Portzamparc et Tillich [FOP$^+$16a]. En effet, l'attaque décrite dans [FOP$^+$16a] fonctionne uniquement pour les automorphismes induits par les transformations affines de la droite projective. D'autres permutations existent, ce sont celles induites par l'action du groupe linéaire projectif. On peut construire un code alternant plus petit à partir de la matrice génératrice d'un code alternant quasi-cyclique dont la permutation est induite par une transformation linéaire projective. Ce résultat est possible grâce à l'utilisation du *code invariant*. En utilisant une approche plus simple grâce à la géométrie algébrique, nous sommes capable d'améliorer le résultat du cas affine et d'étendre l'attaque à tous les codes alternants dont le support est globalement invariant par une homographie. Ce cas n'était pas traité dans [FOP$^+$16a]. De plus, nous proposons un algorithme efficace qui permet, en temps polynomial, de retrouver la structure d'un code alternant quasi-cyclique à partir de la structure de son code invariant. La structure d'un code invariant étant défini par son *support* et son *multiplier*, ce sont ces éléments qui seront considérés comme secrets.

Le sous-code invariant. Dans le but d'étudier la sécurité des codes alternants quasi-cycliques construits à partir de l'action de $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ sur le support et le multiplier, nous utilisons le code invariant. Ce code a été introduit par Loidreau dans [Loi01] et est le sous-code strictement invariant par la permutation. Ce code peut facilement être construit à partir de la matrice génératrice du code original $\mathscr{C}$, puisqu'il s'agit du noyau de l'application linéaire: $\mathbf{c} \in \mathscr{C} \mapsto \mathbf{c} - \sigma(\mathbf{c})$, où σ désigne la permutation qui agit sur $\mathscr{C}$.

Notre principale contribution, en plus de considérer le code invariant, est d'utiliser des outils provenant de la géométrie algébrique. Cette approche a deux avantages. Tout d'abord, le point de vue géométrique simplifie l'analyse en donnant une définition unifiée des codes alternants quasi-cycliques. De plus ce point de vue géométrique permet de simplifier certaines preuves en considérant les codes alternants comme des codes géométriques sur la droite projective. En étudiant le code invariant, nous avons pu démontrer les résultats suivants.

Théorème 3.7. *Soit $\mathbf{GRS}_k(x, y) \subset \mathbb{F}_{q^m}^n$ un code GRS quasi-cyclique, avec $\sigma \in \mathfrak{S}_n$ d'ordre ℓ, tel que $\ell \mid n$, la permutation qui agit sur le code $\mathbf{GRS}_k(x, y)$. Alors le code invariant $\mathbf{GRS}_k(x, y)^\sigma$ est un code GRS de longueur n/ℓ et de dimension $\lfloor k/\ell \rfloor$.*

Corollary 3.8. *Soit $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}) \cap \mathbb{F}_q^n$ un code alternant quasi-cyclique, avec $\sigma \in \mathfrak{S}_n$ d'ordre ℓ, tel que $\ell \mid n$, la permutation qui agit sur le code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$. Alors le code invariant $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})^\sigma$ est un code alternant de longueur n/ℓ et de degré $\lfloor r/\ell \rfloor$.*

Réduction de sécurité. La structure du code invariant peut être utilisée pour retrouver une partie de la structure du code alternant quasi-cyclique choisi pour le schéma de chiffrement. Il

reste à savoir si cette partie est suffisante pour retrouver complètement la structure du code choisi, c'est à dire son support et son multiplier. Cette question est étudiée dans la section 3.3. Dans cette section, nous montrons qu'une attaque sur la clé publique d'un schéma de McEliece utilisant des codes alternants quasi-cycliques se réduit à une attaque sur le code invariant. C'est à dire qu'à partir du support et du multiplier du code invariant nous pouvons retrouver, en temps polynomial, le support et le multiplier du code original. Cependant, on peut noter que retrouver le support et le multiplier du code invariant peut s'avérer difficile si on choisit bien les paramètres du code.

La compétition du NIST pour une cryptographie post-quantique

Au cours de la dernière décennie, des progrès ont été faits dans le domaine du calcul quantique et l'existence d'un ordinateur quantique menacerait la sécurité d'une longue liste de systèmes de chiffrement à clé publique. Cela explique l'intérêt croissant pour la cryptographie post-quantique et ainsi pour la cryptographie basée sur les codes. Ce changement d'intérêt a été prouvé par le récent appel à contribution du National Institut for Standards and Technology (NIST) pour la standardisation de schémas cryptographiques post-quantique [2].

Contribution à la proposition BIG QUAKE. Dans ce contexte, j'ai eu l'occasion de participer à la soumission BIG QUAKE [3] (BInary Goppa QUAsi–cyclic Key Encapsulation). Plus précisément, la proposition est un schéma d'encapsulation de clé utilisant des codes de Goppa quasi-cycliques et réduisant la taille de la clé publique par un facteur $\ell \in [3..19]$, qui est l'ordre de la permutation utilisée. Nous proposons une analyse de la sécurité du schéma basée sur la sécurité du code invariant. Nous avons en particulier étudié le coût d'une attaque algébrique sur le code invariant et proposé des paramètres en fonction de ces données. Les paramètres sont choisis dans le but de ne pas donner plus d'information que le code invariant lui-même. C'est à dire, nous avons évité les cas où un autre groupe d'automorphismes agit sur le code publique. De cette manière, une attaquant ne peut pas construire d'autres sous-codes invariants.

Attaque structurelle de DAGS, une soumission à l'appel du NIST. Parmi les soumissions faites au NIST pour la standardisation de schéma post-quantique, une proposition, DAGS [BBB+17a], est basée sur l'utilisation de codes de Srivastava quasi-dyadiques. Les codes de Srivastava forment une sous-famille des codes alternants. Le terme quasi-dyadique signifie que les codes sont invariants par un groupe de permutations de la forme $(\mathbb{Z}/2\mathbb{Z})^s$, avec $s \in \mathbb{N}^*$. L'étude de DAGS a révélé une faille provenant de la taille trop importante du groupe de permutations choisi et du degré d'extension 2. Une attaque exploitant cette faille permet de retrouver la clé secrète en $\mathcal{O}(n^{3+\frac{2q}{|\mathcal{G}|}})$ opérations dans $\mathbb{F}_q$, avec n la longueur du code utilisé et $\mathcal{G}$ le groupe de permutations qui agit sur le code. Cette attaque est basée sur la recherche d'un sous-code particulier du code invariant, le code *norme-trace*. Une fois ce sous-code obtenu, le support et le multiplier du code de Srivastava, qui sont les éléments secrets, se calculent facilement grâce à l'utilisation du produit de codes et de l'algèbre linéaire. La partie difficile est de retrouver ce sous-code, pour cela une première approche est la recherche exhaustive. Avec cette méthode il est possible de retrouver le code norme-trace pour les paramètres de clés proposés dans DAGS, de sécurité 128, 192 et 256, en effectuant une recherche sur respectivement 2^{70}, 2^{80} et 2^{58} éléments. Une seconde méthode, basée sur la résolution d'un système bilinéaire, a aussi été étudiée pour retrouver le code norme-trace mais nous n'avons pas pu produire une analyse de sa complexité. En revanche cette seconde méthode a été implémentée, avec l'aide du logiciel MAGMA, est permet de retrouver la clé secrète en moins d'une minute pour les paramètres DAGS_5, d'une sécurité de 256 bits, et moins de 20 minutes pour les paramètres DAGS_1, d'une sécurité de 128 bits. Toutefois cette

[2] `https://csrc.nist.gov/Projects/Post-Quantum-Cryptography`
[3] `https://bigquake.inria.fr/`

seconde méthode n'a pas permis d'attaquer les paramètres proposés pour le niveau de sécurité 192 bits.

Étude de la sécurité du schéma de McEliece utilisant des codes géométriques quasi-cycliques

À l'instar des codes alternants quasi-cycliques, nous avons montré que dans le cas général des codes géométriques quasi-cycliques, le sous-code invariant est aussi un code géométrique. On peut alors décrire ce sous-code invariant comme un code géométrique sur la courbe quotient $\mathcal{X}/\langle\sigma\rangle$. Plus précisément, nous avons pu montrer les deux résultats suivants :

Théorème 5.2. *Soit $\mathcal{X}$ une courbe algébrique et G un diviseur de $\mathcal{X}$ invariant par un automorphisme $\sigma \in Aut(\mathcal{X})$. Soit $\mathcal{P}$ un ensemble de n places distinctes de $\mathcal{X}$, de degré 1, tel que $\sigma(\mathcal{P}) = \mathcal{P}$. Alors le code invariant $\mathscr{C}_L(\mathcal{X}, \mathcal{P}, G)^\sigma$ est un code géométrique $\mathscr{C}_L(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$, pour un certain support $\tilde{\mathcal{P}} \subseteq \mathcal{X}/\langle\sigma\rangle$ et un certain diviseur $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$.*

Corollaire 5.3. *En conservant les mêmes notations que le théorème 5.2, soit $SSAG_q(\mathcal{X}, \mathcal{P}, G)$ le sous-code sur un sous-corps d'un code géométrique globalement invariant par σ. Alors le code invariant $SSAG_q(\mathcal{X}, \mathcal{P}, G)^\sigma$ est le code $SSAG_q(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$ pour un certain support $\tilde{\mathcal{P}} \subset \mathcal{X}/\langle\sigma\rangle$ et un certain diviseur $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$*

Des codes construits sur des revêtements de la droite projective. Un premier exemple de codes géométriques a été étudié, celui des codes construits à partir de revêtement cyclique de la droite projective, ie : des courbes d'équation algébrique de la forme $y^\ell = f(x)$. L'étude de cette famille de courbes est particulièrement intéressante pour deux raisons. La première est que l'on connaît au moins un automorphisme σ de cette courbe, c'est celui qui au couple (x, y) associe le couple $(x, \xi y)$, où ξ est une racine ℓ-ième de l'unité. Il est donc possible de construire un code alternant géométrique sur cette courbe qui admette σ comme automorphisme. La seconde raison est que l'on sait facilement décrire les espaces de Riemann-Roch globalement invariants par σ, en prenant par exemple un espace de Riemann-Roch associé au point à l'infini sur cette courbe. On peut donc aisément construire des sous-codes sur un sous-corps de codes géométriques, avec une description compacte, sur ces courbes. Dans ce cas particulier, le code invariant est alors un code alternant puisqu'il s'agit d'un sous-code sur un sous-corps de codes géométriques sur la droite projective. Dans cette thèse, nous montrons qu'il est possible de retrouver le support et le diviseur du code géométrique original à partir des éléments secrets du code invariant. Cela signifie que la sécurité de la clé des sous-codes sur un sous-corps de codes géométriques quasi-cycliques sur la courbe « $y^\ell = f(x)$ », se réduit à la sécurité d'un code alternant plus petit.

Codes quasi-cycliques sur la courbe Hermitienne. La courbe Hermitienne est une courbe souvent étudiée mais elle présente deux caractéristiques intéressantes dans le cadre de la théorie des codes correcteurs. Tout d'abord c'est une courbe maximale, c'est à dire que son nombre de points rationnels atteint la borne supérieure du nombre de points rationnels des courbes de même genre. Cela permet de construire des codes plus longs que la taille de l'alphabet choisi et d'obtenir des paramètres meilleurs que pour des codes construits sur des courbes aléatoires. Le second intérêt de la courbe Hermitienne est la connaissance de son groupe d'automorphismes. Ce groupe d'automorphismes est de taille assez importante et a été bien étudié. On peut construire des codes quasi-cycliques grâce à l'action d'un automorphisme de la courbe Hermitienne. De plus, les quotients de la courbe Hermitienne ont aussi été bien étudié dans [GSX00] et les auteurs proposent une formule pour calculer le genre de ces courbes quotient. En étudiant ces courbes quotient, nous sommes capable de construire des sous- codes sur un sous-corps de code AG sur la courbe Hermitienne dont le code invariant n'est pas un code alternant classique. Le choix de

l'automorphisme est important dans cette étape. À partir de cela, nous fournissons une analyse de la sécurité structurelle d'un schéma utilisant ces codes, basée sur la sécurité du code invariant.

Amélioration de la borne inférieure sur la distance minimale de certains codes géométriques

Au mois de mars 2017 j'ai eu l'occasion de travailler au sein d'une équipe de recherche du laboratoire COMPUTE de l'Université de Lyngby au Danemark. Cette collaboration a donné lieu à un article [BBD+17].

Ce travail est une amélioration de la borne inférieure sur la distance minimale de certains codes géométriques « deux points » sur les courbes de Giulietti-Korchmaros généralisées. Ces codes admettent un diviseur dont le support ne comporte que deux points et l'amélioration de la borne sur la distance minimale repose sur l'étude du semi-groupe de Weierstrass de ces deux points. Ce résultat n'est pas directement lié à la cryptographie sur les codes, j'ai donc fait le choix de le détailler dans l'annexe A.

Contents

Introduction **5**

Résumé **11**

1 Algebraic geometry codes **19**
1.1 Coding theory 19
1.2 Introduction to algebraic geometry 22
1.3 AG codes: definition and properties 29

2 Code-based cryptography **37**
2.1 McEliece encryption scheme 37
2.2 Information Set Decoding (ISD) 38
2.3 Key recovery problem 39
2.4 Algebraic cryptanalysis of McEliece schemes using alternant codes 40

3 McEliece scheme using quasi-cyclic alternant codes **43**
3.1 Quasi-cyclic alternant codes 44
3.2 Structural analysis of the invariant code 48
3.3 Security reduction to the key security of the invariant code 54
3.4 Proposition of a scheme: BIG QUAKE 59

4 A structural attack on a scheme using quasi-dyadic alternant codes **69**
4.1 Presentation of the NIST submission: DAGS 70
4.2 Schur products and conductors 72
4.3 Using the QD structure to compute a conductor 77
4.4 Description of the attack 78
4.5 Algorithm, work factor and implementation 84

5 Short McEliece keys from codes on curves with positive genus **87**
5.1 Construction of quasi-cyclic SSAG codes 87
5.2 Structure of the invariant code 88
5.3 QC codes from a cyclic cover of the projective line 90
5.4 The McEliece scheme with QC Hermitian codes 95

Bibliography **108**

List of Notations **108**

List of Algorithms **109**

List of Tables **111**

A Two-Point Codes for the Generalised GK curve **113**
A.1 Introduction 113
A.2 Preliminaries 114
A.3 The generalized Giulietti–Korchmáros function field 118
A.4 Two-point AG codes on the generalized GK curve. 121
A.5 Computation of the order bound and results 124

Chapter 1

Algebraic geometry codes

1.1 Coding theory

Let $\mathbb{F}_q$ be a finite field of q elements, with q a power of a prime p. For the following definitions and properties we refer to [MS86] and [HP03].

1.1.1 Linear codes

Definition 1.1 (Linear code)**.** Let n and k be two non-negative integers with $k \leq n$. A *linear* $[n, k]$ *code* over $\mathbb{F}_q$ is a subset $\mathscr{C} \subseteq \mathbb{F}_q^n$ of dimension k. The integer n is the *length* of $\mathscr{C}$, k its *dimension* and a vector of $\mathscr{C}$ will be called a *codeword*. The *information rate* of the code is the ratio $R := \frac{k}{n}$.

A *generator matrix* $\mathbf{G}$ of the code $\mathscr{C}$ is a matrix whose rows are formed by a basis of $\mathscr{C}$. Then we have

$$\mathscr{C} = \{\mathbf{mG} \mid \mathbf{m} \in \mathbb{F}_q^k\}.$$

A *parity check matrix* $\mathbf{H}$ of the code $\mathscr{C}$ is a $(n-k) \times k$ matrix over $\mathbb{F}_q$ of rank $(n-k)$ such that

$$\forall \mathbf{c} \in \mathscr{C}, \mathbf{Hc}^\top = \mathbf{0}.$$

There exist several generator matrices, such matrices are not unique and it is the same for parity check matrices. However, it is convenient to have a generator matrix with the specific form

$$\mathbf{G} = \left(\mathbf{I}_k \mid A\right),$$

for some $k \times (n-k)$ matrix A over $\mathbb{F}_q$. If it is the case, the matrix $\mathbf{G}$ is said to be *systematic*. A linear code does not always admit a systematic generator matrix but, if it is the case, this matrix is unique and we say that the code is *systematic*. In this thesis we assume on many occasions that linear codes considered are systematic.

Definition 1.2 (Dual code)**.** Let $\mathscr{C}$ be a linear code over $\mathbb{F}_q$, its *dual* or *orthogonal* code, denoted $\mathscr{C}^\perp$, is the set of all vectors which are orthogonal to all codewords of $\mathscr{C}$, that is

$$\mathscr{C}^\perp := \left\{\mathbf{y} \in \mathbb{F}_q^n \mid \mathbf{xy}^\top = 0 \text{ for all } \mathbf{x} \in \mathscr{C}\right\}.$$

Any parity check matrix of a linear code $\mathscr{C}$ is a generator matrix for its dual $\mathscr{C}^\perp$. Then the dual code $\mathscr{C}^\perp$ has the same length than $\mathscr{C}$, that is n, and dimension $n-k$ with k the dimension of $\mathscr{C}$.

Definition 1.3. The *Hamming distance* between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_q^n$, denoted $d_H(\mathbf{x}, \mathbf{y})$, is the number of positions where they differ, that is

$$d_H(\mathbf{x}, \mathbf{y}) := \left| \{x_i \neq y_i \mid i \in \{1, \ldots, n\}\} \right|.$$

The *Hamming weight* of a vector $\mathbf{x} \in \mathbb{F}_q^n$, denoted $w_H(\mathbf{x})$, is the number of non-zero component x_i, that is

$$w_H(\mathbf{x}) := \big|\, \{x_i \neq 0 \mid i \in \{1, \ldots, n\}\} \,\big| = d_H(\mathbf{x}, \mathbf{0}).$$

In this thesis, we work only with the Hamming metric then for short we speak about the distance between two words and the weight of a word.

Definition 1.4. The *minimum distance* of a code $\mathscr{C}$, denoted d, is the minimum (Hamming) distance between its codewords, that is

$$d := \min\{d_H(\mathbf{x}, \mathbf{y}) \mid \mathbf{x} \neq \mathbf{y} \in \mathscr{C}\}.$$

For a linear code it is also the minimum weight of its non-zero codewords, that is to say $d = \min\{w_H(\mathbf{x}) \mid \mathbf{x} \in \mathscr{C} \setminus \{\mathbf{0}\}\}$.

Later on, a linear code of length n, dimension k and minimum distance d will be called an $[n, k, d]$ code. The following theorem makes a link between these three parameters.

Theorem 1.1 (The Singleton bound)**.** *If $\mathscr{C}$ is an $[n, k, d]$ code, then*

$$n - k \geq d - 1.$$

1.1.2 Permutation and automorphism groups

Let $\mathfrak{S}_n$ be the group of permutations of $\{1, \ldots, n\}$. For this section, definitions and details can be also found in [Dür87].

Definition 1.5 (Permutation group)**.** Let $\mathscr{C}$ be a linear code of length n over $\mathbb{F}_q$. Let $\sigma \in \mathfrak{S}_n$ be a permutation, acting on $\mathscr{C}$ via $\sigma(\mathbf{c}) = (c_{\sigma(1)}, \ldots, c_{\sigma(n)})$. The code $\mathscr{C}$ is said to be *invariant* by σ if $\sigma(\mathscr{C}) = \mathscr{C}$. The *permutation group* of the code $\mathscr{C}$, is

$$\mathrm{Perm}(\mathscr{C}) := \{\sigma \in \mathfrak{S}_n \mid \sigma(\mathscr{C}) = \mathscr{C}\}.$$

More generally, the semi-direct product $(\mathbb{F}_q^*)^n \rtimes \mathfrak{S}_n$ acts on a linear code $\mathscr{C}$ of length n by

$$(\mathbf{a}, \sigma)(\mathbf{c}) := (a_1 c_{\sigma(1)}, \ldots, a_n c_{\sigma(n)}).$$

Moreover, the group law in $(\mathbb{F}_q^*)^n \rtimes \mathfrak{S}_n$ is defined by

$$(\mathbf{a}, \sigma) \cdot (\mathbf{b}, \tau) := (\mathbf{a} \star \sigma(\mathbf{b}), \sigma \circ \tau),$$

where the symbol $\star$ define the multiplication component by component. If we consider a linear code $\mathscr{C}$ over a prime field $\mathbb{F}_p$ then such elements $(\mathbf{a}, \sigma) \in (\mathbb{F}_p^*)^n \rtimes \mathfrak{S}_n$ which let $\mathscr{C}$ invariant form a group that we call the *automorphism group of $\mathscr{C}$*. Now considering codes over a field $\mathbb{F}_q$ with q a prime power, the automorphism group of $\mathscr{C}$ also contains any field automorphisms of $\mathbb{F}_q$. Then we consider the most general definition of the automorphism group of a linear code as follows.

Definition 1.6 (Automorphism group)**.** Let $\mathscr{C}$ be a linear code of length n over $\mathbb{F}_q$. Let π be any field automorphism of $\mathbb{F}_q$, $\mathbf{a}$ a vector of $(\mathbb{F}_q^*)^n$ and $\sigma \in \mathfrak{S}_n$ a permutation. Then we define the action on $\mathscr{C}$ of the semi-direct product $(\mathbb{F}_q^*)^n \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathfrak{S}_n)$ by

$$(\mathbf{a}, \pi, \sigma)(\mathbf{c}) := (\pi(a_1 c_{\sigma(1)}), \ldots, \pi(a_n c_{\sigma(n)})),$$

where $\mathbf{c} \in \mathscr{C}$. Moreover, the group law in $(\mathbb{F}_q^*)^n \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathfrak{S}_n)$ is defined by

$$(\mathbf{a}, \pi, \sigma) \cdot (\mathbf{b}, \gamma, \tau) := (\mathbf{a} \star \pi(\sigma(\mathbf{b})), \pi \circ \gamma, \sigma \circ \tau).$$

The group of elements $(\mathbf{a}, \pi, \sigma) \in (\mathbb{F}_q^*)^n \rtimes (\mathrm{Aut}(\mathbb{F}_q) \times \mathfrak{S}_n)$ which let $\mathscr{C}$ invariant is called the *automorphism group* of $\mathscr{C}$ and denoted by $\mathrm{Aut}(\mathscr{C})$.

Remark 1. In the binary case, that is when we consider codes over $\mathbb{F}_2$, the automorphism group and permutation group coincide. Otherwise we have only the inclusion $\mathrm{Perm}(\mathscr{C}) \subseteq \mathrm{Aut}(\mathscr{C})$.

Proposition 1.2. *Let $\mathscr{C}$ be a linear code, then we have*

$$\mathit{Perm}(\mathscr{C}) = \mathit{Perm}(\mathscr{C}^\perp).$$

1.1.3 Construction of new codes from old

Puncturing and shortening codes. The two operations, puncturing or shortening a code, result in a reduction of the length of the code. Moreover for the shortening, the dimension is also reduced, in general.

Definition 1.7 (Puncturing)**.** Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a code and $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a set of coordinates. The puncturing of the code $\mathscr{C}$ at $\mathcal{I}$, denoted $\mathrm{Punct}_{\mathcal{I}}(\mathscr{C})$, is defined by

$$\mathrm{Punct}_{\mathcal{I}}(\mathscr{C}) := \{(c_i)_{i \in \{1,\ldots,n\} \setminus \mathcal{I}} \mid \mathbf{c} \in \mathscr{C}\}.$$

This is a code of length $n - |\mathcal{I}|$.

Proposition 1.3. *Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a $[n, k, d]$ code and $\mathcal{I} \subseteq \{1, \ldots, n\}$. Then the code $Punct_{\mathcal{I}}(\mathscr{C})$ is an $[n - |\mathcal{I}|, k', d']$ code with*

$$k - |\mathcal{I}| \leq k' \leq k \quad \textit{and} \quad d - |\mathcal{I}| \leq d' \leq d.$$

Definition 1.8 (Shortening)**.** Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a code and $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a set of coordinates. The shortening of the code $\mathscr{C}$ at $\mathcal{I}$, denoted $\mathrm{Short}_{\mathcal{I}}(\mathscr{C})$, is defined by

$$\mathrm{Short}_{\mathcal{I}}(\mathscr{C}) := \mathrm{Punct}_{\mathcal{I}}\left(\{\mathbf{c} \in \mathscr{C} \mid c_i = 0 \text{ for all } i \in \mathcal{I}\}\right).$$

This is a code of length $n - |\mathcal{I}|$.

Proposition 1.4. *Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a $[n, k, d]$ code and $\mathcal{I} \subseteq \{1, \ldots, n\}$. Then the code $Short_{\mathcal{I}}(\mathscr{C})$ is an $[n - |\mathcal{I}|, k', d']$ code with*

$$k - |\mathcal{I}| \leq k' \leq k \quad \textit{and} \quad d \leq d'.$$

The following result makes a link between these two constructions.

Theorem 1.5. *Let $\mathscr{C} \subseteq \mathbb{F}_q^n$ be a code and $\mathcal{I} \subseteq \{1, \ldots, n\}$ be a set of coordinates. Then*

$$Short_{\mathcal{I}}(\mathscr{C}^{\perp}) = (Punct_{\mathcal{I}}(\mathscr{C}))^{\perp}.$$

Codes over subfield. In what follows we deal with codes over a finite extension $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$. For some codewords of these codes it is possible that all their components lie over the subfield $\mathbb{F}_q$. This leads to the following definition.

Definition 1.9. Let $\mathscr{C} \subset \mathbb{F}_{q^m}^n$ be a code of length n over $\mathbb{F}_{q^m}$. The *subfield subcode* of $\mathscr{C}$ over $\mathbb{F}_q$, denoted $\mathscr{C}|_{\mathbb{F}_q}$, is the subcode of all the codewords of $\mathscr{C}$ whose all entries lie in $\mathbb{F}_q$, i.e.:

$$\mathscr{C}|_{\mathbb{F}_q} := \mathscr{C} \cap \mathbb{F}_q^n.$$

Theorem 1.6. *Let $\mathscr{C}$ be an $[n, k, d]$ code over $\mathbb{F}_{q^m}$, then $\mathscr{C}|_{\mathbb{F}_q}$ is an $[n, k', d']$ code over $\mathbb{F}_q$ with*

$$k' \geq n - m(n - k) \quad \textit{and} \quad d' \geq d.$$

Another construction permits to have a subcode of a code $\mathscr{C} \in \mathbb{F}_{q^m}$ defined over the subfield $\mathbb{F}_q$. This use the trace function as follows.

Definition 1.10. Let $\mathscr{C} \subset \mathbb{F}_{q^m}^n$ be a code of length n over $\mathbb{F}_{q^m}$. The *trace code* of $\mathscr{C}$ over $\mathbb{F}_q$ is the code defined by:

$$\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}) := \left\{(\mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(c_1), \ldots, \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(c_n)) \mid c = (c_1, \ldots, c_n) \in \mathscr{C}\right\}.$$

Remark 2. When there is no ambiguity on the subfield $\mathbb{F}_q$, we denote the trace code $\mathrm{Tr}(\mathscr{C})$.

Theorem 1.7 (Delsarte Theorem [Del75])**.** *Let $\mathscr{C} \subset \mathbb{F}_{q^m}^n$ be a linear code, then:*

$$(\mathscr{C} \cap \mathbb{F}_q^n)^{\perp} = \mathrm{Tr}_{\mathbb{F}_{q^m}/\mathbb{F}_q}(\mathscr{C}^{\perp}).$$

1.2 Introduction to algebraic geometry

In this section, we introduce some definitions and properties about algebraic curves that will be used in this thesis, we refer to [Mor93] and [Ful08]. This background will permit us to define algebraic geometry (AG) codes and some properties of these codes. Details for this part can be found in [TVN07] and [HvLP98]. For some technical results, we will also use the language of function field theory, for more details see [Sti09].

Later on, we will denote $\mathbb{F}$ any field algebraically closed. We denote by $\mathbf{A}^n$ the n-dimensional affine space over $\mathbb{F}$.

1.2.1 Algebraic curves

In this section, we only treat *projective curve* and in order to give a definition for this object, we need first to define the notion of projective space. The n-dimensional *projective space* over $\mathbb{F}$, denoted by $\mathbf{P}^n(\mathbb{F})$ or for short $\mathbf{P}^n$, consists in all equivalence classes of $(n+1)$-tuples, denoted $P := (x_1 : \cdots : x_{n+1})$, with $x_i \in \mathbb{F}$ and not all zero, under the relation:

$$(x_1 : \cdots : x_{n+1}) \equiv (y_1 : \cdots : y_{n+1}) \iff \exists \lambda \in \mathbb{F}^* \text{ s.t } \forall i \in \{1, \ldots, n+1\},\ x_i = \lambda y_i.$$

The equivalent classes P, described just above, are called *points* of the projective space $\mathbf{P}^n$ and any $(n+1)$-tuple defining a point P is called a set of *homogeneous coordinates* of P. There is a natural embedding $\mathbf{A}^n \hookrightarrow \mathbf{P}^n$ given by $(x_1, \ldots, x_n) \mapsto (x_1 : \cdots : x_n : 1)$. The points of the complementary set of $\mathbf{A}^n$ in $\mathbf{P}^n$, that is the points of $\mathbf{P}^n$ such that $x_{n+1} = 0$, are called *points at infinity*.

Definition 1.11. A polynomial $F \in \mathbb{F}[X_1, \ldots, X_{n+1}]$ is *homogeneous* of degree d if for any $(n+1)$-tuple $(x_1, \ldots, x_{n+1}) \in \mathbb{F}^{n+1}$ and any scalar $\lambda \in \mathbb{F}^*$, we have

$$F(\lambda x_1, \lambda x_2, \ldots, \lambda x_{n+1}) = \lambda^d F(x_1, \ldots, x_{n+1}).$$

The "evaluation" of a polynomial $F \in \mathbb{F}[X_1, \ldots, X_{n+1}]$ in a point P of $\mathbf{P}^n$ **does not make sense** since it depends on the choice of the set of homogeneous coordinate defining P. However, if the polynomial is homogeneous then its *zero set* is well-defined. Let $S \subseteq \mathbb{F}[X_1, \ldots, X_{n+1}]$ be any set of homogeneous polynomials, we associate with S the following set, called the zero set of S:

$$Z(S) := \{P \in \mathbf{P}^n \mid f(P) = 0 \text{ for every } f \in S\}.$$

A subset $\mathcal{Y} \subseteq \mathbf{P}^n$ is a *projective algebraic set* if there exists $S \subseteq \mathbb{F}_q[X_1, \ldots, X_{n+1}]$ of homogeneous polynomials such that $\mathcal{Y} = Z(S)$. The set $\mathcal{Y}$ is called *irreducible* if $\mathcal{Y}$ is non-empty and cannot be written as the union of two proper algebraic subset $\mathcal{Y} = \mathcal{Y}_1 \cup \mathcal{Y}_2$, such that $\mathcal{Y}_1 \nsubseteq \mathcal{Y}_2$ and $\mathcal{Y}_2 \nsubseteq \mathcal{Y}_1$. A topology can be defined on projective algebraic sets as follows:

Definition 1.12 (Zariski topology)**.** The Zariski topology on $\mathbf{P}^n$ is defined by taking the open sets to be the complement of algebraic sets.

Definition 1.13. A *projective variety* is an irreducible closed subset of $\mathbf{P}^n$. An open subset of a projective variety is a *quasi-projective* variety.

Let $\mathcal{Y}$ be an algebraic set, the homogeneous ideal of $\mathcal{Y}$ is the ideal $I(\mathcal{Y})$ generated by the set $\{f \in \mathbb{F}[X_1, \ldots, X_{n+1}]$ homogeneous polynomial s.t. $f(P) = 0,\ \forall P \in \mathcal{Y}\}$.

Definition 1.14 (Coordinate ring)**.** Let $\mathcal{Y}$ be an algebraic set, we define the coordinate ring of $\mathcal{Y}$ to be $\mathbb{F}_q[\mathcal{Y}] := \mathbb{F}_q[X_1, \ldots, X_{n+1}]/I(\mathcal{Y})$.

Later on, a projective or a quasi-projective variety will be referred to as a variety. Let $\mathcal{Y}$ be a variety and $F, G \in \mathbb{F}[X_1, \ldots, X_{n+1}]$ be homogeneous polynomials of same degree with $G \notin I(\mathcal{Y})$. Then the fraction $\frac{F}{G} \in \mathbb{F}(X_1, \ldots, X_{n+1})$ is called a *rational function* on $\mathcal{Y}$. The elements $\frac{F}{G}$ and $\frac{F'}{G'}$ define the same rational function if the polynomial $FG' - F'G$ vanishes at all $P \in \mathcal{Y}$.

Definition 1.15. The *function field* $\mathbb{F}(\mathcal{Y})$ of a variety $\mathcal{Y}$ is the field of rational functions on $\mathcal{Y}$ and the *dimension* of $\mathcal{Y}$ is the transcendence degree of $\mathbb{F}(\mathcal{Y})$ over $\mathbb{F}$.

Then we can define what is a projective algebraic curve as follows.

Definition 1.16 (Projective curve)**.** A projective curve over $\mathbb{F}$, denoted $\mathcal{X}/\mathbb{F}$ or for short $\mathcal{X}$, is defined as a variety of dimension one over the field $\mathbb{F}$.

Example 1. In the affine plane over a finite field $\mathbb{F}_q$, we consider the variety $\mathcal{X}$ defined by the homogeneous polynomial $Y^3 - X^3 - Z^3$. Let $x := \frac{X}{Z}$ and $y := \frac{Y}{Z}$, the function field $\mathbb{F}(\mathcal{X})$ consists of all elements of the form $\frac{P}{Q}$ with $P, Q \in \mathbb{F}_q[x, y]$. Since y satisfies $y^3 = x^3 + 1$ the transcendence degree of $\mathbb{F}(\mathcal{X})$ over $\mathbb{F}$ is 1, that is the variety $\mathcal{X}$ is a curve.

As we see in Definition 1.15, we can associate to any curve $\mathcal{X}$ an object that we call *function field of* $\mathcal{X}$. The function fields will be studied in the following section. For now, we speak about a notion that we use all the time in this thesis, the smoothness of a curve. The first definition holds for projective plane curves, that is projective curve $\mathcal{X} \subseteq \mathbf{P}^2$. This case is easier to understand and actually for results presented in this thesis it is the only one that we consider.

Definition 1.17. Let $\mathcal{X}$ be a projective plane curve defined by the homogeneous polynomial $F \in \mathbb{F}[X, Y, Z]$. Let P be a point on $\mathcal{X}$. Then P is said to be *nonsingular* if at least one of the partial derivative $\frac{\partial F}{\partial X}, \frac{\partial F}{\partial Y}, \frac{\partial F}{\partial Z}$ is not zero at P. The curve $\mathcal{X}$ is called *nonsingular* or *smooth* if all its points are nonsingular.

Example 2. Consider the plane curve $\mathcal{X}$ defined by the polynomial $F(X, Y, Z) := Y^3 - X^3 - Z^3$ over a finite field $\mathbb{F}_q$. The partials derivatives are $-3X^2$, $3Y^2$ and $-3Z^2$, so the curve is smooth if the characteristic is not 3.

Definition 1.18. Let $\mathcal{X}$ be a projective curve and P be a point of this curve. Then a function $f \in \mathbb{F}(\mathcal{X})$ is called *regular* at the point P if it admits a representation $f = \frac{F}{G}$ with G non zero at P.

Definition 1.19 (Local ring of a point)**.** Let P be a point on a projective curve $\mathcal{X}$. The set of regular functions at the point P from a ring, denoted $\mathcal{O}_P$, and called the local ring of P.

The terminology for the ring $\mathcal{O}_P$ makes sense since it is a *local* ring, that is $\mathcal{O}_P$ has a unique maximal ideal. More precisely, we have the following proposition.

Proposition 1.8. *The subset $\mathfrak{m}_P \subseteq \mathcal{O}_P$ consisting of functions $f \in \mathcal{O}_P$ such that $f(P) = 0$, is the unique maximal ideal of $\mathcal{O}_P$.*

Proof. The subset $\mathfrak{m}_P \subseteq \mathcal{O}_P$ is obviously an ideal, we show that it is maximal. Let N be an ideal of $\mathcal{O}_P$ such that $\mathfrak{m}_P \subsetneq N$. We consider $f \in N$ such that $f \notin \mathfrak{m}_P$ and we write $f = \frac{F}{G}$. Since $f \in \mathcal{O}_P$ we have $G(P) \neq 0$. Moreover $f \notin \mathfrak{m}_P$, then $F(P) \neq 0$. That is the function $\frac{G}{F} \in \mathcal{O}_P$ and then $1 = \frac{G}{F} \cdot f \in \mathbb{N}$. Hence $N = \mathcal{O}_P$ and $\mathfrak{m}_P$ is maximal.

To complete the proof we notice that actually $\mathcal{O}_P \setminus \mathfrak{m}_P = {\mathcal{O}_P}^\times$, that is any invertible element of $\mathcal{O}_P$. Then $\mathfrak{m}_P$ contains any proper ideal of $\mathcal{O}_P$. □

Curve over finite field. In our application, we need to consider finite field $\mathbb{F}_q$ and not its algebraic closure $\overline{\mathbb{F}}_q$. A way to define curve over finite field is the following.

A projective curve $\mathcal{X}/\mathbb{F}_q$ *over the finite field* $\mathbb{F}_q$ is a projective curve $\mathcal{X} \subseteq \mathbf{P}^n(\overline{\mathbb{F}}_q)$ such that the homogeneous polynomials defining this curve have coefficients in $\mathbb{F}_q$. The field of rational functions on $\mathcal{X}$ with coefficients in $\mathbb{F}_q$ will be denoted as $\mathbb{F}_q(\mathcal{X})$ and we say it is the function field of $\mathcal{X}/\mathbb{F}_q$. Moreover, we denote by $\mathcal{X}(\mathbb{F}_q)$ the set of points of $\mathcal{X}$ with coordinates lying in $\mathbb{F}_q$, such points are called *rational points*.

Example 3. Consider the Klein curve $\mathcal{K}_3$ defined, over $\mathbb{F}_4$, by the homogeneous equation

$$X^3Y + Y^3Z + Z^3X = 0.$$

We denote the element of $\mathbb{F}_4$ as $0, 1, \alpha, \alpha+1$, then we have

$$\mathcal{K}_3(\mathbb{F}_4) = \{(0:0:1), (\alpha:\alpha+1:1), (\alpha+1:\alpha:1), (1:0:0), (0:1:0)\}.$$

Example 4 (Hermitian curve)**.** Let $q = q_0^2$ and consider the curve $\mathcal{H}$ over $\mathbb{F}_q$ defined by the equation

$$Y^{q_0}Z + YZ^{q_0} - X^{q_0+1} = 0.$$

This curve is called the *Hermitian curve* over the field $\mathbb{F}_q$. The curve $\mathcal{H}$ has q_0^3+1 rational points described by

$$\mathcal{H}(\mathbb{F}_q) = \left\{P_{\alpha,\beta} := (\alpha:\beta:1) \mid \alpha^{q_0+1} = \beta^{q_0} + \beta\right\} \cup P_\infty,$$

where $P_\infty := (0:1:0) \in \mathbf{P}^2$.

Definition 1.20. A *closed point* P of a projective plane curve $\mathcal{X}$, defined over $\mathbb{F}_q$, is an orbit under the Frobenius automorphism $\mathfrak{f}_q : (x:y:z) \mapsto (x^q:y^q:z^q)$. The *degree* of a closed point is defined as the cardinality of the orbit.

1.2.2 Function fields

In this section, it is not necessary that the field $\mathbb{F}$ is algebraically closed. To avoid misunderstanding we keep notation of previous section and in any of the following results and definitions, $\mathbb{F}$ can be replaced by a finite field $\mathbb{F}_q$. This algebraic point of view will be useful in Section 5 that is why here we give a small dictionary between projective curves and function fields in one variable. If there is no mention, details about this section can be found in [Sti09]. First, let us recall what is a function field in one variable.

Definition 1.21. An algebraic function field $F/\mathbb{F}$ in one variable over the field $\mathbb{F}$ is an extension field $\mathbb{F} \subseteq F$ such that F is a finite algebraic extension of $\mathbb{F}(x)$, where $x \in F$ is transcendental over $\mathbb{F}$.

By Definitions 1.15 and 1.16, we know that the field of rational functions $\mathbb{F}(\mathcal{X})$ of a projective curve is a function field in one variable. The converse is also true. This result can be found in [TVN07] as follows and a proof is given in [Mor93].

Theorem 1.9 ([Mor93, Theorem 1.1])**.** *Let $F/\mathbb{F}$ be a function field in one variable, then there exist a smooth irreducible projective curve $\mathcal{X}$ such that the field $\mathbb{F}(\mathcal{X})$ is isomorphic (as an extension of $\mathbb{F}$) to F.*

Definition 1.22. A *valuation ring* of the function field $F/\mathbb{F}$ is a ring $\mathcal{O} \subseteq F$ such that

(1) $\mathbb{F} \subsetneq \mathcal{O} \subsetneq F$

(2) for any $x \in F$, we have $x \in \mathcal{O}$ or $x^{-1} \in \mathcal{O}$.

In Section 1.2.1 we defined the local ring $\mathcal{O}_P$ of a point $P \in \mathcal{X}$. It is a valuation ring of the function field $\mathbb{F}(\mathcal{X})$, that is why the notations are similar. Let us give some properties of valuation rings.

Proposition 1.10. *Let $\mathcal{O}$ be a valuation ring of a function field $F/\mathbb{F}$.*

(1) $\mathcal{O}$ is a local ring, i.e. $\mathcal{O}$ has a unique maximal ideal $P = \mathcal{O}\backslash\mathcal{O}^\times$.

(2) Let $x \in F$, then $x \in P \iff x^{-1} \notin \mathcal{O}$.

(3) The maximal ideal P of $\mathcal{O}$ is a principal ideal.

Now we define the notion of *place*, which replaces that of points of a curve (or closed points if we consider a curve over $\mathbb{F}_q$) when we use the language of function field.

Definition 1.23. A *place* P of the function field $F/\mathbb{F}$ is the maximal ideal of some valuation ring $\mathcal{O}$ of F. We denote by $\mathbb{P}_F$ the set of all places of the function field F. Every element t such that $P = t\mathcal{O}$ is called a *local parameter* for P.

By Proposition 1.10 (2), we know that the valuation ring $\mathcal{O}$ is uniquely determined by its maximal ideal P. We can denote by $\mathcal{O}_P = \{x \in F \mid x^{-1} \notin P\}$ the *valuation ring of the place* P.

Remark 3. For the case where $\mathbb{F}$ is algebraically closed, let $F = \mathbb{F}(\mathcal{X})$ for some smooth projective curve $\mathcal{X}$. There is a one-to-one correspondence between the set of places $\mathbb{P}_F$ and the points of $\mathcal{X}$. Then the valuation ring of a place $P \in \mathbb{P}_F$ coincides with the local ring of a point $Q \in \mathcal{X}$. That is why there is no ambiguity in the notation $\mathcal{O}_P$ in this case.

By this correspondence and Proposition 1.10 (3), the maximal ideal $\mathfrak{m}_P$, where P is a point of a smooth curve $\mathcal{X}$, is principal. As in Definition 1.23 we call a local parameter of the point P, any element $t \in \mathbb{F}(\mathcal{X})$ such that $\mathfrak{m}_P = t\mathcal{O}_P$.

Proposition 1.11. *If $P = t\mathcal{O}_P$, then any function $x \in F$ has a unique representation of the form $x = t^n u$, with $n \in \mathbb{Z}$ and $u \in \mathcal{O}_P^\times$.*

We consider $x \in \mathcal{O}_P$ a function such that its unique representation is $x = t^n u$, where t is a local parameter of P. If $n > 0$ then we say that P is a *zero of* x, if $n < 0$ we say that P is a *pole of* x. We denote $v_P(x) = n$ the *valuation of* x *at* P. For any place P, v_P is a *discrete valuation* of $F/\mathbb{F}$, that is it satisfies the following properties.

(1) $v_P(x) = \infty \iff x = 0$.

(2) $v_P(xy) = v_P(x) + v_P(y)$ for all $x, y \in F$.

(3) $v_P(x + y) \geq \min\{v_P(x), v_P(y)\}$ for all $x, y \in F$.

(4) There exists an element $z \in F$ such that $v_P(z) = 1$.

(5) $v_P(a) = 0$ for all $a \in \mathbb{F}^*$.

Note that the definition of v_P depends only of the place P and not on the choice of the local parameter. Indeed, if we consider t' another local parameter of P, then we have $t = t'w$, with $w \in \mathcal{O}_P^\times$. Hence $x = (t'w)^n u = t'^n (w^n u)$ and $w^n u \in \mathcal{O}_P^\times$.

Proposition 1.12. *Any non-zero function x of a function field F, has only finitely many zeros and poles.*

Definition 1.24. The field $F_P := \mathcal{O}_P/P$ is called the *residue class field of* P. Then the *degree* of P is defined as $\deg(P) := [F_P : \mathbb{F}]$. A place of degree 1 is also called a *rational place*.

The degree of a place is always finite (see [Sti09, Proposition 1.1.15]).

Correspondence with curves over a finite field. In Remark 3, we talked about the one-to-one correspondence between points and places in the case where the defining field $\mathbb{F}$ is algebraically closed. What happens now if we consider curve defined over a finite field $\mathbb{F}_q$? Let $\mathcal{X}/\mathbb{F}_q$ be a projective curve defined over $\mathbb{F}_q$ and $\mathbb{F}_q(\mathcal{X})$ its function field. In this case the places of $\mathbb{F}_q(\mathcal{X})$ are in one-to-one correspondence with the closed points of $\mathcal{X}$. This correspondence permits to transfer notions from the language of curve to the language of function field (and reciprocally).

Automorphisms and fixed field. Let $F/\mathbb{F}_q$ be a function field, we consider $\mathrm{Aut}(F)$ the (field) automorphism group of F. For a finite subgroup $\mathcal{G} \subset \mathrm{Aut}(F)$, we denote by $F^{\mathcal{G}}$ the function field consisting in all functions of F fixed by every element of $\mathcal{G}$. Then the field extension $F/F^{\mathcal{G}}$ is Galois of degree $|\mathcal{G}|$. Let $\mathcal{X}$ be a smooth projective curve associated to F. The curve associated to $F^{\mathcal{G}}$ is called the *quotient curve* of $\mathcal{X}$ by $\mathcal{G}$ and denoted by $\mathcal{X}/\mathcal{G}$.

Definition 1.25. Let F be a function field and $\mathcal{G} \subseteq \mathrm{Aut}(F)$. Let $P \in \mathbb{P}_F$ and $P' \in \mathbb{P}_{F^{\mathcal{G}}}$ be two places.

(i) P is said to *lie over* P' if $P' \subseteq P$, and we write $P|P'$.

(ii) The integer $e(P|P')$ such that

$$v_P(x) = e(P|P')v_{P'}(x),\ \forall x \in F$$

is called the *ramification index* of P over P'. We say that $P|P'$ is *ramified* if $e(P|P') > 1$ and *unramified* otherwise.

The following property will be useful in Section 5.

Proposition 1.13. *Let F be a function field and $\sigma \in Aut(F)$. Let $P \in \mathbb{P}_F$ and $P' \in \mathbb{P}_{F^{\langle\sigma\rangle}}$ such that $P|P'$, then $\sigma(P) := \{\sigma(z) \mid z \in P\}$ is a place of F such that*

$$\sigma(P)|P' \text{ and } e(\sigma(P)|P') = e(P|P').$$

1.2.3 Divisors

In this section we choose to present the different notions in the geometric point of view, that is we speak about curve and points rather than function field and places. As we saw in the previous section all theses notions can be transferred in the algebraic language. Through this section $\mathcal{X}$ denotes a smooth irreducible projective curve over a finite field $\mathbb{F}_q$.

Definition 1.26 (Divisors)**.** The *divisor group* of a projective curve $\mathcal{X}$ is defined as the free abelian group which is generated by the closed points of $\mathcal{X}$. This group is denoted by $\mathrm{Div}(\mathcal{X})$. A *divisor* G on $\mathcal{X}$, is an element of $\mathrm{Div}(\mathcal{X})$ and so has the following form

$$G = \sum_{P \in \mathcal{X}} n_P P,$$

where P are closed points of $\mathcal{X}$ and n_P are integers all zero but finitely many. The *support* of G is defined as

$$\mathrm{Supp}(G) := \{P \in \mathcal{X} \mid n_P \neq 0\}.$$

For a place $Q \in \mathcal{X}$ and a divisor $G = \sum n_P P$, we define $v_Q(D) := n_Q$. If for all place $P \in \mathrm{Supp}(G)$, we have $v_P(G) > 0$ then G is called an *effective* divisor. Finally we define the *degree* of G as

$$\deg(G) := \sum_{P \in \mathcal{X}} v_P(G) \deg(P).$$

The group $\mathrm{Div}(\mathcal{X})$ has a partial order defined as

$$\sum_{P \in \mathcal{X}} n_P P \geq \sum_{P \in \mathcal{X}} m_P P \iff \forall P \in \mathcal{X},\ n_P \geq m_P.$$

In particular, if $G \in \mathrm{Div}(\mathcal{X})$ is an effective divisor we note $G \geq 0$. Now consider a non-zero function $x \in \mathbb{F}_q(\mathcal{X})^*$, we assign to x the divisor

$$(x) := \sum_{P \in \mathcal{X}} v_P(x) P.$$

This definition is consistent since, by Proposition 1.12, x has a finite number of zeros and poles on $\mathcal{X}$. Such divisors are called *principal divisors* and form a subgroup of $\mathrm{Div}(\mathcal{X})$, denoted by $\mathrm{Princ}(\mathcal{X})$, since for $x, y \in \mathbb{F}_q(X)^*$ we have $(xy) = (x) + (y)$.

Example 5. Let $f \in \mathbb{F}_q[x]$ be a polynomial function and $Z := \{x_1, \ldots, x_s\}$ be the set of zero of f. We denote m_i the multiplicity of the zero x_i for all $i \in \{1, \ldots, s\}$. Then the divisor of f on the projective line is

$$(f) := \sum_{i=0}^{s} m_i P_i + \deg(f) P_\infty,$$

where $P_i := (x_i : 1)$ and P_∞ is the unique pole of x on the projective line.

Definition 1.27. Two divisors $G_1, G_2 \in \mathrm{Div}(\mathcal{X})$, are said to be *linearly equivalent* if $G_1 - G_2 \in \mathrm{Princ}(\mathcal{X})$, that is to say there exists a function $x \in \mathbb{F}_q(\mathcal{X})^*$ such that $G_1 = G_2 + (x)$. Then we write $G_1 \sim G_2$. This is an equivalence relation and the *class group of* $\mathcal{X}$ is

$$\mathrm{Cl}(\mathcal{X}) := \mathrm{Div}(\mathcal{X}) / \mathrm{Princ}(\mathcal{X}).$$

Proposition 1.14. *The degree of a principal divisor is zero.*

Let us denote by $\mathrm{Div}^0(\mathcal{X})$ the group of divisors of $\mathcal{X}$ of degree 0. By the previous proposition $\mathrm{Princ}(\mathcal{X}) \subseteq \mathrm{Div}^0(\mathcal{X})$ and then we define the group of divisor classes of degree 0 as $\mathrm{Cl}^0(\mathcal{X}) := \mathrm{Div}^0(\mathcal{X}) / \mathrm{Princ}(\mathcal{X})$. Moreover, Proposition 1.14 implies that all divisors in a same equivalent class have the same degree.

