Efficient Decoding of Cyclic Codes and Applications in Cryptography

Dissertation
zur
Erlangung der naturwissenschaftlichen Doktorwürde
(Dr. sc. nat.)
vorgelegt der
Mathematisch-naturwissenschaftlichen Fakultät
der
Universität Zürich
von
Davide Schipani
aus
Italien

Promotionskomitee
Prof. Dr. Joachim Rosenthal (Vorsitz)
Prof. Dr. Ing. Michele Elia
Prof. Dr. Juan Antonio Lopez-Ramos

Zürich, 2012

To my family

Abstract

Cyclic codes, like BCH or Reed-Solomon codes, are very widespread and important tools to enable error-correction in many current communication systems. The standard algebraic decoding allows to efficiently use codes whose length is not too large, while for larger lengths the computational costs would be excessive, also in comparison with the iterative decoding algorithms for LDPC codes, for example.
Aim of this dissertation is to show that further computational gains in the algebraic decoding of cyclic codes are possible, so that it can become competitive also for very large lengths. A new decoding method is therefore presented, which makes use of a novel polynomial evaluation algorithms for finite fields instead of the classic Horner scheme, and of Cantor-Zassenhaus polynomial factorization algorithm together with Shanks' algorithm instead of the traditional Chien search. Cantor-Zassenhaus algorithm has been further deeply analysed, in order to obtain deterministic estimates on the maximum number of attempts needed to split a polynomial of a fixed degree or on the minimum degree such that a polynomial is split with one or two attempts.
The complexity of decoding a t-error correcting code of length n can thus be reduced from O(tn) to O($t\sqrt{n}$) for BCH codes or O($t\sqrt{n}\log n$) for Reed-Solomon codes.
We have also addressed in particular two applications where such computational improvements would be desirable: a variant of the McEliece cryptosystem which requires multiple decoding attempts to achieve a higher security level, and some code-based solutions for the secure storage of biometric passwords which involve codes with very large length.
In Appendix we have also included some more detailed analysis and results on Gauss sums and additive decompositions by multiplicative characters that have been exploited within the body of the thesis.

Zusammenfassung

Zyklische Codes, wie BCH oder Reed-Solomon-Codes, sind sehr weit verbreitet und wichtige Instrumente, die die Fehlerkorrektur in vielen aktuellen Kommunikations-Systeme ermöglichen. Die gewöhnliche algebraische Decodierung ermöglicht es, Codes effizient zu benutzen, deren Länge nicht zu gross ist, während für grössere Längen der Rechenaufwand übertrieben wäre, auch im Vergleich mit den iterativen Decodier-Algorithmen für z.B. LDPC Codes.
Ziel dieser Dissertation ist es zu zeigen, dass weitere rechnerische Gewinne in das algebraische Decodierung von zyklischen Codes möglich ist, so dass sie auch bei sehr grossen Längen wettbewerbsfähig werden. Ein neues Decodierverfahren wird deshalb vorgestellt, das einen neuartigen Polynom Evaluierung-Algorithmus für endliche Körper anstelle des klassischen Horner-Schemas und den Cantor-Zassenhaus Polynom-Faktorisierung-Algorithmus zusammen mit Shanks Algorithmus anstelle der herkömmlichen Chien-Suche nutzt.
Desweiteren wurde der Cantor-Zassenhaus Algorithmus analysiert, um deterministische Schätzungen über die maximale Anzahl der Versuche zu erhalten die man benötigt, um ein Polynom festen Grades zu zerlegen, oder über den minimalen Grad eines Polynoms, so dass dieses mit einem oder zwei Versuchen zerlegt wird.
Die Komplexität der Decodierung eines t-Fehler-korregierenden Codes der Länge n kann somit von O(tn) auf O($t\sqrt{n}$) für BCH-Codes oder O($t\sqrt{n}\log n$) für Reed-Solomon-Codes reduziert werden.
Wir betrachten zwei Anwendungen, bei denen solche rechnerische Verbesserungen wünschenswert wären: eine Variante des McEliece Kryptosystem, das mehrere Entschlüsselung Versuche erfordert, um eine höhere Sicherheitsstufe zu erreichen, und einige Code-basierte Lösungen für die sichere Speicherung von biometrischen Passwörtern, die Codes mit sehr grosser Länge beinhalten.
Des Anhang beinhaltet eine detailliertere Analyse und ein Paar Resultate zu Gaussschen Summen und additiven Zerlegungen durch multiplikative Charaktere, die innerhalb des Haupteils dieser Arbeit benutzt wurden.