Proposition 1.15 ([Mor93, Chap. 3])**.** *For any integer r, the number of divisor classes in $\mathrm{Cl}(\mathcal{X})$ of degree r is independent of r and is equal to the cardinality of $\mathrm{Cl}^0(\mathcal{X})$.*

For the following result, we need to know what is the *genus* of a curve. We choose to give this definition later when it will be possible (Definition 1.31). For now, we admit that the genus is an invariant of the curve $\mathcal{X}$ and it is a non negative integer.

Theorem 1.16 ([TVN07, Proposition 3.1.23])**.** *Let $\mathcal{X}$ be a projective curve defined over $\mathbb{F}_q$. The number of divisor classes in $\mathrm{Cl}^0(\mathcal{X})$, called the* class number *and denoted by $h(\mathcal{X})$, satisfies:*

$$(\sqrt{q} - 1)^{2g} \leq h(\mathcal{X}) \leq (\sqrt{q} + 1)^{2g},$$

where g is the genus of $\mathcal{X}$.

The following consequence about the specific case of divisors on the projective line will be useful in Section 5.

Corollary 1.17. *Let $\mathbf{P}^1$ be the projective line and $G_1, G_2 \in \mathrm{Div}(\mathbf{P}^1)$, such that $\deg(G_1) = \deg(G_2)$. Then $G_1 \sim G_2$.*

The following definition will be useful to define algebraic geometry codes on a projective curve $\mathcal{X}$. Algebraic geometry (AG) codes are evaluation codes on a curve $\mathcal{X}$, the vector space which is evaluated is the following.

Definition 1.28. Let G be a divisor on $\mathcal{X}/\mathbb{F}_q$, we define $L(G)$ the *Riemann-Roch space* of G by:

$$L(G) := \{f \in \mathbb{F}(\mathcal{X}) \mid (f) \geq -G\} \cup \{0\}.$$

$L(G)$ is a vector space over $\mathbb{F}_q$ and its dimension over $\mathbb{F}_q$ is denoted by $\ell(G) := \dim_{\mathbb{F}_q} L(G)$.

The computation of the dimension $\ell(G)$ of a Riemann-Roch space $L(G)$ is very important to know the dimension of an AG code. For now, we cannot estimate exactly this dimension but we provide the following bound.

Lemma 1.18. *For a projective curve $\mathcal{X}$ and a divisor $G \in \mathrm{Div}(\mathcal{X})$, we have*

$$\ell(G) \leq \max\{0, \deg(G) + 1\}.$$

In particular, if $\deg(G) < 0$ *then* $\ell(G) = 0$*, that is* $L(G) = \{0\}$*.*

Example 6. Consider the point at infinity P_∞ on the projective line over $\mathbb{F}_q$, and let $k > 0$ be an integer. Then the Riemann-Roch space $L(kP_\infty) =< 1, x, \ldots, x^k >$ has dimension $k + 1$ over $\mathbb{F}_q$.

Lemma 1.19. *Let $G_1, G_2 \in \mathrm{Div}(\mathcal{X})$ be two linearly equivalent divisors. By definition, there exists $h \in \mathbb{F}_q(\mathcal{X})$ such that $G_1 = G_2 + (h)$. Now, consider the following $\mathbb{F}_q$-linear map*

$$\begin{array}{ccc} L(G_1) & \longrightarrow & L(G_2) \\ f & \longmapsto & hf \end{array},$$

it defines an isomorphism between $L(G_1)$ and $L(G_2)$. Then for a divisor G the value of $\ell(G)$ depends only on its linear equivalence class.

1.2.4 Riemann-Roch Theorem

In this section we introduce some definitions and results in order to present the Riemann-Roch theorem. This theorem gives an explicit expression of the dimension $\ell(G)$ for any divisor G. Before presenting this result we need to give the definition of differential forms on a curve and some basic properties of these objects. In the following, $\mathcal{X}$ denotes a smooth projective plane curve over $\mathbb{F}_q$. For this part we refer to [HvLP98].

A *derivation over* $\mathbb{F}_q(\mathcal{X})$ is an $\mathbb{F}_q$-linear map $D : \mathbb{F}_q(\mathcal{X}) \to \mathbb{F}_q(\mathcal{X})$, satisfying the Leibnitz rule $D(xy) = xD(y) + yD(x)$ for any function $x, y \in \mathbb{F}_q(\mathcal{X})$. The set of derivations, that we denote by $Der(\mathbb{F}_q(\mathcal{X}))$, is a vector space over the field $\mathbb{F}_q(\mathcal{X})$.

Definition 1.29. A *differential form*, or *differential*, on $\mathcal{X}$ is an $\mathbb{F}_q(\mathcal{X})$-linear map from $Der(\mathbb{F}_q(\mathcal{X}))$ to $\mathbb{F}_q(\mathcal{X})$. The set of all differentials on $\mathcal{X}$ is denoted by $\Omega(\mathcal{X})$.

In the following we consider the map

$$d : \left\{ \begin{array}{ccc} \mathbb{F}_q(\mathcal{X}) & \longrightarrow & \Omega(\mathcal{X}) \\ f & \longmapsto & df \end{array} \right.$$

which associate to any function f a differential $df : Der(\mathcal{X}) \to \mathbb{F}_q(\mathcal{X})$ defined by $df(D) = D(f)$ for all $D \in Der(\mathcal{X})$.

Theorem 1.20. *The dimension of $\Omega(\mathcal{X})$ over $\mathbb{F}_q(\mathcal{X})$ is 1. Moreover, for any point P and any local parameter t_P at P, the differential dt_P is a basis of $\Omega(\mathcal{X})$ over $\mathbb{F}_q(\mathcal{X})$.*

The previous theorem can be reformulated as follows. For every point $P \in \mathcal{X}$ and local parameter t_P, a differential $\omega \in \Omega(\mathcal{X})$ can be represented in a unique way as $\omega = f dt_P$, where $f \in \mathbb{F}_q(\mathcal{X})$. We **cannot define** the "value" $\omega(P)$ of ω in P since it depends on the choice of the local parameters t_P. However, it is possible to define zeros and poles of a differential.

Definition 1.30. Let $\omega \in \Omega(\mathcal{X})$ be a differential and $P \in \mathcal{X}$ a point. We denote t_P a local parameter of P and we write $\omega = f dt_P$, with $f \in \mathbb{F}_q(\mathcal{X})$. We say that P is *a zero of* ω (resp. *a pole of* ω) if P is a zero of f (resp. a pole of f). Moreover we define the *valuation* of ω in P as $v_P(\omega) := v_P(f)$.

As in the case of valuations of rational functions, this definition does not depend on the choice of the local parameter t_P. Similarly to the case of principal divisors we can define the divisor of a differential $\omega \in \Omega(\mathcal{X}) \setminus \{0\}$ as

$$(\omega) := \sum_{P \in \mathcal{X}} v_P(\omega) P.$$

Such a divisor is called a *canonical* divisor. Let $f \in \mathbb{F}_q(\mathcal{X})$ be a function and $\omega \in \Omega(\mathcal{X})$ a differential, then $(f\omega) = (f) + (\omega)$. By Theorem 1.20, for any $\omega \in \mathbb{F}_q(\mathcal{X})$ we can write $\omega = f\omega'$, with $f \in \mathbb{F}_q(\mathcal{X})$ and $\omega' \in \Omega(\mathcal{X}) \setminus \{0\}$. Then we have the following proposition.

Proposition 1.21. *Canonical divisors are linearly equivalent.*

Definition 1.31. Let $\mathcal{X}$ be a projective curve, we define the *genus* of $\mathcal{X}$ by $g_{\mathcal{X}} := \ell(W)$, with W a canonical divisor on $\mathcal{X}$. If there is no ambiguity we only write g.

By Proposition 1.21 and Lemma 1.19, the genus does not depend on the choice of the canonical divisor W. The genus of a curve $\mathcal{X}$ is crucial in the Riemann-Roch Theorem. This number is not always easy to compute. Let us give a simple formula in the case where $\mathcal{X}$ is a smooth plane curve, which is the case of interest in this thesis.

Proposition 1.22 (Plücker formula, [TVN07, Corollary 2.2.8])**.** *Let $\mathcal{X}$ be a smooth projective plane curve of degree r, then the genus is given by*

$$g = \frac{(r-1)(r-2)}{2}.$$

Theorem 1.23 (Riemann-Roch Theorem)**.** *Let $\mathcal{X}$ be a smooth projective curve and W a canonical divisor on this curve. For any divisor $G \in \mathrm{Div}(\mathcal{X})$, we have*

$$\ell(G) = \deg(G) + 1 - g + \ell(W - G),$$

with g the genus of the curve $\mathcal{X}$.

We can deduce from the Riemann-Roch theorem that for a canonical divisor W, we have $\deg(W) = 2g - 2$. A particular case of this theorem will be important for properties of AG code, it is the following.

Corollary 1.24. *If $\deg(G) > 2g - 2$, then $\ell(G) = \deg(G) + 1 - g$.*

1.3 AG codes: definition and properties

In this section, we define the algebraic geometry (AG) codes and their subfield subcode (SSAG). We present two constructions for AG codes which are dual each other. If there is no mention, details about this section can be found in [TVN07] or [Sti09].

1.3.1 Algebraic geometry codes

In the following, $\mathcal{X}$ will denote a smooth plane projective curve over $\mathbb{F}_{q^m}$. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n distinct rational points of $\mathcal{X}$, ie: places of degree 1 of $\mathbb{F}_{q^m}(\mathcal{X})$. Let G be a divisor such that $\deg(G) < n$, and the support of G does not contain any place of $\mathcal{P}$. We consider the following map:

$$\begin{array}{rccc} \mathrm{Ev}_{\mathcal{P}} : & \mathcal{O}_{\mathcal{P}} & \longrightarrow & \mathbb{F}_{q^m}^n \\ & f & \longmapsto & (f(P_1), \ldots, f(P_n)). \end{array}$$

Here $\mathcal{O}_{\mathcal{P}}$ denotes the subset of $\mathbb{F}_{q^m}(\mathcal{X})$ consisting of functions regular at any $P \in \mathcal{P}$, i.e. $\mathcal{O}_{\mathcal{P}} = \bigcap_{P \in \mathcal{P}} \mathcal{O}_P$.

Definition 1.32. With the previous notations, the algebraic geometry (AG) code $C_L(\mathcal{X}, \mathcal{P}, G)$ is defined by:

$$C_L(\mathcal{X}, \mathcal{P}, G) := \{\mathrm{Ev}_{\mathcal{P}}(f) \mid f \in L(G)\}.$$

The set $\mathcal{P}$ will be called the *support* of the code $C_L(\mathcal{X}, \mathcal{P}, G)$.

Theorem 1.25. *The code $\mathscr{C} := C_L(\mathcal{X}, \mathcal{P}, G)$ is an $[n, k, d]$ code over $\mathbb{F}_{q^m}$ with:*

$$k = \ell(G) \geq \deg(G) - g + 1 \text{ and } d \geq n - \deg(G).$$

If in addition $\deg(G) > 2g - 2$ then $k = \deg(G) - g + 1$. The integer $\delta := n - \deg(G)$ is called the designed distance *of the code $\mathscr{C}$.*

Let $\{f_1, \ldots, f_k\}$ be a basis of $L(G)$, then the matrix

$$\mathbf{G} := \begin{pmatrix} f_1(P_1) & \ldots & f_1(P_n) \\ \vdots & & \vdots \\ f_k(P_1) & \ldots & f_k(P_n) \end{pmatrix} \tag{1.1}$$

is a generator matrix for the code $C_L(\mathcal{X}, \mathcal{P}, G)$.

1.3.2 Duality

Another kind of algebraic geometry codes can be constructed from differential forms. Consider a divisor G on $\mathcal{X}$ and the following space of differential forms

$$\Omega(G) := \{\omega \in \Omega(\mathcal{X}) \mid (\omega) \geq G\} \cup \{0\}.$$

The dimension of $\Omega(G)$ will be denoted $i(G)$ and called the *index of speciality of G*.

Theorem 1.26 ([Sti09, Theorem 1.5.14])**.** *Let g be a divisor on $\mathcal{X}$ and $W = (\omega)$ be a canonical divisor. Then the following map*

$$\begin{array}{ccc} L(W - G) & \longrightarrow & \Omega(G) \\ x & \longmapsto & x\omega \end{array}$$

is an isomorphism of $\mathbb{F}_{q^m}(\mathcal{X})$-vector spaces. Thus, the index of speciality of G satisfies

$$i(G) = \ell(G) - \deg(G) + g - 1$$

with g the genus of the curve $\mathcal{X}$.

Definition 1.33 (Residue)**.** Let $\omega \in \Omega(\mathcal{X})$ be a differential, P be a point on $\mathcal{X}$ and t_P a local parameter at P. Let $\omega = f dt_P$ for some $f \in \mathbb{F}_{q^m}(\mathcal{X})$. Expanding f into a Laurent power series in t_P, we have

$$f = \sum_{i=-M}^{\infty} a_i t_P^i,$$

with $M \in \mathbb{Z}$. The *residue of ω at P*, denoted $\mathrm{Res}_\omega(P)$, is the coefficient a_{-1}. It is independent of the choice of t_P.

Consider the set of rational points $\mathcal{P}$ defined as previously, we define the divisor $D_{\mathcal{P}} := \sum_{P \in \mathcal{P}} P$ associated to the set $\mathcal{P}$. Let G be a divisor on $\mathcal{X}$ with a support disjoint of $\mathcal{P}$ and such that $\deg(G) > 2g - 2$, we consider the following map

$$\mathrm{Res}_{\mathcal{P}} : \left\{ \begin{array}{ccc} \Omega(\mathcal{X}) & \longrightarrow & \mathbb{F}_{q^m} \\ \omega & \longmapsto & (\mathrm{Res}_\omega(P_1), \ldots, \mathrm{Res}_\omega(P_n)) \end{array} \right. .$$

Definition 1.34. With the previous notations, the Ω-AG code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$ is defined by:

$$C_\Omega(\mathcal{X}, \mathcal{P}, G) := \{\mathrm{Res}_\omega(\mathcal{P}) \mid \omega \in \Omega(G - D_\mathcal{P})\}.$$

Theorem 1.27. *The code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$ is an $[n, k, d]$ code over $\mathbb{F}_{q^m}$ with:*

$$k = i(G - D_\mathcal{P}) \geq n - \deg(G) + g - 1 \text{ and } d \geq \deg(G) - 2g + 2.$$

If in addition $\deg(G) < n$ then $k = n - \deg(G) + g - 1$. The integer $\delta_\Omega := \deg(G) - 2g + 2$ is called the designed distance of the code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$.

There is an important relation between the code $C_L(\mathcal{X}, \mathcal{P}, G)$ and the code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$.

Theorem 1.28. *The codes $C_L(\mathcal{X}, \mathcal{P}, G)$ and $C_\Omega(\mathcal{X}, \mathcal{P}, G)$ are dual to each other, i.e.*

$$C_L(\mathcal{X}, \mathcal{P}, G)^\perp = C_\Omega(\mathcal{X}, \mathcal{P}, G).$$

A proof of this result can be found in [Sti09] and comes from the following formula.

Proposition 1.29 (Residue formula)**.** *For any differential form $\omega \in \Omega(\mathcal{X})$, we have*

$$\sum_{P \in \mathcal{X}} \mathrm{Res}_\omega(P) = 0.$$

As a consequence of Theorem 1.28, the matrix defined in (1.1) is a parity check matrix for the code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$. The next result show that an Ω-AG code $C_\Omega(\mathcal{X}, \mathcal{P}, G)$ can be represented as a code $C_L(\mathcal{X}, \mathcal{P}, H)$, for some divisor H. This result will be very useful in Section 5.3 for the example of codes on cyclic cover of $\mathbf{P}^1$. Before stating it, we need an intermediary result which is the following.

Lemma 1.30. *Let $\mathcal{P} := \{P_1, \ldots, P_n\}$ be a set of distinct rational points of $\mathcal{X}$. There exists a differential $\omega \in \Omega(\mathcal{X})$ such that*

$$v_{P_i} = -1 \text{ and } \mathrm{Res}_\omega(P_i) = 1,$$

for all $i \in \{1, \ldots, n\}$.

Theorem 1.31. *Let $\mathcal{P} := \{P_1, \ldots, P_n\}$ be a set of distinct rational points of $\mathcal{X}$, and G be a divisor with support disjoint from $\mathcal{P}$. Let $\omega \in \Omega(\mathcal{X})$ be a differential such that $v_{P_i} = -1$ and $\mathrm{Res}_\omega(P_i) = 1$, for all $i \in \{1, \ldots, n\}$. We denote by $W := (\omega)$. Then*

$$C_\Omega(\mathcal{X}, \mathcal{P}, G) = C_L(\mathcal{X}, \mathcal{P}, W + D_\mathcal{P} - G).$$

1.3.3 Subfield subcodes of AG codes

We keep the notation of the previous section, in particular $\mathcal{X}$ is a smooth plane curve over $\mathbb{F}_{q^m}$.

Definition 1.35. With the notation of Definition 1.32, we define the *subfield subcode* of $C_L(\mathcal{X}, \mathcal{P}, G)$ over $\mathbb{F}_q$, denoted by $\mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$, as follows

$$\mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G) := C_L(\mathcal{X}, \mathcal{P}, G) \cap \mathbb{F}_q^n.$$

For these codes there is an obvious bound on the dimension which is

$$\dim_{\mathbb{F}_q}(\mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)) \leq \dim_{\mathbb{F}_{q^m}}(\mathrm{C_L}(\mathcal{X}, \mathcal{P}, G)).$$

This comes from the fact that a basis of $C_L(\mathcal{X}, \mathcal{P}, G) \cap \mathbb{F}_q^n$ over $\mathbb{F}_q$ is also linearly independent over $\mathbb{F}_{q^m}$. In order to introduce a lower bound on the dimension, we recall the following result.

Let us denotes $\mathrm{Tr}(\mathscr{C})$ the trace code, over $\mathbb{F}_q$, of an AG code $\mathscr{C}$ (see Definition 1.10), by Delsarte Theorem (Theorem 1.7) we have

$$\mathrm{SSAG}_q(\mathcal{X},\mathcal{P},G)^\perp = \mathrm{Tr}(\mathrm{C_L}(\mathcal{X},\mathcal{P},G)^\perp)$$

Moreover, we know that $\dim_{\mathbb{F}_q}(\mathrm{Tr}(\mathrm{C_L}(\mathcal{X},\mathcal{P},G)^\perp)) \leq m \dim_{\mathbb{F}_q}(\mathrm{C_L}(\mathcal{X},\mathcal{P},G)^\perp)$, where m denotes the extension degree of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. This permits to have the following bound

$$\dim_{\mathbb{F}_q}(\mathrm{SSAG}_q(\mathcal{X},\mathcal{P},G)) \geq n - m\dim_{\mathbb{F}_q}(\mathrm{C_L}(\mathcal{X},\mathcal{P},G)^\perp)$$

with n the length of the code. Moreover, since $\mathrm{SSAG}_q(\mathcal{X},\mathcal{P},G) \subseteq \mathrm{C_L}(\mathcal{X},\mathcal{P},G)$, we have the following bound for the minimum distance

$$d_{\min}(\mathrm{SSAG}_q(\mathcal{X},\mathcal{P},G)) \geq d_{\min}(\mathrm{C_L}(\mathcal{X},\mathcal{P},G)).$$

This lead to the following proposition.

Proposition 1.32. *Let $SSAG_q(\mathcal{X},\mathcal{P},G)$ be a subfield subcode as in Definition 1.35, then it is an $[n,k,d]$ code over $\mathbb{F}_q$, with*

$$k \geq n - m(n-\ell(G)) \text{ and } d \geq n - \deg(G).$$

The integer $\delta := n - \deg(G)$ is called the designed distance of the code $SSAG_q(\mathcal{X},\mathcal{P},G)$.

1.3.4 The particular case of AG codes on the projective line

In this section, we speak about AG codes, and their subfield subcodes, defined on the projective line. This case is known as *Generalized Reed Solomon*(GRS) code for AG codes and *alternant* codes for the subfield subcode of AG codes. First let us give some precisions about the projective line.

Let $\mathbf{P}^1$ be the projective line over the field $\mathbb{F}_{q^m}$, then its function field is $\mathbb{F}_{q^m}(\mathbf{P}^1) = \mathbb{F}_{q^m}(x)$ where x is transcendental over $\mathbb{F}_{q^m}$. This function space is called the *rational function field* over $\mathbb{F}_{q^m}$. The unique pole of the function x is called the *point at infinity* of $\mathbf{P}^1$, it is denoted P_∞. Other points of the projective line comes from the embedding $\mathbf{A}^1 \hookrightarrow \mathbf{P}^1$, that is there are points with homogeneous coordinates of the form $(\alpha : 1)$, with $\alpha \in \mathbb{F}_{q^m}$. We denote these points by P_α. Actually there are no other rational points of $\mathbf{P}^1$ than the points P_α, $\alpha \in \mathbb{F}_{q^m}$, and P_∞. That is we have the following proposition.

Proposition 1.33. *The rational points of $\mathbf{P}^1$ over $\mathbb{F}_{q^m}$ are in one-to-one correspondence with $\mathbb{F}_{q^m} \cup \infty$, where ∞ is a notation for P_∞.*

The generalized Reed Solomon codes. Now, let $\mathcal{P} \subseteq \mathbf{P}^1$ be a set of distinct rational points and G be a divisor on $\mathbf{P}^1$ with support disjoint of $\mathcal{P}$. Then, we can define an AG code $\mathscr{C}$ over $\mathbf{P}^1$ as in Definition 1.32:

$$\mathscr{C} := \mathrm{C_L}(\mathbf{P}^1,\mathcal{P},G).$$

We call these codes *rational AG codes*. In the following we assume that $P_i = (x_i : 1)$ for all $i \in \{1,\dots,n\}$. In order to make a correspondence with the classical definition of GRS codes, we construct a generator matrix of $\mathscr{C}$. By Proposition 1.17, we know that $G \sim \deg(G)P_\infty$, that is $G + (h) = \deg(G)P_\infty$, for some $h \in \mathbb{F}_{q^m}(x)$. Moreover, for any $s \in \mathbb{N}$, we have

$$L(sP_\infty) = \langle x^i \mid i \in 0,\dots,s\rangle,$$

then we can write

$$L(G) = \langle hx^i \mid i \in 0,\dots,s\rangle.$$

Let us denote by $k = \deg(G)+1$, $\mathbf{x} := (x(P_1), \ldots, x(P_n)) = (x_1, \ldots, x_n)$ and $\mathbf{y} := (h(P_1), \ldots, h(P_n))$, then the matrix

$$\mathbf{V}_r(\mathbf{x}, \mathbf{y}) := \begin{pmatrix} y_1 & \ldots & y_n \\ y_1x_1 & \ldots & y_nx_n \\ y_1x_1^2 & \ldots & y_nx_n^2 \\ \vdots & & \vdots \\ y_1x_1^{k-1} & \ldots & y_nx_n^{k-1} \end{pmatrix} \tag{1.2}$$

is a generator matrix of the code $\mathscr{C}$.

Definition 1.36 (Generalized Reed Solomon code)**.** Let $\mathbf{x} := (x_1, \ldots, x_n) \in \mathbb{F}_{q^m}^n$ be an n-tuple of distinct elements, $\mathbf{y} := (y_1, \ldots, y_n) \in (\mathbb{F}_{q^m}^*)^n$ be an n-tuple of non-zero elements, and k be a positive integer. The *Generalized Reed Solomon* (GRS) code of dimension k, is defined as follows

$$\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) := \{ \ (y_1f(x_1), \ldots, y_nf(x_n)) \mid f \in \mathbb{F}_{q^m}[z]_{<k} \} .$$

The vector $\mathbf{x}$ is called the *support* and $\mathbf{y}$ a *multiplier* of the code $\mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$. The matrix (1.2) is also a generator matrix for this code and it is an $[n, k]$ code. This leads that any rational AG code is a GRS code.

Actually the converse is also true. Let $\mathscr{C} := \mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$ be a GRS code as defined just above. Consider the function field $\mathbb{F}_{q^m}(x)$, denote by P_i the zeros of $x - x_i$ for $i \in \{1, \ldots, n\}$, and keep P_∞ for the pole of x. We set $\mathcal{P} := (P_1, \ldots, P_n)$. Now, we can choose $h \in \mathbb{F}_{q^m}[x]$ such that

$$h(P_i) = y_i \text{ for } i \in \{1, \ldots, n\} \text{ and } \deg(h) < n.$$

This is possible by the Lagrange interpolation method. We set $G := (k-1)P_\infty + (h)$ and then we have

$$\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) = \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G).$$

By Theorems 1.28 and 1.31, we know that the dual of an AG code is also an AG code. In the case of GRS codes, that is rational AG codes, this result can be reformulated as follows.

Notation 1. Let $\mathbf{x} \subseteq \mathbb{F}_{q^m}^n$ be a vector with distinct entries, we define the *locator polynomial* $\pi_{\mathbf{x}} \in \mathbb{F}_{q^m}[x]$ as

$$\pi_{\mathbf{x}}(x) := \prod_{i=0}^{n-1} (x - x_i).$$

Proposition 1.34. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^m}^n$ be a support and a multiplier of length n and $k \leq n$. Then*

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

where

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

and $\pi'_{\mathbf{x}}$ denotes the derivative of the polynomial $\pi_{\mathbf{x}}$.

The alternant codes. Finally let us give the definition of alternant codes.

Definition 1.37 (Alternant code)**.** Let $\mathbf{x} \in \mathbb{F}_{q^m}^n$ be a support and $\mathbf{y} \in (\mathbb{F}_{q^m}^*)^n$ be a multiplier, then the *alternant* code over $\mathbb{F}_q$ is defined as follows

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

The integer r is referred as the *degree* of the code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$.

By Proposition 1.34, the alternant codes are subfield subcode of GRS codes, and then are subfield subcode of rational AG codes. In the following, we use the notation

$$\mathscr{A}_{r,q}(\mathcal{P}, G) := \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^{\perp} \cap \mathbb{F}_q^n$$

when we want to use the representation of an alternant code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ as an SSAG code.

Proposition 1.35. *The code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$, of length n, has a dimension*

$$k \geq n - mr.$$

We conclude this subsection on alternant codes by a definition which is useful in Chapter 4.

Definition 1.38. An alternant code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ is said to be *fully non degenerate* if it satisfies the two following conditions.

(i) A generator matrix of $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ has no zero column.

(ii) $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}) \neq \mathscr{A}_{r+1,q}(\mathbf{x}, \mathbf{y})$.

Most of the time, an alternant code is fully non degenerate.

The Goppa codes. A special case of alternant codes is often used, it is the Goppa codes. Later on, the notation $P(\mathbf{x})$ for any polynomial P and any vector $\mathbf{x}$ means that we apply the polynomial P on each entry of $\mathbf{x}$.

Definition 1.39 (Classical Goppa codes)**.** Let $\mathbf{x} \in \mathbb{F}_{q^m}^n$ be a vector with pairwise distinct entries and $\Gamma \in \mathbb{F}_{q^m}[z]$ be a polynomial such that $\Gamma(x_i) \neq 0$ for all $i \in \{0, \ldots, n-1\}$. The Goppa code $\mathscr{G}_q(\mathbf{x}, \Gamma)$ associated to Γ and supported by $\mathbf{x}$ is defined as

$$\mathscr{G}_q(\mathbf{x}, \Gamma) := \mathscr{A}_{r,q}(\mathbf{x}, \Gamma(\mathbf{x})^{-1}),$$

with $r := \deg \Gamma$. We call Γ the *Goppa polynomial* and m the *extension degree* of the Goppa code.

1.3.5 Decoding AG codes

In order to use AG codes in a cryptographic context we need to know efficient decoding algorithms for these codes. Several algorithms are known, the survey of Høhold and Pellikaan [HP95] describes some of them. For arbitrary AG codes $C_L(\mathcal{X}, \mathcal{P}, G)$, there exists an algorithm, called the *basic algorithm* [SV90, JLJ+89], which corrects up to $\lfloor \frac{\delta - g - 1}{2} \rfloor$ errors, where δ is the designed distance of the code and g is the genus of the curve $\mathcal{X}$. This algorithm works in $O(n^3)$ operations in the field $\mathbb{F}_{q^m}$, with n the length of the code. In [SV90] the authors presented another algorithm, the *modified algorithm*, which is an improvement of the basic algorithm. For AG codes on plane curves, the modified algorithm can correct up to $\lfloor \frac{\delta-1}{2} - \frac{g}{4} \rfloor$ errors in $O(n^3)$ operations in the field $\mathbb{F}_{q^m}$. The basic and modified algorithms do not correct up to the designed distance, this problem can be avoided by using another algorithm more specific.

In the case of "one-point" codes, i.e. codes constructed from a divisor with one point, on regular planes curves, Feng and Rao proposed, in [FR93] an algorithm which corrects $\lfloor \frac{\delta_{FR}-1}{2} \rfloor$ errors in $O(n^3)$ operations on the base field, and where δ_{FR} is the *Feng-Rao distance* which verifies $\delta_{FR} \geq \delta$. Sakata et al. [SJM+95] improve the complexity of the previous algorithm and gave a decoding algorithm which correct up to $\lfloor \frac{\delta-1}{2} \rfloor$ errors in complexity $O(n^{\frac{7}{3}})$ operations in the base field.

The previous case can be too restrictive for some cryptographic applications. There are other algorithms for specific curves and any divisors. For instance, in [Pel89] Pellikaan gives an algorithm for codes on maximal curves which can correct $\lfloor \frac{\delta-1}{2} \rfloor$ errors in $O(n^4)$. This case is useful in Section 5 where we use codes on Hermitian curve which are maximal curves.

In the case of codes over the projective line, that is GRS and alternant codes, there exists also a specific decoding algorithm: the Berlekamp Welch algorithm (see [Bla08, Chap.3,§12] for details). This algorithm decodes up to $\frac{d-1}{2}$ errors in $O(n^3)$ operation in the base field, with d the minimum distance and n the length of the code. Actually there exists a more efficient algorithm with use Extended Euclid algorithm and works in $O(n^2)$ (see [MS86] for more details).

Chapter 2

Code-based cryptography

2.1 McEliece encryption scheme

The encryption scheme of McEliece, presented in [McE78], is the first cryptosystem based on error correcting codes. The idea of the McEliece cryptosystem is to use a family $\mathcal{F}$ of structured codes for which we know an efficient decoding algorithm. Then the public key will be a random looking generator matrix of a code in $\mathcal{F}$ and the trapdoor will be the decoding algorithm for the chosen code. More precisely the McEliece scheme can presented as follows.

The McEliece scheme

- **Key generation :**
  Input: The parameters $n, k, t \in \mathbb{N}$ and a finite field $\mathbb{F}_q$.
  1. Choose a family $\mathcal{F}$ of linear codes over $\mathbb{F}_q$ with an efficient decoding algorithm $\mathcal{D}$.
  2. Pick any $[n, k]$ code $\mathscr{C} \in \mathcal{F}$ over $\mathbb{F}_q$, correcting t errors. Let $\mathbf{G}$ be a random looking generator matrix of $\mathscr{C}$ and $\mathcal{D}_{\mathscr{C}}$ be a decoding algorithm associated to $\mathscr{C}$ and correcting t errors.

  Output : The **public key** $(\mathbf{G}, t)$ and the **private key** $\mathcal{D}_{\mathscr{C}}$.
- **Encryption :**
  Input: A message $\mathbf{m} \in \mathbb{F}_q^n$.
  1. Pick a random vector $\mathbf{e} \in \mathbb{F}_q^n$ of weight t.
  2. Compute $\mathbf{y} := \mathbf{mG} + \mathbf{e}$.

  Output : The cypher text $\mathbf{y}$.
- **Decryption :**
  Input: A cypher text $\mathbf{y} = \mathbf{c} + \mathbf{e}$, with $\mathbf{c} \in \mathscr{C}$ and $w_H(\mathbf{e}) \leq t$.
  1. Compute $\mathbf{c} = \mathcal{D}_{\mathscr{C}}(\mathbf{y})$.
  2. Deduce $\mathbf{m}$, from the knowledge of $\mathbf{c} = \mathbf{mG}$ and $\mathbf{G}$, by Gaussian elimination.

  Output : The plaintext $\mathbf{m}$.

Before speaking about the security, we precise what does performing an attack against the McEliece scheme mean. We define two kinds of attack against this scheme: *message recovery attacks* and *key recovery attacks*. Message recovery attacks consist in recovering the plaintext $\mathbf{m}$ from the knowledge of the cypher text $\mathbf{y}$ and the public key $(\mathbf{G}, t)$. Once the attack is performed we know the message $\mathbf{m}$ but we do not know the private key $\mathcal{D}_{\mathscr{C}}$. If we want to recover several

messages, the cost will be the cost of the attack on each cypher text. The second kind of attacks, that is the key recovery attacks, consists in recovering an efficient decoding algorithm $\mathcal{D}_{\mathscr{C}}$ for the code $\mathscr{C}$ from the knowledge of a generator matrix $\mathbf{G}$. When we speak about an efficient decoding algorithm we mean that this algorithm works in polynomial time in the parameters n and k of the code. Once the attack is performed, the cost of recovering any message is the cost of the decoding algorithm $\mathcal{D}_{\mathscr{C}}$.

In both cases, we say that there exists an attack if there exists an algorithm, which recover the message or the key, in polynomial time in n and k or which can be performed in less than 2^{80} binary operations.

The security of the scheme depends on the choice of the family $\mathcal{F}$ and on the hardness of the *syndrome decoding problem* (see Section 2.2, Problem 1). The syndrome decoding (SD) problem is independent from the choice of $\mathcal{F}$ and is related to the security of the message. That is to say, the problem to recover the message $\mathbf{m}$ from the cypher text $\mathbf{y}$ and the public key $\mathbf{G}$ reduces to solve an instance of the SD problem. We will see in details this problem in the following section. The choice of the family $\mathcal{F}$ influences the security of the private key. Indeed, for some family $\mathcal{F}$ of linear codes there exists an algorithm, working in polynomial time in parameters n and k, which permits to recover a decoder $\mathcal{D}_{\mathscr{C}}$ from a generator matrix $\mathbf{G}$ of a code $\mathscr{C}$. We speak more precisely of the choice of $\mathcal{F}$ in Section 2.3.

2.2 Information Set Decoding (ISD)

The difficult problem to which refers most of the time code-based cryptosystems is the *syndrome decoding problem*. It is the case for the McEliece cryptosystem. This problem can be defined as follows.

Problem 1 ((Search) Syndrome Decoding Problem)**.** *Let* $\mathbf{H}$ *be a parity check matrix of a random* $[n, k]$ *code* $\mathscr{C}$ *over* $\mathbb{F}_q$*. Let* t *be an integer and* $\mathbf{s} \in \mathbb{F}_q^{n-k}$ *be a uniformly random vector, then the Syndrome Decoding Problem is to find a vector* $\mathbf{e} \in \mathbb{F}^n$ *of weight* $w_H(\mathbf{e}) \leq t$ *such that* $\mathbf{H}\mathbf{e}^\top = \mathbf{s}$*.*

The decisional version of this problem was proved NP-complete by Berlekamp, McEliece and van Tilbørd in [BMvT78]. Moreover, this problem is directly related to the decoding problem for a random linear code. Consider a random linear code $\mathscr{C}$ over $\mathbb{F}_q$, a parity check matrix $\mathbf{H}$ for this code, and denote by t the correction capability of $\mathscr{C}$. Let $\mathbf{y} := \mathbf{c} + \mathbf{e}$ be a vector of $\mathbb{F}_q^n$, with $\mathbf{c} \in \mathscr{C}$ and $\mathbf{e} \in \mathbb{F}_q^n$ an error vector with $w_H(e) \leq t$. We want to decode the vector $\mathbf{y}$, that is to recover $\mathbf{c} \in \mathscr{C}$, without knowing a specific decoding algorithm. From the knowledge of the matrix $\mathbf{H}$ we can compute the vector $\mathbf{s} := \mathbf{H}\mathbf{y}^\top$. The vector $\mathbf{s}$ is called a *syndrome*. This syndrome depends only on the error vector $\mathbf{e}$. Indeed, we have

$$\mathbf{s} := \mathbf{H}\mathbf{y}^\top = \underbrace{\mathbf{H}\mathbf{c}^\top}_{=0} + \mathbf{H}\mathbf{e}^\top = \mathbf{H}\mathbf{e}^\top.$$

If we are able to solve the SD problem for the parameters $\mathbf{H}$, $\mathbf{s}$ and t, that is to recover $\mathbf{e}$, then we are able to decode $\mathbf{c} = \mathbf{y} - \mathbf{e}$.

In order to choose parameters for the McEliece scheme we need to have an estimation of the cost of solving the SD problem. To solve the SD problem, the naive approach consists in performing a brute force search on errors vectors with weight t. The method does not provide a good approximation of the real cost of an attack on the message. Indeed, there exists more efficient algorithm to solve this problem. Of course the complexity of these algorithms is not polynomial in the parameters. Except the brute force approach, all algorithms to solve the SD problem are ameliorations of the attack by *information set decoding* (ISD), introduced by Prange in [Pra62]. This idea is to find a set of positions of the noisy vector which are without errors and such that the sub-matrix of the generator matrix associated to these positions is invertible.

Improvements of this first algorithm exist [Ste88, Dum91, MMT11, BJMM12]. We do not details here any algorithms, some of these methods are well explained in the paper of Peters [Pet10]. Moreover Peters provides a software to compute the best work factor of an ISD attack. To find parameters resistant to ISD algorithms, we use an improve version of this software proposed in [CT16a], called **CaWoF** (for **Ca**culate **Wo**rk **F**actor), which tests all the most efficient variants of the ISD algorithm ([Ste88, Dum91, MMT11, BJMM12]).

2.3 Key recovery problem

The complexity of a key recovery attack is more complicated to provide in the general case since it depends on the choice of the family $\mathcal{F}$. As we saw previously, a key recovery attack consists in recovering a decoding algorithm for a code $\mathscr{C}$ knowing only a generator matrix of $\mathscr{C}$. Actually the decoder recovery problem for a given family of linear codes reduces to an other problem which is what follows. Consider a family $\mathcal{F}$ of linear codes over $\mathbb{F}_q$ and $\mathbf{G}$ a $n \times k$ matrix over $\mathbb{F}_q$. The problem consists in finding the structure of the code $\mathscr{C} \in \mathcal{F}$ defined by $\mathbf{G}$, that is the secret elements which permit to construct an efficient decoding algorithm for the code $\mathscr{C}$. Depending on the choice of $\mathcal{F}$, this problem can be easy to solve. For instance if the chosen family is not large enough, a brute force attack can solve the decoder recovery problem in less than 2^{80} binary operations. The historical choice made by McEliece in [McE78] was the family of classical Goppa codes (see Definition 1.39) with parameters $[1024, 524, 101]$. For now, we do not know a efficient algorithm to solve the decoder recovery problem for the family of Goppa codes. That is, for cryptographic parameters, from a generator matrix of a Goppa code we are not able to recover the support and the Goppa polynomial defining the code. That is why we assume that this family is secure for the McEliece scheme and in code-based cryptography in general. The main problem is that the size of keys is large. Since 1978, several families have been proposed for the McEliece scheme, in order to reduce the key size. However, for some proposals the code recovery problem is not so hard. It is the case for Generalized Reed Solomon (GRS) codes (Definition 1.36), since Sildelnikov and Shestakov provide, in [SS92], an algorithm which recovers in polynomial time in the parameters of the code the GRS code from a given generator matrix. For AG codes on a curve with genus ≥ 1, there exists two kind of attack. In [FM08], Faure and Minder presented an attack against AG codes over curve of genus ≤ 2. They can recover the AG code chosen from its generator matrix. This attack cannot be extend to curve with high genus due to the complexity which is exponential in the genus. In 2014, Couvreur, Márquez-Corbella and Pellikaan presented an attack against AG codes over curve with higher genus [CMCP14]. They do not solve the code recovery problem but solve directly the decoder recovery problem in building a decoder for the chosen AG code from only a generator matrix.

In this thesis, the principal interest for us is the case of alternant codes and subfield subcode of AG (SSAG) codes. For the key security of these two cases the following fact is important.

Fact 1. Let $\mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$ be an AG code and assume that the curve $\mathcal{X}$ is known. It is possible to have a polynomial time decoding algorithm, from the knowledge of the support $\mathcal{P}$ and the divisor G.

That is to say, the security of a SSAG code depends on the knowledge of its support an divisor. In this thesis, when we speak about the *secret elements* of an SSAG code we mean the support and the divisor of this code. In the particular case of alternant codes this also means the support as a vector and the multiplier.

2.4 Algebraic cryptanalysis of McEliece schemes using alternant codes

In this section, we present a general framework of attack, called *algebraic attack*, against the private key of a McEliece scheme using rational SSAG codes, that is alternant codes. In 2010, Faugère, Otmani, Perret and Tillich proposed in [FOPT10] a new approach to study the key security of McEliece schemes using alternant codes. They prove that the secret elements, that is the support and the multiplier of the private key, satisfy a system of polynomial equations. This approach is also well described in the Portzamparc's thesis [dP15]. We explain here, how this polynomial system is built.

Let $\mathscr{C} := \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ be an alternant code over $\mathbb{F}_q$, with a support $\mathbf{x}$ and a multiplier $\mathbf{y}$ of length n defined over $\mathbb{F}_{q^m}$. From the knowledge of $\mathbf{x}$ and $\mathbf{y}$ it is possible to decode the code $\mathscr{C}$. That is the security of the private key reduces to the knowledge of the pair $(\mathbf{x}, \mathbf{y})$. Let us explain how to build a polynomial system whose solutions are $\mathbf{x}$ and $\mathbf{y}$.

As we saw in Section 1.3.4, alternant codes are defined as subfield subcode of the dual of a GRS code, that is $\mathscr{C} = \mathbf{GRS}_r(\mathbf{x}, \mathbf{y})^\perp \cap \mathbb{F}_q^n$. Moreover the following matrix

$$\mathbf{V}_r(\mathbf{x}, \mathbf{y}) = \begin{pmatrix} y_1 & \dots & y_n \\ y_1 x_1 & \dots & y_n x_n \\ \vdots & & \vdots \\ y_1 x_1^{r-1} & \dots & y_n x_n^{r-1} \end{pmatrix}$$

is a generator matrix for the code $\mathbf{GRS}_r(\mathbf{x}, \mathbf{y})$. Then, for any codeword $\mathbf{c} \in \mathscr{C}$, we have

$$\mathbf{V}_r(\mathbf{x}, \mathbf{y}) \cdot \mathbf{c}^\top = 0.$$

We say that $\mathbf{V}_r(\mathbf{x}, \mathbf{y})$ is a parity check matrix for the code $\mathscr{C}$, but we warn the reader that the entries of $\mathbf{V}_r(\mathbf{x}, \mathbf{y})$ are in $\mathbb{F}_{q^m}$ while the code $\mathscr{C}$ is defined over $\mathbb{F}_q$. Let $\mathbf{G}$ be a generator matrix of the code $\mathscr{C}$, then $\mathbf{V}_r(\mathbf{x}, \mathbf{y})\mathbf{G}^\top = 0$. Let us introduce $2n$ formal variables $\mathbf{X} := (X_1, \dots, X_n)$ and $\mathbf{Y} := Y_1, \dots, Y_n$ corresponding to the support $\mathbf{x}$ and the multiplier $\mathbf{y}$. To find the vectors $\mathbf{x}$ and $\mathbf{y}$ we need to solve the system $\mathbf{V}_r(\mathbf{X}, \mathbf{Y})\mathbf{G}^\top = 0$, that is:

$$\begin{pmatrix} Y_1 & \dots & Y_n \\ Y_1 X_1 & \dots & Y_n X_n \\ \vdots & & \vdots \\ Y_1 X_1^{r-1} & \dots & Y_n X_n^{r-1} \end{pmatrix} \mathbf{G}^\top = 0.$$

This system is bi-homogeneous in the variables $\mathbf{X}$ and $\mathbf{Y}$. That is to say, any equation of this system is a bi-homogeneous polynomial $f \in \mathbb{F}_{q^m}[X_1, \dots, X_n, Y_1, \dots, Y_n]$ of bi-degree $(s, 1)$, for some $s \in \{0, \dots, r-1\}$, i.e. $f(\alpha\mathbf{X}, \beta\mathbf{Y}) = \alpha^s \beta f(\mathbf{X}, \mathbf{Y})$ for any $(\alpha, \beta) \in \mathbb{F}_{q^m}^2$. It is easy to notice that the system becomes linear if we fix the variables $\mathbf{X}$. In the case where we fix the variables $\mathbf{Y}$ the system is not directly linear. However, the fact that the system is defined over the finite field $\mathbb{F}_{q^m}$ permits us to extract a linear system by considering the equations of degree $s = p^u$ in $\mathbf{X}$, with $u \in \{0, \dots, \log_p(r-1)\}$.

Solving the system. To solve polynomial systems over finite fields, a method consists to use *Gröbner bases*, for more details about this method see for instance [Bar04]. The complexity analysis of solve a polynomial system is based, in general, on this method. However, to provide a complexity analysis of algebraic attacks against alternant codes is more complicated. The complexity is difficult to estimate for many reasons:

- The choice of the algebraic modelling, i.e. the polynomial system we have to solve by Gröbner bases methods is not unique. We will suggest here some modelling which have been proposed in the literature but cannot assert that they are the only possible modellings;

- The choice of the monomial ordering has no influence on the theoretical complexity in the worst case, but may have a significant influence on practical complexities.
- Theoretical results on the complexity of Gröbner bases suppose the polynomial system to be semi–regular (see [BFS04, Bar04] for theoretical complexity of computing Gröbner bases) which is not true for the algebraic systems to follow.

Chapter 3

McEliece scheme using quasi-cyclic alternant codes

Contents

3.1 Quasi-cyclic alternant codes 44
3.1.1 Quasi-cyclic codes 44
3.1.2 Induced permutations of alternant codes 46
3.1.3 A construction of QC alternant codes 48
3.2 Structural analysis of the invariant code 48
3.2.1 Case σ diagonalizable over $\mathbb{F}_{q^m}$ 50
3.2.2 Case σ trigonalizable over $\mathbb{F}_{q^m}$ 52
3.2.3 Case σ diagonalizable in $\mathbb{F}_{q^{2m}} \backslash \mathbb{F}_{q^m}$ 53
3.3 Security reduction to the key security of the invariant code 54
3.3.1 Recover the divisor and guess the support 54
3.3.2 Recover the permutation 57
3.3.3 Case σ diagonalizable in $\mathbb{F}_{q^{2m}} \backslash \mathbb{F}_{q^m}$ 58
3.4 Proposition of a scheme: BIG QUAKE 59
3.4.1 Quasi-cyclic classical Goppa codes 59
3.4.2 The public key encryption scheme (PKE) 59
3.4.3 Description of the Key Encapsulation Mechanism (KEM) 60
3.4.4 Key recovery attacks and countermeasures 61
3.4.5 Message recovery attacks and countermeasure 64
3.4.6 Suggested parameters 64

In this chapter, we present a structure analysis of quasi-cyclic alternant codes. This permits to show that it is possible to reduce the key-recovery problem on the original quasi-cyclic code to the same problem on a smaller code derived from the public key. This smaller code is the *invariant* code and turns out to be also an alternant code. It is the first result we show. In a second time we prove that it is possible to recover the secret elements of a QC alternant code from the knowledge of the secret elements of its invariant code. This means that the key security of compact McEliece scheme based on alternant codes with some induced permutation reduces to the key security of the invariant code, which has smaller parameters. These two results have been presented at Workshop on Coding and Cryptography (WCC) in September 2017 and can be found in the paper "On the security of Some Compact Keys for McEliece Scheme" [Bar17].

In despite of the hight structure of QC alternant codes, we proposed a scheme using these codes. Indeed, we can notice that key-recovery is generally more expensive than message recovery. With a good choice of parameters it is still possible to construct quasi-cyclic codes with high

complexity of key recovery attack on the invariant code. Then in the context of the recent call of the National Institute for Standards and Technology (NIST) for post quantum cryptography, we proposed a key encapsulation scheme based on QC binary Goppa codes which belongs in the family of QC alternant codes [BBB+17b]. This proposal, called BIG QUAKE, is a collaboration with Magali Bardet, Olivier Blazy, Rodolfo Canto-Torres, Alain Couvreur, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich, and is also available on the NIST's web site.

3.1 Quasi-cyclic alternant codes

3.1.1 Quasi-cyclic codes

In what follows $\mathbb{F}$ denotes a finite field and ℓ denotes a positive integer.

Definition 3.1. Let ℓ be a positive integer and $\sigma : \mathbb{F}^\ell \to \mathbb{F}^\ell$ be the cyclic shift map:

$$\sigma_\ell : \begin{array}{ccc} \mathbb{F}^\ell & \longrightarrow & \mathbb{F}^\ell \\ (x_0, x_1, \ldots, x_{\ell-1}) & \longmapsto & (x_{\ell-1}, x_0, x_1, \ldots, x_{\ell-2}) \end{array}$$

Now, let n be an integer divisible by ℓ, we define the ℓ–th quasi–cyclic shift σ as the map obtained by applying σ_ℓ block-wise:

$$\sigma : \begin{array}{ccc} \mathbb{F}^n & \longrightarrow & \mathbb{F}^n \\ \left(\mathbf{b}_1 \mid \ldots \mid \mathbf{b}_{\frac{n}{\ell}}\right) & \longmapsto & \left(\sigma_\ell(\mathbf{b}_1) \mid \ldots \mid \sigma_\ell\left(\mathbf{b}_{\frac{n}{\ell}}\right)\right) \end{array},$$

where $\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_{\frac{n}{\ell}}$ denote blocks of length ℓ.

This notion is illustrated by Figure 3.1.

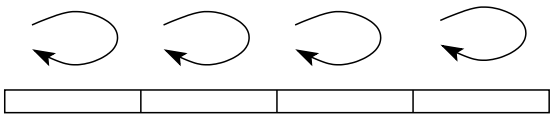

Figure 3.1: Illustration of the quasi–cyclic shift

Definition 3.2. A code $\mathscr{C} \subseteq \mathbb{F}^n$ is said to be *ℓ–quasi–cyclic* (ℓ–QC) if the code is stable by the quasi–cyclic shift map σ. We say that ℓ is the *order* of quasi–cyclicity of the code.

Such codes are used in cryptography, and it is the main point in this thesis, to reduce the key size of the McEliece scheme. To do this, we want to use generator matrix of quasi-cyclic codes which admits a compact representation. Let us introduce such a matrix.

Definition 3.3. Let $\mathbf{M}$ be a matrix. The matrix is said to be ℓ–block–circulant if it splits into $\ell \times \ell$ circulant blocks, i.e.

$$\mathbf{M} = \left(\begin{array}{c|c|c} & & \\ \hline \ldots & \mathbf{M}_i & \ldots \\ \hline & & \end{array}\right) \text{ with } \mathbf{M}_i := \begin{pmatrix} a_0 & a_1 & \cdots & a_{\ell-1} \\ a_{\ell-1} & a_0 & \cdots & a_{\ell-2} \\ \vdots & \ddots & \ddots & \vdots \\ a_1 & \cdots & a_{\ell-1} & a_0 \end{pmatrix}$$

This kind of matrix can be represented by the set of the first row of each block of size ℓ. That is, we can reconstruct an ℓ–block–circulant matrix $\mathbf{M}$ of k rows, from the knowledge of its rows with index $1, \ell+1, \ldots, k-\ell+1$.