Acknowledgements

First I would like to acknowledge the contribution of all the coauthors in the papers which build up the content of this thesis, namely Marco Baldi, Marco Bianchi, Franco Chiaraluce, Michele Elia and Joachim Rosenthal. Not only it has been very helpful for me to work together with them, as they have given me the opportunity to learn a lot researchwise, but it has been also and first of all a really pleasant and nice experience in my life.

I am in particular indebted to Joachim Rosenthal for supervising me throughout the doctorate, encouraging and supporting me in all situations, and Michele Elia for helping also as a sort of guide in the last years.

I would also like to thank the Institute of Mathematics of the University of Zurich for the excellent work environment, my colleagues, friends and family for their support, and everyone who has somehow helped to make all this come true.

Contents

Chapter 1

Introduction

Cyclic codes, like BCH and Reed-Solomon codes, are very widespread tools for error-correction in many important applications in the current society, where information and communication are playing a more and more relevant role. The traditional algebraic decoding algorithm for these codes is on the other hand only efficient if their length is not too large, otherwise computational complexity issues would arise. That is also one of the reason why in the last decades the focus has shifted more on iteratively decodable codes, like LDPC codes, which can also outperform in terms of gap to capacity.

Aim of this dissertation is to show that it is possible to make the classic algebraic decoding scheme more efficient and competitive also for very large lengths. We will also deal with a few current applications where such performance enhancements are in fact required.

1.1 Outline of the thesis

In Chapter 2 we present a new evaluation algorithm for polynomials in finite fields, whose complexity is competitive as soon as the degree of the polynomial is large enough compared to the field characteristic. Specifically, if n is the degree of the polynomial, the asymptotic complexity is shown to be $O(\sqrt{n})$, versus $O(n)$ of classical algorithms. While this is an important achievement in itself, we will focus in the end on applications to the syndrome computation in the decoding of cyclic codes.

Chapter 3 deals with an accurate analysis of the Cantor-Zassenhaus algorithm for factoring polynomials, which will be used in the new decoding algorithm to find the roots of the error-locator polynomial. After giving a description of the Cantor-Zassenhaus algorithm, we show that restricting to linear polynomials as test polynomials is a convenient choice and that this instance of the algorithm can be shown to be almost deterministic. In fact we can precisely compute in many scenarios how many attempts are needed to split a polynomial of given degree or the minimum degree such that a polynomial is split with one or two attempts.

Chapter 4 is devoted to the complete new decoding algorithm for cyclic codes, where the syndrome computation is performed with the new polynomial evaluation algorithm of Chapter 2, while the Chien search for the root finding step is replaced by a combination of Cantor-Zassenhaus algorithm and Shanks' algorithm. A practical numerical example of a BCH code is given for an overall picture.

In Chapter 5 we deal with a first application in cryptography, that is a new variant of the McEliece cryptosystem, where we managed to obtain a higher public key security or equivalently a smaller key size, by using MDS codes like Reed-Solomon codes. The cost to pay, although not the most relevant issue in this context, is an increased decoding complexity with respect to the traditional scheme, whereby having an efficient decoding algorithm at our disposal is extremely important. The new variant is presented in full details from the point of view of the system design, possible attacks and comparison with other variants.

Chapter 6 is concerned with a second cryptographic application, that is the authentication of users by means of biometric features, like fingerprints or irises. Because of the error-tolerance intrinsicly required by the application, error-correcting codes play a fundamental role in most of the systems which have been proposed to this end. In particular, codes of rather large lengths are required because of the typical characteristics of the biometrics in use. This makes the choice and design of the codes a crucial issue for the performance of these types of systems. In our analysis we consider two of these constructions, that are essentially error-tolerant hash functions: the fuzzy commitment scheme and a syndrome based fuzzy hash construction. We propose either the use of classic algebraic codes or that of iteratively soft-decoded LDPC codes, for which a thorough design and simulation analysis is presented.

The Appendix contains results on the computation of Gauss sums of cubic characters and additive decompositions induced by quadratic or cubic characters in finite fields. Some of these results have been used for the analysis of the Cantor-Zassenhaus algorithm in Chapter 3. Many of them are also independently relevant either by their novelty or for the methods used.

Chapter 2

On polynomial evaluation in finite fields

This chapter reports the description given in [40] of a new evaluation method for polynomials in finite fields. Its complexity is shown to be lower than that of standard techniques, when the degree of the polynomial is large enough compared to the field characteristic. Specifically, if n is the degree of the polynomial, the asymptotic complexity is shown to be $O(\sqrt{n})$, versus $O(n)$ of classical algorithms. Applications to the syndrome computation in the decoding of Reed-Solomon codes are highlighted.