Remark 4. A code $\mathscr{C}$ which has a ℓ–block–circulant generator matrix is an ℓ-QC code. Now, for a given ℓ-QC code $\mathscr{C}$, with dimension k, it is not sure that it admits a ℓ–block–circulant generator matrix. However there exists an ℓ–block–circulant matrix $\mathbf{G}$, with $k' \geq k$ rows, which generates the code $\mathscr{C}$.

In in order to reduce optimally the size of the public key in McEliece scheme, we want to consider codes which satisfy the following condition.

Condition 1. The $[n, k]$ ℓ–QC code $\mathscr{C}$ admits a generator matrix (or a parity check matrix) of the form

$$\mathbf{G} = \left(\mathbf{I}_k \mid \mathbf{M} \right)$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{M}$ is an ℓ–block–circulant matrix. We say that $\mathscr{C}$ is a *systematic quasi-cyclic code*.

Remark 5. To satisfy the previous condition, the dimension k of the ℓ–QC code should be a multiple of ℓ. Each time we will use this condition, we will ensure that it holds.

Definition 3.4. Given a matrix $\mathbf{G}$ as in Condition 1, we define $\rho(\mathbf{G})$ as the matrix obtained from $\mathbf{M}$ by extracting only the first row of each block. That is, $\rho(\mathbf{G})$ is obtained by stacking rows of $\mathbf{M}$ with indexes $1, \ell+1, 2\ell+1, \ldots, (n-k)-\ell+1$. Hence, we have the following ratio:

$$\text{NumberOfRows}(\mathbf{G}) = \ell \times \text{NumberOfRows}(\rho(\mathbf{G})).$$

For the analysis of some attacks against QC codes and hence for the security analysis of these codes, we need to introduce the notions of *invariant* code and *folded* code. We detail further the structure of these particular subcodes in the case where $\mathscr{C}$ is an algebraic code (see Sections 3.2 and 5.2).

Definition 3.5 (Folding map)**.** Let n be a positive integer such that ℓ divides n and $\sigma \in \mathfrak{S}_n$ be a the ℓ–th quasi–cyclic shift. The *folding map* on $\mathbb{F}$ is the following map:

$$\begin{array}{rccc} \varphi_\ell : & \mathbb{F}^n & \longrightarrow & \mathbb{F}^{\frac{n}{\ell}} \\ & (x_1, \ldots, x_n) & \longmapsto & \left(\sum_{i=0}^{\ell-1} x_{\sigma^i(1)}, \sum_{i=0}^{\ell-1} x_{\sigma^i(\ell+1)}, \ldots, \sum_{i=0}^{\ell-1} x_{\sigma^i(n-\ell+1)} \right). \end{array}$$

Equivalently, $\varphi_\ell := \text{Punct}_{\mathcal{I}_\ell}\left(\text{id} + \sigma + \cdots + \sigma^{\ell-1}\right)$, where $\mathcal{I}_\ell := \{1, \ldots, n\} \backslash \{1, \ell+1, \ldots, n-\ell+1\}$.

Definition 3.6 (Folded code)**.** Let $\mathscr{C} \subseteq \mathbb{F}^n$ be a ℓ–QC code, then the folded code is $\varphi_\ell(\mathscr{C}) \subseteq \mathbb{F}^{\frac{n}{\ell}}$.

Definition 3.7 (Invariant code)**.** Let $\mathscr{C} \subseteq \mathbb{F}^n$ be a ℓ–QC code, then the invariant code is defined by

$$\mathscr{C}^\sigma := \{c \in \mathscr{C} \mid \sigma(c) = c\}.$$

This code has repeated entries, then we use another code: the *punctured invariant code* which is defined by

$$\overline{\mathscr{C}}^\sigma := \text{Punct}_{\mathcal{I}_\ell}\left(\mathscr{C}^\sigma\right),$$

where $\mathcal{I}_\ell := \{1, \ldots, n\} \backslash \{1, \ell+1, \ldots, n-\ell+1\}$. Let $\sigma_{\mathscr{C}}$ be the ℓ-QC shift σ restricted to the code $\mathscr{C}$, we can write $\overline{\mathscr{C}}^\sigma = \text{Punct}_{\mathcal{I}_\ell}\left(\ker(\sigma_{\mathscr{C}} - \text{id})\right)$.

Later on, we speak about invariant code for both the invariant code and the punctured invariant code, but the notation remain different.

The folded code and the invariant codes are not equal in the general case but we have the following lemma.

Lemma 3.1. *Let ℓ be a positive integer and $\mathscr{C}$ be a ℓ-QC code, then we have*

$$\varphi_\ell(\mathscr{C}) \subseteq \overline{\mathscr{C}}^{\sigma}.$$

Moreover, if $\mathrm{char}(\mathbb{F}) \nmid \ell$ *then* $\varphi_\ell(\mathscr{C}) = \mathscr{C}^{\sigma}$.

Proof. Let $\varphi_{\mathscr{C}} := \mathrm{id}_{\mathscr{C}} + \sigma_{\mathscr{C}} + \cdots + \sigma_{\mathscr{C}}^{\ell-1}$ where $\sigma_{\mathscr{C}}$ and $\mathrm{id}_{\mathscr{C}}$ are the maps σ and id restricted to $\mathscr{C}$, then we have:

$$\varphi_\ell(\mathscr{C}) = \mathrm{Im}(\varphi_{\mathscr{C}}) \subseteq \ker(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}}).$$

This proves the first statement.
Now, we assume that $\mathrm{char}(\mathbb{F}) \nmid \ell$. We will show that $\dim(\varphi_\ell(\mathscr{C})) = \dim(\ker(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}}))$. By the rank–nullity theorem we know that

$$\dim(\mathrm{Im}(\varphi_{\mathscr{C}})) = \dim(\mathscr{C}) - \dim(\ker(\varphi_{\mathscr{C}})). \tag{3.1}$$

Moreover, $\sigma_{\mathscr{C}}^{\ell} - \mathrm{id}_{\mathscr{C}} = (\mathrm{id}_{\mathscr{C}} + \sigma_{\mathscr{C}} + \cdots + \sigma_{\mathscr{C}}^{\ell-1})(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}})$. Since $p \nmid \ell$, we have:

$$\gcd(\sum_{i=0}^{\ell-1} X^i, X - 1) = 1.$$

Hence, $\mathscr{C} = \ker(\varphi_{\mathscr{C}}) \oplus \ker(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}})$ and

$$\dim(\ker(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}})) = \dim(\mathscr{C}) - \dim(\ker(\varphi_{\mathscr{C}})). \tag{3.2}$$

Therefore from (3.1) and (3.2), $\ker(\sigma_{\mathscr{C}} - \mathrm{id}_{\mathscr{C}}) = \mathrm{Im}(\varphi_{\mathscr{C}})$ and $\varphi_\ell(\mathscr{C}) = \mathscr{C}^{\sigma}$. □

Throughout this thesis, we talk about subfield subcodes of algebraic codes. The following remark about invariant codes will be very useful for these algebraic codes.

Remark 6. We notice that the invariant operation commutes with the subfield subcode operation. Indeed, if $\mathscr{C}$ is a linear code over $\mathbb{F}_{q^m}$, stable under a permutation σ then:

$$(\mathscr{C} \cap \mathbb{F}_q^n)^{\sigma} = \{c \in \mathscr{C} \mid c \in \mathbb{F}_q^n \text{ and } \sigma(c) = c\} = \mathscr{C}^{\sigma} \cap \mathbb{F}_q^n.$$

3.1.2 Induced permutations of alternant codes

In what follows, q denotes a power of a prime p and m refers to the extension degree of the finite field $\mathbb{F}_{q^m}$. In this section we explain how we can construct an alternant code, defined over $\mathbb{F}_q$, stable under a prescribed permutation of the support $\{1, \ldots, n\}$, where n denotes the length of the code. Since alternant codes are subfield subcodes of GRS codes, we first study GRS code with permutations.

GRS code with permutations. In [Dür87], Dür determines the automorphism group of GRS codes. She shows that, for appropriate dimension, the whole permutation group is induced by the action of the projective linear group on the support of the code. The same property has been shown by Stichtenoth [Sti90], with the representation of GRS codes as AG codes. More precisely, for appropriate parameters, every permutation of $\mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)$ is induced by a projective linear transformation. First of all, we recall the definition of the projective linear group $\mathrm{PGL}_2(\mathbb{F}_{q^m})$. Then we recall the main definitions and theorems of [Sti90].

The projective linear group $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ is the automorphism group of the projective line $\mathbf{P}^1$ defined by

$$\mathrm{PGL}_2(\mathbb{F}_{q^m}) := \left\{ \begin{array}{ccc} \mathbf{P}^1 & \longrightarrow & \mathbf{P}^1 \\ (x:y) & \longmapsto & (ax+by : cx+dy) \end{array} \;\middle|\; \left\{ \begin{array}{l} a,b,c,d \in \mathbb{F}_{q^m}, \\ ad - bc \neq 0 \end{array} \right. \right\}.$$

The elements of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ have also a matrix representation, i.e.

$$\forall \sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m}),\ \text{we write } \sigma := \begin{pmatrix} a & b \\ c & d \end{pmatrix},\ \text{with } ad - bc \neq 0. \tag{3.3}$$

Where the elements a, b, c and d are defined up to a multiplication by a non-zero scalar. That is to say:

$$\mathrm{PGL}_2(\mathbb{F}_{q^m}) \simeq \mathrm{GL}_2(\mathbb{F}_{q^m}) / \left\{ \begin{pmatrix} \alpha & 0 \\ 0 & \alpha \end{pmatrix}, \alpha \in \mathbb{F}_{q^m}^* \right\}.$$

Definition 3.8. Let $\mathcal{P} \subseteq \mathbf{P}^1$ be a set of n distinct points on the projective line. Let G and G' be divisors of $\mathbf{P}^1$. Then we denote $G \sim_{\mathcal{P}} G'$ if there exists $f \in \mathbb{F}_{q^m}(\mathbf{P}^1)$, $f \neq 0$, such that $G - G' = (f)$ and $f(P) = 1$, for all $P \in \mathcal{P}$.

This is an equivalence relation and we have the following result.

Lemma 3.2 ([Sti90, Lemma 2.1])**.** *Let $\mathcal{P} \subseteq \mathbf{P}^1$ and G, G' be two divisors on $\mathbf{P}^1$ such that $Supp(G) \cap \mathcal{P} = \emptyset$ and $Supp(G') \cap \mathcal{P} = \emptyset$. If $G \sim_{\mathcal{P}} G'$ then $C_L(\mathbf{P}^1, \mathcal{P}, G) = C_L(\mathbf{P}^1, \mathcal{P}, G')$.*

The following definition and theorem give all the possible permutations for AG codes on the projective line.

Theorem 3.3 ([Sti90, Theorem 3.1])**.** *Let $\mathscr{C} = C_L(\mathbf{P}^1, \mathcal{P}, G) \subseteq \mathbb{F}_{q^m}^n$ be an AG code with divisor G such that $1 \leq deg(G) \leq n - 3$. Then*

$$Perm(\mathscr{C}) = \{\sigma \in Aut(\mathbf{P}^1) \mid \sigma(\mathcal{P}) = \mathcal{P} \text{ and } \sigma(G) \sim_{\mathcal{P}} G\}$$

.

Now, to construct GRS codes with permutations is very easy. Let $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ be an automorphism acting on the support $\mathcal{P} \subseteq \mathbf{P}^1$ and the divisor $G \in \mathrm{Div}(\mathbf{P}^1)$, then σ induces a permutation on the code $\mathscr{C} := \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)$. The induced permutation $\tilde{\sigma} \in \mathfrak{S}_n$ is defined by

$$\begin{array}{rccc} \tilde{\sigma}: & \mathscr{C} & \longrightarrow & \mathscr{C} \\ & (f(P_1), \ldots, f(P_n)) & \longmapsto & (f(\sigma(P_1)), \ldots, f(\sigma(P_n))). \end{array}$$

The case of alternant code. Since alternant codes are subfield subcodes of GRS codes, they inherit the permutation group of corresponding GRS codes. More precisely we have the following property.

Lemma 3.4. *Let $\mathscr{C} := \mathscr{A}_{r,q}(\mathcal{P}, G)$ be an alternant code over $\mathbb{F}_q$ and $\sigma \in Perm(C_L(\mathbf{P}^1, \mathcal{P}, G))$, then $\sigma \in Perm(\mathscr{C})$.*

Proof. By definition $\mathscr{C} = \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^{\perp} \cap \mathbb{F}_q^n$ and by Proposition 1.2, we know that $\sigma \in \mathrm{Perm}(\mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^{\perp})$. Let $c := (c_1, \ldots, c_n) \in \mathscr{C}$, then $\sigma(c) \in \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^{\perp}$, since $\mathscr{C}$ is a subcode of $\mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^{\perp}$. For all $i \in \{1, \ldots, n\}$, $c_{\sigma(i)} \in \mathbb{F}_q$, hence $\sigma(c) \in \mathscr{C}$ and $\sigma \in \mathrm{Perm}(\mathscr{C})$. □

This permits us to construct alternant codes with permutation. Unlike the case of GRS codes, there is no equivalence between the permutations of alternant codes and action of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ on the support and the divisor. In [Ber00b, Ber00a], Berger composes the action of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ with the Frobenius automorphism to construct alternant codes stable under a permutation induced by the action of an element of the projective semi- linear group $\mathrm{P\Gamma L}_2(\mathbb{F}_{q^m})$ on the support and the divisor. This group is defined as

$$\mathrm{P\Gamma L}_2(\mathbb{F}_{q^m}) := \left\{ \begin{array}{rcl} \mathbf{P}^1 & \longrightarrow & \mathbf{P}^1 \\ (x:y) & \longmapsto & (ax^{q^i} + by^{q^i} : cx^{q^i} + dy^{q^i}) \end{array} \;\middle|\; \left\{ \begin{array}{l} a, b, c, d \in \mathbb{F}_{q^m}, \\ ad - bc \neq 0 \end{array} \right., \text{ and } 0 \leq i < m \right\}.$$

Let $\mathfrak{f}_q : x \mapsto x^q$ be the Frobenius automorphism of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$ and $\rho \in \mathrm{P\Gamma L}_2(\mathbb{F}_{q^m})$, then we can write $\rho = \sigma \circ \mathfrak{f}_q^i$, for some $i \in \{0, \dots, m-1\}$ and $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$. Using this composition, Berger construct GRS codes stable under a permutation induced by an element $\rho \in \mathrm{P\Gamma L}_2(\mathbb{F}_{q^m})$. Then the subfield subcode a such a GRS code is also stable by a permutation induced by an element $\rho \in \mathrm{P\Gamma L}_2(\mathbb{F}_{q^m})$. This case is more complicated than the action of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ and we do not treat it in this thesis.

3.1.3 A construction of QC alternant codes

Now we have all the properties required to construct some alternant codes invariant under a permutation. We only consider the action of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$, the action of $\mathrm{P\Gamma L}_2(\mathbb{F}_{q^m})$ will not be treated.

Let $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ be an automorphism and ℓ be its order. For a point $P \in \mathbf{P}^1$, we denote $\mathrm{Orb}_\sigma(P) := \{\sigma^i(P) \mid i \in \{1, \dots, \ell-1\}\}$, it is the orbit of P under the action of σ. Let n be a positive integer divisible by ℓ. We define the support:

$$\mathcal{P} := \coprod_{i=1}^{n/\ell} \mathrm{Orb}_\sigma(P_i), \tag{3.4}$$

where the points $P_i \in \mathbf{P}^1(\mathbb{F}_{q^m})$ are pairwise distinct with trivial stabilizer subgroup. Then we define the divisor:

$$G := \sum_{i=1}^{s} t_i \sum_{Q \in Orb_\sigma(Q_i)} Q, \tag{3.5}$$

with Q_i closed points of $\mathbf{P}^1$ (see Definition 1.20), $s \in \mathbb{N}$, $t_i \in \mathbb{Z}$ for $i \in \{1, \dots, s\}$ and

$$\deg(G) = \sum_{i=1}^{s} t_i \ell \deg(Q_i).$$

As we saw in Section 3.1.2, the automorphism σ of $\mathbf{P}^1$ induces a permutation of $\mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)$. For short, we denote by σ both the automorphism of $\mathbf{P}^1$ and the induced permutation. Then, by Lemma 3.4, σ is also a permutation of $\mathscr{A}_{r,q}(\mathcal{P}, G) := \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)^\perp \cap \mathbb{F}_q^n$.

3.2 Structural analysis of the invariant code

In this section, we show that the invariant code of a QC alternant code is also an alternant code. As we noticed in Remark 6, the invariant operation commutes with the subfield subcode operation. This means that to analyse the structure of invariant code of alternant codes it suffices to look at invariant code of GRS codes, i.e. AG codes on the projective line.

Lemma 3.5. *Let $\mathscr{C} := C_L(\mathbf{P}^1, \mathcal{P}, G)$ and $\sigma \in Perm(\mathscr{C})$. If $c = \mathrm{Ev}_\mathcal{P}(f) \in \mathscr{C}$ is such that $\sigma(c) = c$, then f is σ-invariant, i.e. $f \circ \sigma = f$.*

Proof. Let $c = (f(P_1), \dots, f(P_n))$ such that $\sigma(c) = c$, then:

$$\begin{aligned} \forall i \in \{1, \dots, n\},\ f(P_{\sigma(i)}) = f(P_i) &\Leftrightarrow \forall i \in \{1, \dots, n\},\ f \circ \sigma(P_i) = f(P_i) \\ &\Leftrightarrow \forall i \in \{1, \dots, n\},\ (f \circ \sigma - f)(P_i) = 0. \end{aligned}$$

Since $\sigma(G) = G$, $f \circ \sigma \in L(G)$, and then $(f \circ \sigma - f) \in L(G)$. Hence if $(f \circ \sigma - f)$ was non-zero, it should have at most $\deg(G) < n$ zeros on $\mathbf{P}^1$, which is a contradiction. Therefore $(f \circ \sigma - f) \equiv 0$ and f is σ-invariant. □

Later on, to simplify the proofs we assume that G is constructed from single rational point Q, that is $G = t \sum_{R \in \mathrm{Orb}_\sigma(Q)} R$. The result remains true in the general case. We denote:

$$\begin{array}{ll} \sigma^j(P_i) & := (\alpha_{i\ell+j} : \beta_{i\ell+j}), \text{ for } i \in \{0, \ldots, \frac{n}{\ell} - 1\}, j \in \{0, \ldots, \ell - 1\} \\ \sigma^j(Q) & := (\gamma_j : \delta_j), \text{ for } j \in \{0, \ldots, \ell - 1\}. \end{array} \tag{3.6}$$

Lemma 3.6. *Let $G = t \sum_{R \in Orb_\sigma(Q)} R$, then we have:*

$$L(G) = \left\{ \frac{F(X,Y)}{\prod_{j=0}^{\ell-1} (\delta_j X - \gamma_j Y)^t} \;\middle|\; F \in \mathbb{F}_{q^m}[X,Y] \text{ homogeneous polynomial of degree } t\ell. \right\}$$

In the following, we study the action of an automorphism $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ on a Riemann-Roch space $L(G)$ as described just above. The goal of this part is to show the following result.

Theorem 3.7. *Let $C_L(\mathbf{P}^1, \mathcal{P}, G) \subseteq \mathbb{F}_{q^m}^n$ be an AG code of length n and dimension k, and $\sigma \in PGL_2(\mathbb{F}_{q^m})$ of order ℓ acting on it, with $\ell | n$. Let $\mathcal{P}$ and G as in (3.4) and (3.5). Then the invariant code $\overline{C_L(\mathbf{P}^1, \mathcal{P}, G)}^\sigma$ is an AG code of length n/ℓ and dimension $\lfloor k/\ell \rfloor$.*

The most important result in our security analysis of a scheme using QC alternant codes is the following consequence. This result comes from Theorem 3.7 and Remark 6.

Corollary 3.8. *Let $\mathscr{A}_{r,q}(\mathcal{P}, G) := C_L(\mathbf{P}^1, \mathcal{P}, G) \cap \mathbb{F}_q^n$ be an alternant code of length n and order r, and $\sigma \in PGL_2(\mathbb{F}_{q^m})$ of order ℓ acting on it, with $\ell | n$. Let $\mathcal{P}$ and G as in (3.4) and (3.5). Then the invariant code $\overline{\mathscr{A}_{\lfloor r/\ell \rfloor, q}(\mathcal{P}, G)}^\sigma$ is an alternant code of length n/ℓ and order $\lfloor r/\ell \rfloor$.*

In order to prove Theorem 3.7, we consider $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ with $\ell = ord(\sigma)$ and we define the support $\mathcal{P}$ and the divisor G as in (3.4) and (3.5). Moreover we keep the notation (3.6).

To the automorphism $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$, we associate a matrix $M \in \mathrm{GL}_2(\mathbb{F}_{q^m})$ as in (3.3). Later on, the notation $M \sim N$, with $M, N \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$, means there exists $P \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ such that $M = PNP^{-1}$. Three cases are possibles, depending on the eigenvalues of the matrix M:

1. $M \sim \begin{pmatrix} a & 0 \\ 0 & 1 \end{pmatrix}$, with $a \in \mathbb{F}_{q^m}$ (case diagonalizable in $\mathbb{F}_{q^m}$),
2. $M \sim \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix}$, with $b \in \mathbb{F}_{q^m}$ (case trigonalizable in $\mathbb{F}_{q^m}$),
3. $M \sim \begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{q^m} \end{pmatrix}$, with $\alpha \in \mathbb{F}_{q^{2m}}$ (case diagonalizable in $\mathbb{F}_{q^{2m}}$).

The following lemma shows that the study of GRS codes invariant under an induced permutation $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ can be reduced to the study of the three cases: (1), (2) and (3).

Lemma 3.9. *Let $\mathscr{C} := C_L(\mathbf{P}^1, \mathcal{P}, G)$ be an AG code such that $\sigma(\mathscr{C}) = \mathscr{C}$ and $\rho \in PGL_2(\mathbb{F}_{q^m})$. Then $\sigma' := \rho \circ \sigma \circ \rho^{-1}$ induces the same permutation on $\mathscr{C}$ as σ.*

Proof. We first prove that:

$$\mathrm{C_L}(\mathbf{P}^1, \rho^{-1}(\mathcal{P}), \rho^{-1}(G)) = \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G).$$

Let $c = (f(P_1), \ldots, f(P_n))$ be a codeword of $\mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)$. Then, we have

$$c = (f \circ \rho \circ \rho^{-1}(P_1), \ldots, f \circ \rho \circ \rho^{-1}(P_n)).$$

As $f \in L(G)$, the function $h = f \circ \rho \in L(\rho^{-1}(G))$. Hence, $c \in \{\mathrm{Ev}_{\rho^{-1}(\mathcal{P})}(h) \mid h \in L(\rho^{-1}(G))\} = \mathrm{C_L}(\mathbf{P}^1, \rho^{-1}(\mathcal{P}), \rho^{-1}(G))$.
Now, for all $c = (f(P_1), \ldots, f(P_n)) \in \mathscr{C}$, we have:

$$\begin{aligned}\sigma'(c) &= (f \circ \rho \circ \sigma \circ \rho^{-1}(P_1), \ldots, f \circ \rho \circ \sigma \circ \rho^{-1}(P_n)) \\ &= (h \circ \sigma(\rho^{-1}(P_1)), \ldots, h \circ \sigma(\rho^{-1}(P_n)))\end{aligned}$$

with $h = f \circ \rho \in L(\rho^{-1}(G))$. Since $\mathrm{C_L}(\mathbf{P}^1, \rho^{-1}(\mathcal{P}), \rho^{-1}(G)) = \mathrm{C_L}(\mathbf{P}^1, \mathcal{P}, G)$, σ' induces the same permutation of the code $\mathscr{C}$ as σ. □

3.2.1 Case σ diagonalizable over $\mathbb{F}_{q^m}$

In this section, we consider $\sigma := \rho \circ \sigma_d \circ \rho^{-1}$ with $\sigma_d \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$ diagonal and $\rho \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$. W.l.o.g and by Lemma 3.9, one can assume that:

$$\begin{array}{rccc}\sigma\colon & \mathbf{P}^1 & \to & \mathbf{P}^1 \\ & (x:y) & \mapsto & (ax:y),\end{array} \tag{3.7}$$

with $a \in \mathbb{F}_{q^m}^*$. As we saw in Lemma 3.5, the codewords of an AG code fixed by the automorphism σ acting on it, is directly related to the functions invariant by σ. Moreover, Lemma 3.6 describes the Riemann-Roch spaces associated to a divisor globally stable by an automorphism of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$. Functions of the Riemann-Roch space are fractions of homogeneous polynomials. The following proposition gives us the structure of homogeneous polynomials fixed by the diagonal automorphism σ (3.7).

Proposition 3.10. *Let $F \in \mathbb{F}_{q^m}[X, Y]$ be a homogeneous polynomial of degree $t\ell$, and $a \in \mathbb{F}_{q^m}$ of order ℓ. If $F(aX, Y) = F(X, Y)$, then $F(X, Y) = R(X^\ell, Y^\ell)$, with $R \in \mathbb{F}_{q^m}[X, Y]$ a homogeneous polynomial of degree t.*

A proof is given in [FOP+16a, Prop 4]. Here, we present a simpler proof.

Proof. The homogeneous polynomial F can be written as:

$$F(X, Y) = \sum_{i+j=t\ell} f_{ij} X^i Y^j,$$

with $f_{ij} \in \mathbb{F}_{q^m}$. Since $F(aX, Y) = F(X, Y)$, we have:

$$\sum_{i+j=t\ell} f_{ij} X^i Y^j = \sum_{i+j=t\ell} f_{ij} a^i X^i Y^j.$$

Hence $f_{ij} = a^i f_{ij}, \forall i, j \in \mathbb{N}$ such that $i + j = t\ell$. As the order of a is ℓ, we have $a^i \neq 1, \forall i \in \mathbb{N}$ such that $\ell \nmid i$. Therefore $f_{ij} = 0, \forall i \in \mathbb{N}$ such that $\ell \nmid i$. So $F(X, Y) = R(X^\ell, Y^\ell)$, with $R \in \mathbb{F}_{q^m}[X, Y]$ an homogeneous polynomial of degree t. □

Thanks to the previous proposition we are able to prove Theorem 3.7 in the case where σ is diagonal. This is the following proposition.

Proposition 3.11. *Let $\mathscr{C} := C_L(\mathbf{P}^1, \mathcal{P}, G)$ be an AG code as in Theorem 3.7, with σ as in (3.7). Let $\tilde{P}_i = (\alpha_i^\ell : \beta_i^\ell)$ and $\tilde{G} = t\tilde{Q}$, where either $\tilde{Q} = ((-1)^{\ell-1} a^{\frac{\ell(\ell-1)}{2}} (\frac{\gamma_0}{\delta_0})^\ell : 1)$ or $\tilde{Q} = P_\infty$. Then $\overline{\mathscr{C}}^\sigma = C_L(\mathbf{P}^1, \tilde{\mathcal{P}}, \tilde{G})$, which is a GRS code.*

Proof. Let $c := \mathrm{Ev}_{\mathcal{P}}(f) \in \mathscr{C}$ such that $\sigma(c) = c$, we denote $c_\ell := \mathrm{Punct}_{\mathcal{I}_\ell}(c)$, with the set $\mathcal{I}_\ell \subseteq \{1, \ldots, n\}$ defined as in Definition 3.7. The codeword c_ℓ is in the invariant code $\overline{\mathscr{C}}^\sigma$ and

by definition $c_\ell = \mathrm{Punct}_{\mathcal{I}_\ell}(\mathrm{Ev}_{\mathcal{P}}(f))$. By Lemma 3.5, $f \in L(G)$ is fixed by σ, i.e. $f(aX, Y) = f(X, Y)$. Then, by Lemma 3.6, we have

$$f(aX,Y) = f(X,Y) \iff \frac{F(aX,Y)}{\left(\prod_{j=0}^{\ell-1}(a\delta_j X - \gamma_j Y)\right)^t} = \frac{F(X,Y)}{\left(\prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y)\right)^t} \tag{3.8}$$

with $F \in \mathbb{F}_{q^m}[X,Y]$ a homogeneous polynomial of degree $t\ell$. Since the support of G is stable under the action of σ, we can write

$$\begin{aligned}\prod_{j=0}^{\ell-1}(a\delta_j X - \gamma_j Y) &= \prod_{j=0}^{\ell-1}(a\delta_j X - a\gamma_j Y)\\ &= a^\ell \prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y)\\ &= \prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y)\cdot\end{aligned}$$

Hence, the right side of (3.8) becomes $F(aX, Y) = F(X, Y)$. By Proposition 3.10, we know that $F(X, Y) = R(X^\ell, Y^\ell)$ with $R \in \mathbb{F}_{q^m}[X, Y]$ an homogeneous polynomial of degree t. As we see just above, the product $\prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y)$ is also invariant. Again Proposition 3.10 gives:

$$\prod_{j=0}^{\ell-1}(\delta_j X - \gamma_j Y) = \big(\prod_{j=0}^{\ell-1}\delta_j\big)X^\ell + (-1)^\ell\big(\prod_{j=0}^{\ell-1}\gamma_j\big)Y^\ell\cdot$$

Therefore:

$$f(X,Y) = \frac{R(X^\ell, Y^\ell)}{\Big(\big(\prod_{j=0}^{\ell-1}\delta_j\big)X^\ell - (-1)^{\ell-1}\big(\prod_{j=0}^{\ell-1}\gamma_j\big)Y^\ell\Big)^t}\cdot \tag{3.9}$$

For all $i \in \{1, \ldots, \frac{n}{\ell}\}$, we have $f(P_i) = \tilde{f}(\tilde{P}_i)$, with

$$\tilde{f}(X,Y) := \frac{R(X,Y)}{\Big(\big(\prod_{j=0}^{\ell-1}\delta_j\big)X - (-1)^{\ell-1}\big(\prod_{j=0}^{\ell-1}\gamma_j\big)Y\Big)^t} \quad \text{and} \quad \tilde{P}_i := (\alpha_i^\ell : \beta_i^\ell). \tag{3.10}$$

We denote $\tilde{\delta} = \big(\prod_{j=0}^{\ell-1}\delta_j\big)$ and $\tilde{\gamma} = (-1)^{\ell-1}\big(\prod_{j=0}^{\ell-1}\gamma_j\big)$. Let $\tilde{G} := t\tilde{Q}$ be a divisor with $\tilde{Q} := (\tilde{\gamma} : \tilde{\delta})$, by Lemma 3.6, we have:

$$L(\tilde{G}) := \left\{ \frac{R(X,Y)}{\Big(\tilde{\delta}X - \tilde{\gamma}Y\Big)^t} \;\middle|\; R \in \mathbb{F}_{q^m}[X,Y] \text{ homogeneous polynomial of degree } t \right\} \cup \{0\}\cdot$$

Hence the codeword $c_\ell \in \mathrm{C_L}(\mathbf{P}^1, \tilde{\mathcal{P}}, \tilde{G})$.
If $\tilde{Q} \neq P_\infty$, then $\forall j$, $\delta_j \neq 0$ and we can write:

$$\tilde{Q} = \big((-1)^{\ell-1}\prod_{j=0}^{\ell-1}\frac{\gamma_j}{\delta_j} : 1\big).$$

Moreover we have:

$$\prod_{j=0}^{\ell-1}\frac{\gamma_j}{\delta_j}=\prod_{j=0}^{\ell-1}a^j\frac{\gamma_0}{\delta_0}=(\prod_{j=0}^{\ell-1}a^j)(\frac{\gamma_0}{\delta_0})^\ell=a^{\frac{\ell(\ell-1)}{2}}(\frac{\gamma_0}{\delta_0})^\ell.$$

We notice that $\prod_{j=0}^{\ell-1}\frac{\gamma_j}{\delta_j}=(\frac{\gamma_0}{\delta_0})^\ell$, if ℓ is odd.

Conversely, let $c_\ell := \mathrm{Ev}_{\tilde{\mathcal{P}}}(\tilde{f}) \in \mathrm{C_L}(\mathbf{P}^1,\tilde{\mathcal{P}},\tilde{G})$ be a codeword with $\tilde{f}$ as in (3.10). We define $f := \tilde{f}(X^\ell,Y^\ell) \in L(G)$, then $c=\mathrm{Ev}_{\mathcal{P}}(f)\in\mathscr{C}$. The function f is invariant by σ, then $\sigma(c)=c$ and $c_\ell\in\overline{\mathscr{C}}^\sigma$. □

3.2.2 Case σ trigonalizable over $\mathbb{F}_{q^m}$

Here, we consider the case (2) where σ is trigonalizable in $\mathbb{F}_{q^m}$. As in the previous section we only have to treat the case where σ is upper triangular. W.l.o.g and by Lemme 3.9, one can assume that:

$$\begin{array}{rccc}\sigma: & \mathbf{P}^1 & \to & \mathbf{P}^1\\ & (x:y) & \mapsto & (x+by:y)\end{array} \tag{3.11}$$

with $b\in\mathbb{F}_{q^m}^*$. In this case, we have $\ell=\mathrm{ord}(\sigma)=p$. As previously we need to know the structure of the homogeneous polynomials of $\mathbb{F}_{q^m}[X,Y]$ fixed by the automorphism σ.

Proposition 3.12 ([FOP+16a, Prop 4])**.** *Let $F\in\mathbb{F}_{q^m}[X,Y]$ be a polynomial of degree $\deg(F)\le tp$ and $b\in\mathbb{F}_q^*$. If $F(X+bY,Y)=F(X,Y)$, then $F(X,Y)=R(X^p-b^{p-1}XY^{p-1},Y^p)$, with p the characteristic of $\mathbb{F}_{q^m}$ and $R\in\mathbb{F}_{q^m}[X,Y]$ an homogeneous polynomial of degree $\deg(R)\le t$.*

The case of Theorem 3.7 with an automorphism σ upper triangular is treated in the following proposition.

Proposition 3.13. *Let $\mathscr{C}:=C_L(\mathbf{P}^1,\mathcal{P},G)$ be an AG code as in Theorem 3.7, with σ as in (3.11). Let $\tilde{P}_i=(\alpha_i^p-b^{p-1}\alpha_i\beta_i^{p-1}:\beta_i^p)$ and $\tilde{G}=t\tilde{Q}$, where either $\tilde{Q}=((\frac{\gamma_0}{\delta_0})^p-b^{p-1}\frac{\gamma_0}{\delta_0}:1)$ or $\tilde{Q}=P_\infty$. Then $\overline{\mathscr{C}}^\sigma=C_L(\mathbf{P}^1,\tilde{\mathcal{P}},\tilde{G})$, which is a GRS code.*

The proof of this proposition is similar to the proof of Proposition 3.11.

Proof. Let $c=\mathrm{Ev}_{\mathcal{P}}(f)\in\mathscr{C}$ such that $\sigma(c)=c$, we denote $c_\ell:=\mathrm{Punct}_{\mathcal{I}_\ell}(c)$, with the set $\mathcal{I}_\ell\subseteq\{1,\dots,n\}$ defined as in Definition 3.7. The codeword c_ℓ is in the invariant code $\overline{\mathscr{C}}^\sigma$ and by definition $c_\ell=\mathrm{Punct}_{\mathcal{I}_\ell}(\mathrm{Ev}_{\mathcal{P}}(f))$. By Lemma 3.5, f is fixed by σ, i.e. $f(X+bY,Y)=f(X,Y)$. Then, by Lemma 3.6, we have:

$$f(X+bY,Y)=f(X,Y)\iff\frac{F(X+bY,Y)}{\Big(\prod_{j=0}^{p-1}\big(\delta_j(X+bY)-\gamma_jY\big)\Big)^t}=\frac{F(X,Y)}{\Big(\prod_{j=0}^{p-1}\big(\delta_jX-\gamma_jY\big)\Big)^t}, \tag{3.12}$$

with $F\in\mathbb{F}_q[X,Y]$ an homogeneous polynomial of degree tp. Since the support of G is globally stable under the action of σ, we have

$$\begin{aligned}\prod_{j=0}^{p-1}(\delta_j(X+bY)-\gamma_jY)&=\prod_{j=0}^{p-1}(\delta_jX-(\gamma_j-b\delta_j)Y)\\&=\prod_{j=1}^{p}(\delta_{j-1}X-\gamma_{j-1}Y)\\&=\prod_{j=0}^{p-1}(\delta_jX-\gamma_jY).\end{aligned}$$

Hence, the right side of (3.12) becomes $F(X+bY,Y)=F(X,Y)$. By Proposition 3.12, we have $F(X,Y)=R(X^p-b^{p-1}XY^{p-1},Y^p)$, with $R\in\mathbb{F}_{q^m}[X,Y]$ an homogeneous polynomial of degree $\deg(R)\leq t$. The product $\prod_{j=0}^{p-1}(\delta_j X-\gamma_j Y)$ is also invariant by σ, by Proposition 3.12, we have

$$\prod_{j=0}^{p-1}(\delta_j X-\gamma_j Y)=\big(\prod_{j=0}^{p-1}\delta_j\big)\big(X^p-b^{p-1}XY^{p-1},Y^p\big)+\big((-1)^p\prod_{j=0}^{p-1}\gamma_j\big)Y^p.$$

Hence:

$$f(X,Y)=\frac{R(X^p-b^{p-1}XY^{p-1},Y^p)}{\Big(\big(\prod_{j=0}^{p-1}\delta_j\big)\big(X^p-b^{p-1}XY^{p-1}\big)-\big((-1)^{p-1}\prod_{j=0}^{p-1}\gamma_j\big)Y^p\Big)^t}.$$

The arguments to conclude this proof are the same as in Proposition 5.4. Moreover, if $\tilde{Q}\neq P_\infty$, then for all j, $\delta_j\neq 0$ and we can write:

$$\tilde{Q}=\big(\prod_{j=0}^{p-1}\frac{\gamma_j}{\delta_j}:1\big).$$

We have:

$$\prod_{j=0}^{p-1}\frac{\gamma_j}{\delta_j}=\prod_{j=0}^{p-1}\big(\frac{\gamma_0}{\delta_0}+jb\big)=\big(\frac{\gamma_0}{\delta_0}\big)^p-b^{p-1}\frac{\gamma_0}{\delta_0}.$$ □

3.2.3 Case σ diagonalizable in $\mathbb{F}_{q^{2m}}\backslash\mathbb{F}_{q^m}$

In this section we consider the case (3), that is $\sigma:=\rho\circ\sigma_d\circ\rho^{-1}$ with σ_d diagonal in $\mathrm{GL}_2(\mathbb{F}_{q^{2m}})$ and $\rho\in\mathrm{PGL}_2(\mathbb{F}_{q^{2m}})$. Here we cannot apply directly Lemma 3.9 since the code $\mathscr{C}$ is defined over $\mathbb{F}_{q^m}$ and ρ acts on $\mathbf{P}^1(\mathbb{F}_{q^{2m}})$. To overcome this difficulty, we want to extend the code $\mathscr{C}$, defined on $\mathbb{F}_{q^m}$, to the field $\mathbb{F}_{q^{2m}}$. Then we consider the code $\mathscr{C}\otimes\mathbb{F}_{q^{2m}}:=\langle\mathscr{C}\rangle_{\mathbb{F}_{q^{2m}}}$.

$$\begin{array}{ccc}\mathscr{C}\otimes\mathbb{F}_{q^{2m}} & \xrightarrow{\mathrm{Inv}_\sigma} & \overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma}\\ \text{Subfield Subcode}\uparrow & & \uparrow\text{Subfield Subcode}\\ \mathscr{C} & \xrightarrow{\mathrm{Inv}_\sigma} & \overline{\mathscr{C}}^{\sigma}\end{array}$$

Let us denote $L_{\mathbb{F}}(G)$ the Riemann-Roch space included in the function field $\mathbb{F}(x)$. Then we have $\mathscr{C}\otimes\mathbb{F}_{q^{2m}}=\{\mathrm{Ev}_{\mathcal{P}}(f)|f\in L_{\mathbb{F}_{q^{2m}}}(G)\}$. Since $\mathcal{P}\subseteq\mathbf{P}^1(\mathbb{F}_{q^m})$ in particular $\mathcal{P}\subseteq\mathbf{P}^1(\mathbb{F}_{q^{2m}})$ and we have $\mathscr{C}\otimes\mathbb{F}_{q^{2m}}=\mathrm{C_L}(\mathbf{P}^1_{\mathbb{F}_{q^{2m}}},\mathcal{P},G)$. Now it is possible to apply Lemma 3.9 to $\mathscr{C}\otimes\mathbb{F}_{q^{2m}}$. Then the invariant code $\overline{\mathscr{C}\otimes\mathbb{F}_{q^{2m}}}^{\sigma}$ equals the invariant code $(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})^{\sigma_d}$. By Section 3.2.1, we know that the code $\overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma_d}$ is a GRS code. Now we have to show that $\overline{\mathscr{C}}^{\sigma}$ is also a GRS code. First we know that, by definition, $\mathscr{C}\otimes\mathbb{F}_{q^{2m}}$ has a basis in $\mathbb{F}_{q^m}^n$. Then the following lemma permits us to show that the code $\overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma}$ also has a basis in $\mathbb{F}_{q^m}^n$.

Lemma 3.14. *The characteristic p of $\mathbb{F}_{q^m}$ does not divide the order ℓ of σ.*

Proof. The automorphism σ is defined as $\sigma:=\rho\circ\sigma_d\circ\rho^{-1}$ with $\sigma_d:(x:y)\mapsto(\alpha x:\alpha^{q^m}y)$ diagonal in $\mathrm{GL}_2(\mathbb{F}_{q^{2m}})$, that is $\alpha\in\mathbb{F}_{q^{2m}}$, and $\rho\in\mathrm{PGL}_2(\mathbb{F}_{q^{2m}})$. Then the order ℓ of σ is the order of α and we have $\ell\mid(q^{2m}-1)$. Since $q=p^s$, for some $s\in\mathbb{N}^*$, we have $\ell\mid(p^{s2m}-1)$ and so $p\nmid\ell$. □

With the previous lemma, and by Lemma 3.1, we have $\varphi_\ell(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})=\overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma}$. Since the application φ_ℓ is $\mathbb{F}_q$-linear, the code $\overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma}$ has a basis in $\mathbb{F}_{q^m}^n$. Therefore, the subfield subcode on $\mathbb{F}_{q^m}$ of the GRS code $\overline{(\mathscr{C}\otimes\mathbb{F}_{q^{2m}})}^{\sigma_d}$ is a GRS code. That is $\overline{\mathscr{C}}^{\sigma}$ is a GRS code.

3.3 Security reduction to the key security of the invariant code

In this section, we show that the key security of a QC alternant codes, as described in Section 3.1.3, reduces to the key security of its invariant code. We consider an automorphism $\sigma \in \mathrm{PGL}_2(\mathbb{F}_{q^m})$, and an alternant code

$$\mathscr{A}_{r,q}(\mathcal{P}, G) := \mathrm{C}_{\mathrm{L}}(\mathbf{P}^1, \mathcal{P}, G)^{\perp} \cap \mathbb{F}_q^n$$

which is stable under the action of σ. In particular, the support $\mathcal{P}$ and the divisor G are defined as (3.4) and (3.5). From the knowledge of a generator matrix of $\mathscr{A}_{r,q}(\mathcal{P}, G)$ and the induced permutation σ, it is possible to compute the invariant code $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)}^{\sigma}$. By Section 3.2 and Corollary 3.8, we know that this invariant code is an alternant code $\mathscr{A}_{r,q}(\tilde{\mathcal{P}}, \tilde{G})$, for some support and divisor, with smaller parameters. We will show that thanks to the knowledge of $\tilde{\mathcal{P}}$ and $\tilde{G}$ we are able to recover $\mathcal{P}$ and G. We already know that there is a link between the form of $\tilde{\mathcal{P}}$ and $\tilde{G}$ of the invariant code and the form of $\mathcal{P}$ and G of the original alternant code. This link is described by Propositions 3.11 and 3.13 of the previous section.

Here we assume that the support $\tilde{\mathcal{P}}$ and the divisor $\tilde{G}$ of the invariant code $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)}^{\sigma}$ are known. Later on, we denote by $\tilde{\mathcal{P}} := \big\{(\tilde{\alpha}_i : \tilde{\beta}_i) \mid i \in \{1, \ldots, \frac{n}{\ell}\}\big\}$. Moreover we assume that G is constructed from the orbit of one rational point Q, this permits to simplify the proofs and algorithms but the result remains true for the general case. The following notation for the support $\mathcal{P}$ and the divisor G of the public code, will be used. We denote

$$\begin{aligned} \mathcal{P} &:= \big\{(\alpha_{i,j} : 1) \mid i \in \{1, \ldots, \tfrac{n}{\ell}\},\ j \in \{0, \ldots, \ell-1\}\big\} \\ G &:= t \sum_{j=0}^{\ell-1} \sigma^j(Q) \end{aligned} \tag{3.13}$$

with $\sigma^j(Q) := (\gamma_j : \delta_j)$, for all $j \in \{0, \ldots, \ell-1\}$.

3.3.1 Recover the divisor and guess the support

As previously, we study three cases: σ is diagonalizable over $\mathbb{F}_{q^m}$ (1), σ is trigonalizable (2), or σ is diagonalizable over $\mathbb{F}_{q^{2m}}$ (3). In this section, we treat the two first cases, the third case is more particular and it will be treated in Section 3.3.3. The order ℓ of σ is known, hence we know the form of σ.

Case σ diagonalizable over $\mathbb{F}_{q^m}$. In this case, we recall that the form is:

$$\begin{array}{rccc} \sigma\colon & \mathbf{P}^1 & \to & \mathbf{P}^1 \\ & (x : y) & \mapsto & (ax : y), \end{array}$$

with $a \in \mathbb{F}_{q^m}^{\times}$, a primitive ℓ-th root of unity. There exist only $\varphi(\ell) < n$ possibilities for a, where φ is the Euler's phi function, hence we are able to test all the possibilities in a reasonable time. W.l.o.g, we assume for now that we know the element a. The first step is to recover G from $\tilde{G}$.

Keeping the notation (3.13), by Proposition 3.11, we know that

$$\tilde{G} = t\tilde{Q}, \text{ with } \tilde{Q} := \begin{cases} \big((-1)^{\ell-1} a^{\frac{\ell(\ell-1)}{2}} (\frac{\gamma_0}{\delta_0})^{\ell} : 1\big) \\ \text{or } P_{\infty}. \end{cases}$$

Since we know a, we can recover the support of G thanks to the support $\tilde{Q}$ of $\tilde{G}$.

Remark 7. In the case where $\tilde{Q} \neq P_{\infty}$, for all $i \in \{0, \ldots, \ell-1\}$, we have

$$\tilde{Q} = \big((-1)^{\ell-1} (a^i)^{\frac{\ell(\ell-1)}{2}} (\frac{\gamma_i}{\delta_i})^{\ell} : 1\big).$$

We denote $\mu_\ell := \{\sigma^i(a) \mid i \in \{0, \ldots, \ell-1\}\}$ and then from every $a \in \mu_\ell$ we are able to recover the support of G. The set μ_ℓ is exactly the set of the ℓ-th roots of the unity.

Algorithm 1 describes the computation of the divisor G from the knowledge of $\tilde{G}$. The cost of this algorithm will be negligible compared to the cost of the support recovering (see Algorithm 3). The main step of this Algorithm 1 is to find the root of a polynomial $p(X) := a^{\frac{\ell(\ell-1)}{2}} X^\ell - \tilde{\gamma} \in \mathbb{F}_{q^m}[X]$ and this can be done with Berlekamp algorithm.

Algorithm 1: Recover the divisor in the case σ diagonalizable in $\mathbb{F}_{q^m}$

Input : The divisor $\tilde{G}$ of the invariant code $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)^\sigma}$.
Output: Return the support G

$a \leftarrow$ a primitive ℓ-th root of $\mathbb{F}_{q^m}$
if $\tilde{Q} \neq P_\infty$ **then** // $\tilde{G} = t * \tilde{Q}$, with $\tilde{Q} = (\tilde{\gamma} : 1)$
  $\Gamma \leftarrow \text{roots}(a^{\frac{\ell(\ell-1)}{2}} X^\ell - \tilde{\gamma})$ // $a \in \mu_\ell$
  $G \leftarrow t \sum_{\gamma \in \Gamma} (\gamma : 1)$
else
  $G = t\ell P_\infty$
return G.

The second step is to recover a support $\mathcal{P}'$ such as $\mathscr{A}_{r,q}(\mathcal{P}', G) = \mathscr{A}_{r,q}(\mathcal{P}, G)$. By Proposition 3.11, we know that a point $P := (x : y)$ in $\mathcal{P}$ satisfies:

$$\begin{cases} x^\ell - \tilde{\alpha}_i = 0 \\ y^\ell - \tilde{\beta}_i = 0, \end{cases} \tag{3.14}$$

for some $i \in \{1, \dots, \frac{n}{\ell}\}$ such that $(\tilde{\alpha}_i : \tilde{\beta}_i) = \tilde{P}_i$. Since we know $\tilde{\mathcal{P}}$, we are able to recover all elements of $\mathcal{P}$ but **as an unordered set**. We choose one solution (α_i, β_i) of (3.14) for each $i \in \{1, \dots, \frac{n}{\ell}\}$ and we choose $a \in \mu_\ell$, then the set:

$$\mathcal{P}' := \left\{ \left(a^j \frac{\alpha_i}{\beta_i} : 1\right) \mid j \in \{0, \dots, \ell - 1\}, i \in \{1, \dots, \frac{n}{\ell}\} \right\} \tag{3.15}$$

is a support such as $\mathscr{A}_{r,q}(\mathcal{P}', G)$ is a permutation of the code $\mathscr{A}_{r,q}(\mathcal{P}, G)$. For each choice of a set of solutions $S := \{(\alpha_i, \beta_i) \mid i \in \{1, \dots, \frac{n}{\ell}\}\}$ and each choice of $a \in \mu_\ell$, we have a different support $\mathcal{P}'$. In Section 3.3.2 we give an algorithm to find a good choice for S and a and hence the permutation between $\mathscr{A}_{r,q}(\mathcal{P}', G)$ and $\mathscr{A}_{r,q}(\mathcal{P}, G)$.

Case σ trigonalisable over $\mathbb{F}_{q^m}$. In the case where σ is trigonalisable, it is more complicated to know exactly σ. In this case we know that σ has the following form:

$$\begin{array}{rccc} \sigma : & \mathbf{P}^1 & \to & \mathbf{P}^1 \\ & (x : y) & \mapsto & (x + by : y), \end{array}$$

with $b \in \mathbb{F}_{q^m}^\times$. Here, the order of σ is $\ell = p := \text{Char}(\mathbb{F}_{q^m})$ and the first step is to recover b.

Lemma 3.15. *b is a root of the polynomial:*

$$P_b := \gcd\left(\left\{ \text{Res}_X(X^p - Y^{p-1}X - \tilde{\alpha}_i,\ X^{q^m} - X) \mid i \in \{1, \dots, \frac{n}{\ell}\} \right\},\ Y^{q^m} - Y \right),$$

where $\text{Res}_X(P, Q)$ denotes the resultant of the two polynomials P and Q with respect to X.

Proof. We recall that we use notation (3.13). By Proposition 3.13, b is a root of the polynomial $\alpha_{i,j}^p - Y^{p-1}\alpha_{i,j} - \tilde{\alpha}_i \in \mathbb{F}_{q^m}[Y]$ for all $i \in \{1, \dots, \frac{n}{p}\}$ and $j \in \{0, \dots, p-1\}$. As $\alpha_{i,j} \in \mathbb{F}_{q^m}^\times$ for all i, j and polynomials $X^p - Y^{p-1}X - \tilde{\alpha}_i$ are monic in the variable X, we can also write that b is a root of the polynomial $\text{Res}_X(X^p - Y^{p-1}X - \tilde{\alpha}_i,\ X^{q^m} - X) \in \mathbb{F}_{q^m}[Y]$ for all $i \in \{1, \dots, \frac{n}{p}\}$. □

All the elements of the orbit of b under the action of σ are roots of the polynomial P_b defined in Lemma 3.15, i.e.: the set of roots of P_b contains $B := \{b, 2b, \ldots, (\ell-1)b\}$. In practice, and according to computer aided experiments, the degree of the polynomial P_b is ℓ and the set B is exactly the set of its roots. Then there exist only $\ell < n$ possibilities for b, so we are able to test all the possibilities in a reasonable time. As in §3.3.1, we assume for now that we know the element b.

Now we are able to recover the divisor G from $\tilde{G}$. By Proposition 3.13, we know that

$$\tilde{G} = t\tilde{Q}, \text{ where } \tilde{Q} = \begin{cases} \left((\frac{\gamma_0}{\delta_0})^p - b^{p-1}\frac{\gamma_0}{\delta_0} : 1 \right) \\ \text{or } P_\infty. \end{cases}$$

Since we know b, we can recover the support of G thanks to the support $\tilde{Q}$ of $\tilde{G}$.

Algorithm 2: Recover the divisor in the case σ trigonalizable over $\mathbb{F}_{q^m}$

Input : The divisor $\tilde{G}$ of the invariant code $\overline{\mathscr{A}_{r,q}(\mathcal{P}, G)}^{\sigma}$.
Output: G and the set B of possible values of b.

1 `/* Recover` $B := \{b, 2b, \ldots, (p-1)b\}$ `*/` `//` $\tilde{\mathcal{P}} = \{(\tilde{\alpha}_i : \tilde{\beta}_i) \mid i \in \{1, \ldots, \frac{n}{\ell}\}\}$
2 $P_b \leftarrow \gcd\left(\{\mathrm{Res}_X(X^p - Y^{p-1}X - \tilde{\alpha}_i,\ X^{q^m} - X) \mid i \in \{1, \ldots, \frac{n}{p}\}\},\ Y^{q^m} - Y\right)$
3 $B \leftarrow \mathrm{roots}(P_b)$
4 `/* Recover` G `from` $\tilde{G}$ `*/`
5 **if** $\tilde{Q} \neq P_\infty$ **then** `//` $\tilde{G} = t * \tilde{Q}$, `with` $\tilde{Q} = (\tilde{\gamma} : 1)$
6 $\quad \Gamma \leftarrow \mathrm{roots}(X^p - b^{p-1}X - \tilde{\gamma})$ `//` $b \in B$
7 $\quad G \leftarrow t \sum_{\gamma \in \Gamma} (\gamma : 1)$
8 **else**
9 $\quad G = tP_\infty$
10 **return** G, B

Proposition 3.16. *Algorithm 2 finds the set B and the divisor G in $O(nq^m(q^m+p)^\omega)$ operations in $\mathbb{F}_{q^m}$, where ω is the exponent of the linear algebra.*

Proof. We only prove the cost of the algorithm, its correctness is a consequence of Lemma 3.15 and Proposition 3.13.

The resultant in the variable X of two polynomials in $\mathbb{F}_{q^m}[X, Y]$, is the determinant of a matrix with entries in $\mathbb{F}_{q^m}[Y]$. It can be computed with an "evaluation-interpolation" method, which reduces to compute determinants of scalar matrices and one interpolation. Here the degree of the resultant that we must compute is at most $q^m(p-1)$, so we must compute $q^m(p-1)+1$ determinants with scalar coefficients. The only polynomial to evaluate here is $Y^{(p-1)}$, which can be done by fast exponentiation, so the cost of the evaluation is $O(q^m(p-1)\log_2(p-1))$ operations in $\mathbb{F}_{q^m}$. Computing determinants costs $O((q^m+p)^\omega(q^m(p-1)))$ operations in $\mathbb{F}_{q^m}$, where ω is the exponent of the cost of linear algebra. Then the interpolation costs $(q^m)^2(p-1)^2$ operations in $\mathbb{F}_{q^m}$, using Lagrange interpolation, but this is negligible compared to the cost of the previous determinants.
With Euclid Algorithm we can compute the gcd of two polynomials in $\mathbb{F}_{q^m}[Y]$ of degree at most $q^m(p-1)$ in $O\left(q^{2m}p^2\right)$ operations in $\mathbb{F}_{q^m}$. We compute at most $\frac{n}{p}$ resultants and gcd's, so the cost of the first step is $O(nq^m(q^m+p)^\omega)$ operations in $\mathbb{F}_{q^m}$.

The second step is negligible behind the first step, the computation of the roots of the polynomial $X^p - b^{p-1}X - \tilde{\gamma}$ can be done with Berlekamp algorithm. □

The third step is to recover a support $\mathcal{P}'$ such as $\mathscr{A}_{r,q}(\mathcal{P}', G) = \mathscr{A}_{r,q}(\mathcal{P}, G)$. By Proposition

3.13, we know that a point $P = (x : y)$ in $\mathcal{P}$ satisfies:

$$\begin{cases} x^p - b^{p-1}x - \tilde{\alpha}_i = 0 \\ y^p - \tilde{\beta}_i = 0 \end{cases}$$

for $i \in \{1, \ldots, \frac{n}{\ell}\}$, such that $(\tilde{\alpha}_i : \tilde{\beta}_i) = \tilde{P}_i$. Since we know $\tilde{\mathcal{P}}$, we are able to recover all elements of $\mathcal{P}$ but **as an unordered set**. Hence we know a support $\mathcal{P}'$ such that $\mathscr{A}_{r,q}(\mathcal{P}', G)$ is a permutation of the code $\mathscr{A}_{r,q}(\mathcal{P}, G)$.

3.3.2 Recover the permutation

At this point, the problem is to recover the permutation between $\mathscr{A}_{r,q}(\mathcal{P}', G)$ and $\mathscr{A}_{r,q}(\mathcal{P}, G)$. Let $\mathbf{G}$ be a generator matrix of the code $\mathscr{A}_{r,q}(\mathcal{P}, G)$, and $\mathbf{H}'$ be a parity check matrix of the code $\mathscr{A}_{r,q}(\mathcal{P}', G)$. The permutation between $\mathscr{A}_{r,q}(\mathcal{P}, G)$ and $\mathscr{A}_{r,q}(\mathcal{P}', G)$ is represented by matrix $\mathbf{\Pi}$ such that

$$\mathbf{G} \cdot \mathbf{\Pi} \cdot \mathbf{H}'^{\top} = 0. \quad (3.16)$$

If we had no assumption on the permutation between $\mathcal{P}'$ and $\mathcal{P}$, to find the permutation $\mathbf{\Pi}$ we must resolve a linear system with n^2 unknowns, while (3.16) is a system of $k(n-k)$ linear equations which is not enough to find an unique solution.

Now, we assume that we made the good choice for $a \in \mu_\ell$ (or $b \in B$) in the previous section. Then the permutation matrix $\mathbf{\Pi}$ has the following form:

$$\mathbf{\Pi} = \begin{pmatrix} \sum_{i=1}^{\ell} x_{1,i}\mathbf{J}^i & & (0) \\ & \ddots & \\ (0) & & \sum_{i=1}^{\ell} x_{\frac{n}{\ell},i}\mathbf{J}^i \end{pmatrix} \text{ where } \mathbf{J} := \begin{pmatrix} 0 & \cdots & 0 & 1 \\ 1 & \ddots & \ddots & 0 \\ 0 & \ddots & \ddots & \vdots \\ 0 & \cdots 0 & 1 & 0 \end{pmatrix}. \quad (3.17)$$

$\mathbf{J}$ is an $\ell \times \ell$ matrix, and $x_{j,i} \in \{0, 1\}$ are unknowns, for $j \in \{1, \ldots, \frac{n}{\ell}\}$ and $i \in \{1, \ldots, \ell\}$. With this form, the linear system (3.16) has n unknowns. If we assume that $k \leq n - \frac{1}{n}$, then $n \leq (n-k)k$. In this case, we can hope to find a unique solution for $\mathbf{\Pi}$. In all our computer aided experiments we got a unique solution for $\mathbf{\Pi}$. When the choice for $a \in \mu_\ell$ (or $b \in B$) is wrong, then there was no solution for the system in all our experiments.

We present an algorithm to recover the permutation matrix $\mathbf{\Pi}$ and so the good choice for $a \in \mu_\ell$ (or $b \in B$). The algorithm is only written for the first case, where σ is diagonalizable over $\mathbb{F}_{q^m}$, but the other one is similar.

Proposition 3.17. *Let $\mathscr{C} := \mathscr{A}_{r,q}(\mathcal{P}, G)$ be a QC alternant code stable under a permutation σ. We denote n its length, k its dimension, and ℓ the order of σ. Then Algorithm 3 finds a support $\mathcal{P}'$ such that $\mathscr{A}_{r,q}(\mathcal{P}', G) = \mathscr{C}$ in $O(\ell n^2(n-k)k)$ operations in $\mathbb{F}_{q^m}$.*

Proof. We only prove the cost of the algorithm. We must to resolve a linear system of $(n-k)k$ equations with n unknowns, this is possible in $O(n^2(n-k)k)$ operations in $\mathbb{F}_{q^m}$. This step is repeated at most ℓ times so the cost is in $O(\ell n^2(n-k)k)$ operations in $\mathbb{F}_{q^m}$. □

In order to give practical running times for this part of the attack, we implemented Algorithm 3 in Magma [BCP97]. The platform used in the experiments is a 2.27GHz Intel® Xeon® Processor E5520. For each set of parameters, we give the average time obtained after 10 tests. In the following table we use notation:

- m : extension degree of the field of definition of the support and divisor over $\mathbb{F}_q$
- n : length of the quasi–cyclic code

Algorithm 3: Recover the support

Input : A generator matrix $\mathbf{G}$ of a quasi-cyclic alternant code, the divisor G, and the support $\tilde{\mathcal{P}}$ of the invariant code.

Output: $\emptyset$ if no solution is found. Else, $\mathcal{P}'$ such that $\mathscr{A}_{r,q}(\mathcal{P}', G) = \mathscr{A}_{r,q}(\mathcal{P}, G)$

for $i \in \{1, \dots, \frac{n}{\ell}\}$ **do**
  $\alpha_i \leftarrow \text{roots}(x^\ell - \tilde{\alpha}_i)[1]$ `// cf` (3.15)
  $\beta_i \leftarrow \text{roots}(y^\ell - \tilde{\beta}_i)[1]$
for $a \in \mu_\ell$ **do**
  `/* Guess` $\mathcal{P}'$ `*/`
  $\mathcal{P}' \leftarrow \{(a^j \frac{\alpha_i}{\beta_i} : 1) \mid j \in \{0 \dots \ell - 1\}, i \in \{1 \dots \frac{n}{\ell}\}\}$
  $\mathscr{C} \leftarrow \mathscr{A}_{r,q}(\mathcal{P}', G)$
  if $\mathscr{C} = \mathscr{A}_{r,q}(\mathcal{P}, G)$ **then**
    return TRUE, $\mathcal{P}'$
  else
    $\mathbf{H} \leftarrow \text{ParityCheckMatrix}(\mathscr{C})$
    $S \leftarrow$ **solve**($\mathbf{G} \cdot \mathbf{\Pi} \cdot \mathbf{H}^\top = 0$, with $\mathbf{\Pi}$ a permutation matrix of the form (3.17))
    if $\dim(S) = 1$ **then**
      return $(\pi(\mathcal{P}'))$ `// with` $\pi \in \mathfrak{S}_n$ `associated to` $\mathbf{\Pi}$
return $\emptyset$

- k : dimension of the quasi–cyclic code
- ℓ : order of quasi–cyclicity of the code
- w_{ISD} denotes the logarithm of the work factor for message recovery attacks. It is computed using **CaWoF** library [CT16a].

q	m	n	k	ℓ	w_{ISD}	Algorithm 3
2	12	3600	2825	3	129	1659 s ($\approx$ 27 min)
2	12	3500	2665	5	130	2572 s ($\approx$ 42 min)
2	12	3510	2579	13	132	8848 s ($\approx$ 2h27)

Table 3.1: Average time for Algorithm 3 with σ diagonalizable in $\mathbb{F}_{q^m}$

3.3.3 Case σ diagonalizable in $\mathbb{F}_{q^{2m}} \backslash \mathbb{F}_{q^m}$

In this case, we recall that $\sigma = \rho \circ \sigma_d \circ \rho^{-1}$ with $\rho \in \mathrm{GL}_2(\mathbb{F}_{q^{2m}})$ and:

$$\begin{array}{rccc} \sigma_d\colon & \mathbf{P}^1 & \to & \mathbf{P}^1 \\ & (x:y) & \mapsto & (\alpha x : \alpha^{q^m} y), \end{array}$$

where $\alpha \in \mathbb{F}_{q^{2m}}$ is an ℓ-th root of unity. As σ_d is diagonal in $\mathbb{F}_{q^{2m}}$, we can recover a support $\mathcal{P}'$ and a divisor G' in $\mathbb{F}_{q^{2m}}$, using the same method as in Sections 3.3.1 and 3.3.2.

Now we want to recover a support $\mathcal{P}$ and a divisor G in $\mathbb{F}_{q^m}$. We consider $\pi_\alpha := X + aX + b$ the minimal polynomial of α, with $a, b \in \mathbb{F}_{q^m}$. Then:

$$M_{\sigma_d} = \begin{pmatrix} \alpha & 0 \\ 0 & \alpha^{q^m} \end{pmatrix} \sim \begin{pmatrix} 0 & -b \\ 1 & -a \end{pmatrix} = M_{\sigma'}$$

and there exist $\rho' \in \mathrm{GL}_2(\mathbb{F}_{q^{2m}})$ such that $\sigma_d = \rho' \circ \sigma' \circ \rho'^{-1}$, where σ' is the element of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$ associated to $M_{\sigma'}$. By Lemma 3.9, we can assume that $\sigma = \sigma'$, then we want to recover ρ'. Thanks to Section 3.3.2, we know α and it is easy to compute a and b. To recover ρ' it suffices to diagonalize the matrix $M_{\sigma'}$. From ρ' and a support $\mathcal{P}'$ and a divisor G' in $\mathbb{F}_{q^{2m}}$, we can recover a support $\mathcal{P} = \rho'^{-1}(\mathcal{P}')$ and a divisor $G = \rho'^{-1}(G')$ in $\mathbb{F}_{q^m}$.

3.4 Proposition of a scheme: BIG QUAKE

In this section, we propose a key encapsulation scheme based on binary quasi-cyclic classical Goppa codes. To do this, we propose a public key encryption scheme (PKE) which is converted into a key encapsulation mechanism (KEM) using a generic transformation described in [HHK17]. This transformation permits us to get an IND–CCA2 security, for more details see the NIST submission [BBB+17b]. The PKE used is a Niederreiter-like scheme based on binary QC classical Goppa codes, it is described in Section 3.4.2. Throughout this part, we use classical notation of alternant codes (see Section 1.3.4), in particular the support is a vector of ${\mathbb{F}_{2^m}}^n$ and we speak about multiplier rather than divisor to describe the code.

3.4.1 Quasi-cyclic classical Goppa codes

The Goppa codes are defined in Definition 1.39. In the proposed scheme we use *binary Goppa codes*, that is the base field will be the finite field $\mathbb{F}_2$. In [Ber00b, Ber00a] Berger described several manners to construct ℓ-QC binary classical Goppa codes. Here we use the following manner:

- Let ℓ be a prime dividing $2^m - 1$. Let ζ_ℓ be a primitive ℓ–th root of unity;
- Let n, t be positive integers divisible by ℓ and set $r := \frac{t}{\ell}$;
- The support $\mathbf{x} = (x_0, \ldots, x_{n-1})$ is a vector of elements of $\mathbb{F}_{2^m}$ whose entries are pairwise distinct. It splits into $\frac{n}{\ell}$ blocks of length ℓ of the form $(x_{i\ell}, x_{i\ell+1}, \ldots, x_{(i+1)\ell-1})$ such that for any $j \in \{1, \ldots, \ell-1\}$, $x_{i\ell+j} = \zeta_\ell^j x_{i\ell}$. That is, the support is a disjoint union of orbits under the action of the cyclic group generated by ζ_ℓ. From now on, such blocks are referred to as ζ–**orbits**.
- The Goppa polynomial $\Gamma(z)$ is chosen as $\Gamma(z) = g(z^\ell)$ for some monic polynomial $g \in \mathbb{F}_{2^m}[z]$ of degree $r = \frac{t}{\ell}$ such that $g(z^\ell)$ is irreducible.

Then, the binary Goppa code $\mathscr{G}_2(\mathbf{x}, \Gamma)$ is ℓ-QC.

Remark 8. The construction described above is similar to the description in Section 3.1.3. We saw in Section 1.3.4 that to the support $\mathbf{x} \in \mathbb{F}_{q^m}^n$ and multiplier $\Gamma(\mathbf{x})^{-1}$ we can associate a support $\mathcal{P} \in \mathbb{P}_{\mathbf{P}^1}$ and a divisor G such that $\mathscr{G}_2(\mathbf{x}, \Gamma) = \mathscr{A}_{r,2}(\mathcal{P}, G)$. Moreover, let $\sigma_\xi : x \mapsto \xi x$ be an automorphism of $\mathrm{PGL}_2(\mathbb{F}_{q^m})$. Then the support $\mathcal{P}$ is a disjoint union of orbits under the automorphism σ_ξ and G has a support which is also an union of orbits under the automorphism σ_ξ. That is the support $\mathcal{P}$ and the divisor G have the form described in Section 3.1.3.

3.4.2 The public key encryption scheme (PKE)

In the Key Encapsulation Mechanism (KEM) proposed in the following section we use a public key encryption scheme. This encryption scheme is a Niederreiter-like scheme but avoids the computation of a bijection between words of fixed length and constant weight words. In this scheme we use a hash function, denoted Hash. In practice we use SHA3 but it can be replaced by any secure hash function. Moreover, we assume that this hash function returns a vector in $\mathbb{F}_2^s$ where s is a parameter of the scheme to determine.

Key generation We consider the ℓ–QC binary Goppa code

$$\mathscr{C}_{\text{pub}} := \mathscr{G}_2(\mathbf{x}, \Gamma)$$

of length n and dimension k with ℓ dividing n, k and satisfying Condition 1. Let $\mathbf{H}_{\text{pub}}$ be a systematic parity–check matrix for this code:

$$\mathbf{H}_{\text{pub}} = (\mathbf{I}_{n-k} \mid \mathbf{M}) \tag{3.18}$$

where $\mathbf{M}$ is an ℓ–blocks–circulant matrix (see Definition 3.3). The matrix $\mathbf{M}$ is determined by the set of first rows of each block of length ℓ. Then $\mathbf{H}_{\text{pub}}$ is entirely determined by $\rho(\mathbf{H}_{\text{pub}})$ (Definition 3.4).

- **Public key**: The matrix $\rho(\mathbf{H}_{\text{pub}})$ and the integer t.
- **Private key**: The support $\mathbf{x}$ and the Goppa polynomial $\Gamma(z) = g(z^\ell)$.

Encryption A plaint text $\mathbf{m} \in \mathbb{F}_2^s$, with s a parameter of the scheme, is encrypted as the pair

$$\mathbf{c} := (\mathbf{m} \oplus \mathsf{Hash}(\mathbf{e}),\ \mathbf{H}_{\text{pub}} \cdot \mathbf{e}^\top)$$

where e is a uniformly random word of weight t in $\mathbb{F}_2^n$.

Decryption Denote by $(\mathbf{c}_1,\ \mathbf{c}_2)$ the received pair. Using the private key, one can compute $\mathbf{e} \in \mathbb{F}_2^n$ as the word of weight $\leq t$ such that $\mathbf{c}_2 = \mathbf{H}_{\text{pub}} \cdot \mathbf{e}^\top$. Then, it is easy to compute the message $\mathbf{m} := \mathbf{c}_1 \oplus \mathsf{Hash}(\mathbf{e})$.

3.4.3 Description of the Key Encapsulation Mechanism (KEM)

Context Two participants, Alice and Bob, want to share a common session secret key K. The KEM has three parts. The first part is the key generation computed by Bob. This part is the same as for the PKE described in the above section. Then Bob publishes his public key $(\rho(\mathbf{H}_{\text{pub}}), t)$. His private key is denoted as the pair $(\mathbf{x}, g(z^\ell))$. The second part is the encapsulation mechanism. This part is done by Alice who computes a session key K, encrypts it with Bob's public key and sends, the cyphertext to Bob. Finally the last part is the decapsulation mechanism. Bob decrypts the message sent by Alice and makes a verification. If the verification is correct then he can compute the session key K.

Before to describe the encapsulation and decapsulation mechanisms, we introduce two objects.

- Let s be a positive integer, it denotes the number of bits of security. That is, $s = 128$ (resp. 192, resp. 256) for a 128 (resp. 192, resp. 256) bits security proposal.
- Let $\mathcal{F} : \mathbb{F}_2^s \to \{\mathbf{x} \in \mathbb{F}_2^n \mid w_H(\mathbf{x}) = t\}$ be a function taking a binary vector of length s as input and returning a word of weight t. This function permits to de–randomize the PKE, which is needed for the KEM. The construction of this function is detailed just after the KEM.

Encapsulation. Alice generates a random $\mathbf{m} \in \mathbb{F}_2^s$ and converts it into a word $\mathbf{e} := \mathcal{F}(\mathbf{m})$ of weight t. Then, Alice sends

$$\mathbf{c} := (\mathbf{m} \oplus \mathsf{Hash}(\mathbf{e}),\ \mathbf{H}_{\text{pub}} \cdot \mathbf{e}^\top,\ \mathsf{Hash}(m))$$

to Bob. The session key is defined as:

$$K := \mathsf{Hash}(\mathbf{m}, \mathbf{c}).$$

Decapsulation. We denote $\mathbf{c} = (\mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3)$ the received tuple. Using his private key, Bob can find $\mathbf{e}'$ of weight $\leq t$ such that $\mathbf{c}_2 = \mathbf{H}_{\text{pub}} \cdot \mathbf{e}'^{\top}$. Then Bob computes

$$\mathbf{m}' := \mathbf{c}_1 \oplus \mathsf{Hash}(\mathbf{e}') \text{ and } \mathbf{e}'' = \mathcal{F}(\mathbf{m}').$$

If $\mathbf{e}'' \neq \mathbf{e}'$ or $\mathsf{Hash}(\mathbf{m}') \neq \mathbf{c}_3$ then the algorithm aborts. Else, Bob can compute the session key, that is:

$$K = \mathsf{Hash}(\mathbf{m}', \mathbf{c}).$$

The function $\mathcal{F}$. This function takes a vector of $\mathbb{F}_2^s$ as input and returns a word of length n and weight t. The function depends on the choice of the hash function Hash. In practice we chose SHA3.

The construction of the constant weight word, is performed in constant time using an algorithm close to Knuth shuffle algorithm [Knu97] which generates a uniformly random permutation of the set $\{0, \ldots, n-1\}$. Here, the randomness is replaced by calls of the hash function Hash. The algorithm of evaluation of $\mathcal{F}$ is detailed in Algorithm 4.

Algorithm 4: The function $\mathcal{F}$: construction of a word of weight t

Input : A binary vector $\mathbf{m}$, integers n, t
Output: A word of weight t in $\mathbb{F}_2^n$

1 $\mathbf{u} \leftarrow (0, 1, 2, \ldots, n-2, n-1)$;
2 $\mathbf{b} \leftarrow \mathbf{m}$;
3 **for** i **from** 0 **to** $t-1$ **do**
4 $\quad j \leftarrow \mathsf{Hash}(\mathbf{b}) \mod (n-i-1)$;
5 $\quad$ **Swap** entries u_i and u_{i+j} in u;
6 $\quad \mathbf{b} \leftarrow \mathsf{Hash}(\mathbf{b})$;
7 **end**
8 $\mathbf{e} \leftarrow$ vector with 1's at positions $u_0, \ldots, u_{t-1}$ and 0's elsewhere;
9 **return e**

Further details about line 4 If the hash function Hash outputs 256 or 512 bit strings, this must be converted into a big integer and then reduced modulo $(n-i-1)$. To do this we use the following approach.

Step 1. First, we truncate $\mathsf{Hash}(b)$ to a string of r bytes, where r is larger than the byte size of n. In our proposal, $n < 2^{14}$, hence taking $r = 3$ is reasonable and is the choice we made in practice.

Step 2. Then we convert this r–bytes string to an integer A:

(a) If $A > 2^{8r} - (2^{8r} \mod n-i-1)$ then go to Step 1 (this should be done to assert a uniformity of the drawn integers in $\{0, \ldots, n-i-2\}$);

(b) else set $j = A \mod (n-i-1)$.

3.4.4 Key recovery attacks and countermeasures

In Section 3.3, it has been proved that the key security of quasi-cyclic alternant codes reduces to that of the invariant code. Moreover, Corollary 3.8 shows that the invariant code of a QC alternant code is also an alternant code. In the case of binary quasi-cyclic Goppa code we have the following result.

Theorem 3.18 ([FOP+16a]). *Let $\mathscr{G}_2\left(\mathbf{x}, g(z^\ell)\right)$ be a binary ℓ–QC–Goppa code, and σ the permutation acting on it. Then,*

$$\overline{\mathscr{G}_2\left(\mathbf{x}, g(z^\ell)\right)}^{\sigma} = \mathscr{G}_2\left(Punct_{\mathcal{I}_\ell}\left(\mathbf{x}^\ell\right), g(z)\right)$$

where

$$\mathbf{x}^\ell := (x_0^\ell, x_1^\ell, \ldots, x_{n-1}^\ell)$$

and the set $\mathcal{I}_\ell$ is defined in Definition 3.7.

The above theorem can be also viewed as a consequence of Corollary 3.8. This result is very important to evaluate the key security of the PKE scheme described in Section 3.4.2 and so the key security of the proposed KEM. Indeed, since the invariant code can be constructed in polynomial time from the public key, anybody can compute it. Moreover, this invariant code has a specific structure that is a structure of binary classical Goppa codes. In this section we investigate the different known attacks against binary Goppa code. In order to have a secure scheme we choose parameters of the QC Goppa code so that attacks against the invariant code cost at least 2^s binary operations, where s is the security parameters (see Section 3.4.3).

Exhaustive search on Goppa polynomial and support. Here we detail the exhaustive search on the invariant code $\mathscr{G}_2\left(\mathrm{Punct}_{\mathcal{I}_\ell}\left(\mathbf{x}^\ell\right), g(z)\right)$. A brute force attack could consist in enumerating all the possible polynomials $g(z)$ and then guess the support $\tilde{\mathbf{x}} := \mathrm{Punct}_{\mathcal{I}_\ell}\left(\mathbf{x}^\ell\right)$. If we guessed the good support as a non-ordered set, it is possible to get the good permutation using Sendrier's Support Splitting Algorithm (SSA in short, [Sen00]). If it fails, then try with another support until the good ordering of the support is obtained thanks to SSA. The algorithm for the brute force search on the invariant code can described as follows.

- Perform brute force search among monic irreducible polynomials $g(z)$ of degree r;
- Guess the support $\tilde{\mathbf{x}} = \mathrm{Punct}_{\mathcal{I}_\ell}\left(x_0^\ell, x_1^\ell, \ldots, x_{n-1}^\ell\right)$. Note that the elements of the support set are ℓ–th power. Hence there exists only $\frac{2^m-1}{\ell}$ such powers and we need to guess a good subset of length $\frac{n}{\ell}$ among them.
- Perform SSA to check whether the support set is the good one and, if it is, get the permutation and hence the ordered support;

In order to provide a lower bound of the cost of the brute force search described just above, we estimate the maximum number of guesses of polynomials we need to perform. Indeed, to perform the brute force we need to estimate the number of all possible pairs $(\tilde{\mathbf{x}}, g)$.

To estimate the number of possible polynomial for the invariant code, we need to count the number of monic polynomials $g(z) \in \mathbb{F}_{2^m}[z]$ of degree r such that $g(z^\ell)$ is irreducible. Before this particular case, we recall the well-known formula for the number of monic irreducible polynomials of degree r over $\mathbb{F}_{2^m}$, denoted $m_r(2^m)$:

$$m_r(2^m) = \frac{1}{r}\sum_{d|r} \mu\left(\frac{r}{d}\right) 2^{md},$$

where μ denotes the Möbius function defined by

$$\mu(r) := \begin{cases} 1 & \text{if } r = 1 \\ (-1)^a & \text{if } r = p_1 \ldots p_a, \text{ with } p_i \text{ distinct} \\ 0 & \text{otherwise } (r \text{ is not squarre free}). \end{cases}$$

For the polynomials of interest here, we have the following lemma.

Lemma 3.19. *Let* $s_r(2^m) := \#\{g(z) \in \mathbb{F}_{2^m}[z] \mid \deg(g) = r \text{ and } g(z^\ell) \text{ is irreducible}\}$, *then*

$$s_r(2^m) \geq \left(1 - \frac{1}{\ell}\right) m_r(2^m).$$

Proof. First we notice that, if $g(z^\ell)$ is irreducible, then g is also irreducible. Now we consider a polynomial $g \in \mathbb{F}_{2^m}[z]$ such that $g(z^\ell)$ is reducible. We suppose that $g(z^\ell) = \prod h_i(z)$, with h_i irreducible polynomials over $\mathbb{F}_{2^m}$.

Let ξ be a ℓ-th root of unity in $\mathbb{F}_{2^m}$ and consider the finite group of order ℓ spanned by the map

$$\sigma_\zeta : \begin{array}{ccc} \mathbb{F}_{2^m}[z] & \longrightarrow & \mathbb{F}_{2^m}[z] \\ z & \longmapsto & \zeta z \end{array}.$$

The group $\langle \sigma_\zeta \rangle$ acts on polynomials as $f(z) \mapsto f(\zeta z)$ and so fixes the polynomial $g(z^\ell)$. Hence the polynomials h_i of its irreducible decomposition form orbits under the action of σ_ξ and these orbits have size ℓ since ℓ is prime. Moreover, there is only one orbit otherwise the polynomial g cannot be irreducible. Thus the polynomial $g(z^\ell)$ as the following form:

$$g(z^\ell) = \prod_{i=0}^{\ell-1} h(\zeta^i z) \tag{3.19}$$

for some irreducible polynomial h. The number of polynomials of the form (3.19) is bounded below by $m_r(2^m)/\ell$. This leads to

$$s_r(2^m) \geq \left(1 - \frac{1}{\ell}\right) m_r(2^m).$$

□

Consequently, the number of public keys is bounded below by

$$\#\text{public keys} \geq \#\text{support} \times \left(1 - \frac{1}{\ell}\right) m_r(2^m).$$

We estimate the number of possible supports in estimating the number of possible orbits of size ℓ in $\mathbb{F}_{q^m}$, that is

$$\#\text{support} = \binom{\frac{q^m-1}{\ell}}{\frac{n}{\ell}},$$

where n denotes the length of the code and ℓ the order of quasi-cyclicity. Finally, we have the following bound

$$\#\text{public keys} \geq \binom{\frac{q^m-1}{\ell}}{\frac{n}{\ell}} \left(1 - \frac{1}{\ell}\right) m_r(2^m). \tag{3.20}$$

Algebraic attacks. As we saw in Section 2.4, the point of such an attack is to recover the support and the multiplier of an alternant code. In the case of our proposal, the codes used are binary Goppa codes which belongs in the family of alternant codes. Moreover, the codes used are quasi-cyclic and by Corollary 3.8 we know that the invariant code of a QC alternant is also an alternant code. By Section 3.3, we know that the key security of an QC alternant code reduces to the key security of its invariant code. Then, in the context of algebraic attacks, the key security depends on the cost of algebraic attack against the invariant code.

Distinguisher for high rate Goppa codes. In [FGO$^+$10], it is proved that high rate Goppa codes are distinguishable from random ones in polynomial time. To assert the security of the system, the public Goppa code **and** the invariant code should be indistinguishable from random ones. Hence, the parameters of the code should be chosen in order to be out of the reach of this distinguisher.

The following statement rephrases the results of [FGO$^+$10] in a simpler manner.

Proposition 3.20. *Consider an irreducible binary Goppa code of length n, extension degree m, associated to an irreducible polynomial of degree r. Then the Goppa code is distinguishable in polynomial time from a random code of the same length and dimension as soon as*

$$n > \max_{2 \leq s \leq r} \frac{ms}{2} \left(s(m - 2e - 1) + 2^e + 2 \right),$$

where $e = \lceil \log_2 s \rceil + 1$.

3.4.5 Message recovery attacks and countermeasure

In Section 2.2, we spoke about known generic decoding algorithms which are different variants of the ISD algorithm. These algorithms are described in [Pet10]. In order provide parameters resistant to ISD and their variants we test all the most efficient generic decoding algorithms with the software **CaWoF** [CT16a].

Moreover, there exists a message recovery attack using the ℓ–quasi-cyclicity of the code and in particular the folding operation. This attack is a work in progress, due to J.P. Tillich (see [BBB$^+$17b, Section 4.2.2]). The parameters are chosen in order to avoid this attack.

3.4.6 Suggested parameters

In this section, we propose some parameters for various security levels. We start with informal discussions about the order of quasi-cyclicity ℓ and the extension degree m which explain in which range we choose our parameters. We first investigate the choice of ℓ.

Choice of the quasi-cyclicity order ℓ The key size depends on the quasi-cyclic order ℓ. A large ℓ permits to have a small public key compared to non quasi-cyclic AG codes. The choice of ℓ can also influence the security of the scheme. As we saw in Section 3.2, the security of the public code reduces to the security of the invariant code. We recall that it is always possible to build the invariant code from the knowledge of a generator matrix $\mathbf{G}_{\text{pub}}$ (or a parity check matrix $\mathbf{H}_{\text{pub}}$) of the public code and the permutation σ. Here we want to chose ℓ such that it is not possible to construct another intermediary code. We have two criteria to avoid this.

(i) **ℓ should be prime.** If there exists $s|\ell$, then the permutation σ^s acts on the ℓ-QC code $\mathscr{C}_{\text{pub}}$. That is $\mathscr{C}_{\text{pub}}$ is also a s-quasi-cyclic code and we can construct the invariant code ${\mathscr{C}_{\text{pub}}}^{\sigma^s}$. The permutation $\sigma^s \in \mathfrak{S}_n$ acts on the support and the multiplier and so, by Corollary 3.8, the code ${\mathscr{C}_{\text{pub}}}^{\sigma^s}$ is an alternant code. Thus for any divisor of ℓ, it is possible to construct an intermediary code which is a alternant code smaller than the public code. We do not know if it is easier to attack the public key with the knowledge of several invariant codes. However each intermediary code may provide information about the support and the divisor. We want to give the less information as possible, hence we require ℓ to be prime.

(ii) **ℓ should be such that 2 is an $(\ell - 1)$-th primitive root of 1**, which leads that the polynomial $1 + z + \cdots + z^{\ell-1}$ is irreducible in $\mathbb{F}_2[z]$. Here we describe another way to construct intermediary codes. There exists another code which can be constructed from the knowledge of the public code $\mathscr{C}_{\text{pub}}$ and the permutation σ, it is the "folded" code. We

recall that this folded code, introduced in [FOP+16a], is the image of the public code $\mathscr{C}_{\text{pub}}$ by the map $\mathrm{id} + \sigma + \cdots + \sigma^{\ell-1}$ (see Definition 3.6). Now, if the polynomial $1 + z + \cdots + z^{\ell-1}$ is reducible over $\mathbb{F}_2$, then it is possible to construct an intermediary subcode of $\mathscr{C}_{\text{pub}}$ by computing the image of $\mathscr{C}_{\text{pub}}$ by the map $P(\sigma)$, where P is a divisor of $1 + z + \cdots + z^{\ell-1}$. As in the previous case, the folded code has a special structure related to the support and the multiplier of $\mathscr{C}_{\text{pub}}$ (see [FOP+16a]). Again, we cannot prove that the knowledge of several folded subcodes leads to an attack on $\mathscr{C}_{\text{pub}}$, but we guess that it could be helpful for an attacker.

Remark 9. Among the odd prime numbers below 20, the ones satisfies condition (ii) are

$$\ell \in \{3, 5, 11, 13, 19\}.$$

For these choices of the order, then the only subcode that we can compute is the invariant code.

Remark 10. At several places in the discussion above we suggest that having some data could "help an attacker". We emphasize that these arguments are only precautions, we actually do **not** know how to use such data for cryptanalysis. In particular, the second condition is more a precaution than a necessary condition for the security.

Choice of the field extension m To provide a binary Goppa code, we need to choose a finite extension $\mathbb{F}_{2^m}$ of $\mathbb{F}_2$. The following discussion about the choice of m is informal, that is to say we do not clarify what we mean by *large* or *small*.

(i) A large m provides codes which are "far from" generalized Reed–Solomon codes. Hence, when m is large Goppa codes have less structure. Note that q–ary Goppa codes with $m = 2$ have been broken by a polynomial-time distinguishing and filtration attack in [COT17] and that rather efficient algebraic attacks for small m ($m = 2$ or 3) over non prime q–ary fields exist [FPdP14]. This encourages to avoid too low values of m. In addition, m should be large enough to have a large enough code length.

(ii) On the other hand m should not be too large since it has a negative influence on the rate of the code. That is to say, for a fixed error correcting capacity t an a fixed code length n, the dimension is $n - mt$, hence the rate is $1 - m\frac{t}{n}$.

(iii) Finally, to get ℓ–quasi–cyclic codes, ℓ should divide $2^m - 1$ and ℓ should not be too large to prevent algebraic attacks as [FOPT10, FOP+16b, FOP+16a]. Thus, $2^m - 1$ should have small factors.

In practice. In this proposal we suggest that a good trade-off between (i) and (ii) would be $m \in \{12, \ldots, 18\}$. This choice leads to the following decompositions of $2^m - 1$.

- $\mathbb{F}_{2^{12}} : 2^{12} - 1 = 3^2 \cdot 5 \cdot 7 \cdot 13$.
- $\mathbb{F}_{2^{13}} : 2^{13} - 1$ is prime.
- $\mathbb{F}_{2^{14}} : 2^{14} - 1 = 3 \cdot 43 \cdot 127$.
- $\mathbb{F}_{2^{15}} : 2^{15} - 1 = 7 \cdot 31 \cdot 151$.
- $\mathbb{F}_{2^{16}} : 2^{16} - 1 = 3 \cdot 5 \cdot 17 \cdot 257$.
- $\mathbb{F}_{2^{17}} : 2^{17} - 1$ is prime.
- $\mathbb{F}_{2^{18}} : 2^{18} - 1 = 3^3 \cdot 7 \cdot 19 \cdot 73$.

We can immediately exclude $m = 13$ and 17. To prevent algebraic attacks, we prefer avoiding ℓ's larger than 20 and, as explained above and since we look only for ℓ satisfies condition (ii), the proposal will focus on

- 3, 5 and 13–quasi–cyclic Goppa codes with $m = 12$,
- 3–quasi–cyclic Goppa codes with $m = 14$,
- 5–quasi–cyclic Goppa codes with $m = 16$,
- 19–quasi–cyclic Goppa codes with $m = 18$.

Parameters. In the following tables we use notation

- m : extension degree of the field of definition of the support and Goppa polynomial over $\mathbb{F}_2$;
- n length of the quasi–cyclic code;
- k dimension of the quasi–cyclic code;
- ℓ denotes the order of quasi–cyclicity of the code;
- r denotes the degree of $g(z)$;
- t denotes error–correcting capacity, which is nothing but the degree of $g(z^\ell)$;
- w_{msg} denotes the logarithm of the work factor for message recovery attacks. It is computed using **CaWoF** library;
- Keys is a lower bound for the logarithm of the number of possible keys (see (3.20));
- Max Dreg denotes the maximal degree of regularity that such a system could have in order that the size of the Macaulay matrix does not exceed 2^{128} bits under the assumption that Gaussian elimination's cost on $n \times n$ matrices is $\Omega(n^2)$.

m	n	k	ℓ	Size (bytes)	r	$t = r\ell$ ($\deg g(z^\ell)$)	w_{msg}	Keys	Max Dreg
12	3600	2664	3	103896	26	78	129	1027	8
12	3500	2480	5	63240	17	85	130	684	9
12	3510	2418	13	25389	7	91	132	263	11

Table 3.2: Suggested parameters for security 128

m	n	k	ℓ	Size (bytes)	r	$t = r\ell$ ($\deg g(z^\ell)$)	w_{msg}	Keys	Max Dreg
14	6000	4236	3	311346	42	126	193	5751	11
16	7000	5080	5	243840	24	120	195	6798	12
18	7410	4674	19	84132	8	152	195	2696	16

Table 3.3: Suggested parameters for security 192

m	n	k	ℓ	Size bytes	r	$t = r\ell$ ($\deg g(z^\ell)$)	w_{msg}	Keys	Max Dreg
14	9000	7110	3	559913	45	135	257	6039	14
16	9000	6120	5	440640	36	180	260	8129	15
18	10070	6650	19	149625	10	190	263	3412	20

Table 3.4: Suggested parameters for security 256

Chapter 4

A structural attack on a scheme using quasi-dyadic alternant codes

Contents

4.1 Presentation of the NIST submission: DAGS 70

4.2 Schur products and conductors 72

4.2.1 Schur products of codes 72

4.2.2 Conductors 73

4.2.3 The case of alternant codes 74

4.2.4 The norm-trace code 75

4.3 Using the QD structure to compute a conductor 77

4.4 Description of the attack 78

4.4.1 Brute force search on $\mathscr{D}$ 79

4.4.2 How to recover $\mathscr{D}$ and $\mathscr{NT}(\mathbf{x})$ by solving a system? 80

4.4.3 Finishing the attack 82

4.5 Algorithm, work factor and implementation 84

4.5.1 The first variant of the attack 84

4.5.2 The second variant of the attack 85

The result of this chapter is described in the paper *An efficient structural attack on NIST submission DAGS*, written with Alain Couvreur [BC18].

The purpose of this chapter is to detail a key recovery attack against a McEliece–like encryption scheme using quasi–dyadic (QD) alternant codes with extension degree 2. In particular, this attack can be applied to the NIST submission DAGS [BBB$^+$17a]. Indeed, the proposal DAGS suggests to use a McEliece encryption scheme based on QD generalised Strivastava codes which form a subfamily of QD alternant codes. We describe their proposal in a first part.

To perform the attack, we exploit two properties of the proposed scheme DAGS: the large automorphism group and the small size of the extension degree. The first one permits us to compute explicitly a subcode of the public code, referred to as $\mathscr{D}$. Actually, this code $\mathscr{D}$ is a subcode of the *invariant code*. The second one, permits us to construct, from the code $\mathscr{D}$, a small code, referred to as the *norm trace code*, with a structure close to GRS structure. From this small code, the secret key can easily be recovered. The key step of the attack consists in computing the *conductor* of $\mathscr{D}$ into the public code to build the norm-trace code. We introduce this central operation in Section 4.2.4.

4.1 Presentation of the NIST submission: DAGS

Among the schemes recently submitted to NIST, the submission DAGS [BBB+17a] uses as a primitive a McEliece encryption scheme based on QD generalised Srivastava codes. It is well-known that generalised Srivastava codes form a subclass of alternant codes [MS86, Chapter 12]. Therefore, this proposal lies in the scope of the attack presented in what follows. The authors of the proposal DAGS chose to present quasi-dyadic (QD) generalised Strivastava codes, here we speak about QD alternant codes. First we recall how to construct QD alternant codes. The construction of such a code is also well described in Portzamparc's thesis [dP15, Chapter 5].

Quasi-dyadic alternant codes. Quasi-dyadic (QD) codes are codes over a field of characteristic 2 with an automorphism group $\mathcal{G}$ isomorphic to $(\mathbb{Z}/2\mathbb{Z})^\gamma$, for some $\gamma \in \mathbb{N}^*$. Such a code has length $n = 2^\gamma n_0$ and we call a *block of positions*, 2^γ consecutive positions such that the group $\mathcal{G}$ act on it.

Example 7. The matrix

$$\mathbf{G} := \left(\begin{array}{cc|cc||cc|cc} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right)$$

defined a QD linear code over $\mathbb{F}_2$, with an automorphism group $\mathcal{G}$ isomorphic to $(\mathbb{Z}/2\mathbb{Z})^4$. This code has 2 blocks of positions of length 4: $\{1, \ldots, 4\}$ and $\{5, \ldots, 8\}$.

Similarly to the construction of QC alternant codes in Section 3.1.3, to construct QD alternant code we chose a support and a multiplier such that a group $\mathcal{G}$ acting on it. In the following we denote $\mathbb{F}_{q^m}$ an extension of $\mathbb{F}_q$ where $q = 2^t$ is a power of 2.

Let $\mathcal{G} \subset \mathbb{F}_{q^m}$ be an additive subgroup with γ generators, i.e. $\mathcal{G}$ is an $\mathbb{F}_2$–vector subspace of $\mathbb{F}_{q^m}$ of dimension γ with an $\mathbb{F}_2$–basis $a_1, \ldots, a_\gamma$. Clearly, as an additive group, $\mathcal{G}$ is isomorphic to $(\mathbb{Z}/2\mathbb{Z})^\gamma$. The group $\mathcal{G}$ acts on $\mathbb{F}_{q^m}$ by translation: for any $a \in \mathcal{G}$, we denote by τ_a the translation

$$\tau_a : \begin{array}{ccc} \mathbb{F}_{q^m} & \longrightarrow & \mathbb{F}_{q^m} \\ x & \longmapsto & x + a \end{array}.$$

In order to construct a support stable under the group $\mathcal{G}$, we fix an ordering in $\mathcal{G}$. Using the basis $a_1, \ldots, a_\gamma$, we represent any element $a = u_1 a_1 + \cdots + u_\gamma a_\gamma \in \mathcal{G}$ as a γ-tuple $(u_1, \ldots, u_\gamma) \in (\mathbb{Z}/2\mathbb{Z})^\gamma$. Then we sort them by lexicographic order.

Example 8. If $\gamma = 3$ and $\mathcal{G} := \langle a_1, a_2, a_3 \rangle$, then we have

$$0 < a_1 < a_2 < a_1 + a_2 < a_3 < a_1 + a_3 < a_2 + a_3 < a_1 + a_2 + a_3.$$

Now we can define a support for a QD alternant code. Let $\mathbf{x} \in \mathbb{F}_{q^m}^n$ be a support which splits into n_0 blocks of 2^γ elements of $\mathbb{F}_{q^m}$. Each block is an orbit under the action of $\mathcal{G}$ by translation on $\mathbb{F}_{q^m}$ sorted using the previously described ordering. We say that $\mathbf{x}$ is a *QD support*.

Example 9. If $\gamma = 2$ and $\mathcal{G} := \langle a_1, a_2 \rangle$, then a QD support $\mathbf{x}$ is of the form,

$$\begin{aligned} \mathbf{x} = \quad & (t_1,\ t_1 + a_1,\ t_1 + a_2,\ t_1 + a_1 + a_2, \ldots, \\ & \ldots, t_{n_0},\ t_{n_0} + a_1,\ t_{n_0} + a_2,\ t_{n_0} + a_1 + a_2), \end{aligned} \tag{4.1}$$

where the t_i's are chosen to have disjoint orbits under the action of $\mathcal{G}$ by translation on $\mathbb{F}_{q^m}$.

The chose of a multiplier $\mathbf{y}$ is easier, it suffices to chose a vector in $(\mathbb{F}_{q^m}^*)^n$ which also splits into n_0 blocks of length 2^γ whose entries are equal on a same block. Such a multiplier $\mathbf{y}$ has the form

$$\mathbf{y} = (\underbrace{y_1, \ldots, y_1}_{2^\gamma \text{ times}}, \ldots, y_{n_0}, \ldots, y_{n_0}),$$

where all the y_i are non-zero

Proposition 4.1. *For any positive integer r, the alternant code $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ is quasi–dyadic.*

Proof. See for instance [dP15, Chapter 5]. □

Parameters proposed in DAGS submission. The proposed parameters in [BBB+17a] are listed in Table 4.1.

Name	q	m	n	n_0	k	k_0	γ	r_0
DAGS_1	2^5	2	832	52	416	26	4	13
DAGS_3	2^6	2	1216	38	512	16	5	11
DAGS_5	2^6	2	2112	33	704	11	6	11

Table 4.1: Parameters proposed in DAGS.

Let us remind what parameters $q, m, n, n_0, k, k_0, \gamma, r_0$ stand for:

- q denotes the size of the base field of the alternant code;
- m denotes the extension degree. Hence the GRS code above the alternant code is defined over $\mathbb{F}_{q^m}$;
- n denotes the length of the QD alternant code;
- n_0 denotes the length of the invariant subcode, i.e. $\overline{\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})}^{\mathcal{G}}$, where $\mathcal{G}$ denotes the permutation group.
- k denotes the dimension of the QD alternant code;
- k_0 denotes the dimension of the invariant code;
- γ denotes the number of generators of $\mathcal{G}$, *i.e.* $\mathcal{G} \simeq (\mathbb{Z}/2\mathbb{Z})^\gamma$;
- r_0 denotes the degree of the invariant code, which is alternant according to Corollary 3.8.

Remark 11. The indexes 1, 3 and 5 in the parameters names, in Table 4.1, correspond to security levels according to NIST's call. Level 1, corresponds to 128 bits security with a classical computer, Level 3 to 192 bits security and Level 5 to 256 bits security.

Invariant code of QD code. It turns out the invariant code (see Definition 3.7) of a QD alternant code is an alternant code too. Actually the proof of this result is similar to the quasi-cyclic case in Section 3.2.2. First we need to to recall some basic notions of additive polynomials.

Definition 4.1. An *additive polynomial* $P \in \mathbb{F}_{q^m}[z]$ is a polynomial whose monomials are all of the form z^{2^i} for $i \geq 0$. Such a polynomial satisfies $P(a+b) = P(a) + P(b)$ for any $a, b \in \mathbb{F}_{q^m}$.

Proposition 4.2 ([Gos96, Proposition 1.3.5])**.** *Let $\mathcal{G} \subset \mathbb{F}_{q^2}$ be an additive group of cardinality 2^γ. There exists a unique additive polynomial $\psi_\mathcal{G} \in \mathbb{F}_{q^m}[z]$ which is monic of degree 2^γ and vanishes at any element of $\mathcal{G}$.*

Notation 2. From now on, given an additive subgroup $\mathcal{G} \subseteq \mathbb{F}_{q^m}$, we always denote by $\psi_{\mathcal{G}}$ the unique monic additive polynomial of degree $|\mathcal{G}|$ in $\mathbb{F}_{q^m}[z]$ which vanishes on $\mathcal{G}$.

Then the result about invariant code of QD alternant codes is given by the following theorem.