2.1 Introduction

The direct evaluation of a polynomial $P(x) = a_n x^n + a_{n-1}x^{n-1} \cdots + a_0$ of degree n over a ring or a field in a point α may be performed computing the n powers α^i recursively as $\eta_{i+1} = \alpha\eta_i$, for $i = 1, \ldots, n-1$, starting with $\eta_1 = \alpha$, obtaining $P(\alpha)$ as

$$P(\alpha) = a_0 + a_1\eta_1 + a_2\eta_2 + \cdots + a_n\eta_n \ .$$

This method requires $2n-1$ multiplications and n additions. However, Horner's rule (e.g. [67]), which has become a standard, is more efficient and computes the value $P(\alpha)$ iteratively as

$$P(\alpha) = (\cdots((a_n\alpha + a_{n-1})\alpha + a_{n-2})\alpha + \cdots)\alpha + a_1)\alpha + a_0 \ .$$

This method requires n multiplications and n additions. In particular scenarios, for example when the number of possible values of the coefficients is finite, more advantageous procedures can be used, as it will be shown in this chapter.

We point out that what is usually considered in the literature to establish upper and lower bounds to the minimum number of both "scalar" and "nonscalar" multiplications refers, sometimes implicitly, to polynomials with coefficients taken from an infinite set, e.g. fields of characteristic zero, or algebraically closed fields. In fact, in [20, 89, 125], Horner's rule is proved to be optimal assuming that the field of coefficients is infinite; instead, we show that this is not the case if the coefficients belong to a finite field. Furthermore, in [90], restricting the field of coefficients to the rational field, and converting multiplications by integers into iterated sums (therefore scalar multiplications are not counted in that model), it is shown that the number of required multiplications is less than that required by Horner's rule, although the number of sums can grow unboundedly.

In the following we describe a method to evaluate polynomials with coefficients over a finite field $\mathbb{F}_{p^s}$, and estimate its complexity in terms of field multiplications and sums. However, as is customary, we only focus on the number of multiplications, that are more expensive operations than additions: in $\mathbb{F}_{2^m}$, for example, the cost of an addition is $O(m)$ in space and 1 clock in time, while the cost of a multiplication is $O(m^2)$ in space and $O(\log_2 m)$ in time ([39]). Clearly, field multiplication by look-up tables may be faster, but this approach is only possible for small values of m. We also keep track of the number of additions, so as to verify that a reduction in the number of multiplications does not bring with it an exorbitant increase in the number of additions.
Our approach exploits the Frobenius automorphism and its group properties, therefore we call it "polynomial automorphic evaluation".

The next Section describes the principle of the algorithm, with two different methods, referring to the evaluation in a point of $\mathbb{F}_{p^m}$ of a polynomial with coefficients in the prime field $\mathbb{F}_p$. The complexity is carefully estimated in order to make the comparisons self-evident. Section 2.3 concerns the evaluation in $\mathbb{F}_{p^m}$ of polynomials with coefficients in $\mathbb{F}_{p^s}$, for any $s > 1$ dividing m: different approaches will be described and their complexity compared. Section 2.4 includes examples concerning the syndrome computation in the algebraic decoding of error-correcting codes (cf. also [106] and Chapter 4), and some final remarks.

2.2 Polynomial automorphic evaluation: basic principle

Consider a polynomial $P(x)$ of degree $n > p$ over a prime field $\mathbb{F}_p$, and let α be an element of $\mathbb{F}_{p^m}$. We write $P(x)$ as a sum of p polynomials

$$P(x) = P_{1,0}(x^p) + xP_{1,1}(x^p) \cdots + x^{p-1}P_{1,p-1}(x^p) \ , \tag{2.1}$$

where $P_{1,0}(x^p)$ collects the powers of x with exponent a multiple of p and in general $x^i P_{1,i}(x^p)$ collects the powers of the form x^{ap+i}, with $a \in \mathbb{N}$ and $0 \leq i \leq p-1$.