Theorem 4.3. *Let $\mathscr{C} = \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$ be a QD–alternant code with permutation group $\mathcal{G}$ of order 2^γ. Set $r' = \left\lfloor \frac{r}{2^\gamma} \right\rfloor$. Then,*

$$\overline{\mathscr{C}}^{\mathcal{G}} = \mathscr{A}_{r',q}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}),$$

where

$$\begin{cases} \tilde{\mathbf{x}} := Punct_{\mathcal{I}_\gamma}\left(\psi_{\mathcal{G}}(\mathbf{x})\right) \\ \tilde{\mathbf{y}} := Punct_{\mathcal{I}_\gamma}\left(\mathbf{y}\right) \end{cases}$$

with $\mathcal{I}_\gamma := \{1, \ldots, n\} \setminus \{1, 2^\gamma + 1, \ldots, n - 2^\gamma + 1\}$.

4.2 Schur products and conductors

The component-wise product of two vectors $\mathbf{u}, \mathbf{v} \in \mathbb{F}_q^n$, also called the *Schur product* $\mathbf{u} \star \mathbf{v}$, is defined as:

$$\mathbf{u} \star \mathbf{v} := (u_1 v_1, \ldots, u_n v_n).$$

The i-th power $\mathbf{u} \star \cdots \star \mathbf{u}$ will be denoted by $\mathbf{u}^i$. More generally, given a polynomial $P \in \mathbb{F}_q[z]$ we define $P(\mathbf{u})$ as the vector $(P(u_1), \ldots, P(u_n))$. In particular, the norm and trace from $\mathbb{F}_{q^2}$ to $\mathbb{F}_q$ can be viewed as polynomials and applied component by component to vectors in $\mathbb{F}_{q^2}$. Given $\mathbf{u} \in \mathbb{F}_{q^2}^n$, we denote by $\mathrm{Tr}(\mathbf{u})$ and $\mathrm{N}(\mathbf{u})$ the vectors obtained by applying respectively the trace and the norm map :

$$\begin{aligned} \mathrm{Tr}(\mathbf{u}) &:= (u_1 + u_1^q, \ldots, u_n + u_n^q) \\ \mathrm{N}(\mathbf{u}) &:= (u_1^{q+1}, \ldots, u_n^{q+1}). \end{aligned}$$

The unit vector of the algebra $\mathbb{F}_q^n$ with operations $+$ and $\star$ is the all-ones vector $(1, \ldots, 1)$, denoted by $\mathbf{1}$.

4.2.1 Schur products of codes

The *Schur product* of two codes $\mathscr{A}$ and $\mathscr{B} \subseteq \mathbb{F}_q^n$ is defined as

$$\mathscr{A} \star \mathscr{B} := \langle \mathbf{a} \star \mathbf{b} \mid \mathbf{a} \in \mathscr{A},\ \mathbf{b} \in \mathscr{B} \rangle_{\mathbb{F}_q}.$$

In particular, $\mathscr{A}$ denotes the *square code* of a code $\mathscr{A}$: $\mathscr{A}^2 := \mathscr{A} \star \mathscr{A}$.

The case of GRS codes. Since GRS codes have more structure that alternant codes, it seems more convenient to study first the GRS codes. The behaviour of GRS codes with respect to the Schur product is very different from that of random codes. Indeed the following theorem shows that the set of GRS codes of fixed support is stable by Schur product.

Theorem 4.4 ([CGG+14, Proposition 6])**.** *Let $\mathbf{x} \in \mathbb{F}_{q^m}^n$ be a support and $\mathbf{y}, \mathbf{y}' \in \mathbb{F}_{q^m}^n$ be multipliers. Let k, k' be two positive integers, then*

$$\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) \star \mathbf{GRS}_{k'}(\mathbf{x}, \mathbf{y}') = \mathbf{GRS}_{k+k'-1}(\mathbf{x}, \mathbf{y} \star \mathbf{y}').$$

4.2.2 Conductors

In this section, we define the *conductor* of two codes. This operation is central in the attack on the scheme DAGS.

Definition 4.2. Let $\mathscr{C}$ and $\mathscr{D}$ be two codes of length n over $\mathbb{F}_q$. The *conductor of $\mathscr{D}$ into $\mathscr{C}$* is defined as the largest code $\mathscr{Z} \subseteq \mathbb{F}_q^n$ such that $\mathscr{D} \star \mathscr{Z} \subseteq \mathscr{C}$. That is:

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}) := \{\mathbf{u} \in \mathbb{F}_q^n \mid \mathbf{u} \star \mathscr{D} \subseteq \mathscr{C}\}.$$

Let us give an explicit formula to compute the conductor. The following proposition will be very useful in the main step of the attack.

Proposition 4.5 ([CMCP17, COT17])**.** *Let $\mathscr{D}, \mathscr{C} \subseteq \mathbb{F}_q^n$ be two codes, then*

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}) = \left(\mathscr{D} \star \mathscr{C}^\perp\right)^\perp.$$

Remark 12. We notice that the conductor of a code $\mathscr{C}_1$ into a code $\mathscr{C}_2$ can be computed from the knowledge of a generator matrix for each code.

As in §4.2.1, before discussing behaviour of alternant codes, we provide a result about GRS codes.

Proposition 4.6. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^m}^n$ be a support and a multiplier. Let $k \leq k'$ be two integers less than n. Then,*

$$\mathbf{Cond}(\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}), \mathbf{GRS}_{k'}(\mathbf{x}, \mathbf{y})) = \mathbf{RS}_{k'-k+1}(\mathbf{x}).$$

Proof. From Proposition 4.5, the dual of the above conductor is a Schur product between to GRS codes. Then, by Proposition 1.34 and Theorem 4.4, we have

$$\begin{aligned}\mathbf{Cond}(\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}), \mathbf{GRS}_{k'}(\mathbf{x}, \mathbf{y}))^\perp &= \mathbf{GRS}_k(\mathbf{x}, \mathbf{y}) \star \mathbf{GRS}_{n-k'}(\mathbf{x}, \mathbf{y}^\perp) \\ &= \mathbf{GRS}_{n-k'+k-1}(\mathbf{x}, \mathbf{y} \star \mathbf{y}^\perp).\end{aligned}$$

Note that

$$\mathbf{y} \star \mathbf{y}^\perp = \left(\frac{1}{\pi'_{\mathbf{x}}(\mathbf{x})(x_1)}, \ldots, \frac{1}{\pi'_{\mathbf{x}}(\mathbf{x})(x_n)}\right).$$

Then, using again Proposition 1.34, we get

$$\mathbf{Cond}(\mathbf{GRS}_k(\mathbf{x}, \mathbf{y}), \mathbf{GRS}_{k'}(\mathbf{x}, \mathbf{y})) = \mathbf{GRS}_{k'-k+1}(\mathbf{x}, (\mathbf{y} \star \mathbf{y}^\perp)^\perp) = \mathbf{RS}_{k'-k+1}(\mathbf{x}).$$

□

In Proposition 4.6, we notice that the conductor of two GRS codes with same support $\mathbf{x}$ and multiplier $\mathbf{y}$ **does not depend on** $\mathbf{y}$. This point is important since the cryptanalysis of a Reed-Solomon code is easier than that of a GRS code. Moreover, if we can extract $\mathbf{x}$ then we know that recovering $\mathbf{y}$ from the knowledge of a generator matrix of the public code can be done with linear algebra. The crucial step in our attack is to provide a "good" conductor, that is a conductor with a simple structure from which one can extract the support $\mathbf{x}$. In the case of GRS codes, it is easy to provide such a conductor from the knowledge of a GRS subcode of codimension 1 in the public code. Indeed, by Proposition 4.6, the computed conductor will always be a Reed-Solomon code. Let us give an illustrative example which is inspired of [COT17].

Illustrative example with GRS codes. Let $\mathbf{x}, \mathbf{y}$ be a support and a multiplier in $\mathbb{F}_{q^2}^n$. The pair $(\mathbf{x}, \mathbf{y})$ is **secret**. Let $k < n$ be a positive integer, we consider the code $\mathscr{C} := \mathbf{GRS}_k(\mathbf{x}, \mathbf{y})$. A generator matrix of $\mathscr{C}$ is **public**. Now we assume that we also know a generator matrix of the code $\mathscr{C}_{-1} := \mathbf{GRS}_{k-1}(\mathbf{x}, \mathbf{y})$. By Proposition 4.6, we have:

$$\begin{aligned}\mathbf{Cond}(\mathbf{GRS}_{k-1}(\mathbf{x}, \mathbf{y}), \mathbf{GRS}_k(\mathbf{x}, \mathbf{y})) &= \mathbf{RS}_2(\mathbf{x}) \\ &= \langle \mathbf{1}, \mathbf{x} \rangle_{\mathbb{F}_{q^2}}.\end{aligned}$$

Then from $\mathbf{Cond}(\mathscr{C}_{-1}, \mathscr{C})$ it is easy to recover a support $\mathbf{x}'$ and a multiplier $\mathbf{y}'$ such that $\mathscr{C} = \mathbf{GRS}_k(\mathbf{x}', \mathbf{y}')$ (see Section 4.4.3 for more details). However there is no reason to be able to compute a generator matrix of the code $\mathbf{GRS}_{k-1}(\mathbf{x}, \mathbf{y})$. In [COT17], the authors propose to use another subcode of $\mathscr{C}$ of codimension 1. In our case, the first difficulty comes from the fact that we have subfield subcodes of GRS codes. We will see in the following section that it is more difficult to provide information on the support $\mathbf{x}$ with conductor of alternant codes. However, the quasi-dyadic structure permits us to construct a subcode whose the conductor into the public code has a structure revealing the support $\mathbf{x}$.

4.2.3 The case of alternant codes

Since an alternant code is a subfield subcode of a GRS code, we hope that the behaviour of conductor of alternant codes is similar to the case of GRS codes. Unfortunately when dealing with alternant codes, proving equalities becomes difficult. We can at least prove an attenuated form of Proposition 4.6.

Theorem 4.7. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^2}^n$ be a support and a multiplier. Let $r' \geq r$ be two positive integers. Then,*

$$\mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n \subseteq \mathbf{Cond}(\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y}), \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})).$$

Proof. Consider the Schur product

$$\begin{aligned}\left(\mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n\right) &\star \mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y}) \\ &= \left(\mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n\right) \star \left(\mathbf{GRS}_{n-r'}(\mathbf{x}, \mathbf{y}^\perp) \cap \mathbb{F}_q^n\right) \\ &\subseteq \left(\mathbf{RS}_{r'-r+1}(\mathbf{x}) \star \mathbf{GRS}_{n-r'}(\mathbf{x}, \mathbf{y}^\perp)\right) \cap \mathbb{F}_q^n.\end{aligned}$$

Next, using Theorem 4.4,

$$\begin{aligned}\left(\mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n\right) \star \mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y}) &\subseteq \mathbf{GRS}_{n-r}(\mathbf{x}, \mathbf{y}^\perp) \cap \mathbb{F}_q^n \\ &\subseteq \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}).\end{aligned}$$

The last inclusion is a direct consequence of Proposition 1.34 and Definition 1.37. □

Remark 13. We cannot extends directly the result of Proposition 4.6 to alternant codes. However, it turns out that equality frequently holds in Theorem 4.7.

On the other hand, if the degree r' of the subcode $\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ is not large enough then the code $\mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n$ can be trivial. This is illustrate by the following example.

Illustrative example with alternant codes. As in the context of the example of §4.2.2, we assume that we know the subcode $\mathscr{A}_{r+1,q}(\mathbf{x}, \mathbf{y})$ of $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$. Then, by Theorem 4.7

$$\mathbf{Cond}(\mathscr{A}_{r+1,q}(\mathbf{x}, \mathbf{y}), \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})) \supseteq \mathbf{RS}_2(\mathbf{x}) \cap \mathbb{F}_q^n.$$

Now assume that the equality holds, then

$$\mathbf{Cond}(\mathscr{A}_{r+1,q}(\mathbf{x}, \mathbf{y}), \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})) = \mathbf{RS}_2(\mathbf{x}) \cap \mathbb{F}_q^n.$$

However, if the vector $\mathbf{x}$ has one entry in $\mathbb{F}_{q^2}\backslash\mathbb{F}_q$ then the code $\mathbf{RS}_2(\mathbf{x}) \cap \mathbb{F}_q^n$ provides no information about the support $\mathbf{x}$. Indeed this code is trivial, that is $\mathbf{RS}_2(\mathbf{x}) \cap \mathbb{F}_q^n = \langle \mathbf{1} \rangle_{\mathbb{F}_q}$. To avoid this case, we need to consider an alternant of much larger codimension in $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$, ie: with a higher degree.

4.2.4 The norm-trace code

The strategy of the attack consists in recovering a particular code which does not depend on the multiplier $\mathbf{y}$ in order to obtain a code which is easier to study. This code will be determined by the computation of a conductor of a subcode $\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ into $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$. By Theorem 4.7, a good candidate for this particular code is a subfiled subcode of a Reed-Solomon code. At this point, the main problem is to find a small enough subfiled subcode of a Reed-Solomon code which is not trivial. Once this is done, we extract information about $\mathbf{x}$. Let us now investigate further the structure of the code $\mathbf{RS}_k(\mathbf{x}) \cap \mathbb{F}_q^n$, for an integer $k < n$. Since the subfield subcode operation keeps only words with all components in $\mathbb{F}_q$, we know that if the dimension of the RS code is high enough then $\mathrm{Tr}(\mathbf{x})$ will be in the code $\mathbf{RS}_k(\mathbf{x}) \cap \mathbb{F}_q^n$. More precisely, if $k \geq q+1$ then $\mathrm{Tr}(\mathbf{x}) \in \mathbf{RS}_k(\mathbf{x}) \cap \mathbb{F}_q^n$ and so the code is not trivial.

Actually the code that we recover is generated by the trace and the norm of the support vector $\mathbf{x}$. We call this code the *norm-trace code* and it is defined as follows. Later on, we consider a $\mathbb{F}_q$-basis of $\mathbb{F}_{q^2}$ denoted by $(1, \alpha)$, where $\alpha \in \mathbb{F}_{q^2}$ is such that $\mathrm{Tr}(\alpha) = 1$.

Definition 4.3 (Norm trace code)**.** Let $\mathbf{x} \in \mathbb{F}_{q^2}^n$ be a support. The *norm–trace code* $\mathscr{NT}(\mathbf{x}) \subseteq \mathbb{F}_q^n$ is defined as

$$\mathscr{NT}(\mathbf{x}) := \langle \mathbf{1}, \mathrm{Tr}(\mathbf{x}), \mathrm{Tr}(\alpha\mathbf{x}), \mathrm{N}(\mathbf{x}) \rangle_{\mathbb{F}_q}.$$

Under the assumption that $\mathbf{x}$ is full, we can prove that the norm-trace code has a structure of subfield subcode of Reed-Solomon code. In the general case we have only the following lemma.

Lemma 4.8. *Let $\mathbf{x} \in \mathbb{F}_{q^2}^n$ be a support. Then, for any $k > q+1$, we have*

$$\mathscr{NT}(\mathbf{x}) \subseteq \mathbf{RS}_k(\mathbf{x}) \cap \mathbb{F}_q^n.$$

Proof. Since $k > q+1$, $\mathrm{Tr}(\mathbf{x}) = \mathbf{x} + \mathbf{x}^q$, $\mathrm{Tr}(\alpha\mathbf{x})$ and $\mathrm{N}(\mathbf{x}) = \mathbf{x}^{q+1}$ are codewords of $\mathbf{RS}_k(\mathbf{x})$. □

Remark 14. We observed experimentally that for $2q+1 > k > q+1$ the above inclusion is in general an equality.

The link with conductors of alternant codes The purpose of the attack is to provide information on the support $\mathbf{x}$ from only the computation of a conductor. If we know a subcode $\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ of $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$, such that $q < r' - r$, then by Theorem 4.7 and Lemma 4.8 we have:

$$\mathscr{NT}(\mathbf{x}) \subseteq \mathbf{Cond}(\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y}), \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})).$$

Moreover, experimentally (see Remark 13 and 14), we have almost every time

$$\mathscr{NT}(\mathbf{x}) = \mathbf{Cond}(\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y}), \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})). \tag{4.2}$$

The most difficult part remains to find a code $\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ with $q < r' - r$. However, the quasi-dyadic structure of the public code permits us to construct a subcode $\mathscr{D} \subseteq \mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ with $q < r' - r$. That is why, we will use the following heuristic which extends (4.2). This heuristic is discussed just after the wording.

Heuristic 4.9. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^2}^n$ be a support and a multiplier, and $r' \geq r$ be two positive integers such that $q < r' - r < 2q$. Suppose that $n \geq 4q$. Let $\mathscr{D}$ be a subcode of $\mathscr{A}_{r',q}(\mathbf{x}, \mathbf{y})$ such that*

(i) $\dim \mathscr{D} \cdot \dim \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp} \geq n$*;*

(ii) $\mathscr{D} \not\subset \mathscr{A}_{r'+1,q}(\mathbf{x},\mathbf{y})$*;*

(iii) a generator matrix of $\mathscr{D}$ *has no zero column.*

Then, with high probability we have

$$\mathbf{Cond}(\mathscr{D}, \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})) = \mathbf{RS}_{r'-r}(\mathbf{x}) \cap \mathbb{F}_q^n$$

This heuristic extends the result of Theorem 4.7. Let us give some explanations about this heuristic. From Proposition 4.5,

$$\mathbf{Cond}(\mathscr{D}, \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})) = \left(\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp}\right)^{\perp}.$$

Next, from Delsarte theorem (Theorem 1.7), we have

$$\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp} = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}(\mathbf{GRS}_r(\mathbf{x},\mathbf{y})).$$

Since $\mathscr{D}$ is a code over $\mathbb{F}_q$ and by the $\mathbb{F}_q$–linearity of the trace map, we get

$$\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp} = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathscr{D} \star \mathbf{GRS}_r(\mathbf{x},\mathbf{y})\right).$$

Now, we use the fact that $\mathscr{D}$ is contained in $\mathscr{A}_{r',q}(\mathbf{x},\mathbf{y})$ and hence is a subset of a GRS code. Namely,

$$\mathscr{D} \subset \mathbf{GRS}_{n-r'}(\mathbf{x},\mathbf{y}^{\perp}), \quad \text{where} \quad \mathbf{y}^{\perp} = \left(\frac{1}{\pi'_{\mathbf{x}}(x_1)y_1}, \ldots, \frac{1}{\pi'_{\mathbf{x}}(x_n)y_n}\right).$$

Therefore, thanks to Theorem 4.4, we get

$$\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp} \subseteq \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathbf{GRS}_{n-r'+r-1}(\mathbf{x},\mathbf{y}\star\mathbf{y}^{\perp})\right). \tag{4.3}$$

Here, let us note that $\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp}$ is spanned by $\dim \mathscr{D} \cdot \dim \mathscr{A}_{r,\mathbf{x}}(\mathbf{y},) \perp$ generators which are obtained by computing the Schur products of elements of a basis of $\mathscr{D}$ by elements of a basis of $\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp}$. By (i), the number of such generators exceeds n. For this reason, it is very reasonable to hope that this Schur product will fill in the target code and hence that actually,

$$\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp} = \mathrm{Tr}_{\mathbb{F}_{q^2}/\mathbb{F}_q}\left(\mathbf{GRS}_{n-r'+r-1}(\mathbf{x},\mathbf{y}\star\mathbf{y}^{\perp})\right).$$

Next, we have

$$\mathbf{y}\star\mathbf{y}^{\perp} = \left(\frac{1}{\pi'_{\mathbf{x}}(x_1)}, \ldots, \frac{1}{\pi'_{\mathbf{x}}(x_n)}\right).$$

Therefore, using Proposition 1.34, we conclude that

$$\left(\mathscr{D} \star \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\perp}\right)^{\perp} = \mathbf{RS}_{r'-r+1}(\mathbf{x}) \cap \mathbb{F}_q^n.$$

Remark 15. Assumption (ii) permits to avoid the situation where the conductor could be the subfield subcode of a larger Reed–Solomon code. Assumption (iii) permits to avoid the presence of words of weight 1 in the conductor that would not be elements of a Reed-Solomon code.

4.3 Using the QD structure to compute a conductor

The purpose of the attack is to find a subcode $\mathscr{D}$ of $\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})$ such that Heuristic 4.9 is satisfied. The knowledge of a such code $\mathscr{D}$, permits us to recover the support $\mathbf{x}$. We recall that, we need a gap of codimension of more than q to catch the norm-trace code. In the general case, there is no reason to be able to compute subcodes of such a large codimension. In the present case, it is possible to compute some subcode with the good codimension from invariant codes. By Theorem 4.3, we already know that the invariant code of a QD alternant code has a particular structure. The following statement, which is crucial for our attack, completes the result of this theorem.

Theorem 4.10. *Let $\mathbf{x},\mathbf{y} \in \mathbb{F}_{q^2}$ be a support and a multipier, and s be an integer of the form $s = 2^\gamma s_0$. Let $\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})$ be a QD alternant code stable by the action of a group $\mathcal{G}$, where $|\mathcal{G}| = 2^\gamma$. Suppose that $\overline{\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})}^{\mathcal{G}}$ is fully non degenerate (see Definition 1.38). Then,*

(a) $\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}} \subseteq \mathscr{A}_{s+|\mathcal{G}|-1,q}(\mathbf{x},\mathbf{y})$;

(b) $\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}} \not\subseteq \mathscr{A}_{s+|\mathcal{G}|,q}(\mathbf{x},\mathbf{y})$.

Proof. We know that:

$$\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y}) = \left\{ \left(\frac{1}{y_i \pi'_{\mathbf{x}}(x_i)} f(x_i) \right)_{i=1,\dots,n} \;\middle|\; f \in \mathbb{F}_{q^2}[z]_{<n-s} \right\} \cap \mathbb{F}_q^n.$$

This code is obtained by evaluation of polynomials of degree up to

$$n - s - 1 = (2^\gamma(n_0 - s_0) - 1).$$

Next, from Theorem 4.3 and Proposition 4.2, the invariant codewords of $\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})$ come from evaluations of polynomials of the form $h \circ \psi_{\mathcal{G}}$. Such polynomials have a degree which is a multiple of $\deg \psi_{\mathcal{G}} = 2^\gamma$ and hence their degree cannot exceed $2^\gamma(n_0 - s_0 - 1)$. Thus, they should lie in $\mathbb{F}_{q^2}[z]_{\leq n-s-|\mathcal{G}|} = \mathbb{F}_{q^2}[z]_{<n-s-|\mathcal{G}|+1}$. This leads to

$$\begin{aligned}\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}} &\subseteq \left\{ \left(\frac{1}{y_i \pi'_{\mathbf{x}}(x_i)} f(x_i) \right)_{i=1,\dots,n} \;\middle|\; f \in \mathbb{F}_{q^2}[z]_{<n-s-|\mathcal{G}|+1} \right\} \cap \mathbb{F}_q^n \\ &\subseteq \mathscr{A}_{s+|\mathcal{G}|-1,q}(\mathbf{x},\mathbf{y}).\end{aligned}$$

To prove (b), note that the assumption on $\mathscr{A}_{s_0,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}}$ asserts the existence of $f \in \mathbb{F}_{q^2}[z]_{<n_0-s_0}$ such that $\deg f = n_0 - s_0 - 1$ and $f(\psi_{\mathcal{G}}(\mathbf{x})) \in \mathbb{F}_q^n$. Thus, $f(\psi_{\mathcal{G}}(\mathbf{x})) \in \mathbb{F}_q^n$ and $\deg(f \circ \psi_{\mathcal{G}}) = n - s - |\mathcal{G}|$. Therefore $f(\psi(\mathbf{x})) \in \mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}}$ and $\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}}$ contains an element of $\mathscr{A}_{s+|\mathcal{G}|-1,q}(\mathbf{x},\mathbf{y})$ which is not in $\mathscr{A}_{s+|\mathcal{G}|,q}(\mathbf{x},\mathbf{y})$. □

This result shows that the invariant code is contained in a smaller alternant code, i.e. a subfield subcode of a code corresponding to the evaluation of polynomials of lower degree. Statement (b) is useful to constrain the dimension of the conductor $\mathbf{Cond}(\mathscr{A}_{s,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}}, \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y}))$. This is avoid to have a code larger than $\mathbf{RS}_{s+|\mathcal{G}|-r}(\mathbf{x}) \cap \mathbb{F}_q^n$, which is the expected code under the assumptions of Heuristic 4.9.

Remark 16. In the case where $q < |\mathcal{G}|$, which never holds in DAGS suggested parameters, we have a particularly simple manner to compute $\mathscr{NT}(\mathbf{x})$. In such a situation, replacing possibly $\mathcal{G}$ by a subgroup, one can suppose that $|\mathcal{G}| = 2q$. Then, according to Theorem 4.10 and Heuristic 4.9, we have

$$\mathbf{Cond}(\mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})^{\mathcal{G}}, \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})) = \mathbf{RS}_{|\mathcal{G}|-1}(\mathbf{x}) \cap \mathbb{F}_q^n$$

By Remark 14, we get the norm-trace code.

From now on $q \geq |\mathcal{G}|$, let us give the definition of the code $\mathscr{D}$ that we will look for.

Definition 4.4. Suppose that $|\mathcal{G}| \leq q$. We define the code $\mathscr{D}$ as

$$\mathscr{D} := \mathscr{A}_{r+q,q}(\mathbf{x}, \mathbf{y})^{\mathcal{G}}.$$

Assuming that $\overline{\mathscr{A}_{r+q,q}(\mathbf{x}, \mathbf{y})}^{\mathcal{G}}$ is fully non degenerate, by Theorem 4.10, the code $\mathscr{D}$ satisfies the assumptions of Heuristic 4.9. Hence, under Heuristics 4.9, we have

$$\mathbf{Cond}(\mathscr{D}, \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})) = \mathbf{RS}_{q+|\mathcal{G}|-1}(\mathbf{x}) \cap \mathbb{F}_q^n.$$

However, when starting the attack, the code $\mathscr{D}$ is unknown. The following property will be useful to find $\mathscr{D}$.

Proposition 4.11. *The code $\mathscr{D}$ has codimension $\leq \frac{2q}{|\mathcal{G}|}$ in $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})^{\mathcal{G}}$.*

Proof. Using Theorem 4.3, we know that $\mathscr{D}$ has the same dimension as $\mathscr{A}_{r_0+\frac{q}{|\mathcal{G}|},q}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}})$, for some $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{y}}$. This code has dimension $\geq n_0 - 2(r_0 + \frac{q}{|\mathcal{G}|})$. Since $\dim \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})^{\mathcal{G}} = k_0 = n_0 - 2r_0$, we get the result. □

Remark 17. Actually the codimension equals $\frac{2q}{|\mathcal{G}|}$ almost all the time.

4.4 Description of the attack

We recall that the extension degree is always $m = 2$ and we set $q := 2^\ell$ for some positive integer ℓ. The public code is a QD alternant code defined by:

$$\mathscr{C}_{\text{pub}} := \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}).$$

The permutation group acting on $\mathscr{C}_{\text{pub}}$ is $\mathcal{G}$ with cardinality $|\mathcal{G}| = 2^\gamma$. The code has length $n = n_0 2^\gamma$, dimension k and is defined over a field $\mathbb{F}_q$. The degree r of the alternant code is also a multiple of $|\mathcal{G}| = 2^\gamma$ and hence is of the form $r = r_0 2^\gamma$. We suppose from now on that the lower bound on the dimension k given by Proposition 1.35 is reached. Namely that $k = n - 2r$. This always holds in the parameters proposed in [BBB+17a]. We finally set $k_0 = \frac{k}{2^\gamma}$. In summary, we have the following notation:

$$n = n_0 2^\gamma, \quad k = k_0 2^\gamma, \quad r = r_0 2^\gamma. \tag{4.4}$$

We can describe the attack in three steps, as follows:

(1) Compute $(\mathscr{C}_{\text{pub}})^{\mathcal{G}}$;

(2) Guess the subcode $\mathscr{D}$ of $(\mathscr{C}_{\text{pub}})^{\mathcal{G}}$ of codimension $\frac{2q}{|\mathcal{G}|}$ such that

$$\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\text{pub}}) = \mathbf{RS}_{q+|\mathcal{G}|-1}(\mathbf{x}) \cap \mathbb{F}_q^n;$$

(3) Determine $\mathbf{x}$ from $\mathscr{NT}(\mathbf{x})$ and then $\mathbf{y}$ from $\mathbf{x}$.

The first step can be done with the computation of the kernel of a map (see Definition 3.7). The second one is the most expensive part. A first way to guess $\mathscr{D}$ and than compute $\mathscr{NT}(\mathbf{x})$ consists in performing exhaustive search on subcodes of codimension $\frac{2q}{|\mathcal{G}|}$ of ${\mathscr{C}_{\text{pub}}}^{\mathcal{G}}$. We give the complexity of this approach and then we present two manners to reduce the cost of brute force search. A second approach consists in getting both $\mathscr{D}$ and $\mathscr{NT}(\mathbf{x})$ by solving a polynomial system using Gröbner bases. For this approach, we did not succeed to get a relevant estimate of the complexity. However, its practical implementation permits to break the instances of `DAGS_1` and `DAGS_5` in few minutes but not `DAGS_3`. We give details in §4.5.2. Finally, the last step is easy and actually is negligible compared to the second one (see §4.5.1).

4.4.1 Brute force search on $\mathscr{D}$

To perform a brute search on $\mathscr{D}$ consists in enumerating all the subspaces $\mathscr{X} \subseteq (\mathscr{C}_{\text{pub}})^{\mathcal{G}}$ of codimension $\frac{2q}{|\mathcal{G}|}$ until we find one such that $\mathbf{Cond}(\mathscr{X}, \mathscr{C}_{\text{pub}})$ is not trivial, that is its dimension is not 1. In general, if $\mathbf{Cond}(\mathscr{X}, \mathscr{C}_{\text{pub}})$ is not trivial its dimension is 4. Indeed, for an arbitrary $\mathscr{X}$ the conductor will have no particular structure and we can suppose that it is generated by $\mathbf{1}$, so its dimension is 1. In contrast, for $\mathscr{X} = \mathscr{D}$ the conductor will be $\mathbf{RS}_{q+|\mathcal{G}|-1}(\mathbf{x}) \cap \mathbb{F}_q^n$ which typically has dimension 4 by Remark 14. The number of subspaces to test is in general much too large to make the search practical. Indeed we need to enumerate $O\big(q^{\frac{2q}{|\mathcal{G}|}(k_0 - \frac{2q}{|\mathcal{G}|})}\big)$ subspaces. In the DAGS case, this number is out of reach. However, it is possible to reduce the cost of the brute force. The two manners presented here are very similar in terms of complexity.

Picking random subcodes of codimension 2. Here we use Heuristic 4.9 with a random subcode $\mathscr{D}_0$ of $\mathscr{D}$ of smaller dimension. For any parameter proposed in DAGS, the code ${\mathscr{C}_{\text{pub}}}^{\perp}$ has rate lager that $\frac{1}{2}$. Therefore, it suffices to take a random subcode $\mathscr{D}_0$ of dimension 2, to satisfies the assumption (i) of Heuristic 4.9. Then, with a high probability, we have $\mathbf{Cond}(\mathscr{D}_0, \mathscr{C}_{\text{pub}}) = \mathscr{NT}(\mathbf{x})$. We do not know the code $\mathscr{D}$ so we cannot directly pick random subcodes of $\mathscr{D}$. We know that $\mathscr{D} \subseteq {\mathscr{C}_{\text{pub}}}^{\mathcal{G}}$ and then we can pick two independent vectors $\mathbf{c}, \mathbf{c}'$ in ${\mathscr{C}_{\text{pub}}}^{\mathcal{G}}$ and compute the conductor $\mathbf{Cond}(\langle \mathbf{c}, \mathbf{c}' \rangle, \mathscr{C}_{\text{pub}})$. If the conductor has dimension 4, we can assume that we found the code $\mathscr{NT}(\mathbf{x})$, else we try again with another pair $\mathbf{c}, \mathbf{c}'$. The probability that $\mathbf{c}, \mathbf{c}' \in \mathscr{D}$ equals $q^{-\frac{4q}{|\mathcal{G}|}}$. Therefore, one may have found $\mathscr{NT}(\mathbf{x})$ after $O(q^{\frac{4q}{|\mathcal{G}|}})$ computations of conductors.

Shortening codes. Another way consists in performing the brute force on a shortened code. For that we replace the public code and the unknown code $\mathscr{D}$ by one of their shortenings. In order to keep the quasi-dyadic structure we choose a set $\mathcal{I}$ of $a := a_0 2^\gamma$ positions which is a union of blocks (see Section 4.1). The integer a must be chosen such that the shortening of $\mathscr{D}$ has dimension 2. Actually, it suffices to chose a such that the dimension of the invariant code $\text{Short}_{\mathcal{I}}(\mathscr{C}_{\text{pub}})^{\mathcal{G}}$ is $2 + \frac{2q}{|\mathcal{G}|}$. To determine $\text{Short}_{\mathcal{I}}(\mathscr{D})$, we can enumerate any subspace $\mathscr{X} \subseteq \text{Short}_{\mathcal{I}}(\mathscr{C}_{\text{pub}})^{\mathcal{G}}$ of dimension 2 and compute $\mathbf{Cond}(\mathscr{X}, \text{Short}_{\mathcal{I}}(\mathscr{C}_{\text{pub}}))$. In general, we get the trivial code spanned by the all–one codeword $\mathbf{1}$. If the conductor has dimension 4 it is highly likely that we found $\text{Short}_{\mathcal{I}}(\mathscr{D})$. In this case the conductor will be $\text{Punct}_{\mathcal{I}}\left(\mathbf{RS}_{q+|\mathcal{G}|-1}(\mathbf{x})\right) = \text{Punct}_{\mathcal{I}}\left(\mathscr{NT}(\mathbf{x})\right) = \mathscr{NT}(\mathbf{x}_{\mathcal{I}})$. We need to enumerate $O(q^{\frac{4q}{|\mathcal{G}|}})$ subspaces.

Complexity of a single operation of the brute force search. In each iteration we compute the conductor $\mathbf{Cond}(\mathscr{X}, \mathscr{C}_{\text{pub}})$ and this consists in computing the code $(\mathscr{X} \star {\mathscr{C}_{\text{pub}}}^{\perp})^{\perp}$. Since our attack consists in computing such conductors for various $\mathscr{X}$'s, one can compute a generator matrix of ${\mathscr{C}_{\text{pub}}}^{\perp}$ once for good. Hence, one can suppose a generator matrix for ${\mathscr{C}_{\text{pub}}}^{\perp}$ is known. Then, the calculation of a generator matrix of $\mathscr{X} \star {\mathscr{C}_{\text{pub}}}^{\perp}$ is reduced to the computation of the Schur product of two codes. We use the following lemma.

Lemma 4.12. *Let $\mathscr{A}, \mathscr{B}$ be two codes on $\mathbb{F}_{q^m}$ of length n and respective dimensions k_a, k_b. The computation of $\mathscr{A} \star \mathscr{B}$ can be performed in*

$$nk_ak_b + nk_ak_b \min(n, k_ak_b)$$

operations in $\mathbb{F}_{q^m}$. Moreover if $k_ak_b \geq n$, then the costs of $\mathscr{A} \star \mathscr{B}$ can be done in less than $2n^3$ operations in $\mathbb{F}_{q^m}$.

Proof. To compute the Schur product $\mathscr{A} \star \mathscr{B}$ we can proceed as follows.

1. Take bases $\mathbf{a}_1, \ldots, \mathbf{a}_{k_a}$ and $\mathbf{b}_1, \ldots, \mathbf{b}_{k_b}$ of $\mathscr{A}$ and $\mathscr{B}$ respectively and construct a matrix $\mathbf{M}$ whose rows are all the possible products $\mathbf{a}_i \star \mathbf{b}_j$, for $1 \leq i \leq k_a$ and $1 \leq j \leq k_b$. This matrix has k_ak_b rows and n columns.

2. Perform Gaussian elimination to get a reduced echelon form of $\mathbf{M}$.

The cost of the computation of a reduced echelon form of a $s \times n$ matrix is $ns\min(n,s)$ operations in the base field. The cost of the computation of the matrix $\mathbf{M}$ is the cost of $k_a k_b$ Schur products of vectors, i.e. nk_ak_b operations in the base field. This leads to an overall calculation of the Schur product equal to

$$nk_ak_b + nk_ak_b\min(n, k_ak_b)$$

operations in the base field. When $k_ak_b \geq n$, the cost of the Schur product can be reduced using a probabilistic shortcut described in [CMCP17]. It consists in computing an $n \times n$ submatrix of $\mathbf{M}$ by choosing some random subset of products $\mathbf{a}_i \star \mathbf{b}_j$. This permits to reduce the cost of computing a generator matrix in row echelon form of $\mathscr{A} \star \mathscr{B}$ to $2n^3$ operations in the base field. □

The calculation of a generator matrix of $\mathscr{X} \star \mathscr{C}_{\text{pub}}{}^{\perp}$ costs at most $2n^3$ operations. Next, note that for most of the iterations, there is no need to deduce a generator matrix in reduced echelon form of $(\mathscr{X} \star \mathscr{C}_{\text{pub}}{}^{\perp})^{\perp}$, since it suffices to evaluate the dimension of $\mathscr{X} \star \mathscr{C}_{\text{pub}}{}^{\perp}$, which is immediate from the generator matrix in reduced echelon form. If the dimension of the code is not the expected one, namely $n - \dim\mathscr{D} \neq n-4$, then we skip to the next iteration. Hence, the overall cost of a single iteration of the brute force search is bounded above by $2n^3$ operations in $\mathbb{F}_q$.

Complexity of finding $\mathbf{RS}_{q+|\mathcal{G}|-1}(\mathbf{x})$. To summarise, the cost of the brute force search on $\mathscr{D}$ can be bounded from above by

$$2n^3 q^{\frac{4q}{|\mathcal{G}|}} \text{ operations in } \mathbb{F}_q.$$

Since, $n = \Theta(q^2)$, we get a complexity in $O(n^{3+\frac{2q}{|\mathcal{G}|}})$ operations in $\mathbb{F}_q$ for the computation of $\mathscr{NT}(\mathbf{x})$.

4.4.2 How to recover $\mathscr{D}$ and $\mathscr{NT}(\mathbf{x})$ by solving a system?

An alternative to recover $\mathscr{D}$ and $\mathscr{NT}(\mathbf{x})$ consists in solving a polynomial system. The main idea is to use Proposition 4.5. Since $\mathscr{NT}(\mathbf{x}) \subseteq \mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\text{pub}})$ and $\mathbf{Cond}(\mathscr{D}, \mathscr{C}_{\text{pub}}) = (\mathscr{D} \star \mathscr{C}_{\text{pub}}{}^{\perp})^{\perp}$, we have

$$\forall \mathbf{c} \in \mathscr{NT}(\mathbf{x}),\ \mathbf{G}_{\mathscr{D}\star\mathscr{C}_{\text{pub}}{}^{\perp}} \cdot \mathbf{c}^{\top} = \mathbf{0}, \tag{4.5}$$

where $\mathbf{G}_{\mathscr{D}\star\mathscr{C}_{\text{pub}}{}^{\perp}}$ denotes a generator matrix of $\mathscr{D}\star\mathscr{C}_{\text{pub}}{}^{\perp}$. The above identity provides the system we wish to solve. The first step is to provide a formal generator matrix of $\mathscr{D}$. Later on we set $c := \frac{2q}{|\mathcal{G}|}$ and for an integer t we denote by $\mathbf{I}_t$ the $t \times t$ identity matrix. Let $\mathbf{G}^{\text{inv}}$ be a generator matrix of $\mathscr{C}_{\text{pub}}{}^{\mathcal{G}}$, we introduce $(k_0 - c)k_0$ formal variables $U_{1,1}, \ldots, U_{1,c}, \ldots, U_{k_0-c,1}, \ldots, U_{k_0-c,c}$ and define

$$\mathbf{G}(U_{i,j}) := \left(\begin{array}{c|ccc} & U_{11} & \cdots & U_{1,c} \\ \mathbf{I}_{k_0-c} & \vdots & & \vdots \\ & U_{k_0-c,1} & \cdots & U_{k_0-c,c} \end{array}\right) \cdot \mathbf{G}_{\text{inv}}.$$

There exist a generator matrix $\mathbf{G}_{\text{inv}}$ and some values $u_{1,1}, \ldots, u_{k_0-c,c} \in \mathbb{F}_q$, such that $\mathbf{G}(u_{i,j})$ is a generator matrix of $\mathscr{D}$. Almost all choices of $\mathbf{G}^{\text{inv}}$ suit. We notice that the entries of the matrix $\mathbf{G}(U_{i,j})$ are polynomials of $\mathbb{F}_q[U_{1,1}, \ldots, U_{k_0-c,c}]$ of degree 1.

Now we have to construct a generator matrix of the Schur product $\mathscr{D} \star \mathscr{C}_{\text{pub}}{}^{\perp}$. Let $\mathbf{H}$ be a parity-check matrix of the public code $\mathscr{C}_{\text{pub}}$. We can obtain a formal matrix of the Schur product

in generating all possible Schur products between one row of $\mathbf{G}(U_{i,j})$ and one row of $\mathbf{H}$. We define

$$\mathbf{R}(U_{i,j}) := \Big(g \star h\Big)_{g\in \mathrm{Rows}(\mathbf{G}(U_{i,j})),\ h\in \mathrm{Rows}(\mathbf{H})}.$$

There exists a specialisation $u_{1,1},\dots,u_{k_0-c,c} \in \mathbb{F}_q$ of the variables $U_{i,j}$ such that $\mathbf{R}(u_{i,j})$ is a generator matrix of $\mathscr{D} \star \mathscr{C}_{\mathrm{pub}}{}^{\perp}$. We notice that the entries of $R(U_{i,j})$ are still polynomials of $\mathbb{F}_q[U_{1,1},\dots,U_{k_0-c,c}]$ of degree 1.

Let us introduce a second set of variables $X_1,\dots,X_n$ which corresponds to the entries of codewords $\mathbf{c} \in \mathscr{NT}(\mathbf{x})$. By (4.5), the system we have to solve is

$$\mathbf{R}(U_{i,j}) \cdot \begin{pmatrix} X_1 \\ \vdots \\ X_n \end{pmatrix} = \mathbf{0}. \tag{4.6}$$

This system is made of polynomials of degree 2. For any such polynomial, the degree in each variable $U_{1,1},\dots,U_{k_0-c,c}$ and $X_1,\dots,X_n$ is at most to 1. Furthermore, we notice that every quadratic monomial is of the form $U_{i,j}X_s$, for some i,j,s. This kind of system is called a system of *bi-degree* $(1,1)$ in the sets of variables $U_{i,j}$ and X_s.

Using the QD structure of the support. We recall that the norm-trace code $\mathscr{NT}(\mathbf{x})$ has dimension 4 over $\mathbb{F}_q$ and is spanned by $\mathbf{1}, \mathrm{Tr}(\mathbf{x}), \mathrm{Tr}(\alpha\mathbf{x})$ and $\mathrm{N}(\mathbf{x})$. We want to use the quasi-dyadic structure to reduce the number of variables. However the quasi-dyadic structure is compatible with the trace map but not with the norm. Actually we not need to have all the code $\mathscr{NT}(\mathbf{x})$ to recover $\mathbf{x}$, it sufficient to know $\mathrm{Tr}(\mathbf{x})$ and $\mathrm{Tr}(\alpha\mathbf{x})$. The addition of structure that we describe here permits us to reduce also the number of solutions of (4.6), i.e. the solutions are only generated by $\mathrm{Tr}(\mathbf{x})$ and $\mathrm{Tr}(\alpha\mathbf{x})$.

Since the code is QD, the vector $\mathbf{x}$ is a union of orbits under the action of the additive group $\mathcal{G}$. We notice that the trace map, viewed as a polynomial, is an additive polynomial (see Definition 4.1) and in particular for any $a \in \mathcal{G}$, $\mathrm{Tr}(z+a) = \mathrm{Tr}(z) + \mathrm{Tr}(a)$, for all $z \in \mathbb{F}_{q^m}$. Let us introduce the vector $\mathbf{a}_{\mathcal{G}} := (0, a_1, a_2, a_1+a_2, \dots, \sum_{i=1}^{\gamma} a_i)$ whose entries are the elements of $\mathcal{G}$ sorted with the lexicographic order. We can write $\mathbf{x} = \mathbf{t} + \mathbf{a}$, with

$$\mathbf{t} := (\underbrace{t_1,\dots,t_1}_{2^\gamma \text{ times}}, t_2, \dots, t_{n_0}, \dots, t_{n_0}) \text{ and } \mathbf{a} := (\underbrace{\mathbf{a}_{\mathcal{G}}, \mathbf{a}_{\mathcal{G}}, \dots, \mathbf{a}_{\mathcal{G}}}_{n_0 \text{ times}}).$$

By the additive property of the trace map, we have $\mathrm{Tr}(\mathbf{x}) = \mathrm{Tr}(\mathbf{t}) + \mathrm{Tr}(\mathbf{a})$. One can introduce formal variables $A_1,\dots,A_\gamma$ corresponding to the trace of the generator of $\mathcal{G}$ and formal variables $T_1,\dots,T_{n_0}$ corresponding to the trace of the element $t_1,\dots,t_{n_0}$. Then, one can replace $(X_1,\dots,X_n)$ by

$$(T_1,\ T_1+A_1,\ \dots\ ,\ T_1+A_1+\cdots+A_\gamma,\ T_2, T_2+A_1,\ \dots\ ). \tag{4.7}$$

We have now $n_0+\gamma$ unknowns for the second set of variables instead of $n_0 2^\gamma$.

Reducing to the case $x_1 = 0$ and $x_2 = 1$. Without loss of generality we can assume that the first entries of $\mathbf{x}$ are 0 and 1 respectively. This fact comes from the 2-transitivity of the affine group over $\mathbb{F}_{q^m}$ and the following lemma.

Lemma 4.13. *Let $\mathbf{x},\mathbf{y} \in \mathbb{F}_{q^m}^n$ be a support and a multiplier, and r be an integer. Let $a,b \in \mathbb{F}_{q^m}$, such that $a \neq 0$ and let $\varphi(z) := az+b$. We have $\mathscr{A}_{r,q}(\varphi(\mathbf{x}),\mathbf{y}) = \mathscr{A}_{r,q}(\mathbf{x},\mathbf{y})$.*

Proof. By definition of alternant code, it suffices to prove that $\mathbf{GRS}_r(\varphi(\mathbf{x}),\mathbf{y})^\perp = \mathbf{GRS}_r(\mathbf{x},\mathbf{y})^\perp$. We only prove that $\mathbf{GRS}_r(\mathbf{x},\mathbf{y})^\perp \subseteq \mathbf{GRS}_r(\varphi(\mathbf{x}),\mathbf{y})^\perp$, the other inclusion is obtained by equality

of the dimensions. Let $\mathbf{c}$ be a codeword of $\mathbf{GRS}_r(\mathbf{x},\mathbf{y})^\perp$. To prove that $c \in \mathbf{GRS}_r(\varphi(\mathbf{x}),\mathbf{y})^\perp$, we have to show that for any polynomial $P \in \mathbb{F}_{q^m}[z]_{<r}$ we have $\mathbf{c}\cdot(\mathbf{y}\star P(\varphi(\mathbf{x}))) = 0$. We notice that $P(az+b)$ can be written as a polynomial $Q(z)$ of same degree than P, whose coefficients depend on a and b. Then $\mathbf{c}\cdot(\mathbf{y}\star P(\varphi(\mathbf{x}))) = \mathbf{c}\cdot(\mathbf{y}\star Q(\mathbf{x})) = 0$, since $\deg(Q) < r$ and $c \in \mathbf{GRS}_r(\mathbf{x},\mathbf{y})^\perp$. □

If the first entry x_1 of $\mathbf{x}$ equals to 0, then $\mathrm{Tr}(t_1) = \mathrm{Tr}(x_1) = 0$ and we can replace in (4.7) the variable T_1 by 0. In the same way, if x_2 equals to 1, then $\mathrm{Tr}(a_1) = 0$ and we can replace A_1 by 0. However, to recover $\mathbf{x}$, it is important to have $\mathrm{Tr}(\mathbf{x})$ and $\mathrm{Tr}(\alpha\mathbf{x})$. Since $\mathrm{Tr}(\alpha a_1) = \mathrm{Tr}(\alpha) = 1$, if we want to recover $\mathrm{Tr}(\alpha\mathbf{x})$ we have to replace A_1 by 1 and not by 0. Actually, we solve two systems: one with $A_1 = 0$ and one with $A_1 = 1$.

Using shortened codes. Similarly to the approach of §4.4.1, one can shorten the codes so that $\mathscr{D}$ has only dimension 2, which reduces the number of variables U_{ij} to $2c$ and also reduces the length of the support we seek and hence reduces the number of the variables T_i.

Name	q	n_0	k_0	γ	c	Number of variables U_{ij}	Number of variables X_i
DAGS_1	2^5	52	26	4	4	8	34
DAGS_3	2^6	38	16	5	4	8	31
DAGS_5	2^6	33	11	6	2	4	30

Table 4.2: Number of variables of the polynomial system after reduction

The number of variables of the polynomial system depends of the ratio $\frac{q}{|\mathcal{G}|}$ and the parameters n_0, k_0 and γ.

4.4.3 Finishing the attack

When the previous step of the attack is over, then, we know at least $\mathscr{NT}(\mathbf{x})$ or $\mathscr{NT}(\mathbf{x}_\mathcal{I})$ for some set $\mathcal{I}$ of positions. Thus, there remains to be able to

(1) recover $\mathbf{x}$ from $\mathscr{NT}(\mathbf{x})$ or $\mathbf{x}_\mathcal{I}$ from $\mathscr{NT}(\mathbf{x}_\mathcal{I})$;

(2) recover $\mathbf{y}$ from $\mathbf{x}$ or $\mathbf{y}_\mathcal{I}$ from $\mathbf{x}_\mathcal{I}$;

(3) recover $\mathbf{x},\mathbf{y}$ from the knowledge of $\mathbf{x}_\mathcal{I},\mathbf{y}_\mathcal{I}$.

Recovering $\mathbf{x}$ from $\mathscr{NT}(\mathbf{x})$. It is not difficult to prove that the same code after base field extension satisfies

$$\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2} = \langle \mathbf{1}, \mathbf{x}, \mathbf{x}^{\star q}, \mathbf{x}^{\star(q+1)} \rangle.$$

Because of the 2–transitivity of the affine group on $\mathbb{F}_{q^2}$, without loss of generality, one can suppose that the first entry of $\mathbf{x}$ is 0 and the second one is 1, as we see in §4.4.2. Therefore, after shortening $\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2}$ we get a code that we call $\mathscr{S}$, which is of the form

$$\mathscr{S} := \mathrm{Short}_{\{1\}}(\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2}) = \langle \mathbf{x}, \mathbf{x}^q, \mathbf{x}^{q+1} \rangle_{\mathbb{F}_{q^2}}.$$

Next, a simple calculation shows that

$$\mathscr{S} \cap \mathscr{S}^2 = \langle \mathbf{x}^{q+1} \rangle.$$

Since, the second entry of $\mathbf{x}$ has been set to 1, we can deduce the value of $\mathbf{x}^{q+1}$.

Now, finding $\mathbf{x}$ is easy: enumerate the affine subspace of $\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2}$ of vectors whose first entry is 0 and second entry is 1 (or equivalently, the affine subspace of vectors of $\mathscr{S}$ whose first entry equals 1). For any such vector $\mathbf{c}$, compute $\mathbf{c}^{q+1}$. If $\mathbf{c}^{q+1} = \mathbf{x}^{q+1}$, then $\mathbf{c}$ equals either $\mathbf{x}$ or $\mathbf{x}^q$. Taking $\mathbf{x}$ or $\mathbf{x}^q$ has no importance, due to the following lemma.

Lemma 4.14. *Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}_{q^2}^n$ be a support and a multiplier, then*

$$\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}) = \mathscr{A}_{r,q}(\mathbf{x}^q, \mathbf{y}^q).$$

Proof. Let $\mathbf{c}$ be a codeword of $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$, then $\mathbf{c} \in \mathbf{GRS}_{n-r}(\mathbf{x}, \mathbf{y})$ and $\mathbf{c} = (y_1 f(x_1), \dots, y_n f(x_n))$ for some $f \in \mathbb{F}_{q^2}[z]_{<n-r}$. Since $\mathbf{c} \in \mathbb{F}_q^n$ for all $i \in \{1, \dots, n\}$, we have

$$\begin{aligned} y_i f(x_i) &= (y_i f(x_i))^q \\ &= y_i^q \Big(\sum_{j=0}^{n-k-1} f_j x_i^j \Big)^q \\ &= y_i^q \Big(\sum_{j=0}^{n-k-1} f_j^q (x_i^q)^j \Big). \end{aligned}$$

Let us denote $f^{(q)}$ the polynomial $\sum_{j=0}^{n-k-1} f_j^q z^j \in \mathbb{F}_{q^2}[z]_{<n-r}$. Then $\mathbf{c} = (y_1^q f^{(q)}(x_1^q), \dots, y_n^q f^{(q)}(x_n^q))$ and $\mathbf{c} \in \mathbf{GRS}_{n-r}(\mathbf{x}^q, \mathbf{y}^q) \cap \mathbb{F}_q^n = \mathscr{A}_{r,q}(\mathbf{x}^q, \mathbf{y}^q)$. We just proved that $\mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}) \subseteq \mathscr{A}_{r,q}(\mathbf{x}^q, \mathbf{y}^q)$. The reciprocal inclusion is obtained by the following result

$$\mathscr{A}_{r,\mathbf{x}^q}(\mathbf{y}^q, \subseteq) \mathscr{A}_{r,q}(\mathbf{x}^{q^2}, \mathbf{y}^{q^2}) = \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y}). \qquad \square$$

Thus, without loss of generality, one can suppose $\mathbf{x}$ has been found. We summarise this section in Algorithm 5

Algorithm 5: Recovering $\mathbf{x}$ from $\mathscr{NT}(\mathbf{x})$

Input : The code $\mathscr{NT}(\mathbf{x})$.
Output: The support $\mathbf{x}$

$\mathscr{S} \leftarrow \mathrm{Short}_{\{1\}}(\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2})$
$\mathbf{s} \xleftarrow{\$} (\mathscr{S} \cap \mathscr{S}^2) \backslash \{\mathbf{0}\}$
$\mathbf{x}^{q+1} \leftarrow s_1^{(-1)} \cdot \mathbf{s}$
for $\mathbf{u} \in \mathscr{S}$ **do**
  if $\mathrm{N}(\mathbf{u}) = \mathbf{x}^{q+1}$ **then**
    return $u_1^{(-1)} \cdot \mathbf{u}$

Remark 18. If we have recovered $\mathscr{T}(\mathbf{x}) = \langle \mathrm{Tr}(\mathbf{x}), \mathrm{Tr}(\alpha \mathbf{x}) \rangle_{\mathbb{F}_q}$ (see §4.4.2), we do not need to shorten the code. Indeed

$$\mathscr{T}(\mathbf{x}) \otimes \mathbb{F}_{q^2} = \langle \mathbf{x}, \mathbf{x}^q \rangle_{\mathbb{F}_{q^2}}$$

and then we have directly $(\mathscr{T}(\mathbf{x}) \otimes \mathbb{F}_{q^2})^2 \cap (\mathscr{NT}(\mathbf{x}) \otimes \mathbb{F}_{q^2}) = \langle \mathbf{x}^{q+1} \rangle$.

Proposition 4.15. *Algorithm 5, return $\mathbf{x}$ (or $\mathbf{x}^q$) in $O(n^4)$ operations in $\mathbb{F}_{q^2}$.*

Proof. The correctness of the algorithm is already proved by the discussion above. We only prove the complexity of the algorithm.

Since the code $\mathscr{S}$ has dimension $3 \ll n$, the computation of $\mathscr{S}^2$ costs $O(n^2)$ operations in $\mathbb{F}_{q^2}$. The computation of $\mathscr{S}^2 \cap \mathscr{S}$ boils down to linear algebra and then costs $O(n^3)$ operations in $\mathbb{F}_{q^2}$.

The enumeration of elements in $\mathscr{S}$ can be performed in $O(q^6) = O(n^3)$ operations in $\mathbb{F}_{q^2}$. Indeed, the code $\mathscr{S}$ has dimension 3 over $\mathbb{F}_{q^2}$ and then has q^6 elements. Assuming that the Frobenius $z \mapsto z^q$ can be computed in constant time in $\mathbb{F}_{q^2}$, the cost of the computation of the norm of a vector is in $O(n)$ operation in $\mathbb{F}_{q^2}$.

Hence the algorithm can be performed in $O(n^4)$ operations in $\mathbb{F}_{q^2}$. $\square$

Recovering y from x. This is very classical calculation. The public code $\mathscr{C}_{\text{pub}}$ is alternant, and hence is well–known to have a parity–check matrix defined over $\mathbb{F}_{q^2}$ of the form

$$\mathbf{H}_{\text{pub}}(\mathbf{x},\mathbf{y}) = \begin{pmatrix} y_1 & \cdots & y_n \\ x_1 y_1 & \cdots & x_n y_n \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_n^{r-1} y_n \end{pmatrix}.$$

Denote by $\mathbf{G}_{\text{pub}}$ a generator matrix of $\mathscr{C}_{\text{pub}}$. Then, since the x_i are known, then the y_i can be computed by solving the linear system

$$\mathbf{H}_{\text{pub}}(\mathbf{x},\mathbf{Y}) \cdot \mathbf{G}_{\text{pub}}^{\top} = 0,$$

where $\mathbf{Y}$ is the formal vector $(Y_1,\ldots,Y_n)$. This step boils down to linear algebra and hence can be done in $O(n^3)$ operations in $\mathbb{F}_{q^2}$.

Recovering $\mathbf{x},\mathbf{y}$ from $\mathbf{x}_{\mathcal{I}},\mathbf{y}_{\mathcal{I}}$. After a suitable reordering of the indexes, one can suppose that $\mathcal{I} = \{s, s+1,\ldots,n\}$. Hence, the entries $x_1,\ldots,x_{s-1}$ of $\mathbf{x}$ and $y_1,\ldots,y_{s-1}$ are known. Let us explain how to compute x_s, y_s. Set $\mathcal{I}' := \mathcal{I} \setminus \{s\}$. Thus, let $\mathbf{G}(\mathcal{I}')$ be a generator matrix of $\mathscr{A}_{r,q}(\mathbf{x}_{\mathcal{I}'},\mathbf{y}_{\mathcal{I}'})$, which is nothing by $\text{Short}_{\mathcal{I}'}(\mathscr{C}_{\text{pub}})$. Using the notation of the previous section, we have

$$\begin{pmatrix} y_1 & \cdots & y_s \\ x_1 y_1 & \cdots & x_s y_s \\ \vdots & & \vdots \\ x_1^{r-1} y_1 & \cdots & x_s^{r-1} y_s \end{pmatrix} \cdot \mathbf{G}(\mathcal{I}') = 0.$$

In the above identity, all the x_i and y_i are known but not x_s, y_s. The entry y_s can be found by solving the linear system

$$\big(y_1,\cdots,y_{s-1},Y\big) \cdot \mathbf{G}(\mathcal{I}') = 0.$$

Then, x_s can be deduced by solving the linear system

$$\big(x_1 y_1,\cdots,x_{s-1} y_{s-1},X y_s\big) \cdot \mathbf{G}(\mathcal{I}') = 0.$$

By this manner, we can iteratively recover the entries $x_{s+1},\ldots,x_n$ and $y_{s+1},\ldots,y_n$. The only constraint is that $\mathcal{I}$ should be small enough so that $\text{Short}_{\mathcal{I}}(\mathscr{C}_{\text{pub}})$ is nonzero. But this always holds true for the choices of $\mathcal{I}$ we made in the previous sections. For each iteration, one equation suffices to find the solution and hence the cost of one iteration is bounded by $O(n^2)$ operations in $\mathbb{F}_{q^2}$. Thus, the final cost of this step is bounded by $O(n^3)$ operations in $\mathbb{F}_{q^2}$.

4.5 Algorithm, work factor and implementation

In this section, we describe the complete algorithm for each version of the attack that we proposed. For the first version, we give also the complexity of the attack and the estimate work factor for the DAGS proposal. For the second version, we are not able to provide a complexity analysis. However we give the result of practical experiments.

4.5.1 The first variant of the attack

Let us summarise the first version of the attack in the following algorithm.

For the complexity analysis, we make the approximation that operations in $\mathbb{F}_q$ can be done in constant time. Indeed, the base fields of the public keys of DAGS proposal are $\mathbb{F}_{32}$ and $\mathbb{F}_{64}$. For such a field, it is reasonable to store a multiplication and inversion table.

Algorithm 6: First version of the attack

Input : A generator matrix $\mathbf{G}_{\text{pub}}$ of $\mathscr{C}_{\text{pub}}$ and the action of $\mathcal{G}$ on $\mathscr{C}_{\text{pub}}$.
Output: The support $\mathbf{x}$ and the multiplier $\mathbf{y}$

1 $\mathscr{C}_{\text{inv}} \leftarrow {\mathscr{C}_{\text{pub}}}^{\mathcal{G}}$
2 $\mathscr{NT} \leftarrow \{0\}$
3 **while** $\dim(\mathscr{NT}) \neq 4$ **do**
4 $\quad c, c' \xleftarrow{\$} \mathscr{C}_{\text{inv}}$
5 $\quad \mathscr{NT} \leftarrow \mathbf{Cond}(\langle c, c' \rangle, \mathscr{C}_{\text{pub}})$
6 $\mathbf{x} \leftarrow \text{Recover\_}\mathbf{x}(\mathscr{NT})$ `// see Algorithm 5`
7 $\mathbf{M}_{\mathbf{x}} \leftarrow \begin{pmatrix} 1 & \cdots & 1 \\ x_1 & \cdots & x_n \\ \vdots & & \vdots \\ x_1^{r-1} & \cdots & x_n^{r-1} \end{pmatrix}$
8 $\mathbf{y} \leftarrow \text{Solve}(\mathbf{M}_{\mathbf{x}} \cdot \mathbf{Y} \cdot \mathbf{G}_{\text{pub}}^{\top} = \mathbf{0})$
9 **return** $\mathbf{x}, \mathbf{y}$

The first step can be performed by the computation of a kernel, then the complexity of line 1 is in $O(n^3)$ operations in $\mathbb{F}_q$. As we see in Section 4.4.1, the complexity of the computation of $\mathscr{NT}(\mathbf{x})$ is in $O(n^{3+\frac{2q}{|\mathcal{G}|}})$ operations in $\mathbb{F}_q$. Next, the Algorithm 5 recover $\mathbf{x}$ in $O(n^4)$ operations in $\mathbb{F}_{q^2}$ and compute $\mathbf{y}$ from the knowledge of $\mathbf{x}$ is performed in $O(n^3)$ operations in $\mathbb{F}_{q^2}$. As a conclusion, the first and the third parts of the attack are negligible compared to the computation of $\mathscr{NT}(\mathbf{x})$. Hence, we have an approximate work factor of the form

$$2n^3 q^{\frac{4q}{|\mathcal{G}|}} \text{operations in } \mathbb{F}_q.$$

Therefore, we list in Table 4.3 some approximate work factors for DAGS. The second column recalls the security levels claimed in [BBB+17a] for the best possible attack. The last column gives the approximate work factors for the first variant of our attack.

Name	Claimed security level	Work factor of our attack
`DAGS_1`	128 bits	$\approx 2^{70}$
`DAGS_3`	192 bits	$\approx 2^{80}$
`DAGS_5`	256 bits	$\approx 2^{58}$

Table 4.3: Work factors of the first variant of the attack

4.5.2 The second variant of the attack

Since the first variant of the attack had too significant costs to be tested on our machines, we only tested the second variant based on solving a polynomial system. All the tests have been done using Magma [BCP97] on an Intel® Xeon 2.27 GHz.

We have not been able to break `DAGS_3` keys using this variant of the attack, on the other hand about 100 tests have been performed for `DAGS_1` and `DAGS_5`. The average running time of the attack for `DAGS_1` keys is about 19 minutes and for `DAGS_5` keys is about 35 seconds.

Name	**Claimed security level**	**Average time**
DAGS_1	128 bits	19 mn
DAGS_5	256 bits	$<$ 1 mn

Table 4.4: Average times for the second variant of the attack.

Chapter 5

Short McEliece keys from codes on curves with positive genus

Contents

5.1 Construction of quasi-cyclic SSAG codes 87
5.2 Structure of the invariant code 88
5.3 QC codes from a cyclic cover of the projective line 90
5.3.1 Description of a public key for the McEliece scheme 90
5.3.2 A key recovery attack 91
5.4 The McEliece scheme with QC Hermitian codes 95
5.4.1 The proposed scheme 95
5.4.2 Key security reduction to the key security of the invariant code. 96
5.4.3 Model of attacks against the invariant code 96
5.4.4 The choice of the automorphism σ. 100
5.4.5 Suggested parameters. 101

5.1 Construction of quasi-cyclic SSAG codes

Let q be a power of prime p and m be a positive integer which refers to the extension degree of the field $\mathbb{F}_{q^m}$. Let $\mathcal{X}$ be an arbitrary curve defined over a finite field $\mathbb{F}_{q^m}$ and $\mathrm{Aut}(\mathcal{X})$ be its field automorphism group. Let $\langle\sigma\rangle$ be a cyclic subgroup of $\mathrm{Aut}(\mathcal{X})$ and $\ell := \mathrm{ord}(\sigma)$ its order. For any place $P \in \mathbb{P}_{\mathcal{X}}$ we denote by $\mathrm{Orb}_\sigma(P) := \left\{\sigma^i(P) \mid i \in \{0, \ldots, \ell-1\}\right\}$ the orbit of P under the automorphism σ.

Let $n_0 \in \mathbb{N}^*$, we define the support:

$$\mathcal{P} := \coprod_{i=1}^{n_0} \mathrm{Orb}_\sigma(R_i), \tag{5.1}$$

where $R_i \in \mathbb{P}_{\mathcal{X}}$ are places of degree 1, pairwise distinct with trivial stabiliser subgroup under the action of σ. Moreover we choose the R_i's such that $\mathrm{Orb}_\sigma(R_i) \neq \mathrm{Orb}_\sigma(R_j)$ for all $i \neq j$. The length of the support $\mathcal{P}$ is $n := \ell n_0$.

Let $s \in \mathbb{N}^*$, we define the divisor:

$$G := \sum_{i=1}^{s} t_i \sum_{Q \in \mathrm{Orb}_\sigma(Q_i)} Q, \tag{5.2}$$

where $Q_i \in \mathbb{P}_{\mathcal{X}}$ are places, possibly of degree bigger than 1 and $t_i \in \mathbb{Z}$ for $i \in \{1, \ldots, s\}$. The degree of the divisor G is

$$\deg(G) = \sum_{i=1}^{s} t_i \cdot \deg(Q_i) \cdot |\mathrm{Orb}_\sigma(Q_i)|.$$

Then the code $\mathrm{C_L}(\mathcal{X}, \mathcal{P}, G)$ is an ℓ-quasi-cyclic code on the curve $\mathcal{X}$ (see Definition 3.2). Indeed the group $\langle\sigma\rangle \subseteq \mathrm{Aut}(\mathcal{X})$ induces a permutation $\tilde{\sigma} \in \mathfrak{S}_n$ on the code $\mathscr{C}$ defined by:

$$\begin{array}{rccc} \tilde{\sigma}\colon & \mathrm{C_L}(\mathcal{X}, \mathcal{P}, G) & \longrightarrow & \mathrm{C_L}(\mathcal{X}, \mathcal{P}, G) \\ & (f(P_1), \ldots, f(P_n)) & \longmapsto & (f(\sigma(P_1)), \ldots, f(\sigma(P_n))). \end{array}$$

The permutation $\tilde{\sigma}$ is cyclic on each orbit of size ℓ of the support $\mathcal{P}$. Since the code $\mathscr{C} := \mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$ is the subcode of $\mathrm{C_L}(\mathcal{X}, \mathcal{P}, G)$ defined by $\mathrm{C_L}(\mathcal{X}, \mathcal{P}, G) \cap \mathbb{F}_q^n$, the permutation $\tilde{\sigma}$ acts also on it. Then the code $\mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$ is a quasi-cyclic code over $\mathbb{F}_q$.

5.2 Structure of the invariant code

We recall that the invariant code of a code stable under a permutation σ is the set of codewords fixed by σ (see Definition 3.7). In this section, we show that the invariant code of a QC SSAG code $\mathscr{C}$ is also an SSAG code with smaller parameters. Depending on the assumptions on the public elements, this result leads to the reduction of the key security of a quasi-cyclic SSAG code to the key security of its invariant code. This result is very important for the security of a McEliece scheme using QC SSAG codes. Indeed, since the invariant code can be constructed from the public key, anybody can compute it. That is why, we discuss also known attacks against this invariant code and countermeasures which can be used.

Invariant of SSAG codes. As in Section 3.2, we begin by studying the case of QC AG codes before talking about SSAG codes. We recall that, by Lemma 3.5, the invariant codewords of an AG code $\mathscr{C}$ correspond to the invariant functions of $L(G)$, where G is the divisor of $\mathscr{C}$. Then we have to look at subspaces of the function field which are fixed by an automorphism. The following proposition gives a description, in terms of Riemann-Roch spaces, of the fixed space $L(G)^\sigma$, where σ is an automorphism acting on G.

Proposition 5.1. *Let F be a function field and G be a divisor of F invariant by an automorphism $\sigma \in Aut(F)$. Then $L(G)^\sigma = L(\tilde{G})$ with $\tilde{G} \in \mathrm{Div}(F^\sigma)$.*

Proof. We denote $\mathrm{Supp}(G) = \coprod_{i=1}^{s} \mathrm{Orb}_\sigma(Q_i)$, where Q_i is a place of F for all $i \in \{1, \ldots, s\}$, i.e.:

$$G = \sum_{i=1}^{s} k_i \sum_{R \in \mathrm{Orb}_\sigma(Q_i)} R,$$

for some $k_i \in \mathbb{Z}$. Let $g \in L(G) \subseteq F$ be a function such that $g \circ \sigma = g$, i.e.: $g \in L(G)^\sigma \subseteq F^\sigma$. We can define its valuation $v_{Q'}(g)$ at any place $Q' \in \mathbb{P}_{F^\sigma}$, and its valuation $v_Q(g)$ at any place $Q \in F$. Let Q be such that $Q|Q'$, we know that:

$$e(Q|Q')v_{Q'}(g) = v_Q(g) \tag{5.3}$$

where $e(Q|Q')$ is the ramification index of Q over Q'.

For each $i \in \{1, \ldots, s\}$ we consider the place $Q'_i \in \mathbb{P}_{F^\sigma}$ such that $Q_i|Q'_i$. Then, for all $R \in \mathrm{Orb}_\sigma(Q_i)$, we have $R|Q'_i$ and $e(R|Q'_i) = e(Q_i|Q'_i)$. Since $g \in L(G)$, we know that the principal divisor over F, $(g)_F$, satisfies:

$$(g)_F \geq -\sum_{i=1}^{s} k_i \sum_{R \in Orb(Q_i)} R.$$

By (5.3), the divisor $(g)_{F^\sigma}$ over F^σ satisfies:

$$(g)_{F^\sigma} \geq -\sum_{i=1}^{s} \frac{k_i}{e(Q_i|Q'_i)} Q'_i.$$

We define $\tilde{G} := -\sum_{i=1}^{s} \left\lfloor \frac{k_i}{e(Q_i|Q'_i)} \right\rfloor Q'_i$ which is a divisor on the function field F^σ and then $g \in L(\tilde{G}) \subseteq F^\sigma$. Hence $L(G)^\sigma \subset L(\tilde{G})$.
Let $g \in F^\sigma$ such that $g \in L(\tilde{G})$, with $\tilde{G}$ defined as previously. By the equation (5.3), we have:

$$(g)_F \geq -\sum_{i=1}^{s} e(Q_i|Q'_i) \left\lfloor \frac{k_i}{e(Q_i|Q'_i)} \right\rfloor \sum_{R \in Orb(Q_i)} R.$$

Thus we have $g \in L(G)^\sigma$. □

As a direct consequence of Proposition 5.1, we have the following result on the QC AG codes (as described in Section 5.1).

Theorem 5.2. *Let $\mathcal{X}$ be an algebraic curve and G be a divisor of $\mathcal{X}$ invariant by an automorphism $\sigma \in Aut(\mathcal{X})$. Let $\mathcal{P}$ be a set of n distinct places of $\mathcal{X}$, of degree 1, such that $\sigma(\mathcal{P}) = \mathcal{P}$. Then the invariant code $C_L(\mathcal{X}, \mathcal{P}, G)^\sigma$ is the AG code $C_L(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$, for some $\tilde{\mathcal{P}} \subseteq \mathcal{X}/\langle\sigma\rangle$ and $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$.*

Proof. It is a consequence of Lemma 3.5 and Proposition 5.1 applied on the function field F of $\mathcal{X}$. □

The result about the invariant code of QC SSAG codes is given in the following corollary and comes from the fact that the invariant operation commutes with the subfield subcode operation (see Remark 6).

Corollary 5.3. *With the notation of Theorem 5.2, let $SSAG_q(\mathcal{X}, \mathcal{P}, G)$ be a subfield subcode of an AG code and σ acting on it. Then the invariant code $SSAG_q(\mathcal{X}, \mathcal{P}, G)^\sigma$ is the code $SSAG_q(\mathcal{X}/\langle\sigma\rangle, \tilde{\mathcal{P}}, \tilde{G})$ for some $\tilde{\mathcal{P}} \subset \mathcal{X}/\langle\sigma\rangle$ and $\tilde{G} \in \mathrm{Div}(\mathcal{X}/\langle\sigma\rangle)$.*

Key security reduction of QC SSAG codes. We consider a $\mathscr{C}$ is a QC SSAG code, stable under a permutation σ, and we assume that we know a generator matrix $\mathbf{G}_{\mathrm{pub}}$ of $\mathscr{C}$. The invariant code of $\mathscr{C}$ can be constructed from the knowledge of $\mathbf{G}_{\mathrm{pub}}$ and this can be done in polynomial time in the parameters of $\mathscr{C}$. It means that the generator matrix of the invariant code must be considered as a public data. As we show just above, the code $\mathscr{C}^\sigma$ has a structure of SSAG code. Now we wonder if the knowledge of secret elements of $\mathscr{C}^\sigma$ permits to recover the secret elements of $\mathscr{C}$, i.e. the support and the divisor. If it is possible in polynomial time, we say that the key security of $\mathscr{C}$ reduces to the key security of $\mathscr{C}^\sigma$.

We can consider two models. In the first one, the curve used to construct $\mathscr{C}$ is unknown, we only know the family of curves used in the scheme. In this case the key security depends on the chosen family. It must be chosen carefully to provide no information on the code $\mathscr{C}$. For instance, the family of cyclic covers of the projective line is not a good choice. First in this case, the invariant code is an SSAG code on the projective line. Moreover, we show in Section 5.3 that, in the case of Kummer extensions it is possible to recover the support and the divisor of $\mathscr{C}$ from the knowledge of the support and the divisor of the invariant code. For this example, the key security of $\mathscr{C}$ reduces to the key security of $\mathscr{C}^\sigma$. The case of Artin-Schreier extensions is similar, indeed it is also a cyclic cover of the projective line and the invariant code provide information on the x-coordinate of the support of the original code. However it is possible to choose other families of curves and avoid the described attack in Section 5.3.

In the second one, the curve $\mathcal{X}$ and the automorphism σ are known. In this case we have directly the reduction of the key security of $\mathscr{C}$ to the key security of $\mathscr{C}^\sigma$. Indeed the quotient curve $\mathcal{X}/\langle\sigma\rangle$ is known and the invariant code $\mathscr{C}^\sigma$ is an SSAG code on $\mathcal{X}/\langle\sigma\rangle$. Then from a place P on $\mathcal{X}/\langle\sigma\rangle$ it is clear that we can recover the orbit $\mathrm{Orb}_\sigma(P)$ on the curve $\mathcal{X}$. Hence the knowledge of the support and the divisor of the invariant code leads to the knowledge of the support and the divisor of $\mathscr{C}$. We will detail this for the proposed scheme in Section 5.4. Here the security of the scheme depends on the quotient curve $\mathcal{X}/\langle\sigma\rangle$ and on the possibles attacks against the invariant code.

5.3 QC codes from a cyclic cover of the projective line

In this section, we discuss the key security of quasi-cyclic codes from cyclic covers of the projective line. In particular we show that, for Kummer extensions, the key security of a code $\mathscr{C}_{\mathrm{pub}}$ reduces to the key security of its invariant code. We present first the construction of QC codes from Kummer extensions of $\mathbb{F}_{q^m}(x)$. Then, from the knowledge of the support $\tilde{\mathcal{P}}$ and the divisor $\tilde{G}$ of the invariant code, we give an algorithm to recover the support $\mathcal{P}$ and the divisor G of the code $\mathscr{C}_{\mathrm{pub}}$.

5.3.1 Description of a public key for the McEliece scheme

Cyclic cover of the projective line. Let $\ell > 1$ be an integer relatively prime to the characteristic p of $\mathbb{F}_{q^m}$, and $f \in \mathbb{F}_{q^m}[X]$ a square free polynomial. We consider the algebraic function field $F := \mathbb{F}_{q^m}(x, y)$ with

$$y^\ell = f(x), \tag{5.4}$$

and we denote by $\mathcal{X}$ the irreducible smooth curve associated to F. We denote by $P_\infty \in \mathbb{P}_F$ a common pole of x and y. The extension $F/\mathbb{F}_{q^m}(x)$ is cyclic of degree ℓ and the automorphisms of $F/\mathbb{F}_{q^m}(x)$ are given by $\sigma : y \mapsto \xi y$, where $\xi \in \mathbb{F}_{q^m}$ is an ℓ-th root of unity, i.e.: $Aut(F) =< \sigma >$. More precisely, it is a Kummer extension.

Quasi-cyclic AG codes. Let $n \in \mathbb{N}^*$, we consider a support $\mathcal{P} := \coprod_{i=1}^{n/\ell} \mathrm{Orb}_\sigma(R_i)$, as in (5.1) and a divisor

$$G := \sum_{i=1}^{s} k_i \sum_{Q \in \mathrm{Orb}_\sigma(Q_i)} Q,$$

as in (5.2). We suppose that $\mathrm{Supp}(G) \cap \mathcal{P} = \emptyset$ and $\deg(G) < n$. Moreover, we assume that $\ell | v_Q(G)$ for any ramified place $Q \in \mathrm{Supp}(G)$. We define the public code as

$$\mathscr{C}_{\mathrm{pub}} := \mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G).$$

Hypothesis for the McEliece scheme. We want to use the code $\mathscr{C}_{\mathrm{pub}}$ as a public key for the McEliece scheme (see Section 2.1). We propose to make the following hypothesis.

- The family of curve is known but not the curve $\mathcal{X}$ itself, that is we know that the equation of $\mathcal{X}$ has the form (5.4) but we **do not know** the polynomial f.
- The support $\mathcal{P}$ and the divisor G are **unknown**.
- The automorphism σ is **known**, that is we know the ℓ-th root of unity ξ. Actually is not really necessary to know σ to attack the McEliece scheme using a QC code over $\mathcal{X}$ as $\mathscr{C}_{\mathrm{pub}}$.
- A generator matrix $\mathbf{G}_{\mathrm{pub}}$ of $\mathscr{C}_{\mathrm{pub}}$ is **known** then we know also a generator matrix of ${\mathscr{C}_{\mathrm{pub}}}^\sigma$.

5.3.2 A key recovery attack

Invariant code on the projective line. From the knowledge of a generator matrix $\mathbf{G}_{\text{pub}}$ of $\mathscr{C}_{\text{pub}}$ we are able to compute a generator matrix $\mathbf{G}_{\text{inv}}$ of the invariant code ${\mathscr{C}_{\text{pub}}}^{\sigma}$. Moreover, by Corollary 5.3, we have

$$ {\mathscr{C}_{\text{pub}}}^{\sigma} := \text{SSAG}_q(\mathbf{P}^1, \tilde{\mathcal{P}}, \tilde{G}), $$

with $\tilde{\mathcal{P}} \subseteq \mathbf{P}^1$ and $\tilde{G} \in \text{Div}(\mathbf{P}^1)$. More precisely, with the previous notation,

$$ \tilde{\mathcal{P}} := \left\{ P' \in \mathbf{P}^1 \mid \exists P \in \mathcal{P} \text{ such that } P|P' \right\} \text{ and } \tilde{G} := \sum_{i=1}^{s} \left\lfloor \frac{k_i}{e(Q_i|Q_i')} \right\rfloor Q_i', $$

where for all $i \in \{1, \ldots, s\}$, $Q_i|Q_i'$. Here the invariant code ${\mathscr{C}_{\text{pub}}}^{\sigma}$ is a special case of AG code since it is a subfield subcode of GRS code. As the dual of a GRS code is a GRS code (Proposition 1.34), the code ${\mathscr{C}_{\text{pub}}}^{\sigma}$ is in fact an alternant code as defined in Definition 1.37. We recall that it is easy to pass from a description of ${\mathscr{C}_{\text{pub}}}^{\sigma}$ as an SSAG code over $\mathbf{P}^1$ to a description as a classical alternant code (see Section 1.3.4). Let us denote ${\mathscr{C}_{\text{pub}}}^{\sigma} = \mathscr{A}_{r,q}(\mathbf{x}, \mathbf{y})$, with $\mathbf{x}$ a support related to $\tilde{\mathcal{P}}$ and $\mathbf{y}$ a multiplier related to $\tilde{G}$. If we recover the vectors $\mathbf{x}$ and $\mathbf{y}$ then we recover the support $\tilde{P}$ and the divisor G.

We saw in Section 2.4 that it is possible to construct a polynomial system to recover the support and the multiplier of an alternant code. However, the cost of solving such a system is hard to estimate. Here we assume that the code ${\mathscr{C}_{\text{pub}}}^{\sigma}$ is small enough to compute a solution and recover $\mathbf{x}$ and $\mathbf{y}$ with less than 2^{80} binary operations. Then from now we assume that the support $\tilde{\mathcal{P}}$ and the divisor $\tilde{G}$ of ${\mathscr{C}_{\text{pub}}}^{\sigma}$ are **known**.

Recovering $\mathscr{C}_{\text{pub}}$ from ${\mathscr{C}_{\text{pub}}}^{\sigma}$. Now we show that it is possible to recover $\mathcal{P}$ and G from the knowledge of $\tilde{\mathcal{P}}$ and $\tilde{G}$. This means that the security of the code $\mathscr{C}_{\text{pub}}$ reduces to that of the invariant code ${\mathscr{C}_{\text{pub}}}^{\sigma}$.

From now on, we denote by $x_i := x(R_i)$, for all $i \in \{1, \ldots, \frac{n}{\ell}\}$. If we know $\tilde{\mathcal{P}}$, we know x_i for all $i \in \{1, \ldots, \frac{n}{\ell}\}$. The approach to recover $y_i := y(P_i)$ for all $i \in \{1, \ldots, n\}$ consists in solving a linear system. By definition, we have $\text{SSAG}_q(\mathcal{X}, \mathcal{P}, G) \subseteq \text{C}_{\text{L}}(\mathcal{X}, \mathcal{P}, G)$ hence $\text{C}_{\text{L}}(\mathcal{X}, \mathcal{P}, G)^{\perp} \subseteq \text{SSAG}_q(\mathcal{X}, \mathcal{P}, G)^{\perp}$. We know that

$$ \text{C}_{\text{L}}(\mathcal{X}, \mathcal{P}, G)^{\perp} = \begin{pmatrix} \text{Res}_{\omega}(P_1) & & 0 \\ & \ddots & \\ 0 & & \text{Res}_{\omega}(P_n) \end{pmatrix} \cdot \text{C}_{\text{L}}(\mathcal{X}, \mathcal{P}, W - G_{\mathcal{P}} + G), $$

with ω a differential such that $v_P(\omega) = -1$, for all $P \in \mathcal{P}$, $W := (\omega)$ and $G_{\mathcal{P}} := \sum_{P \in \mathcal{P}} P$. Then for all $g \in L(W - G_{\mathcal{P}} + G)$,

$$ \mathbf{G}_{\text{pub}} \cdot \begin{pmatrix} \text{Res}_{\omega}(P_1) & & 0 \\ & \ddots & \\ 0 & & \text{Res}_{\omega}(P_n) \end{pmatrix} \cdot \text{Ev}_{\mathcal{P}}(g)^{\top} = 0 \tag{5.5} $$

where $\mathbf{G}_{\text{pub}}$ denotes a generator matrix of $\mathscr{C}_{\text{pub}}$. We set $z_i := \text{Res}_{\omega}(P_i)$ for all $i \in \{1, \ldots, n\}$. If we know a good basis of $L(W - G_{\mathcal{P}} + G)$, the above identity provides a linear system whose products $z_i y_i$'s are solution. The first step is to find a simple basis of $L(G)$ and after that we can provide a simple basis of $L(W - G_{\mathcal{P}} + G)$.

Proposition 5.4. *With the previous notation, if for any ramified place $Q \in Supp(G)$, $\ell | v_Q(G)$, then there exists a function $h \in F$ such that $G = (h) + \deg(G) P_{\infty}$.*

Proof. Let $S := \{Q_i \mid i \in \{1, \dots, s\}\} \subseteq \mathbb{P}_F$ be a set of representatives of orbits in the support G. For each $Q_i \in S$, we consider $Q'_i \in \mathbb{P}_{\mathbf{P}^1}$ such that $Q_i | Q'_i$. By Proposition 1.17 we have $Q'_i \sim \deg(Q'_i)P'_\infty$, where $P'_\infty \in \mathbb{P}_{\mathbf{P}^1}$ is the unique pole of x. We denote by $h_i \in \mathbb{F}_{q^m}(x) \subseteq F$ the function such that:

$$Q'_i = (h_i)_{F^\sigma} + \deg(Q'_i)P'_\infty.$$

For all $Q|Q'_i$ and $R|Q'_i$, we notice that $e(Q|Q'_i) = e(R|Q'_i)$, so later on we denote by $e(Q'_i) := e(Q|Q'_i)$ for all $Q|Q'_i$. Moreover there is only one place lying over P'_∞, it is $P_\infty \in \mathbb{P}_F$ and $e(P_\infty|P'_\infty) = \ell$. By (5.3) we have:

$$e(Q'_i) \sum_{Q|Q'_i} Q = (h_i)_F + \ell \deg(Q'_i)P_\infty.$$

$F/\mathbb{F}_{q^m}(x)$ is a Kummer extension and F is defined by (5.4), so we know that (see [Sti09])

$$e(Q'_i) = \frac{\ell}{\gcd(\ell, v_{Q'_i}(f))}.$$

Since f is a square free polynomial, for $Q'_i \neq P_\infty$ we have

$$e(Q'_i) = \begin{cases} \ell & \text{if } Q'_i \text{ is a zero of } f \\ 1 & \text{else.} \end{cases}$$

If $e(Q'_i) = 1$, then:

$$\sum_{Q \in \mathrm{Orb}_\sigma(Q_i)} Q = \sum_{Q|Q'_i} Q = (h_i)_F + \ell \deg(Q'_i)P_\infty,$$

with $Q_i|Q'_i$.
If $e(Q'_i) = \ell$, then $\mathrm{Orb}_\sigma(Q_i) = \{Q_i\}$. We denote by $p \in \mathbb{F}_{q^m}(x)$ the irreducible factor over $\mathbb{F}_{q^m}$ of f such that $p(Q'_i) = 0$. Then we have $(p)_{F^\sigma} = Q'_i - \deg(Q'_i)P'_\infty$ and, by (5.3)

$$\ell Q_i = (p)_F + \ell \deg(Q'_i)P_\infty.$$

Since we assume that $\ell | v_R(G)$ for all ramified places $R \in \mathrm{Supp}(G)$, we have

$$v_{Q_i}(G)Q_i \sim v_{Q_i}(G)P_\infty.$$

Hence there exists $h \in F$ such that $G = (h)_F + \deg(G)P_\infty$. □

In the previous proof we notice that $h = \prod_i h_i \prod_j p_j$ with $h_i, p_j \in \mathbb{F}_{q^m}(x)$ then in the following we denote by $h_x \in \mathbb{F}_{q^m}(x)$ the rational function corresponding to $h \in \mathbb{F}_{q^m}(x, y)$. By the previous proposition, we have:

$$L(G) := \left\langle h_x(x)x^i y^j \mid i \geq 0, j \geq 0, \text{ and } \ell i + \deg(f)j \leq \deg(G) \right\rangle.$$

We recall that we want to have a simple basis for $L(W + G_\mathcal{P} - G)$. It suffices to chose a differential ω such that $W + G_\mathcal{P} - G \sim tP_\infty$, for some integer t.

Lemma 5.5. *There exits a differential ω such that*

$$L(W + G_\mathcal{P} - G) := \left\langle h_x(x)x^i y^j \;\middle|\; i \geq 0, j \geq 0, \text{ and } \ell i + \deg(f)j \leq 2g - 2 + n - \deg(G) \right\rangle,$$

where $W := (\omega)$.

Proof. Let ω be the following differential

$$\omega := \frac{y}{f(x)\prod_{i=1}^{n/\ell}(x-x_i)}dx,$$

then $W = (2g-2+n)P_\infty - \sum_{i=1}^{n} P_i$. Hence $W + G_{\mathcal{P}} - G = (h_x) + (2g-2+n-\deg(G))P_\infty$. Then we have the result. □

Since we know a simple basis of $L(W + G_{\mathcal{P}} - G)$, now we are able to construct a polynomial system corresponding to the equation (5.5). We introduce $2n$ formal variables $Y_1, \ldots, Y_n$ and $Z_1, \ldots, Z_n$ which correspond to $y(P)$ and $\mathrm{Res}_\omega(P)$, for $P \in \mathcal{P}$. Let $\mathbf{X}$ and $\mathbf{M}_h$ be $n \times n$ diagonal matrices with entries respectively $x(P)$ and $h(P)$, for $P \in \mathcal{P}$, then the system we have to solve, in $\mathbb{F}_{q^m}$, is:

$$\begin{pmatrix} \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \\ \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X} \\ \vdots \\ \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X}^\lambda \end{pmatrix} \cdot \begin{pmatrix} Z_1 Y_1 \\ \vdots \\ Z_1 Y_n \end{pmatrix} = 0$$

with $\lambda = \left\lfloor \frac{2g-2+n-\deg(G)-\deg(f)}{\ell} \right\rfloor$. Since the vector $(y_i)_i = (y(P))_{P\in\mathcal{P}}$ has a specific structure, that is it has a geometric progression, we are able to add some equations. We consider the following bloc matrix:

$$\mathbf{E}(\xi) := \begin{pmatrix} \mathbf{B}(\xi) & 0 & \cdots & 0 \\ 0 & & & 0 \\ \vdots & & & \\ 0 & \cdots & 0 & \mathbf{B}(\xi) \end{pmatrix}, \text{ where } \mathbf{B}(\xi) := \begin{pmatrix} 1 & -\xi & 0 & \cdots & 0 \\ 0 & & & & 0 \\ \vdots & & & & \\ 0 & & & & -\xi \\ -\xi & 0 & \cdots & 0 & 1 \end{pmatrix}$$

Then the vector $(y_i)_i$ is also solution of the system:

$$\mathbf{E}(\xi) \cdot \begin{pmatrix} Y_1 \\ \vdots \\ Y_n \end{pmatrix} = 0$$

We can also show that the vector $(z_i)_i = (\mathrm{Res}_\omega(P))_{P\in\mathcal{P}}$ has the same geometric progression. Indeed for all $P_i \in \mathcal{P}$, we have

$$\mathrm{Res}_\omega(P_i) := \frac{y_i}{f(x_i)\prod_{j\neq i}(x_i - x_j)},$$

then $\mathrm{Res}_\omega(\sigma(P_i)) = \xi\,\mathrm{Res}_\omega(P_i)$. Moreover we want to solve a linear system, so we set $U_i := Z_i Y_i$, for all $i \in \{1, \ldots, n\}$. The linear system we have to solve is

$$\begin{pmatrix} \mathbf{E}(\xi^2) \\ \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \\ \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X} \\ \vdots \\ \mathbf{G}_{\mathrm{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X}^\lambda \end{pmatrix} \cdot \begin{pmatrix} U_1 \\ \vdots \\ U_n \end{pmatrix} = 0, \tag{5.6}$$

where $\lambda := \left\lfloor \frac{2g-2+n-\deg(G)-\deg(f)}{\ell} \right\rfloor$. This system is over-constrained but we know that there exists at least one nonzero solution , i.e. the elements $z_i y_i$'s. Since the system is over-constrained, actually it is very reasonable to hope this system has a unique solution. In all our computer experiments, for cryptographic parameters, we never found more than one solution.

Once we have found the vector $(z_i y_i)_i$, we need to separate $(z_i)_i$ from $(y_i)_i$. This can be done easily in the following manner.

- First we find the vector $(z_i y_i^2)_i$, by solving the same kind of linear system than (5.6). The linear system to solve is obtained by replacing $\mathbf{E}(\xi^2)$ by $\mathbf{E}(\xi^3)$ and the maximum power λ by $\lambda' := \left\lfloor \frac{2g-2+n-\deg(G)-2\deg(f)}{\ell} \right\rfloor$, i.e.:

$$\begin{pmatrix} \mathbf{E}(\xi^3) \\ \mathbf{G}_{\text{pub}} \cdot \mathbf{M}_h \\ \mathbf{G}_{\text{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X} \\ \vdots \\ \mathbf{G}_{\text{pub}} \cdot \mathbf{M}_h \cdot \mathbf{X}^{\lambda'} \end{pmatrix} \cdot \begin{pmatrix} U_1 \\ \vdots \\ U_n \end{pmatrix} = 0. \tag{5.7}$$

- Then we compute the vector $(y_i)_i = (z_i y_i^2)_i \star (z_i y_i)_i^{-1}$, where $\star$ denotes the component-wises product.

To conclude this section, we have found all the places P_i, since we know $x(P_i)$ and $y(P_i)$ for all $i \in \{1, \ldots, n\}$. We found also $G = (h) + \deg(G) P_\infty$, and then we know the secret elements of the code $\mathscr{C}_{\text{pub}}$. We can recover these elements in polynomial time from the knowledge of $\tilde{\mathcal{P}}$ and $\tilde{G}$, that is the security of $\mathscr{C}_{\text{pub}}$ reduces to the security of ${\mathscr{C}_{\text{pub}}}^\sigma$. Let us give an algorithm to summarise this section. In this algorithm, the step " $a \xleftarrow{\$} A$" means that the element a is randomly chosen in A.

Algorithm 7: Recover the code $\mathscr{C}_{\text{pub}}$ from the invariant code

Input : A generator matrice $\mathbf{G}_{\text{pub}}$ of $\mathscr{C}_{\text{pub}}$, the support $\tilde{\mathcal{P}}$ and the divisor $\tilde{G}$ of the invariant code ${\mathscr{C}_{\text{pub}}}^\sigma$

Output: The equation of $\mathcal{X}$, the support $\mathcal{P}$ and the divisor G such that $\mathscr{C}_{\text{pub}} = \text{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$

$\mathbf{x} \leftarrow (\underbrace{x(\tilde{P}_1), \ldots, x(\tilde{P}_1)}_{\ell \text{ times}}, x(\tilde{P}_2), \ldots, x(\tilde{P}_{\frac{n}{\ell}}))$

From $\tilde{G}$ compute $h \in \mathbb{F}_{q^m}(x)$ st. $\tilde{G} = (h) + \deg(\tilde{G}) P_\infty$

$\xi \leftarrow$ an ℓ-th primitive root of unity

for $i = 1$ *to* $\ell - 1$ **do**

  $S \leftarrow$ Solve(System (5.6))

  if $\dim(S) = 1$ **then**

    $s \xleftarrow{\$} S \setminus \{0\}$

    $S' \leftarrow$ Solve(System (5.7))

    $s' \xleftarrow{\$} S'\{0\}$

    $\mathbf{y} \leftarrow s' \star s^{-1}$

    $f \leftarrow$Interpolation$(\mathbf{x}, \mathbf{y}^\ell)$ `// Then` $\mathcal{X}$ `is defined by` $y^\ell = f(x)$

    $\mathcal{P} := \{(x_i, y_i) \mid i \in \{1, \ldots, n\}\}$

    Recover G from the knowledge of $\tilde{G}$ and $\mathcal{X}$

    return $\mathcal{X}, \mathcal{P}, G$

Actually, if the quotient curve is $\mathbf{P}^1$, then the security of the scheme with QC code on a curve of genus > 0 will be the same as the scheme with QC classical Goppa codes. For the same

security there no more advantages to use codes on such algebraic curves than classical Goppa codes.

5.4 The McEliece scheme with QC Hermitian codes

We recall that q denotes a power of a prime p and m refers to the extension degree of a field $\mathbb{F}_{q^m}$. To define the Hermitian function field we need a quadratic extension, hence we set $q^m = p^{2s}$, with p a prime number. To avoid misunderstanding with the field $\mathbb{F}_q$, where the SSAG code is defined, we introduced another field $\mathbb{F}_{q_0}$ with $q_0 := p^s$.

5.4.1 The proposed scheme

The Hermitian function field will be defined on $\mathbb{F}_{q^m} = \mathbb{F}_{q_0^2}$, by $\mathcal{H} := \mathbb{F}_{q_0^2}(x, y)$ with

$$y^{q_0} + y = x^{q_0+1}. \tag{5.8}$$

The number of rational places of $\mathcal{H}$ is $N(\mathcal{H}) = q_0^3 + 1$. We denote by $P_\infty \in \mathbb{P}_\mathcal{H}$ the unique common pole of x and y, it will be named the *place at infinity*. The automorphism group of the Hermitian function field $\mathcal{A} := \mathrm{Aut}(\mathcal{H})$ has order $|\mathcal{A}| = q_0^3(q_0^2 - 1)(q_0^3 + 1)$ (see for instance [Sti09]). We construct a QC SSAG code $\mathscr{C}_{\mathrm{pub}}$ on the Hermitian function field $\mathcal{H}$ in the manner of Section 5.1. In the following, we consider an automorphism $\sigma \in \mathcal{A}$ of order $\ell \in \mathbb{N}^*$. The choice of ℓ and σ will be detailed later. In what follows we fix the notation.

- ℓ denotes a prime divisor of $q_0^2 - 1$, such that q generates the cyclic group $\mathbb{Z}/\ell\mathbb{Z}^\times$ of nonzero elements in $\mathbb{Z}/\ell\mathbb{Z}$. We will explain this point in §3.4.6.
- $\sigma \in \mathcal{A}$ denotes an automorphism of $\mathcal{H}$, of order ℓ.
- n denotes a positive integer $n < N(\mathcal{H})$ which refers to the length of a code. It should be an integer multiple of ℓ.
- $\mathcal{P} \subseteq \mathbb{P}_\mathcal{H}$ denotes the support of the QC code. It has length n and splits into $\frac{n}{\ell}$ blocks of length ℓ such that each block is an orbit under the action of σ (see Section 5.1).
- G denotes a divisor on $\mathcal{H}$ invariant by σ, as defined in 5.1.

Key generation We consider the QC code

$$\mathscr{C}_{\mathrm{pub}} := \mathrm{SSAG}_q(\mathcal{H}, \mathcal{P}, G) \tag{5.9}$$

on the Hermitian curve $\mathcal{H}$, with length n and dimension k. Let t be the correction capability of the code and $\mathbf{G}_{\mathrm{pub}}$ be a quasi-cyclic systematic generator matrix for $\mathscr{C}_{\mathrm{pub}}$, i.e.

$$\mathbf{G}_{\mathrm{pub}} = \left(\mathbf{I}_k \mid \mathbf{M}\right)$$

with $\mathbf{I}_k$ is the $k \times k$ identity matrix and $\mathbf{M}$ a ℓ-blocks-circulant matrix (see Definition 3.3). The matrix $\mathbf{G}_{\mathrm{pub}}$ can be determined by the following set:

$$\rho(\mathbf{G}_{\mathrm{pub}}) := \{\mathbf{M}_i \mid i \in \{1, \ell + 1, 2\ell + 1, \ldots, (n - k) - \ell + 1\}\},$$

where $\mathbf{M}_i$ is the i-th row of the matrix $\mathbf{M}$.

- **Public key**: the set $\rho(\mathbf{G}_{\mathrm{pub}})$ and the integer t.
- **Private key**: the support $\mathcal{P}$ and the divisor G.

Encryption A plain text $\mathbf{m} \in \mathbb{F}_q^k$ is encrypted by

$$\mathbf{y} = \mathbf{m}\mathbf{G}_{\text{pub}} + \mathbf{e}$$

where $\mathbf{G}$ is the public generator matrix and $\mathbf{e} \in \mathbb{F}_q^n$ is a vector of Hamming weight $\leq t$.

Decryption Using a decoding algorithm for Hermitian codes (see Section 1.3.5) to find the codeword $\mathbf{c} = (\mathbf{y} - \mathbf{e}) \in \mathscr{C}_{\text{pub}}$. From $\mathbf{c}$ we can recover the message $\mathbf{m}$.

Remark 19. Using a systematic generator matrix for the public key can seem unsecured. Actually the McEliece scheme is not use in this form and there exist several semantically secure conversions (see [BS08, KI02]). Moreover publishing a systematic generator matrix provides the same security against structural attacks as a random matrix.

5.4.2 Key security reduction to the key security of the invariant code.

Before choosing parameters for the QC code, we recall the hypothesis of the scheme. By Corollary 5.3, we know that the invariant code of a QC SSAG code on the Hermitian curve, will be an SSAG code on a quotient curve $\mathcal{H}/\langle\sigma\rangle$, where σ is the automorphism acting on the support and the divisor of $\mathscr{C}_{\text{pub}}$. In the case of the proposed scheme, the function field $\mathcal{H}$ is known, then recover $\mathscr{C}_{\text{pub}}$ from ${\mathscr{C}_{\text{pub}}}^{\sigma}$ is easier than in Section 5.3. We made the following hypothesis for the scheme.

(i) The curve $\mathcal{H}$ and the automorphism $\sigma \in \text{Aut}(\mathcal{H})$ are **known**.

(ii) The support $\mathcal{P}$ and the divisor G are **secret**.

(iii) A generator matrix $\mathbf{G}_{\text{pub}}$ of $\mathscr{C}_{\text{pub}}$ is **public**, then a generator matrix $\mathbf{G}_{\text{inv}}$ of ${\mathscr{C}_{\text{pub}}}^{\sigma}$ is also **known**.

Actually, these are the same hypothesis than McEliece scheme using classical Goppa codes.