First method. If σ is the Frobenius automorphism of $\mathbb{F}_{p^m}$ mapping γ to γ^p, which leaves invariant the elements of $\mathbb{F}_p$, we write the expression above as

$$P_{1,0}(\sigma(x)) + xP_{1,1}(\sigma(x)) + \cdots + x^{p-1}P_{1,p-1}(\sigma(x)) \ ,$$

where $P_{1,i}(y)$, $i = 0, \ldots, p-1$, are polynomials of degree $\lfloor \frac{n}{p} \rfloor$ at most. Then we may evaluate these p polynomials in the same point $\sigma(\alpha)$, and obtain $P(\alpha)$ as the linear combination

$$P_{1,0}(\sigma(\alpha)) + \alpha P_{1,1}(\sigma(\alpha)) \cdots + \alpha^{p-1}P_{1,p-1}(\sigma(\alpha)) \ .$$

A possible strategy is now to evaluate recursively the powers α^j for j from 2 up to p, and $\sigma(\alpha)^j$ for j from 2 up to $\lfloor \frac{n}{p} \rfloor$, compute the p numbers $P_{1,i}(\sigma(\alpha))$, $i = 0, \ldots, p-1$, using n sums and at most $\lfloor \frac{n}{p} \rfloor (p-2)$ products (the powers of $\sigma(\alpha)$ times their possible coefficients; the multiplications by 0 and 1 are not counted), and obtain $P(\alpha)$ with $p-1$ products and $p-1$ additions. The total number $M_p(n)$ of multiplications is

$$M_p(n) = p - 1 + \lfloor \frac{n}{p} \rfloor - 1 + (p-1) + \lfloor \frac{n}{p} \rfloor (p-2) = 2p - 3 + \lfloor \frac{n}{p} \rfloor (p-1) \ .$$

Then this procedure is more efficient compared to Horner's rule as far as $M_p(n) < n$. For example, if $p = 3$ and $n = 10$ we have $M_3(10) = 9 < 10$, and for every $n > 10$ the outlined method is always more efficient. More in general the condition is certainly satisfied whenever $n > 2p^2 - 3p$, as it can be verified by considering n written in base p.
Let us see an example in detail, for the sake of clarity, in the case $p = 3$ and $n = 10$. Suppose we want to evaluate the polynomial $f(x) = 1 + 2x + x^2 + 2x^4 + x^5 + x^6 + 2x^8 + x^{10}$ in some element $\alpha \in \mathbb{F}_{3^m}$. Writing $f(x)$ as in equation (2.1)

$$f(x) = 1 + x^6 + x(2 + 2x^3 + x^9) + x^2(1 + x^3 + 2x^6),$$

we see that it is sufficient to compute α^2, α^3, α^6, α^9, then $2\alpha^3$, $2\alpha^6$, $2\alpha^9$ (all possible coefficients needed to evaluate the three sub-polymonials), and lastly the two products by α and α^2 in front of the brackets, for a total of 9 multiplications. Note that actually $2\alpha^9$ is not needed for this particular example, but in general we always suppose to have a worst case situation. Clearly α should belong to $\mathbb{F}_{3^m}$ for some m such that $3^m > n$, so that the powers of α up to the exponent n are all different. Note, in particular, that if both the coefficients and the evaluation point are in $\mathbb{F}_p$, then the polynomial has degree at most $p - 1$, and our methods cannot be applied.

Now, the above mechanism can be iterated, and the point is to find the number of steps or iterations yielding the maximum gain. In fact we can prove the following:

Theorem 2.1. *Let L_{opt} be the number of steps of this method yielding the minimum number of products, $G_1(p, n, L_{opt})$, required to evaluate a polynomial of degree n with coefficients in $\mathbb{F}_p$. Then L_{opt} is either the integer which is nearest to $\log_p \sqrt{n(p-1)}$, or this integer minus 1, and asymptotically we have:*

$$G_1(p, n, L_{opt}) \approx 2\sqrt{n(p-1)} \ .$$

PROOF.

At step i, the number of polynomials at step $i - 1$ is multiplied by p since each polynomial $P_{i-1,h}(x)$ is partitioned into p sub-polynomials $P_{i,j+ph}(x)$, j varies between 0 and $p - 1$, of degree roughly equal to the degree of $P_{i-1,h}(x)$ divided by p, that is of degree $\lfloor \frac{n}{p^i} \rfloor$; the number of these polynomials is p^i.
After L steps we need to evaluate p^L polynomials of degree nearly $\frac{n}{p^L}$, then $P(\alpha)$ is reconstructed performing back the linear combinations with the polynomials $P_{i,h}(x)$ substituted by the corresponding values $P_{i,h}(\alpha)$. The total cost of the procedure, in terms of multiplications and additions, is composed of the following partial costs