The first hypothesis permits us to say that the fixed field $\mathcal{H}^\sigma$ is also known. Then when we know a place $P \in \mathcal{H}^\sigma$, we are able to find any place $R \in \mathcal{H}$ such that $R|P$. Then, recovering the divisor G from $\tilde{G}$ will be immediate. For the support, we can recover a set of places $\mathcal{P}' \subseteq \mathcal{H}$ from $\tilde{\mathcal{P}} \subseteq \mathcal{H}^\sigma$, but after that we need to order the set $\mathcal{P}'$ to recover $\mathcal{P}$. This can be done by using the Support Splitting Algorithm (SSA in short, see [Sen00]). Then we have the following result.

Lemma 5.6. *Under the assumption* (i)*, the key security of the QC code* $\mathscr{C}_{pub} := SSAG_q(\mathcal{H}, \mathcal{P}, G)$ *reduces to the key security of* ${\mathscr{C}_{pub}}^{\sigma}$*.*

5.4.3 Model of attacks against the invariant code

In the case where the fixed field is rational, we saw in Section 5.3 and Section 2.4 that is possible to construct an algebraic system to recover the secret elements of ${\mathscr{C}_{\text{pub}}}^{\sigma}$ from a generator matrix $\mathbf{G}_{\text{inv}}$. Then, in this case, the key security of a QC SSAG code on the Hermitian curve is the same than a QC alternant code. That is to say, the key security reduces to solve a bilinear system whose unknowns are the coordinates of points on the projective line. It is possible to propose a scheme whose key security depends on solving this kind of system, as we did it in Section 3.4. However, there is no advantage for the key security to use another SSAG code than alternant code in this context.

The main reason which permits us to construct this algebraic system is that there is only one class of divisors on $\mathbf{P}^1$. Moreover, we know a simple basis for the Riemann-Roch space associated to a multiple of the point at infinity. Hence, we can compute easily a basis for any Riemann-Roch space on the projective line. In the following section, we consider the case where the quotient curve is not rational.

Exhaustive search on the quotient curve

As previously, we assume that the automorphism σ is known, hence the fixed field $\mathcal{H}^\sigma$ is also known. A brute force attack on the invariant code consists in the two following steps.

(i) Enumerating all the possible divisor classes of a given degree on $\mathcal{H}^\sigma$;

(ii) Guess the divisor in the class;

(iii) Then guess the support of the invariant code.

We first speak about the third step and for this we assume that a divisor $\tilde{G}$ was found. Later on, the invariant code of the $\mathscr{C}_{\text{pub}}$ will be $\text{SSAG}_q(\mathcal{H}^\sigma, \tilde{\mathcal{P}}, \tilde{G}) = \text{C}_\text{L}(\mathcal{H}^\sigma, \tilde{\mathcal{P}}, \tilde{G}) \cap \mathbb{F}_q^n$.

Recovering the support of the invariant code

To recover the support there are two ways to proceed. The first way consist in performing an exhaustive search on sets $\mathcal{S} \subseteq \mathcal{H}^\sigma$ of length $n_0 := \frac{n}{\ell}$, then get the good permutation using SSA algorithm [Sen00].

The second way is to solve a polynomial system. Let us explain how to construct a system whose solutions are places of the support $\tilde{\mathcal{P}}$. Here we assume that we guessed a divisor G' such that $\text{C}_\text{L}(\mathcal{H}^\sigma, \tilde{\mathcal{P}}, G') = \text{C}_\text{L}(\mathcal{H}^\sigma, \tilde{\mathcal{P}}, \tilde{G})^\perp$. Since we know the fixed field $\mathcal{H}^\sigma$, we can assume that we are able to compute the Riemann-Roch space $L(G') \subseteq \mathbb{F}_{q^m}(\mathcal{H}^\sigma)$. Then we have

$$\forall f \in L(G'),\ (f(\tilde{P}_1), \ldots, f(\tilde{P}_{n_0})) \cdot \mathbf{G}_{\text{inv}}^\top = 0,$$

where $\mathbf{G}_{\text{inv}}$ denotes a generator matrix of the invariant code. Later on, we denote $L(G') = \langle f_1, \ldots, f_r \rangle_{\mathbb{F}_{q^m}}$. Let us introduce $2n_0$ formal variables $X_1, \ldots, X_{n_0}$ and $Y_1, \ldots, Y_{n_0}$ corresponding to the evaluations in x and y at the places of $\tilde{\mathcal{P}}$. The system we have to solve is

$$\begin{pmatrix} f_1(X_1, Y_1) & \cdots & f_1(X_{n_0}, Y_{n_0}) \\ f_2(X_1, Y_1) & \cdots & f_2(X_{n_0}, Y_{n_0}) \\ \vdots & & \vdots \\ f_r(X_1, Y_1) & \cdots & f_r(X_{n_0}, Y_{n_0}) \end{pmatrix} \cdot \mathbf{G}_{\text{inv}}^\top = 0.$$

We cannot provide a complexity analysis of solving this system for several reasons. First, in the general case, we cannot estimate the form of the rational function f_i and actually the choice of the system we have to solve is not unique. On the other hand, even if the system is polynomial, the complexity of solving this kind of system with Gröbner bases, depends on the form of polynomials and this is difficult to forecast.

Enumeration of divisors

Let us estimate the number of divisors we have to enumerate. First, we notice that, if we fix the support, two different divisors can produce the same code. Then it is not necessary to enumerate all divisors of $\mathcal{H}^\sigma$. Actually, we can define an equivalence relation on the set of codes of same length and dimension, which permits us to reduce the number of divisors to enumerate.

Definition 5.1. Let $\mathscr{C}_1, \mathscr{C}_2 \subseteq \mathbb{F}_q^n$ be two codes, we say that $\mathscr{C}_1$ and $\mathscr{C}_2$ are *diagonal equivalent*, and we denote $\mathscr{C}_1 \sim_{\text{diag}} \mathscr{C}_2$, if there exist $\lambda_1, \ldots, \lambda_n$, n non zero elements of $\mathbb{F}_q$, such that

$$\mathscr{C}_1 = \{(\lambda_1 c_1, \ldots, \lambda_n c_n) \mid (c_1, \ldots, c_n) \in \mathscr{C}_2\}.$$

Remark 20. When $\mathscr{C}_1 = \{(\lambda_1 c_1, \ldots, \lambda_n c_n) \mid (c_1, \ldots, c_n) \in \mathscr{C}_2\}$ for some non zero vector $(\lambda_1, \ldots, \lambda_n)$, we note $\mathscr{C}_1 = (\lambda_1, \ldots, \lambda_n) \star \mathscr{C}_2$.

The diagonal equivalence between two codes $\mathscr{C}_1$ and $\mathscr{C}_2$ can be computed by solving a linear system. Let $\mathbf{G}_{\mathscr{C}_1}$ and $\mathbf{H}_{\mathscr{C}_2}$ be respectively a generator matrix of $\mathscr{C}_1$ and a parity check matrix of $\mathscr{C}_2$. Let $Z_1, \ldots, Z_n$ be n formal variables, then we have to solve the following system

$$\mathbf{G}_{\mathscr{C}_1} \cdot \begin{pmatrix} Z_1 & & 0 \\ & \ddots & \\ 0 & & Z_n \end{pmatrix} \cdot {\mathbf{H}_{\mathscr{C}_2}}^{\top} = 0. \tag{5.10}$$

Hence we have an easy way to know if two codes are diagonal equivalent, it suffices to solve the system (5.10). If there exist a nonzero solution, then the two codes$\mathscr{C}_1$ and $\mathscr{C}_2$ are diagonal equivalent.

The case of AG codes. In the case of AG codes, the following result shows that the diagonal equivalence class of a code depends **only** on the equivalent class of its divisor.

Theorem 5.7 ([MP93, Corollary 4.15])**.** *If $\mathcal{P}$ is a set of $n > 2g-2$ rational places of $\mathcal{X}$, where g is the genus of an algebraic curve $\mathcal{X}$, and G and H are two divisors of the same degree $2g-1 < t < n-1$, then:*

$$C_L(\mathcal{X}, \mathcal{P}, G) \sim_{diag} C_L(\mathcal{X}, \mathcal{P}, H) \Leftrightarrow G \sim H.$$

Let $AG_r(\mathcal{X}, \mathcal{P})$ be the set of algebraic geometry codes on an algebraic curve $\mathcal{X}$ over $\mathbb{F}_{q^m}$, defined by the support $\mathcal{P}$ and a divisor of degree r. Then we have the following result.

Corollary 5.8. *Let $\mathcal{X}$ be an smooth algebraic curve over $\mathbb{F}_{q^m}$ and g its genus. Let $\mathcal{P} \subseteq \mathcal{X}(\mathbb{F}_{q^m})$ be a support of length $n > 2g+2$, and $r \in \mathbb{N}$ such that $2g-1 < r < n-1$. Then*

$$\#(AG_r(\mathcal{X}, \mathcal{P}) / \sim_{diag}) = h(\mathcal{X}),$$

where $h(\mathcal{X})$ is number of divisor classes.

Proof. This result is the consequence of Theorem 5.7 and Proposition 1.15. □

This estimation is sufficient if we want to perform a brute force search on AG codes. We assume that we know a generator matrix of an AG code $\mathscr{C} := \mathrm{C_L}(\mathcal{X}, \mathcal{P}, G)$. We could proceed as follows.

- Perform a brute force search among divisor classes of degree r, then guess a divisor G'.
- Guess the support $\mathcal{P}'$ and solve the system (5.10) to check whether the code $\mathrm{C_L}(\mathcal{X}, \mathcal{P}', G')$ is diagonal equivalent to the code $\mathscr{C}$.
- If the system (5.10) has a solution, then we found a divisor G', a support $\mathcal{P}'$ and a vector $(\lambda_1, \ldots, \lambda_n) \in (\mathbb{F}_{q^m}^*)^n$ such that the code $\mathscr{C}$ is $(\lambda_1, \ldots, \lambda_n) \star \mathrm{C_L}(\mathcal{X}, \mathcal{P}', G')$.

The case of SSAG codes. For subfield subcodes of AG codes, the estimate of the cost of the brute force search is more complicated. Indeed Theorem 5.7 cannot be applied on subfield subcodes. We can use the following result to have a first estimate of the number of SSAG codes.

Proposition 5.9 ([MP93, Corollary 7.4])**.** *If $n > 2g+2$ and $2g-1 < r < n$, then*

$$\#AG_r(\mathcal{X}, \mathcal{P}) = (q^m - 1)^{n-1} h(\mathcal{X}).$$

Let $SSAG_{q,r}(\mathcal{X}, \mathcal{P})$ be the set of subfield subcodes on $\mathbb{F}_q$ of AG codes on an algebraic curve $\mathcal{X}$, defined by the support $\mathcal{P}$ and a divisor of degree r. With the previous proposition, it is clear that

$$\#SSAG_{q,r}(\mathcal{X}, \mathcal{P}) \leq \#AG_r(\mathcal{X}, \mathcal{P}) = (q^m - 1)^{n-1} h(\mathcal{X}).$$

Actually, we can decrease a little the previous bound with the following remark.

Remark 21. If $\mathscr{C}_1 = (\lambda_1, \ldots, \lambda_n) \star \mathscr{C}_2$, with $\mathscr{C}_1, \mathscr{C}_2 \subseteq \mathbb{F}_{q^m}^n$ and $\lambda_1, \ldots, \lambda_n \in \mathbb{F}_q^*$, then

$$\mathscr{C}_1 \cap \mathbb{F}_q^n = \mathscr{C}_2 \cap \mathbb{F}_q^n.$$

This leads to the upper bound

$$\#SSAG_{q,r}(\mathcal{X}, \mathcal{P}) \leq \frac{(q^m - 1)^{n-1}}{(q-1)^{n-1}} h(\mathcal{X}).$$

Brute force algorithm. In Algorithm 8, $\mathrm{Cl}^r(\mathcal{X})$ denotes the group of divisor classes of degree r (see Definition 1.27). For all divisor $G \in \mathrm{Div}(\mathcal{X})$ of degree r, we denote $[G]$ its class in $\mathrm{Cl}^r(\mathcal{X})$.

Algorithm 8: Brute force on SSAG

Input : A generator matrix $\mathbf{G}$ of an SSAG code $\mathscr{C}$ and an integer r which the degree of the divisor

Output: The support $\mathcal{P}$ and the divisor G such that $\mathscr{C} = \mathrm{SSAG}_q(\mathcal{X}, \mathcal{P}, G)$

```
S ← {P ∈ 𝒳 | P of degree 1}                          // The order of S is fixed
for [G] ∈ Cl^r(𝒳) do
    𝒞' ← SSAG(𝒳, S, G)                               // G will be a representative of [G]
    for [v] ∈ F_{q^m}^n / F_q^n do
        /* v will be a representative of [v] */
        for I ⊆ {1, ..., N(𝒳) − 1} with |I| = n do
            𝒞'_I ← Punct_I(𝒞')
            π ← SSA(v ⋆ 𝒞'_I, 𝒞)                     // SSA return a permutation π or '?'
            if π ∈ 𝔖_n then
                S_I ← {P_i ∈ S | i ∈ I}
                return π(S_I), G and v
```

The complexity of Algorithm 8 is at least the cost of the exhaustive search on G and $\mathbf{v}$. Then we estimate that the cost of the brute force search to find the code $\mathscr{C}$ is at least

$$((q^m - 1)^{n-1} - (q-1)^{n-1}) h(\mathcal{X}) \text{ operations over } \mathbb{F}_q.$$

By Theorem 1.16 we know that

$$(\sqrt{q^m} - 1)^{2g} \leq h(\mathcal{X}) \leq (\sqrt{q^m} + 1)^{2g}.$$

Then we can write that $h(\mathcal{X}) \in O(q^{mg})$ and the cost of the enumeration of divisor classes $[G]$ in $\mathrm{Cl}^r(\mathcal{X})$ and vectors in $\mathbb{F}_{q^m} \setminus \mathbb{F}_q$ is in $O(q^{m(n-1)+mg})$.

The case of QC Hermitian codes. In the previous discussion we treat the general case of SSAG codes. Here the case of interest is the invariant code of some QC SSAG codes over the Hermitian curve. That is, we want to estimate the cost of an exhaustive search on SSAG codes over a fixed field $\mathcal{H}^\sigma$ for some $\sigma \in \mathcal{A}$. As we said previously, we only treat the cost of

the enumeration of equivalent divisors in $\mathrm{Cl}^r(\mathcal{H}^\sigma)$, where r is a parameter of the scheme. If $\mathbb{F}_q$ denotes the base field of the SSAG code, in §5.4.3 we estimate this cost to be

$$O(q^{m(n-1)+mg}) \text{ operations in } \mathbb{F}_q,$$

where g denotes the genus of $\mathcal{H}^\sigma$, m the extension degree of the SSAG code and n its length.

We can make an observation about this estimation. It depends on the choice of the automorphism σ, since it depends on the genus of $\mathcal{H}^\sigma$. In the following section we talk about the choice of σ and in particular we give the genus of $\mathcal{H}^\sigma$ for our choice of σ.

5.4.4 The choice of the automorphism σ.

The choice of the automorphism σ is important, as we just explain in the previous section, the structure of the fixed field $\mathcal{H}^\sigma$ can be useful to attack the scheme. In a first time we describe the group where σ will be chosen. In a second part, we discuss the influence of σ in the complexity of attacks against the invariant code which are explained in Section 5.2.

We denote by $\mathcal{A}(P_\infty)$ the subgroup consisting in all automorphisms $\sigma \in \mathcal{A}$ such that $\sigma(P_\infty) = P_\infty$. An element of $\mathcal{A}(P_\infty)$ acts as follows:

$$\begin{cases} \sigma(x) = ax + b, \\ \sigma(y) = a^{q_0+1}y + ab^{q_0}x + c, \end{cases} \tag{5.11}$$

with $a \in \mathbb{F}^*_{q_0^2}$, $b \in \mathbb{F}_{q_0^2}$ and $b^{q_0+1} = c^{q_0} + c$ (see [GSX00]). It has order $|\mathcal{A}(P_\infty)| = q_0^3(q_0^2 - 1)$. In the following, we identify an automorphism $\sigma \in \mathcal{A}(P_\infty)$ with a triple $[a, b, c]$. The order of σ depends only on the order of a and on the choice of c.

Lemma 5.10 ([GSX00, Lemma 4.1])**.** *Let $\sigma := [a, b, c] \in \mathcal{A}(P_\infty)$ with $a \neq 1$. Then we have*

(i) If $\mathrm{ord}(a)$ is not a divisor of $q_0 + 1$, then $\mathrm{ord}(\sigma) = \mathrm{ord}(a)$.

(ii) If $\mathrm{ord}(a)$ divides $q_0 + 1$ then

$$\mathrm{ord}(\sigma) = \begin{cases} \mathrm{ord}(a) & \text{if } c = \frac{ab^{q_0+1}}{a-1} \\ p \cdot \mathrm{ord}(a) & \text{otherwise.} \end{cases}$$

Let $a \in \mathbb{F}^*_{q^2}$ be an element of order $\ell > 2$ with $\ell | (q_0^2 - 1)$ and $b \in \mathbb{F}_{q_0^2}$. If ℓ divides $q_0 + 1$, we chose $c = \frac{ab^{q_0+1}}{a-1}$, else we chose randomly c over the roots of $z^{q_0} + z - b^{q_0+1}$. We consider the automorphism $\sigma := [a, b, c]$, by the previous lemma it has order ℓ.

Complexity of exhaustive search on divisors of $\mathcal{H}^\sigma$. The quotient curves of the Hermitian curve have been studied in [GSX00] and the authors provide a formula to compute the genus of the fixed field $\mathcal{H}^\sigma$. Let us recall this result.

Proposition 5.11. *Let $\sigma := [a, b, c] \in \mathcal{A}(P_\infty)$ be an automorphism of order $\ell > 2$, such that ℓ is prime. Then we have*

(i) If ℓ divides $(q_0 - 1)$, then $g(\mathcal{H}^\sigma) = \frac{(q_0-1)q_0}{2\ell}$.

(ii) If ℓ divides $(q_0 + 1)$ and $c = \frac{ab^{q_0+1}}{a-1}$, then $g(\mathcal{H}^\sigma) = \frac{(q_0-1)(q_0-(\ell-1))}{2\ell}$.

Proof. It is a particular case of [GSX00, Theorem 4.4]. □

We notice that to avoid the case $g(\mathcal{H}^\sigma) = 0$, ℓ should be strictly less than $q_0 + 1$. We will see other conditions on ℓ in Section 5.4.5. With the previous proposition we can estimate the class number $h(\mathcal{H}^\sigma)$. As a consequence of Proposition 5.11 and Theorem 1.16, we have the following result.

Corollary 5.12. *Let $\sigma := [a, b, c] \in \mathcal{A}(P_\infty)$ be an automorphism of order $\ell > 2$, such that ℓ is prime. Then we have*

(i) If ℓ divides $(q_0 - 1)$, then $h(\mathcal{H}^\sigma) = \Theta\Big(q_0^{\frac{q_0^2}{\ell}}\Big)$.

(ii) If ℓ divides $(q_0 + 1)$ and $c = \frac{ab^{q_0+1}}{a-1}$, then $h(\mathcal{H}^\sigma) = \Theta\Big(q_0^{\frac{q_0(q_0-\ell)}{\ell}}\Big)$.

The number of divisor classes $h(\mathcal{H}^\sigma)$ is a lower bound for the cost of Algorithm 8. Actually, in most of parameters which we propose in Section 5.4.5, this number is large enough to reach a complexity larger that 2^{128} operations in the field $\mathbb{F}_q$. This number depends on the field $\mathbb{F}_{q_0}$ and on the order of quasi-cyclicity ℓ. The choice of these two parameters will be discussed in the following section.

5.4.5 Suggested parameters.

The last thing to do is to choose the quasi-cyclicity order ℓ, the field $\mathbb{F}_q$ and the extension degree m. These choices are related between them and to the parameters of the public code. The discussion about the order of quasi-cyclicity was already done in §3.4.6. The choice for ℓ remains the same for condition (i). For the second condition we have to replace the number 2 be q. That is we have the two following conditions.

(i) ℓ **should be prime.**

(ii) ℓ **should be such that** q **is an** $(\ell - 1)$**-th primitive root of** 1, which leads that the polynomial $1 + z + \cdots + z^{\ell-1}$ is irreducible in $\mathbb{F}_q[z]$.

Remark 22. If $p \nmid \ell$ then the folded code equals to the invariant code (see Lemma 3.1). In the following, we always have $p \nmid \ell$. Hence, if we choose ℓ which satisfies (i) and (ii), then the only subcode that an attacker can construct, by the manners described previously, is the invariant code ${\mathscr{C}_{\text{pub}}}^\sigma$.

Choice of the field $\mathbb{F}_q$ and extension degree m. To provide SSAG codes over $\mathbb{F}_q$, defined on a Hermitian function field, we have to choose a finite extension $\mathbb{F}_{q^m}$ of $\mathbb{F}_q$. Let us discuss the choice of q and m.

- First we can notice that q and m must be chosen such that q^m is a square. Indeed the Hermitian function field must be defined on a quadratic extension $\mathbb{F}_{q_0^2}$. In the following, the choice of q_0, which is related to q and m, will also be discussed.
- m should be not too large since it has a negative influence on the dimension of the code. Indeed, for a fixed length n and a fixed divisor G, the lower bound on the dimension is $n - m(n - \ell(G))$. This leads to a negative influence of m on the rate of the code.
- We can make the same remark for q_0, about the negative influence on the dimension. If we look at the lower bound on the dimension we have

$$k \geq n - m(n - \ell(G)) = n - m(n - \deg(G) + g - 1),$$

with $g := \frac{q_0(q_0-1)}{2}$. Then q_0 should not be too large as m.

- On the other hand, the choice $q = q_0$ and $m = 2$, which could be a good choice with respect to the two previous points, is not encouraged. Let us give some informal arguments about this. The smaller the extension degree m, the closer the structure of the SSAG code to that of the AG code. We recall that the AG codes have been broken in polynomial time in [CMCP17], and that for codes on the projective line, some Goppa codes with $m = 2$ have also be broken in polynomial time [COT17]. We actually do not know if the same attacks are possible on SSAG codes on the Hermitian function field. We made the choice to give some parameters with $m = 2$, because it provides the best key sizes. However, we warn the reader that it could be the weakest keys.

Parameters. We recall the notation that will be used in the following table.

- q is a power of a prime. The SSAG code is defined on $\mathbb{F}_q$.
- m is the extension degree of the field of definition of the support and divisor over $\mathbb{F}_q$.
- q_0 is a prime power such that $q_0^2 = q^m$. The Hermitian function field is defined on $\mathbb{F}_{q_0^2}$.
- ℓ is the order of quasi-cyclicity of the SSAG code, satisfying (i) and (ii).
- n is the length of the SSAG code, divisible by ℓ.
- k is the dimension of the SSAG code.
- w_{ISD} is the work factor for message recovery. It is computed using **CaWoF** library [CT16a].

m	q	q_0	n	k	ℓ	Key size (bytes)	w_{ISD}	$g(\mathcal{H}^\sigma)$	$h(\mathcal{H}^\sigma)$
8	2	2^4	4083	2307	3	170718	128	40	$\approx 2^{326}$
8	2	2^4	4085	2315	5	102438	129	24	$\approx 2^{196}$
4	2^2	2^4	3000	1245	3	182416	128	40	$\approx 2^{326}$
4	2^2	2^4	3000	1250	5	109687	129	24	$\approx 2^{196}$
3	3^2	3^3	2996	1337	7	158434	129	39	$\approx 2^{374}$

Table 5.1: Suggested parameters for security 128, with $m > 2$

In the following table, we have $q_0 = q$, since $m = 2$. For some parameters, $h(\mathcal{H}^\sigma) < 2^{128}$, so we precise the number of different SSAG codes over $\mathbb{F}_q$, with the same support $\mathcal{P} \subseteq \mathcal{H}$ and the degree of the divisor equals to r. As in Section 5.4.3, we denote this number $\#SSAG_{q,r}(\mathcal{H}, \mathcal{P})$.

m	q	n	k	ℓ	Key size (bytes)	w_{ISD}	$g(\mathcal{H}^\sigma)$	$h(\mathcal{H}^\sigma)$	$\#SSAG_{q,r}(\mathcal{H}, \mathcal{P})$
2	11	900	513	3	33088	129	15	$\approx 2^{107}$	$\approx 2^{6316}$
2	11	1200	740	5	34040	128	11	$\approx 2^{78}$	$\approx 2^{8360}$
2	13	1299	735	3	69090	131	26	$\approx 2^{197}$	$\approx 2^{9793}$
2	13	994	539	7	17517	130	6	$\approx 2^{45}$	$\approx 2^{7386}$

Table 5.2: Suggested parameters for security 128, with $m = 2$

Bibliography

[ABQ09] Miriam Abdón, Juscelino Bezerra, and Luciane Quoos, *Further examples of maximal curves*, Journal of Pure and Applied Algebra **213** (2009), no. 6, 1192–1196.

[Bar04] Magali Bardet, *Étude des systèmes algébriques surdéterminés. applications aux codes correcteurs et à la cryptographie*, Ph.D. thesis, Université Paris VI, December 2004, http://tel.archives-ouvertes.fr/tel-00449609/en/.

[Bar17] Élise Barelli, *On the security of some compact keys for McEliece scheme*, WCC Workshop on Coding and Cryptography, September 2017.

[BBB+17a] Gustavo Banegas, Paulo S.L.M Barreto, Brice Odilon Boidje, Pierre-Louis Cayrel, Gilbert Ndollane Dione, Kris Gaj, Cheikh Thiécoumba Gueye, Richard Haeussler, Jean Belo Klamti, Ousmane N'diaye, Duc Tri Nguyen, Edoardo Persichetti, and Jefferson E. Ricardini, *DAGS : Key encapsulation for dyadic GS codes*, `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/DAGS.zip`, November 2017, First round submission to the NIST post-quantum cryptography call.

[BBB+17b] Magali Bardet, Elise Barelli, Olivier Blazy, Rodolfo Canto-Torres, Alain Couvreur, Philippe Gaborit, Ayoub Otmani, Nicolas Sendrier, and Jean-Pierre Tillich, *BIG QUAKE*, `https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/BIG_QUAKE.zip`, November 2017, first round submission to the NIST post-quantum cryptography call.

[BBD+17] Elise Barelli, Peter Beelen, Mrinmoy Datta, Vincent Neiger, and Johan Rosenkilde, *Two-point codes for the generalised GK curve*, IEEE Trans. Inform. Theory (2017).

[BC18] Elise Barelli and Alain Couvreur, *An efficient structural attack on nist submission DAGS*, `https://arxiv.org/abs/1805.05429`, 2018.

[BCGO09] Thierry P. Berger, Pierre-Louis Cayrel, Philippe Gaborit, and Ayoub Otmani, *Reducing key length of the McEliece cryptosystem*, Progress in Cryptology - AFRICACRYPT 2009 (Gammarth, Tunisia) (Bart Preneel, ed.), LNCS, vol. 5580, June 21-25 2009, pp. 77–97.

[BCP97] Wieb Bosma, John Cannon, and Catherine Playoust, *The Magma algebra system I: The user language*, J. Symbolic Comput. **24** (1997), no. 3/4, 235–265.

[Bee07] Peter Beelen, *The order bound for general algebraic geometric codes*, Finite Fields and Their Applications **13** (2007), no. 3, 665–680.

[Ber00a] Thierry P. Berger, *Goppa and related codes invariant under a prescribed permutation*, IEEE Trans. Inform. Theory **46** (2000), no. 7, 2628–2633.

[Ber00b] ———, *On the cyclicity of Goppa codes, parity-check subcodes of Goppa codes and extended Goppa codes*, Finite Fields Appl. **6** (2000), no. 3, 255–281.

[BFS04] Magali Bardet, Jean-Charles Faugère, and Bruno Salvy, *On the complexity of Gröbner basis computation of semi-regular overdetermined algebraic equations*, Proceedings of the International Conference on Polynomial System Solving, 2004, pp. 71–74.

[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*, Advances in Cryptology - EUROCRYPT 2012, LNCS, Springer, 2012.

[Bla08] Richard E Blahut, *Algebraic codes on lines, planes, and curves: an engineering approach*, Cambridge University Press, 2008.

[BLM11] Paulo Barreto, Richard Lindner, and Rafael Misoczki, *Monoidic codes in cryptography*, Post-Quantum Cryptography 2011, LNCS, vol. 7071, Springer, 2011, pp. 179–199.

[BMvT78] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg, *On the inherent intractability of certain coding problems*, IEEE Trans. Inform. Theory **24** (1978), no. 3, 384–386.

[BMZ18] Daniele Bartoli, Maria Montanucci, and Giovanni Zini, *Ag codes and ag quantum codes from the ggs curve*, Designs, Codes and Cryptography **86** (2018), no. 10, 2315–2344.

[BS08] Bhaskar Biswas and Nicolas Sendrier, *McEliece cryptosystem implementation: theory and practice*, Post-Quantum Cryptography, Second International Workshop, PQCrypto 2008, Cincinnati, OH, USA, October 17-19, 2008, Proceedings (Johannes Buchmann and Jintai Ding, eds.), LNCS, vol. 5299, Springer, 2008, pp. 47–62.

[BT06] Peter Beelen and Nesrin Tutaş, *A generalization of the weierstrass semigroup*, Journal of Pure and Applied Algebra **207** (2006), no. 2, 243–260.

[CGG+14] Alain Couvreur, Philippe Gaborit, Valérie Gauthier-Umaña, Ayoub Otmani, and Jean-Pierre Tillich, *Distinguisher-based attacks on public-key cryptosystems using Reed-Solomon codes*, Des. Codes Cryptogr. **73** (2014), no. 2, 641–666.

[CMCP14] Alain Couvreur, Irene Márquez-Corbella, and Ruud Pellikaan, *A polynomial time attack against algebraic geometry code based public key cryptosystems*, Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014, June 2014, pp. 1446–1450.

[CMCP17] ———, *Cryptanalysis of McEliece cryptosystem based on algebraic geometry codes and their subcodes*, IEEE Trans. Inform. Theory **63** (2017), no. 8, 5404–5418.

[COT17] Alain Couvreur, Ayoub Otmani, and Jean-Pierre Tillich, *Polynomial time attack on wild McEliece over quadratic extensions*, IEEE Trans. Inform. Theory **63** (2017), no. 1, 404–427.

[CT16a] Rodolfo Canto Torres, *CaWoF, C library for computing asymptotic exponents of generic decoding work factors*, 2016, https://gforge.inria.fr/projects/cawof/.

[CT16b] Alonso Sepúlveda Castellanos and Guilherme Chaud Tizziotti, *Two-point ag codes on the gk maximal curves.*, IEEE Trans. Information Theory **62** (2016), no. 2, 681–686.

[Del75] Philippe Delsarte, *On subfield subcodes of modified Reed-Solomon codes*, IEEE Trans. Inform. Theory **21** (1975), no. 5, 575–576.

[DH76] Whitfield Diffie and Martin Hellman, *New directions in cryptography*, IEEE transactions on Information Theory **22** (1976), no. 6, 644–654.

[DKP11] Iwan Duursma, Radoslav Kirov, and Seungkook Park, *Distance bounds for algebraic geometric codes*, Journal of Pure and Applied Algebra **215** (2011), no. 8, 1863–1878.

[dP15] Frédéric Urvoy de Portzamparc, *Algebraic and physical security in code-based cryptography. (sécurités algébrique et physique en cryptographie fondée sur les codes correcteurs d'erreurs)*, Ph.D. thesis, Pierre and Marie Curie University, Paris, France, 2015.

[Dum91] Ilya Dumer, *On minimum distance decoding of linear codes*, Proc. 5th Joint Soviet-Swedish Int. Workshop Inform. Theory (Moscow), 1991, pp. 50–52.

[Dür87] Arne Dür, *The automorphism groups of Reed-Solomon codes*, Journal of Combinatorial Theory, Series A **44** (1987), 69–82.

[Duu11] Iwan M Duursma, *Two-point coordinate rings for gk-curves*, IEEE Transactions on Information Theory **57** (2011), no. 2, 593–600.

[Fau09] Cédric Faure, *Etudes de systèmes cryptographiques construits à l'aide de codes correcteurs, en métrique de hamming et en métrique rang.*, Ph.D. thesis, Ecole Polytechnique X, 2009.

[FG10] Stefania Fanali and Massimo Giulietti, *One-point ag codes on the gk maximal curves*, IEEE Transactions on Information Theory **56** (2010), no. 1, 202–210.

[FGO+10] Jean-Charles Faugère, Valérie Gauthier, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich, *A distinguisher for high rate McEliece cryptosystems*, IACR Cryptology ePrint Archive, Report2010/331, 2010, http://eprint.iacr.org/.

[Fin09] Matthieu Finiasz, *NP-completeness of certain sub-classes of the syndrome decoding problem*, 2009, arXiv:0912.0453.

[FM08] Cédric Faure and Lorenz Minder, *Cryptanalysis of the McEliece cryptosystem over hyperelliptic curves*, Proceedings of the eleventh International Workshop on Algebraic and Combinatorial Coding Theory (Pamporovo, Bulgaria), June 2008, pp. 99–107.

[FOP+16a] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich, *Folding alternant and Goppa Codes with non-trivial automorphism groups*, IEEE Trans. Inform. Theory **62** (2016), no. 1, 184–198.

[FOP+16b] ———, *Structural cryptanalysis of McEliece schemes with compact keys*, Des. Codes Cryptogr. **79** (2016), no. 1, 87–112.

[FOPT10] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, and Jean-Pierre Tillich, *Algebraic cryptanalysis of McEliece variants with compact keys*, Advances in Cryptology - EUROCRYPT 2010, LNCS, vol. 6110, 2010, pp. 279–298.

[FPdP14] Jean-Charles Faugère, Ludovic Perret, and Frédéric de Portzamparc, *Algebraic attack against variants of McEliece with Goppa polynomial of a special form*, Advances in Cryptology - ASIACRYPT 2014 (Kaoshiung, Taiwan, R.O.C.), LNCS, vol. 8873, Springer, December 2014, pp. 21–41.

[FR93] G-L Feng and Thammavarapu RN Rao, *Decoding algebraic-geometric codes up to the designed minimum distance*, IEEE Transactions on Information Theory **39** (1993), no. 1, 37–45.

[Ful08] William Fulton, *Algebraic curves*, An Introduction to Algebraic Geometry (2008), 54.

[Gab05] Philippe Gaborit, *Shorter keys for code based cryptography*, Proceedings of the 2005 International Workshop on Coding and Cryptography (WCC 2005) (Bergen, Norway), March 2005, pp. 81–91.

[GGS10] Arnaldo Garcia, Cem Güneri, and Henning Stichtenoth, *A generalization of the giulietti–korchmáros maximal curve*, Advances in Geometry **10** (2010), no. 3, 427–434.

[GK09] Massimo Giulietti and Gábor Korchmáros, *A new family of maximal curves over a finite field*, Mathematische Annalen **343** (2009), no. 1, 229.

[GKL93] Arnaldo Garcia, Seon Jeong Kim, and Robert F Lax, *Consecutive weierstrass gaps and minimum distance of goppa codes*, Journal of pure and applied algebra **84** (1993), no. 2, 199–207.

[Gos96] David Goss, *Basic structures of function field arithmetic*, Ergebnisse der Mathematik und ihrer Grenzgebiete (3) [Results in Mathematics and Related Areas (3)], vol. 35, Springer-Verlag, Berlin, 1996.

[GÖS13] Cem Güneri, Mehmet Özdemiry, and Henning Stichtenoth, *The automorphism group of the generalized giulietti–korchmáros function field*, Advances in Geometry **13** (2013), no. 2, 369–380.

[GSX00] Arnaldo Garcia, Henning Stichtenoth, and Chao-Ping Xing, *On subfields of the hermitian function field*, Compositio Mathematica **120** (2000), no. 2, 137–170.

[HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz, *A modular analysis of the Fujisaki-Okamoto transformation*, Theory of Cryptography Conference, Springer, 2017, pp. 341–371.

[HP95] Tom Høholdt and Ruud Pellikaan, *On the decoding of algebraic-geometric codes*, IEEE Transactions on Information Theory **41** (1995), no. 6, 1589–1614.

[HP03] W. Cary Huffman and Vera Pless, *Fundamentals of error-correcting codes*, Cambridge University Press, Cambridge, 2003.

[HvLP98] Tom Høholdt, Jacobus H van Lint, and Ruud Pellikaan, *Algebraic geometry codes*, Handbook of coding theory **1** (1998), no. Part 1, 871–961.

[HY17] Chuangqiang Hu and Shudi Yang, *Multi-point codes from the ggs curves*, arXiv preprint arXiv:1706.00313 (2017).

[JLJ+89] Jørn Justesen, Knud J Larsen, Helge Elbrønd Jensen, Allan Havemose, and Tom Høholdt, *Construction and decoding of a class of algebraic geometry codes*, IEEE Transactions on Information Theory **35** (1989), no. 4, 811–821.

[JM96] Heeralal Janwa and Oscar Moreno, *McEliece public key cryptosystems using algebraic-geometric codes*, Des. Codes Cryptogr. **8** (1996), no. 3, 293–307.

[KI02] Kazukuni Kobara and Hideki Imai, *Semantically secure mceliece public-key cryptosystem*, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences **85** (2002), no. 1, 74–83.

[Knu97] Donald Ervin Knuth, *The art of computer programming: sorting and searching*, vol. 3, Pearson Education, 1997.

[Loi01] Pierre Loidreau, *Codes derived from binary Goppa codes*, Probl. Inf. Transm. **37** (2001), no. 2, 91–99.

[Mat01] Gretchen L Matthews, *Weierstrass pairs and minimum distance of goppa codes*, Designs, Codes and Cryptography **22** (2001), no. 2, 107–121.

[MB09] Rafael Misoczki and Paulo Barreto, *Compact McEliece keys from Goppa codes*, Selected Areas in Cryptography (Calgary, Canada), August 13-14 2009.

[McE78] Robert J. McEliece, *A public-key system based on algebraic coding theory*, pp. 114–116, Jet Propulsion Lab, 1978, DSN Progress Report 44.

[Min] MinT, *The online database for optimal parameters of* (t, m, s)*-nets,* (t, s)*-sequences, orthogonal arrays, linear codes, and ooas.*

[Min07] Lorenz Minder, *Cryptography based on error correcting codes*, Ph.D. thesis, Ecole Polytechnique Fédérale de Lausanne, 2007.

[MMT11] Alexander May, Alexander Meurer, and Enrico Thomae, *Decoding random linear codes in* $O(2^{0.054n})$, Advances in Cryptology - ASIACRYPT 2011 (Dong Hoon Lee and Xiaoyun Wang, eds.), LNCS, vol. 7073, Springer, 2011, pp. 107–124.

[Mor93] Carlos Moreno, *Algebraic curves over finite fields*, no. 97, Cambridge University Press, 1993.

[MP93] Carlos Munuera and Ruud Pellikaan, *Equality of geometric goppa codes and equivalence of divisors*, Journal of Pure and Applied Algebra **90** (1993), no. 3, 229–252.

[MS86] Florence J. MacWilliams and Neil J. A. Sloane, *The theory of error-correcting codes*, fifth ed., North–Holland, Amsterdam, 1986.

[MS07] Lorenz Minder and Amin Shokrollahi, *Cryptanalysis of the Sidelnikov cryptosystem*, Advances in Cryptology - EUROCRYPT 2007 (Barcelona, Spain), LNCS, vol. 4515, 2007, pp. 347–360.

[Nie86] Harald Niederreiter, *Knapsack-type cryptosystems and algebraic coding theory*, Problems of Control and Information Theory **15** (1986), no. 2, 159–166.

[Pel89] Ruud Pellikaan, *On a decoding algorithm for codes on maximal curves*, IEEE transactions on information theory **35** (1989), no. 6, 1228–1232.

[Pet10] Christiane Peters, *Information-set decoding for linear codes over* $\mathbf{F}_q$, Post-Quantum Cryptography 2010, LNCS, vol. 6061, Springer, 2010, pp. 81–94.

[Pra62] Eugene Prange, *The use of information sets in decoding cyclic codes*, IRE Transactions on Information Theory **8** (1962), no. 5, 5–9.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman, *A method for obtaining digital signatures and public-key cryptosystems*, Commun. ACM **21** (1978), no. 2, 120–126.

[Sen94] Nicolas Sendrier, *On the structure of a randomly permuted concatenated code*, EUROCODE'94, 1994, pp. 169–173.

[Sen00] ———, *Finding the permutation between equivalent linear codes: The support splitting algorithm*, IEEE Trans. Inform. Theory **46** (2000), no. 4, 1193–1203.

[Sho94] Peter W. Shor, *Algorithms for quantum computation: Discrete logarithms and factoring*, FOCS (S. Goldwasser, ed.), 1994, pp. 124–134.

[Sid94] Vladimir Michilovich Sidelnikov, *A public-key cryptosytem based on Reed-Muller codes*, Discrete Math. Appl. **4** (1994), no. 3, 191–207.

[SJM+95] Shajiro Sakata, Jørn Justesen, Y Madelung, H Elbrond Jensen, and T Høholdt, *Fast decoding of algebraic-geometric codes up to the designed minimum distance*, IEEE Transactions on Information Theory **41** (1995), no. 6, 1672–1677.

[SS92] Vladimir Michilovich Sidelnikov and S.O. Shestakov, *On the insecurity of cryptosystems based on generalized Reed-Solomon codes*, Discrete Math. Appl. **1** (1992), no. 4, 439–444.

[Ste88] Jacques Stern, *A method for finding codewords of small weight*, Coding Theory and Applications (G. D. Cohen and J. Wolfmann, eds.), LNCS, vol. 388, Springer, 1988, pp. 106–113.

[Sti90] Henning Stichtenoth, *On automorphisms of geometric Goppa codes*, Journal of Algebra **130** (1990), no. 1, 113–121.

[Sti09] ———, *Algebraic function fields and codes*, second ed., Graduate Texts in Mathematics, vol. 254, Springer-Verlag, Berlin, 2009.

[SV90] Alexei N Skorobogatov and Serge G Vlăduţ, *On the decoding of algebraic-geometric codes*, IEEE Transactions on Information Theory **36** (1990), no. 5, 1051–1060.

[TVN07] Michael A. Tsfasman, Serge G. Vlăduţ, and Dmitry Nogin, *Algebraic geometric codes: basic notions*, no. 139, American Mathematical Soc., 2007.

List of Algorithms

1 Recover the divisor in the case σ diagonalizable in $\mathbb{F}_{q^m}$ 55
2 Recover the divisor in the case σ trigonalizable over $\mathbb{F}_{q^m}$ 56
3 Recover the support 58
4 The function $\mathcal{F}$: construction of a word of weight t 61

5 Recovering $\mathbf{x}$ from $\mathscr{NT}(\mathbf{x})$ 83
6 First version of the attack 85

7 Recover the code $\mathscr{C}_{\text{pub}}$ from the invariant code 94
8 Brute force on SSAG 99

9 OrderBoundTable 125

List of Tables

3.1 Average time for Algorithm 3 with σ diagonalizable in $\mathbb{F}_{q^m}$ 58
3.2 Suggested parameters for security 128 66
3.3 Suggested parameters for security 192 66
3.4 Suggested parameters for security 256 67

4.1 Parameters proposed in DAGS. 71
4.2 Number of variables of the polynomial system after reduction 82
4.3 Work factors of the first variant of the attack 85
4.4 Average times for the second variant of the attack. 86

5.1 Suggested parameters for security 128, with $m > 2$ 102
5.2 Suggested parameters for security 128, with $m = 2$ 102

A.1 Estimation of the minimum distance 126

Appendix A

Two-Point Codes for the Generalised GK curve

We improve previously known lower bounds for the minimum distance of certain two-point AG codes constructed using a Generalized Giulietti–Korchmaros curve (GGK). Castellanos and Tizziotti recently described such bounds for two-point codes coming from the Giulietti–Korchmaros curve (GK). Our results completely cover and in many cases improve on their results, using different techniques, while also supporting any GGK curve. Our method builds on the order bound for AG codes: to enable this, we study certain Weierstrass semigroups. This allows an efficient algorithm for computing our improved bounds. We find several new improvements upon the MinT minimum distance tables.

A.1 Introduction

Algebraic geometry (AG) codes are a class of linear codes constructed from algebraic curves defined over a finite field. This class continues to provide examples of good codes when considering their basic parameters: the length n, the dimension k, and the minimum distance d. If the algebraic curve used to construct the code has genus g, the minimum distance d satisfies the inequality $d \geq n - k + 1 - g$. This bound, a consequence of the Goppa bound, implies that the minimum distance of an AG code can be designed. It is well known that the Goppa bound is not necessarily tight, and there are various results and techniques which can be used to improve upon it in specific cases. Such a result has been given in [Mat01, Thm. 2.1], where the Goppa bound is improved by one. Another approach to give lower bounds on the minimum distance of AG codes is described in [HvLP98] and the references therein. This type of lower bound is often called the *order bound*; various refinements and generalizations have been given, for example in [Bee07, DKP11].

To obtain good AG codes, the choice of the algebraic curve in the construction plays a key role. A very good class of curves are the so-called maximal curves, i.e., algebraic curves defined over a finite field having as many rational points as allowed by the Hasse–Weil bound. More precisely, a maximal curve of genus g defined over a finite field $\mathbb{F}_q$ with q elements, has $q+1+2\sqrt{q}g$ $\mathbb{F}_q$-rational points, i.e., points defined over $\mathbb{F}_q$; this only makes sense if the cardinality q is a square number. An important example of a maximal curve is the Hermitian curve, but recently other maximal curves have been described [GK09, GGS10], often called the *generalized Giulietti–Korchmáros (GK) curves*. In this article we continue the study of two-point AG codes coming from the generalized GK curves that was initiated in [CT16b]. However, rather than using the improvement upon the Goppa bound from [Mat01], we use the order bound as given in [Bee07]. As a matter of fact, we also show that the order bound from [Bee07] implies Theorem 2.1 in [Mat01]. Thus, we will automatically recover all the results in [CT16b], but on various occasions we obtain better bounds for the minimum distance than the ones reported in [CT16b].

We will also paraphrase the order bound from [Bee07] and explain how we have computed it. A key object in this computation is a two-point generalization of a Weierstrass semigroup given in [BT06], and therefore some time will be used to describe this semigroup explicitly in the case of certain pairs of points on the generalized GK curve.

After finishing this work, we were made aware of the contemporaneous work [HY17]. In [HY17] multi-point codes and their duals from the generalized GK function field are constructed and investigated. Proposition A.15 is different from, but akin to [HY17, Thm. 2] and similar proof techniques were used. The techniques used in [HY17] to analyse the code parameters are very different from ours and more related to the ones used in [CT16b]. Our main tools, the explicit computation of the map $\tau_{0,\infty}$ in Corollary A.11 and the resulting algorithm to compute the order bound, were not employed in [HY17]. Our improvements on the MinT code tables are not present in [HY17].

A.2 Preliminaries

Though later we will only consider the generalized GK curves, we will in this section consider any algebraic curve χ defined over a finite field $\mathbb{F}_q$. The field of functions on χ, or briefly the function field of χ, will be denoted by $\mathbb{F}_q(\chi)$, while the genus of χ is denoted by $g(\chi)$. Rather than using the language of curves, we will formulate the theory using the language of function fields; see [Sti09] for more details. In particular, we will speak about places of $\mathbb{F}_q(\chi)$ rather than points of χ. For any place Q of $\mathbb{F}_q(\chi)$, we denote by v_Q the valuation map at the place Q. The valuation $v_Q : \mathbb{F}_q(\chi) \setminus \{0\} \to \mathbb{Z}$ sends a nonzero function f to its order of vanishing at Q. If $v_Q(f) < 0$, one also says that f has a pole of order $-v_Q(f)$ at Q.

A divisor of $\mathbb{F}_q(\chi)$ is a finite formal sum $\sum_i n_i Q_i$ of places Q_i of $\mathbb{F}_q(\chi)$, where the n_i's are integers in $\mathbb{Z}$. The support of a divisor $\sum_i n_i Q_i$ is the (finite) set of places $\{Q_i \mid n_i \neq 0\}$. Finally, we call two divisors disjoint if they have disjoint supports. To any nonzero function $f \in \mathbb{F}_q(\chi)$ one can associate two divisors (f) and $(f)_\infty$ known as the divisor of f and the divisor of poles of f respectively, given by:

$$(f) := \sum_Q v_Q(f)\, Q \quad \text{and} \quad (f)_\infty := \sum_{Q; v_Q(f)<0} -v_Q(f)\, Q.$$

If all the coefficients n_i in a divisor $G = \sum_i n_i Q_i$ are nonnegative, we call G an effective divisor; notation $G \geq 0$.

We now recall some notations for AG codes; we again refer to [Sti09] for a more comprehensive exposition. Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ be a set of n distinct rational places of $\mathbb{F}_q(\chi)$, i.e., places of degree 1, and define the divisor $D = P_1 + \cdots + P_n$. Further let G be a divisor such that $\deg(G) < n$ and G does not contain any place of $\mathcal{P}$. We consider the following map:

$$\begin{array}{rccc} Ev_{\mathcal{P}} : & \mathbb{F}_q(\chi)_{\mathcal{P}} & \longrightarrow & \mathbb{F}^n \\ & f & \longmapsto & (f(P_1), \ldots, f(P_n)). \end{array}$$

Here $\mathbb{F}_q(\chi)_{\mathcal{P}}$ denotes the subset of $\mathbb{F}_q(\chi)$ consisting of functions not having a pole at any $P \in \mathcal{P}$. Then we define the AG code $C_L(D, G)$ by $C_L(D, G) := \{Ev_{\mathcal{P}}(f) \mid f \in L(G)\}$. Here $L(G)$ denotes the Riemann–Roch space $L(G) := \{f \in \mathbb{F}_q(\chi) \setminus \{0\} \mid (f) + G \geq 0\} \cup \{0\}$. It is well known that the minimum distance d of $C_L(D, G)$ (resp. $C_L(D, G)^\perp$) satisfies the Goppa bound $d \geq n - \deg(G)$ (resp. $d \geq \deg(G) - 2g(\chi) + 2$).

Here, we will make use of another lower bound for the minimum distance of $C_L(D, G)^\perp$, obtained in [Bee07]. We will use the notions of G-gaps and G-non-gaps at a place Q, which were for example also used in [GKL93].

Definition A.1. Let Q be a rational place and G be a rational divisor of $\mathbb{F}_q(\chi)$. We define $L(G+\infty Q) := \bigcup_{i\in\mathbb{Z}} L(G+iQ)$ and

$$H(Q;G) := \{-v_Q(f) \mid f \in L(G+\infty Q)\setminus\{0\}\}.$$

We call $H(Q;G)$ the set of *G-non-gaps at Q*. The set

$$\Gamma(Q;G) := \mathbb{Z}_{\geq v_Q(G)-\deg(G)} \setminus H(Q;G)$$

is called the set of *G-gaps at Q*.

Note that if $G=0$ we obtain $H(Q;0)=H(Q)$, the Weierstrass semigroup of Q, and $\Gamma(Q;0)=\Gamma(Q)$, the set of gaps at Q. Further, note that if $i\in H(Q;F_1)$ and $j\in H(Q;F_2)$, then $i+j\in H(Q;F_1+F_2)$. Finally, observe that the theorem of Riemann–Roch implies that the number of G-gaps at Q coincides with the genus of χ, that is, $|\Gamma(Q;G)|=g(\chi)$.

Remark 23. If $i<-\deg(G)$ then $\deg(G+iQ)<0$ and $L(G+iQ)=\{0\}$. So in the previous definition we can write $L(G+\infty Q)=\bigcup_{i\geq -\deg(G)} L(G+iQ)$. Further, note that for any $a\in\mathbb{Z}$ we have $L(G+aQ+\infty Q)=L(G+\infty Q)$ and hence $H(Q;G+aQ)=H(Q;G)$ as well as $\Gamma(Q;G+aQ)=\Gamma(Q;G)$.

Definition A.2. Let Q be a rational place and let F_1, F_2 be two divisors of χ. As in [Bee07] we define

$$N(Q;F_1,F_2) := \{(i,j)\in H(Q;F_1)\times H(Q;F_2) \mid i+j=v_Q(G)+1\},$$
$$\nu(Q;F_1,F_2) := |N(Q;F_1,F_2)|.$$

Proposition A.1. *[Bee07, Prop. 4] Let $D=P_1+\cdots+P_n$ be a divisor that is a sum of n distinct rational places of $\mathbb{F}_q(\chi)$, Q be a rational place not occurring in D, and F_1,F_2 be two divisors disjoint from D. Suppose that $C_L(D,F_1+F_2)\neq C_L(D,F_1+F_2+Q)$. Then, for any codeword $c\in C_L(D,F_1+F_2)^\perp\setminus C_L(D,F_1+F_2+Q)^\perp$, we have*

$$w_H(c)\geq \nu(Q;F_1,F_2).$$

In particular, the minimum distance $d(F_1+F_2)$ of $C_L(D,F_1+F_2)^\perp$ satisfies

$$d(F_1+F_2)\geq \min\{\nu(Q;F_1,F_2), d(F_1+F_2+Q)\},$$

where $d(F_1+F_2+Q)$ denotes the minimum distance of $C_L(D,F_1+F_2+Q)^\perp$.

To arrive at a lower bound for the minimum distance of $C_L(D,G)^\perp$, one applies this proposition in a recursive manner. More precisely, one constructs a sequence $Q^{(1)},\ldots,Q^{(N)}$ of not necessarily distinct rational places, none occurring in D, such that $C_L(D,G+Q^{(1)}+\cdots+Q^{(N)})^\perp=0$. Such a sequence exists, since the theorem of Riemann–Roch implies that $C_L(D,G+Q^{(1)}+\cdots+Q^{(N)})=\mathbb{F}_q^n$ as soon as $N\geq 2g(\chi)-1+n-\deg(G)$. Then, the code $C_L(D,G)^\perp$ has for example minimum distance at least $\min \nu(Q^{(i)};G+Q^{(1)}+\cdots+Q^{(i-1)},0)$, where the minimum is taken over all i satisfying $1\leq i\leq N$ and $C_L(D,G+Q^{(1)}+\cdots+Q^{(i-1)})\neq C_L(D,G+Q^{(1)}+\cdots+Q^{(i)})$.

The well known Goppa bound is a direct consequence of Proposition A.1 as shown in [Bee07, Lem. 9]. We will need the following slightly more general version of [Bee07, Lem. 9].

Lemma A.2. *Let $D=P_1+\cdots+P_n$ be a sum of distinct rational places, let Q be a rational place not occurring in D, and let F_1,F_2 be two divisors disjoint from D. Then $\nu(Q;F_1,F_2)\geq \deg(F_1+F_2)-2g+2$.*

Proof. Define the formal Laurent series

$$p_{Q;F_1}(t) := \sum_{i \in H(Q;F_1)} t^i \qquad \text{and} \qquad p_{Q;F_2}(t) := \sum_{i \in H(Q;F_2)} t^i.$$

Then $\nu(Q; F_1, F_2)$ is the coefficient of $t^{v_Q(F_1+F_2)+1}$ in the Laurent series $p_{Q;F_1}(t) \cdot p_{Q;F_2}(t)$. The lemma follows by analyzing this product carefully. First we introduce

$$q_{Q;F_1}(t) := \sum_{i \in \Gamma(Q;F_1)} t^i \qquad \text{and} \qquad q_{Q;F_2}(t) := \sum_{i \in \Gamma(Q;F_2)} t^i.$$

Then

$$p_{Q;F_1}(t) + q_{Q;F_1}(t) = \frac{t^{v_Q(F_1)-\deg(F_1)}}{1-t} \qquad \text{and} \qquad p_{Q;F_2}(t) + q_{Q;F_2}(t) = \frac{t^{v_Q(F_2)-\deg(F_2)}}{1-t},$$

implying that

$$\begin{aligned} p_{Q;F_1}(t) \cdot p_{Q;F_2}(t) = t^{v_Q(F_1+F_2)-\deg(F_1+F_2)} \Bigg(& \frac{1}{(1-t)^2} - \frac{2g(\chi)}{1-t} \\ & + \frac{g(\chi) - t^{-v_Q(F_2)+\deg(F_2)} q_{Q;F_2}(t)}{1-t} + \frac{g(\chi) - t^{-v_Q(F_1)+\deg(F_1)} q_{Q;F_1}(t)}{1-t} \Bigg) + q_{Q;F_1}(t) \cdot q_{Q;F_2}(t). \end{aligned}$$

Since both $t^{-v_Q(F_1)+\deg(F_1)} q_{Q;F_1}(t)$ and $t^{-v_Q(F_2)+\deg(F_2)} q_{Q;F_2}(t)$ are a sum of $g(\chi)$ distinct nonnegative powers of t, the last three Laurent series in the above expression are in fact finite Laurent series with nonnegative coefficients. Hence the coefficient of $t^{v_Q(F_1+F_2)+1}$ in $p_{Q;F_1}(t) \cdot p_{Q;F_2}(t)$ is bounded from below by the corresponding coefficient in

$$t^{v_Q(F_1+F_2)-\deg(F_1+F_2)} \left(1/(1-t)^2 - 2g(\chi)/(1-t)\right),$$

which is $\deg(F_1 + F_2) - 2g(\chi) + 2$. □

In this paper, we are interested in a lower bound on the minimum distance for two-point AG codes. We will typically apply Proposition A.1 to the special setting where $F_1 = 0$ and $F_2 = G = a_1Q_1 + a_2Q_2$, with Q_1, Q_2 two rational places of $\mathbb{F}_q(\chi)$. Hence we want to compute $\nu(Q; G) := \nu(Q; 0, G)$ where $G = a_1Q_1 + a_2Q_2$. Furthermore, we will only consider the case where $Q \in \{Q_1, Q_2\}$. In order to compute the number $\nu(Q; G)$, we need to know the Weierstrass semigroup $H(Q)$ and the set $H(Q; G)$ of G-non-gaps at Q. A very practical object in this setting is a two-point generalization of the Weierstrass semigroup and a map between two Weierstrass semigroups considered in [BT06]:

Definition A.3. Let Q_1, Q_2 be two distinct rational places of $\mathbb{F}_q(\chi)$. We define $R(Q_1, Q_2) := \{f \in \mathbb{F}_q(\chi) \mid \mathrm{Supp}((f)_\infty) \subseteq \{Q_1, Q_2\}\}$, the ring of functions on χ that are regular outside the points Q_1 and Q_2. The two-point Weierstrass semigroup of Q_1 and Q_2 is then defined as:

$$H(Q_1, Q_2) := \{(n_1, n_2) \in \mathbb{Z}^2 \mid \exists f \in R(Q_1, Q_2) \setminus \{0\},\ v_{Q_i}(f) = -n_i, i \in \{1, 2\}\}.$$

Further we define the following map:

$$\begin{array}{rccl} \tau_{Q_1,Q_2}: & \mathbb{Z} & \longrightarrow & \mathbb{Z} \\ & i & \longmapsto & \min\{j \mid (i, j) \in H(Q_1, Q_2)\}. \end{array}$$

Remark 24. Note that $H(Q_1, Q_2) \subseteq \{(i, j) \in \mathbb{Z}^2 \mid i + j \geq 0\}$, since $L(iQ_1 + jQ_2) = \{0\}$ if $i + j < 0$. In particular, we have for any $i \in \mathbb{Z}$ that $\tau_{Q_1,Q_2}(i) \geq -i$. Moreover, the theorem of Riemann–Roch implies that $\tau_{Q_1,Q_2}(a_1) \leq 2g(\chi) - a_1$.

Proposition A.3. *[BT06, Prop. 14] Let Q_1, Q_2 be two distinct rational places of $\mathbb{F}_q(\chi)$. The map τ_{Q_1,Q_2} is bijective and $\tau^{-1}_{Q_1,Q_2} = \tau_{Q_2,Q_1}$.*

By the definitions of τ_{Q_1,Q_2} and $H(Q_1, Q_2)$, for all $i \in \mathbb{Z}$ there exists a function $f^{(i)}_{Q_1,Q_2} \in R(Q_1, Q_2)$ such that $v_{Q_1}(f^{(i)}_{Q_1,Q_2}) = -i$ and $v_{Q_2}(f^{(i)}_{Q_1,Q_2}) = -\tau_{Q_1,Q_2}(i)$. Since τ_{Q_1,Q_2} is a bijection, the functions $f^{(i)}_{Q_1,Q_2}$ have distinct pole orders at Q_1 as well as Q_2.

Theorem A.4. *Let Q_1, Q_2 be two distinct rational places of χ and $a_1, a_2 \in \mathbb{Z}_{\geq 0}$. The Riemann–Roch space $L(a_1Q_1 + a_2Q_2)$ has dimension $|\{i \leq a_1 \mid \tau_{Q_1,Q_2}(i) \leq a_2\}|$ and basis*

$$\{f^{(i)}_{Q_1,Q_2} \mid i \leq a_1 \text{ and } \tau_{Q_1,Q_2}(i) \leq a_2\}.$$

Proof. Consider the filtration of $\mathbb{F}$-vector spaces:

$$L(a_1Q_1+a_2Q_2) \supseteq L((a_1-1)Q_1+a_2Q_2) \supseteq \cdots \supseteq L(-a_2Q_1+a_2Q_2) \supseteq L(-(a_2+1)Q_1+a_2Q_2) = \{0\}.$$

For $-a_2 \leq i \leq a_1$, the strict inequality $\ell(iQ_1 + a_2Q_2) > \ell((i-1)Q_1 + a_2Q_2)$ holds if and only if there exists a function $f \in \mathbb{F}_q(\chi)$ such that $(f)_\infty = iQ_1 + jQ_2$ with $j \leq a_2$. Such a function exists if and only if $\tau_{Q_1,Q_2}(i) \leq a_2$. Hence, $\ell(a_1Q_1 + a_2Q_2) = |\{-a_2 \leq i \leq a_1 \mid \tau_{Q_1,Q_2}(i) \leq a_2\}|$. Since $\tau_{Q_1,Q_2}(i) \geq -i$, we see that $\ell(a_1Q_1 + a_2Q_2) = |\{i \leq a_1 \mid \tau_{Q_1,Q_2}(i) \leq a_2\}|$ as was claimed.

A basis for $L(a_1Q_1 + a_2Q_2)$ can be directly derived from the above, since the set

$$\{f^{(i)}_{Q_1,Q_2} \mid i \leq a_1 \text{ and } \tau_{Q_1,Q_2}(i) \leq a_2\}$$

is a subset of $L(a_1Q_1 + a_2Q_2)$ consisting of $\ell(a_1Q_1 + a_2Q_2)$ linearly independent functions. Note that the linear independence follows from the fact the functions have mutually distinct pole orders at Q_1. □

A direct corollary is an explicit description of the $(a_1Q_1 + a_2Q_2)$-gaps and non-gaps at Q_1.

Corollary A.5. *Let $G = a_1Q_1 + a_2Q_2$. Then the set of G-non-gaps at Q_1 is given by*

$$\{a \in \mathbb{Z} \mid \tau_{Q_1,Q_2}(a) \leq a_2\}$$

and the set of G-non-gaps at Q_2 is given by

$$\{b \in \mathbb{Z} \mid \tau^{-1}_{Q_1,Q_2}(b) \leq a_1\}.$$

Proof. The first part follows directly from the previous theorem by considering basis of $L(aQ_1 + a_2Q_2)$ for a tending to infinity. Reversing the roles of Q_1 and Q_2, the second part follows. □

This corollary implies that for $G = a_1Q_1 + a_2Q_2$, it is not hard to compute the G-gaps at either Q_1 or Q_2 once the bijection τ_{Q_1,Q_2} can be computed efficiently. We show in an example that this does occur in a particular case. Moreover, in the next section we will give a very explicit description of τ_{Q_1,Q_2} for a family of function fields and pairs of rational points Q_1 and Q_2.

Example 10. The Hermitian curve $\mathcal{H}$ is the curve defined over $\mathbb{F}_{q^2}$ by the equation $x^q + x = y^{q+1}$. The corresponding function field $\mathbb{F}_{q^2}(\mathcal{H})$ is called the Hermitian function field. For any two distinct rational places Q_1 and Q_2 of $\mathbb{F}_{q^2}(\mathcal{H})$, the map τ_{Q_1,Q_2} satisfies $\tau_{Q_1,Q_2}(i) = -iq$ for $q \leq i \leq 0$. Since furthermore $\tau_{Q_1,Q_2}(i + q + 1) = \tau_{Q_1,Q_2}(i) - (q + 1)$ for any $i \in \mathbb{Z}$, this describes τ_{Q_1,Q_2} completely. See [BT06] for more details. This example also appears as a special case in the next section.

A.3 The generalized Giulietti–Korchmáros function field

Let $e \geq 1$ be an odd integer. We consider the generalized Giulietti–Korchmáros (GK) curve χ_e, also known as the Garcia–Güneri–Stichtenoth curve [GGS10]. It is defined over the finite field $\mathbb{F}_{q^{2e}}$ by the equations

$$x^q + x = y^{q+1} \text{ and } z^{\frac{q^e+1}{q+1}} = y^{q^2} - y.$$

This is a maximal curve when considered over the finite field $\mathbb{F}_{q^{2e}}$. Indeed, its genus and number of rational points are

$$g(\chi_e) := (q-1)(q^{e+1} + q^e - q^2)/2,$$
$$N_e := q^{2e+2} - q^{e+3} + q^{e+2} + 1.$$

As before, we will use the language of function fields and denote the corresponding function field $\mathbb{F}_{q^{2e}}(\chi_e)$ as the generalized GK function field. For $e = 1$ one simply obtains the Hermitian function field $\mathbb{F}_{q^2}(\mathcal{H})$, while for $e = 3$, one obtains what is known as the Giulietti–Korchmáros function field [GK09].

The function $x \in \mathbb{F}_{q^{2e}}(\chi_e)$ has exactly one zero and one pole, which we will denote by Q_0 and Q_∞ respectively. The functions y and z also have a pole at Q_∞ only. For a given rational place P of $\mathbb{F}_{q^{2e}}(\chi_e)$ different from Q_∞, we call $(x(P), y(P), z(P)) \in \mathbb{F}_{q^{2e}}^3$ the coordinates of P. For the function field $\mathbb{F}_{q^{2e}}(\chi_e)$, rational places are uniquely determined by their coordinates. A place with coordinates $(a, b, c) \in \mathbb{F}_{q^{2e}}^3$ will be denoted by $P_{(a,b,c)}$. In particular, we have $Q_0 = P_{(0,0,0)}$.

With these notations, we can express the divisors of x, y and z as follows:

$$\begin{aligned}
(x) &= (q^e+1)(Q_0 - Q_\infty), \\
(y) &= \sum_{\substack{a \in \mathbb{F}_q \\ a^q + a = 0}} \frac{q^e+1}{q+1} P_{(a,0,0)} \; - \; q\frac{q^e+1}{q+1} Q_\infty, \\
(z) &= \sum_{\substack{(a,b) \in \mathbb{F}_{q^2} \\ a^q + a = b^{q+1}}} P_{(a,b,0)} \; - \; q^3 Q_\infty.
\end{aligned}$$

In each summation, the point $P_{(0,0,0)} = Q_0$ occurs. For future reference we also note that for $k \in \mathbb{Z}$ and $\ell \geq 0, m \geq 0$ we have

$$(x^k y^\ell z^m) = \left(k(q^e+1) + \ell\frac{q^e+1}{q+1} + m\right) Q_0 - \left(k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3\right) Q_\infty + E, \quad \text{(A.1)}$$

with E an effective divisor with support disjoint from $\{Q_0, Q_\infty\}$. The above information is enough to determine that $H(Q_\infty)$, the Weierstrass semigroup of Q_∞, is generated by $q^3, q\frac{q^e+1}{q+1}$ and q^e+1.

Theorem A.6 ([GÖS13], Cor.3.5)**.** *We have* $H(Q_\infty) = \left\langle q^3,\, q\frac{q^e+1}{q+1},\, q^e+1 \right\rangle$.

A direct consequence of this theorem is a description of $\Gamma(Q_\infty)$, the set of gaps of $H(Q_\infty)$.

Corollary A.7. *The set* $\Gamma(Q_\infty)$ *of gaps of* $H(Q_\infty)$ *is given by*

$$\Big\{ k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3 \mid 0 \leq \ell \leq q, 0 \leq m < \frac{q^e+1}{q+1}, k < 0, k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3 \geq 0 \Big\}.$$

Proof. Any integer can uniquely be written in the form $k(q^e+1)+\ell q\frac{q^e+1}{q+1}+mq^3$, with k,ℓ and m integers satisfying $0\le\ell\le q, 0\le m<\frac{q^e+1}{q+1}$. To be an element of $H(Q_\infty)$ the additional requirement is simply that $k\ge 0$. Since $\Gamma(Q_\infty)=\mathbb{N}\setminus H(Q_\infty)$, the corollary follows. □

We now give a further consequence of Theorem A.6: a complete description of the ring of functions that are regular outside Q_∞; that is to say, the functions having no poles except possibly at Q_∞. The next result follows directly from the similar statement in [GÖS13, Prop. 3.4].

Corollary A.8. *The ring $R(Q_\infty)$ of functions in $\mathbb{F}_{q^{2e}}(\chi_e)$ regular outside Q_∞ is given by $\mathbb{F}_{q^{2e}}[x,y,z]$.*

For the AG codes that we wish to study, we in fact need to understand a larger ring of functions, allowing functions that may have a pole in Q_∞ as well as Q_0. An explicit description of this ring is given in the following corollary.

Corollary A.9. *The ring $R(Q_0,Q_\infty)$ of functions in $\mathbb{F}_{q^{2e}}(\chi_e)$ regular outside $\{Q_0,Q_\infty\}$ is given by $\mathbb{F}_{q^{2e}}[x,x^{-1},y,z]$.*

Proof. It is clear from Eq. (A.1) that any function in $\mathbb{F}_{q^{2e}}[x,x^{-1},y,z]$ is regular outside $\{Q_0,Q_\infty\}$. Conversely, if a function f has no pole outside $\{Q_0,Q_\infty\}$, then for a suitably chosen exponent k, the function x^kf has no pole outside Q_∞. Hence $x^kf\in R(Q_\infty)$. Corollary A.8 implies that $f\in\mathbb{F}_{q^{2e}}[x,x^{-1},y,z]$. □

Corollary A.9 implies that the ring $R(Q_0,Q_\infty)$ has a natural module structure over $\mathbb{F}_{q^{2e}}[x,x^{-1}]$. When viewed as such a module, $R(Q_0,Q_\infty)$ is free of rank q^e+1 with basis $y^\ell z^m$ where $0\le\ell<q+1$ and $0\le m<\frac{q^e+1}{q+1}$. For $e=1$, the above theorem and the mentioned consequences are well known. For $e=3$, these results are contained in [GK09, Duu11].

We now turn to the study of the two-point Weierstrass semigroup $H(Q_0,Q_\infty)$. We will determine this semigroup completely. Equation (A.1) will be used to describe the functions $f^{(i)}_{Q_0,Q_\infty}$, resp. the bijection τ_{Q_0,Q_∞}. For convenience, we will use the more compact notation $f^{(i)}_{0,\infty}$, resp. $\tau_{0,\infty}$. Similarly we write $\tau^{-1}_{0,\infty}=\tau_{\infty,0}$.

Theorem A.10. *Let $i\in\mathbb{Z}$ and write $i=-k(q^e+1)-\ell\frac{q^e+1}{q+1}-m$ for a triple $(k,\ell,m)\in\mathbb{Z}^3$ satisfying $0\le\ell<q+1$ and $0\le m<\frac{q^e+1}{q+1}$. Then $f^{(i)}_{0,\infty}=x^ky^\ell z^m$.*

Proof. By definition of $f^{(i)}_{0,\infty}$ we have $-v_{Q_0}(f_i)=i$ and $v_{Q_\infty}(f_i)=\tau_{0,\infty}(i)$. Suppose $f^{(i)}_{0,\infty}$ cannot be chosen as a monomial in x^{-1},x,y and z. Since by Corollary A.9 we have $f^{(i)}_{0,\infty}\in\mathbb{F}_{q^{2e}}[x,x^{-1},y,z]$ we can write

$$f^{(i)}_{0,\infty}=\sum_{\alpha=-N}^{M}\sum_{\beta=0}^{q}\sum_{\gamma=0}^{\frac{q^e-q}{q+1}}a_{\alpha\beta\gamma}x^\alpha y^\beta z^\gamma,$$

for integers N,M and constants $a_{k\ell m}\in\mathbb{F}_{q^{2e}}$. Note that the pole orders at Q_0 of each of the occurring monomials $x^\alpha y^\beta z^\gamma$ are distinct. Since $-v_{Q_0}(f^{(i)}_{0,\infty})=i$, this implies that there exists a uniquely determined triple (k,ℓ,m) such that $a_{k\ell m}\neq 0$ and $i=-k(q^e+1)-\ell\frac{q^e+1}{q+1}-m$, while for all other monomials $x^\alpha y^\beta z^\gamma$ occurring in $f^{(i)}_{0,\infty}$ we have

$$-\alpha(q^e+1)-\beta\frac{q^e+1}{q+1}-\gamma<i.$$

Likewise, the pole orders at Q_∞ of all of the occurring monomials $x^\alpha y^\beta z^\gamma$ are distinct. Since $-v_{Q_\infty}(f^{(i)}_{0,\infty})=\tau_{0,\infty}(i)$ there exists a uniquely determined triple (k',ℓ',m') such that $a_{k'\ell'm'}\neq 0$

and $\tau_{0,\infty}(i) = k'(q^e+1) + \ell' q\frac{q^e+1}{q+1} + m'q^3$, while for all other monomials $x^\alpha y^\beta z^\gamma$ occurring in $f_{0,\infty}^{(i)}$ we have

$$\alpha(q^e+1) + \beta q\frac{q^e+1}{q+1} + \gamma q^3 < \tau_{0,\infty}(i).$$

If $(k,\ell,m) \neq (k',\ell',m')$, the monomial $x^k y^\ell z^m$ would have pole order i in Q_0, but pole order strictly less than $\tau_{0,\infty}(i)$ in Q_∞, which gives a contradiction by the definition of $\tau_{0,\infty}$. Hence we may take $f_{0,\infty}^{(i)} = x^k y^\ell z^m$. □

Corollary A.11. *Let $i \in \mathbb{Z}$, and let $(k,\ell,m) \in \mathbb{Z}^3$ be the unique triple such that $0 \leq \ell < q+1$, $0 \leq m < \frac{q^e+1}{q+1}$ and $i = -k(q^e+1) - \ell\frac{q^e+1}{q+1} - m$. Then*

$$\tau_{0,\infty}(i) = k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3.$$

Proof. For a given $i \in \mathbb{Z}$, the proof of Theorem A.10 implies that $f_{0,\infty}^{(i)} = x^k y^\ell z^m$ for a uniquely determined triple $(k,\ell,m) \in \mathbb{Z}^3$ such that $-i = k(q^e+1)+\ell\frac{q^e+1}{q+1}+m$, $0 \leq \ell \leq q$ and $0 \leq m < \frac{q^e+1}{q+1}$. Hence $\tau_{0,\infty}(i) = -v_{Q_\infty}(f_i) = k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3$ as claimed. □

It is interesting to see what can be said about the Weierstrass semigroups $H(Q_0)$ and $H(Q_\infty)$ using the above tools. First of all, it should be noted that for $e = 1$ and $e = 3$, it is well known that $H(Q_0) = H(Q_\infty)$. The reason is that there exists an automorphism interchanging Q_0 to Q_∞. For $e > 3$, the place Q_∞ is fixed by any automorphism of χ_e and in fact $H(Q_0)$ and $H(Q_\infty)$ were shown to be distinct in [GÖS13]. However, for any $e \geq 1$ the points of the form $P_{(a,b,0)}$ fall within the same orbit under the action of the subgroup of the automorphism group of χ_e consisting of automorphisms fixing Q_∞. This means that later on in the article, one can always exchange the point Q_0 with any point of the form $P_{(a,b,0)}$.

It is easy to describe the set $\Gamma(Q_0)$, but it should first be noted that the precise structure of $H(Q_0)$ (and hence of $\Gamma(Q_0)$) has already been determined in [BMZ18]. For the sake of completeness and since our description of $\Gamma(Q_0)$ is rather compact, we give the following corollary.

Corollary A.12. *The set $\Gamma(Q_0)$ of gaps of the Weierstrass semigroup $H(Q_0)$ of the point Q_0 on χ_e is given by*

$$\Big\{-k(q^e+1) - \ell\frac{q^e+1}{q+1} - m \mid 0 \leq \ell \leq q, 0 \leq m < \frac{q^e+1}{q+1}, k < 0, k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3 \geq 0\Big\}.$$

Proof. We denote by $\Gamma(Q_\infty)$ (resp. $\Gamma(Q_0)$) the set of gaps of Q_∞ (resp. Q_0). It is well known that $\tau_{\infty,0}$ gives rise to a bijection from $\Gamma(Q_\infty)$ to $\Gamma(Q_0)$. Since by Corollary A.7 we have

$$\Gamma(Q_\infty) = \Big\{k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3 \mid 0 \leq \ell \leq q, 0 \leq m < \frac{q^e+1}{q+1}, k < 0, k(q^e+1) + \ell q\frac{q^e+1}{q+1} + mq^3 \geq 0\Big\},$$

Corollary A.11 implies that $\Gamma(Q_0)$ is as stated. □

A.4 Two-point AG codes on the generalized GK curve.

Since the curves χ_e are maximal, they are good candidates to be used for the construction of error-correcting codes. Let the divisor D be the sum of all the rational points of χ_e different from Q_0 and Q_∞. If the support of a divisor G consists of one rational point not in $\mathrm{supp}(D)$, the code $C_L(D,G)$ is called a one-point AG code. Similarly, if $G = a_1Q_0 + a_2Q_\infty$, the code $C_L(D,G)$ is called a two-point code. By slight abuse of notation, the dual of a one-point code (resp. two-point code) are sometimes also called one-point (resp. two-point) codes, but we will only use the terminology for the codes $C_L(D,G)$. The main reason we do this is that for any divisor G with $\mathrm{supp}(G) \cap \mathrm{supp}(D) = \emptyset$, there exists a divisor H with $\mathrm{supp}(H) \cap \mathrm{supp}(D) = \emptyset$ such that $C_L(D,G)^\perp = C_L(D,H)$, but even if the support of G is small, the support of H might be large. Therefore, in our sense of the word, $C_L(D,G)^\perp = C_L(D,H)$ may not be a one-point or two-point code, even if $C_L(D,G)$ is.

Duals of one-point codes with defining divisor of the form aQ_∞ or aQ_0 on the generalized GK curves were investigated in [FG10, BMZ18]. As we will see below, their analysis of the parameters of these codes has direct implications for the study of the one-point codes $C_L(D+Q_0, aQ_\infty)$ and $C_L(D+Q_0, aQ_0)$ themselves. Duals of two-point AG codes on the GK curve (i.e. $e=3$) have been studied in [CT16b]. As we will see, their analysis can be refined significantly, yielding more excellent AG codes. Furthermore, the case $e > 3$ will be considered.

The theorem used in [CT16b] (which comes from [Mat01, Thm. 2.1]) allows one to improve the Goppa bound by one for the minimum distance of a nontrivial code defined on an algebraic curve χ of the form $C_L(D, (a_1+b_1-1)Q_1 + (a_2+b_2-1)Q_2)$, where Q_1 and Q_2 are rational points not in $\mathrm{supp}(D)$. Here, the four nonnegative integers a_1, a_2, b_1, b_2 should satisfy

1. $a_1 \geq 1$,
2. $L((a_1-1)Q_1 + a_2Q_2) = L(a_1Q_1 + a_2Q_2)$,
3. $(b_1, b_2-1-t) \in \Gamma(Q_1;Q_2)$ for all t satisfying $0 \leq t \leq \min\{b_2-1, 2g-1-a_1-a_2\}$.

In the next theorem we show that the order bound in the same situation improves upon the Goppa bound by at least one as well. Therefore, our results will automatically include all results in [CT16b] as a special case. First note that $L((a_1-1)Q_1+a_2Q_2) = L(a_1Q_1+a_2Q_2)$ is equivalent to saying that $\tau_{Q_1,Q_2}(a_1) > a_2$ by Theorem A.4. Further the condition that $(b_1, b_2-1-t) \in \Gamma(Q_1;Q_2)$ for all t satisfying $0 \leq t \leq \min\{b_2-1, 2g-1-a_1-a_2\}$ is equivalent to the statement that $\tau_{Q_1,Q_2}(b_1) \geq b_2$ or $\tau_{Q_1,Q_2}(b_1) < b_2-1-\min\{b_2-1, 2g-1-a_1-a_2\}$. With these reformulations in mind, we now show that Proposition A.1 implies [Mat01, Thm. 2.1].

Theorem A.13. *Let a_1, a_2, b_1, b_2 be nonnegative integers and write $G := (a_1+b_1-1)Q_1 + (a_2+b_2-1)Q_2$. Further suppose that $\tau_{Q_1,Q_2}(a_1) > a_2$.*

1. *If $\tau_{Q_1,Q_2}(b_1) \geq b_2$, then $\nu(Q_1; (b_2-1)Q_2, a_2Q_2) > \deg(G) - 2g(\chi) + 2$.*
2. *If $\tau_{Q_1,Q_2}(b_1) < b_2 - 1 - \min\{b_2-1, 2g-a_1-a_2\}$, then $\nu(Q_2; b_1Q_1, (a_1-1)Q_1) > \deg(G) - 2g(\chi) + 2$.*

In particular, in either case the minimum distance of the code $C_L(D,G)^\perp$ is at least $\deg(G) - 2g + 3$.

Proof. If $\tau_{Q_1,Q_2}(b_1) \geq b_2$, then $a_1 \in \Gamma(Q_1; a_2Q_2)$ and $b_1 \in \Gamma(Q_1; (b_2-1)Q_2)$. Combining Remark 23 with (the proof of) Lemma A.2 we see that $\nu(Q_1; (a_1-1)Q_1 + a_2Q_2, b_1Q_1 + (b_2-1)Q_2) > \deg(G) - 2g(\chi) + 2$. Indeed, the term $q_{Q_1,(a_1-1)Q_1+a_2Q_2}(t) q_{Q_1,b_1Q_1+(b_2-1)Q_2}(t)$ will contribute to the coefficient of $t^{v_Q(G)+1}$ with at least 1. From Proposition A.1 and the Goppa bound applied to $C_L(D, G+Q_1)^\perp$, we see that $C_L(D,G)$ has minimum distance at least $\deg(G) - 2g + 3$.

If $\tau_{Q_1,Q_2}(b_1) < b_2$ and $\min\{b_2 - 1, 2g - 1 - a_1 - a_2\} = b_2 - 1$, then we have $(b_1, b_2 - 1 - t) \in \Gamma(Q_1; Q_2)$ by assumption for all t satisfying $0 \leq t \leq b_2 - 1$. This implies that $\tau_{Q_1,Q_2}(b_1) < 0$. However, since $\tau_{Q_1,Q_2}(0) = 0$ and $b_1 \geq 0$, we see that $(b_1, 0) \in H(Q_1, Q_2)$, giving a contradiction. This situation can therefore not occur.

If $\tau_{Q_1,Q_2}(b_1) < b_2$ and $\min\{b_2 - 1, 2g - 1 - a_1 - a_2\} = 2g - 1 - a_1 - a_2$, then similarly as before we have $\tau(b_1) < b_2 - 2g + a_1 + a_2$. This implies that $b_2 - 1 - t \in \Gamma(Q_2; b_1Q_1)$ for all t satisfying $0 \leq t \leq 2g - 1 - a_1 - a_2$. On the other hand, we have $\tau_{Q_1,Q_2}(a_1) \in \Gamma(Q_2; (a_1 - 1)Q_1)$. Now using Remark 24, note that

$$b_2 - 2g - a_1 - a_2 \leq a_2 + b_2 - \tau_{Q_1,Q_2}(a_1) \leq b_2 - 1.$$

Hence $a_2 + b_2 - \tau_{Q_1,Q_2}(a_1) \in \Gamma(Q_2; b_1Q_1)$. The term $q_{Q_2,(a_1-1)Q_1+a_2Q_2}(t)q_{Q_2,b_1Q_1+(b_2-1)Q_2}(t)$ will then contribute to the coefficient of $t^{v_Q(G)+1}$ with at least 1. Hence $\nu(Q_2; (a_1 - 1)Q_1 + a_2Q_2, b_1Q_1 + (b_2 - 1)Q_2) > \deg(G) - 2g(\chi) + 2$. Proposition A.1 and the Goppa bound applied to $C_L(D, G + Q_2)^\perp$, imply that $C_L(D, G)^\perp$ has minimum distance at least $\deg(G) - 2g + 3$. □

With the above theorem in place, we could in principle start to compute our lower bound on the minimum distance of the duals of two-point codes. Before doing that, we show in the remainder of this section that duals of two-point codes on the generalized GK curve are closely related to two-point codes. This means that our bounds not only can be applied to the duals of two-point codes, but to two-point codes themselves as well. In order to do this, we need to understand the structure of the rational point of χ_e. The structure of these points is described explicitly in [ABQ09, GÖS13]. Since e is odd, we write $e = 2t + 1$ for some nonnegative integer t. Apart from Q_∞, all rational points are of the form $P_{(a,b,c)}$. There are q^3 rational places of the form $P_{(a,b,0)}$ and $q^3(q^e + 1)(q^{e-1} - 1)$ of the form $P_{(a,b,c)}$ with $c \neq 0$. Both for $c = 0$ and $c \neq 0$, the place $P_{(a,b,c)}$ is unramified in the degree q^3 extension $\mathbb{F}_{q^{2e}}(\chi_e)/\mathbb{F}_{q^{2e}}(z)$ by [ABQ09, GÖS13]. This means that there are exactly $(q^e + 1)(q^{e-1} - 1)$ possible nonzero values of $c \in \mathbb{F}_{q^{2e}}$ giving rise to q^3 rational places of $\mathbb{F}_{q^{2e}}(\chi_e)$ if the form $P_{(a,b,c)}$. By [ABQ09] these values of c are exactly the roots of the polynomial

$$f := 1 + \sum_{i=0}^{t-1} z^{\frac{q^e+1}{q+1}\left(q^{2i+2}-1+q^e-q\right)} + \sum_{i=0}^{t-1} z^{\frac{q^e+1}{q+1}\left(q(q^{2i+2}-1)\right)}.$$

Denoting, as before, by D the divisor which is the sum of all rational points distinct from Q_0 and Q_∞, this implies that

$$(zf) = Q_0 + D - q^3(q^{2e-1} - q^e + q^{e-1})Q_\infty. \tag{A.2}$$

This expression is very useful to determine whether or not two two-point codes are equal. This comes in very handy, when computing the order bound using Proposition A.1, since one should only apply this proposition if the codes $C_L(D, G + Q)$ and $C_L(D, G)$ are distinct. We give a criterion in the following lemma.

Lemma A.14. *Let χ_e be the generalized GK curve over $\mathbb{F}_{q^{2e}}$ and let the divisor D be the sum of all its rational places different from Q_0 and Q_∞. Further let $G = a_1Q_0 + a_2Q_\infty$ and $Q \in \{Q_0, Q_\infty\}$. Then*

$$\dim(C_L(D, G)) = \dim(L(G)) - \dim(L(G + Q_0 - q^3(q^{2e-1} - q^e + q^{e-1})Q_\infty)).$$

Furthermore $C_L(D, G + Q) = C_L(D, G)$ if and only if

$$\dim(L(G + Q)) - \dim(L(G + Q + Q_0 - q^3(q^{2e-1} - q^e + q^{e-1})Q_\infty)) = \dim(L(G)) - \dim(L(G + Q_0 - q^3(q^{2e-1} - q^e + q^{e-1})Q_\infty)).$$

Proof. First, note that $\dim(C_L(D,G)) = \dim(L(G)) - \dim(L(G-D))$. Since $\dim(L(G-D)) = \dim(L(G-D+(zf)))$, the first part of the lemma follows from Eq. (A.2). Now applying this formula to compute the dimension of $\dim(C_L(D,G+Q))$, the lemma follows. □

Since we know the map $\tau_{0,\infty}$ explicitly, it is very easy to check the above criterion using Theorem A.4.

The function zf from equation (A.2) is also useful when identifying dual two-point codes and two-point codes. The standard way to identify the dual of an AG code $C_L(D,G)^\perp$ with a code of the form $C_L(D,H)$ is to identify a differential on the curve with simple poles in all evaluation points and residues in these points equal to 1. Equation (A.2) implies that the differential $\omega := \frac{1}{fz}dz$ has simple poles and nonzero residue in all rational points of the form $P_{(a,b,c)}$. More precisely, using the defining equations of the curve χ_e and equation (A.2), we obtain that

$$(\omega) = -Q_0 - D + (q^{2e+2} - q^{e+3} + 2q^{e+2} - q^e + q^2 - 1)Q_\infty. \tag{A.3}$$

Since the differential ω has simple poles in all the points in D, its residues at those points will all be nonzero. However, it turns out that in general these residues are not all 1. Nonetheless, we can identify an explicit relation between the class of codes $C_L(D,G)$ and $C_L(D,G)^\perp$. Two codes C_1 and C_2 are called equivalent up to column multipliers, which we denote by $C_1 \cong C_2$, if there exist nonzero elements $a_1,\ldots,a_n$ such that the map $\phi : \mathbb{F}_{q^e} \to \mathbb{F}_{q^e}$ defined by $\phi(v_1,\ldots,v_n) = (a_1v_1,\ldots,a_nv_n)$ satisfies $\phi(C_1) = C_2$. Note that the basic parameters of such codes C_1 and C_2, such as the minimum distance, are the same.

Proposition A.15. *Let χ_e be the generalized GK curve over $\mathbb{F}_{q^{2e}}$ and let the divisor D be the sum of all its rational places different from Q_0 and Q_∞. Further let $G = a_1Q_0 + a_2Q_\infty$. Then $C_L(D,G)^\perp \cong C_L(D,H)$, where*

$$H = -(a_1+1)Q_0 + (q^{2e+2} - q^{e+3} + 2q^{e+2} - q^e + q^2 - 1 - a_2)Q_\infty.$$

Proof. Let $h := (zf)'$ be the derivative of zf with respect to the variable z. Then the differential $\eta = h\omega = (zf)'/(zf)dz$ has simple poles in all the rational points $P_{(a,b,c)}$ of χ_e. Moreover, in each of those points, the residue of η is equal to 1. Therefore the standard theory of AG codes implies that $C_L(D,G)^\perp = C_L(D,H')$, with $H' = D - G + (\eta) = D - G + (h) + (\omega)$. Since zf has simple roots only, its derivative h is nonzero in Q_0 and the points in D. Hence the codes $C_L(D,H') \cong C_L(D,H)$, with $H = D - G + (\omega)$. Explicitly, the column multipliers are given by $(h(P))_{P\in\mathrm{supp}(D)}$. Using Eq. (A.3), the lemma follows. □

This proposition implies that the class of two-point codes $C_L(D, a_1Q_0 + a_2Q_\infty)$ on the generalized GK curve is essentially the same as the class of codes of the form $C_L(D, a_1Q_0 + a_2Q_\infty)^\perp$. In particular, the bounds on the minimum distance of codes of the form $C_L(D, a_1Q_0 + a_2Q_\infty)^\perp$ will imply bounds for the minimum distance of codes of the form $C_L(D, a_1Q_0 + a_2Q_\infty)$. Note that the above proof shows that $C_L(D,G)^\perp = C_L(D,H')$, where

$$H' = -(a_1+1)Q_0 + (q^{2e+2} - q^{e+3} + 2q^{e+2} - q^e + q^2 - 1 - a_2)Q_\infty + ((zf)').$$

However, for our purposes this is less useful, since the divisor of $(zf)'$ may contains other points of χ_e. Therefore $C_L(D,H')$ is in general not a two-point code, even if $C_L(D,G)$ is.

Using the same differential ω as above, we obtain the following corollary for one-point AG codes on the generalized GK curve.

Corollary A.16. *Let χ_e be the generalized GK curve over $\mathbb{F}_{q^{2e}}$ and let the divisor D be the sum of all its rational places different from Q_0 and Q_∞. Further let $G = aQ_\infty$. Then $C_L(D+Q_0,G)^\perp \cong C_L(D+Q_0,H)$, where*

$$H = (q^{2e+2} - q^{e+3} + 2q^{e+2} - q^e + q^2 - 1 - a)Q_\infty.$$

A.5 Computation of the order bound and results

Now that all the theoretical tools are in place, all that is left is to give the lower bounds that we obtain using the above theory as well as state the improvements on the MinT tables [Min]. We would also like to explain briefly how we computed these bounds. The given algorithm works for any of the generalized GK curves χ_e. We have already seen that the explicit description of the bijection $\tau_{0,\infty}$ in Corollary A.11, implies that for $G = a_1Q_0 + a_2Q_\infty$ and $Q \in \{Q_0, Q_\infty\}$, it is computationally easy to determine:

1. The dimension of $L(G)$, see Theorem A.4.
2. The dimension of $C_L(D, G)$ (and hence of $C_L(D, G)^\perp$), see Lemma A.14.
3. The sets $H(Q; G)$ (and hence the value of $\nu(Q; G)$), see Corollary A.5.

What is left is to describe how to find the best recursive use of Proposition A.1. We do this efficiently by using a dynamic programming approach, in the form of a backtracking algorithm which starts with large degree divisors, where the order bound coincides with the Goppa bound and is easy to compute, and then backtracks to smaller degree divisors. A pseudo-code description is given in Algorithm 9. For $q = 2$ and $e = 3$ our results supplement and improve those in [CT16b], as indicated in Table A.1.

Algorithm 9: OrderBoundTable

Input : parameters q and e.

Output: array containing the order bound for $C_L(D, aQ_0 + bQ_\infty)^\perp$, for $a, b \in \mathbb{Z}_{\geq 0}$ whose sum $a + b$ is at most Δ, a bound beyond which the order bound and the Goppa bound coincide.

$g_e := (q-1)(q^{e+1} + q^e - q^2)/2$ // genus of χ_e
$N_e := q^{2e+2} - q^{e+3} + q^{e+2} + 1$ // number of rational points
$\Delta := N_e + 2g_e$ // if larger degree, order bound coincides with Goppa bound
orderBound := two-dimensional array of size $(\Delta + 1) \times (\Delta + 1)$
for a ***from*** Δ ***to*** 0 **do**
  orderBound$[a, \Delta - a] = \Delta - 2g_e + 2$ // Goppa bound for degree Δ
for δ ***from*** $\Delta - 1$ ***to*** 0 **do**
  // backtrack: iterate on decreasing degree $\delta = a + b$
  for a ***from*** 0 ***to*** δ **do**
    $b := \delta - a$
    /* Walk on the horizontal edge */
    $U := \{$Weierstrass semigroup at $Q_0\} \cap \{0, \ldots, \delta + 1\}$
    $V := \{bQ_\infty$-non-gaps at $Q_0\} \cap \{-b, \ldots, a + 1\}$
    $\overline{U} := \{a + 1 - u, u \in U\}$
    $w :=$ cardinality of $\overline{U} \cap V$
    if $w \neq 0$ *and* $\dim(C_L(D, aQ_0 + bQ_\infty)) \neq \dim(C_L(D, (a+1)Q_0 + bQ_\infty))$ **then**
      hbound := $\min(w,$ orderBound$[a + 1, b])$
    else
      hbound := orderBound$[a + 1, b]$
  /* Walk on the vertical edge */
  $U := \{$Weierstrass semigroup at $Q_\infty\} \cap \{0, \ldots, \delta + 1\}$
  $V := \{aQ_0$-non-gaps at $Q_\infty\} \cap \{-a, \ldots, b + 1\}$
  $\overline{U} := \{b + 1 - u, u \in U\}$
  $w :=$ cardinality of $\overline{U} \cap V$
  if $w \neq 0$ *and* $\dim(C_L(D, aQ_0 + bQ_\infty)) \neq \dim(C_L(D, aQ_0 + (b+1)Q_\infty))$ **then**
    vbound := $\min(w,$ orderBound$[a, b + 1])$
  else
    vbound := orderBound$[a, b + 1]$
  /* Combine the obtained bounds */
  orderBound$[a, b] := \max($hbound, vbound$)$

n	k	(a_1, a_2)	d_{2P}	d_{1P}	n	k	(a_1, a_2)	d_{2P}	d_{1P}
223	222	$(0, 0)$	2	2	223	203	$(22, 7)$	**13**	12
223	221	$(6, 0)$	2	2	223	202	$(22, 8)$	13	12
223	220	$(8, 0)$	2	2	223	201	$(31, 0)$	14	14
223	219	$(11, 0)$	3	3	223	200	$(28, 4)$	**15**	14
223	218	$(13, 0)$	3	3	223	199	$(28, 5)$	16	15
223	217	$(14, 0)$	3	3	223	198	$(28, 6)$	17	16
223	216	$(8, 7)$	4	3	223	197	$(28, 7)$	**18**	17
223	215	$(16, 0)$	4	4	223	196	$(28, 8)$	**19**	18
223	214	$(17, 0)$	5	5	223	195	$(37, 0)$	20	20
223	213	$(19, 0)$	6	6	223	10	$(215, 7)$	205	204
223	212	$(20, 0)$	6	6	223	9	$(216, 7)$	206	206
223	211	$(21, 0)$	6	6	223	8	$(217, 7)$	207	206
223	210	$(22, 0)$	6	6	223	7	$(218, 7)$	208	207
223	209	$(19, 4)$	8	6	223	6	$(219, 7)$	209	208
223	208	$(19, 5)$	9	6	223	5	$(220, 7)$	211	209
223	207	$(19, 6)$	9	7	223	4	$(222, 7)$	214	212
223	206	$(19, 7)$	10	8	223	3	$(231, 0)$	215	215
223	205	$(19, 8)$	11	9	223	2	$(226, 6)$	217	214
223	204	$(28, 0)$	12	12	223	1	$(228, 6)$	223	220

Table A.1: Table A.1 gives for $q = 2$, $e = 3$, $n = 223$ and fixed k a value of (a_1, a_2) for which the estimate d_{2P} for the minimum distance of the code $C_L(D, a_1Q_0 + a_2Q_\infty)^\perp$ is largest. It is compared to the corresponding estimate d_{1P} for the minimum distance of a code of the same length and dimension of the form $C_L(D, a_1Q_0)^\perp$ or $C_L(D, a_2Q_\infty)^\perp$. The four entries in boldface indicate new improvements on the MinT tables. In [CT16b] it was already shown that the entries for $k \in \{198, 199\}$ improve the MinT [Min] table, which is why we have not put those two values in boldface.

université PARIS-SACLAY

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre :Étude de la sécurité de certaines clés compactes pour le schéma de McEliece utilisant des codes géométriques

Mots clés : cryptographie à clé publique, codes correcteurs d'erreurs, géométrie algébrique

Résumé : En 1978, McEliece introduit un schéma de chiffrement à clé publique issu de la théorie des codes correcteurs d'erreurs. L'idée du schéma de McEliece est d'utiliser un code correcteur dont la structure est masquée, rendant le décodage de ce code difficile pour toute personne ne connaissant pas cette structure. Le principal défaut de ce schéma est la taille de la clé publique. Dans ce contexte, on se propose d'étudier l'utilisation de codes dont on connaît une représentation compacte, en particulier le cas de codes quais-cyclique ou quasi-dyadique.

Les deux familles de codes qui nous intéressent dans cette thèse sont: la famille des codes alternants et celle des sous–codes sur un sous–corps de codes géométriques. En faisant agir un automorphisme σ sur le support et le multiplier des codes alternants, on sait qu'il est possible de construire des codes alternants quasi-cycliques. On se propose alors d'estimer la sécurité de tels codes à l'aide du *code invariant*. Ce sous–code du code public est constitué des mots du code strictement invariant par l'automorphisme σ. On montre ici que la sécurité des codes alternants quasi-cyclique se réduit à la sécurité du code invariant. Cela est aussi valable pour les sous–codes sur un sous–corps de codes géométriques quasi-cycliques. Ce résultat nous permet de proposer une analyse de la sécurité de codes quasi-cycliques construit sur la courbe Hermitienne. En utilisant cette analyse nous proposons des clés compactes pour la schéma de McEliece utilisant des sous-codes sur un sous-corps de codes géométriques construits sur la courbe Hermitienne.

Le cas des codes alternants quasi-dyadiques est aussi en partie étudié. En utilisant le code invariant, ainsi que le *produit de Schur* et le *conducteur* de deux codes, nous avons pu mettre en évidence une attaque sur le schéma de McEliece utilisant des codes alternants quasi-dyadique de degré 2. Cette attaque s'applique notamment au schéma proposé dans la soumission DAGS, proposé dans le contexte de l'appel du NIST pour la cryptographie post-quantique.

Title: On the security of short McEliece keys from algebraic and algebraic geometry codes with automorphisms

Keywords: public-key cryptography, code-based cryptography, algebraic geometry codes

Abstract: In 1978, McEliece introduce a new public key encryption scheme coming from errors correcting codes theory. The idea is to use an error correcting code whose structure would be hidden, making it impossible to decode a message for anyone who do not know a specific decoding algorithm for the chosen code. The McEliece scheme has some advantages, encryption and decryption are very fast and it is a good candidate for public-key cryptography in the context of quantum computer. The main constraint is that the public key is too large compared to other actual public-key cryptosystems. In this context, we propose to study the using of some quasi-cyclic or quasi-dyadic codes.

In this thesis, the two families of interest are: the family of alternant codes and the family of subfield subcode of algebraic geometry codes. We can construct quasi-cyclic alternant codes using an automorphism which acts on the support and the multiplier of the code. In order to estimate the securtiy of these QC codes we study the *invariant code*. This invariant code is a smaller code derived from the public key. Actually the invariant code is exactly the subcode of codewords fixed by the automorphism σ. We show that it is possible to reduce the key-recovery problem on the original quasi-cyclic code to the same problem on the invariant code. This is also true in the case of QC algebraic geometry codes. This result permits us to propose a security analysis of QC codes coming from the Hermitian curve. Moreover, we propose compact key for the McEliece scheme using subfield subcode of AG codes on the Hermitian curve.

The case of quasi-dyadic alternant code is also studied. Using the invariant code, with the *Schur product* and the *conductor* of two codes, we show weaknesses on the scheme using QD alternant codes with extension degree 2. In the case of the submission DAGS, proposed in the context of NIST competition, an attack exploiting these weakness permits to recover the secret key in few minutes for some proposed parameters.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France