- Evaluation of p powers of α, this step also produces $\sigma(\alpha) = \alpha^p$, and requires $p - 1$ products.
- Evaluation of $(\sigma^i(\alpha))^j$, $i = 1, \ldots, L - 1$, $j = 2, \ldots, p$; this step also produces $\sigma^L(\alpha)$, and requires $(p - 1)(L - 1)$ products.
- Evaluation of $\lfloor \frac{n}{p^L} \rfloor$ powers of $\sigma^L(\alpha)$, this step requires $\lfloor \frac{n}{p^L} \rfloor - 1$ products.
- Evaluation of p^L polynomials $P_{L,j}(x)$, of degree at most $\lfloor \frac{n}{p^L} \rfloor$, at the same point $\sigma^L(\alpha)$, this step requires n additions and $\lfloor \frac{n}{p^L} \rfloor (p - 2)$ products at most.
- Computation of $p - 1 + (p^2 - p) + \cdots + p^L - p^{L-1} = p^L - 1$ multiplications by powers of $\sigma^i(\alpha)$, $(i = 0, \ldots, L - 1)$.

- Computation of $p-1+(p^2-p)+\cdots+p^L-p^{L-1}=p^L-1$ additions.

The total number of products as a function of n, p and L is then

$$G_1(p,n,L)=\lfloor\frac{n}{p^L}\rfloor(p-1)+L(p-1)+p^L-2 \ ,$$

which should be minimized with respect to L. The values of L that correspond to local minima are specified by the conditions

$$G_1(p,n,L)\leq G_1(p,n,L-1) \quad \text{and} \quad G_1(p,n,L)\leq G_1(p,n,L+1) \ , \tag{2.2}$$

which can be explicitly written in the forms

$$\lfloor\frac{n}{p^L}\rfloor+p^{L-1}\leq\lfloor\frac{n}{p^{L-1}}\rfloor-1 \quad \text{and} \quad \lfloor\frac{n}{p^L}\rfloor-p^L\leq\lfloor\frac{n}{p^{L+1}}\rfloor+1 \ .$$

Let $\{x\}$ denote the fractional part of x, then $\lfloor x\rfloor=x-\{x\}$, thus the last inequalities can be written as

$$1+\{\frac{n}{p^{L-1}}\}-\{\frac{n}{p^L}\}\leq\frac{n}{p^{L-1}}-\frac{n}{p^L}-p^{L-1} \quad \text{and} \quad \frac{n}{p^L}-\frac{n}{p^{L+1}}-p^L\leq 1+\{\frac{n}{p^L}\}-\{\frac{n}{p^{L+1}}\} \ .$$

Since $\{x\}$ is a number less than 1, these inequalities can be relaxed to

$$0<\frac{n}{p^{L-1}}-\frac{n}{p^L}-p^{L-1} \quad \text{and} \quad \frac{n}{p^L}-\frac{n}{p^{L+1}}-p^L<2 \ ,$$

which imply

$$p^{2L}<n(p-1)p \quad \text{and} \quad n(p-1)+p<p^{2L+1}+2p^{L+1}+p=p(p^L+1)^2 \ .$$

Thus, we have the chain of inequalities

$$\frac{1}{\sqrt{p}}\sqrt{n(p-1)+p}-1<p^L<\sqrt{p}\sqrt{n(p-1)} \ ,$$

and taking the logarithm to base p we have

$$-\log_p\left(\sqrt{1+\frac{p}{n(p-1)}}+\sqrt{\frac{p}{n(p-1)}}\right)-\frac{1}{2}+\log_p\sqrt{n(p-1)}<L<\log_p\sqrt{n(p-1)}+\frac{1}{2} \ , \tag{2.3}$$

which shows that at most two values of L satisfy the conditions for a minimum, because L is constrained to be in an interval of amplitude $1+\epsilon$, with $\epsilon=\log_p\left(\sqrt{1+\frac{p}{n(p-1)}}+\sqrt{\frac{p}{n(p-1)}}\right)<1$, around the point of coordinate $\log_p\sqrt{n(p-1)}$. Therefore, the optimal value L_{opt} is either the integer which is nearest to $\log_p\sqrt{n(p-1)}$, or this integer minus 1. Hence, we have the very good asymptotic estimation $L_{opt}\approx\log_p\sqrt{n(p-1)}$, and correspondingly a very good asymptotic estimation for $G_1(p,n,L_{opt})$, that is

$$G_1(p,n,L_{opt})\approx 2\sqrt{n(p-1)} \ .$$

□

Second method. We describe here another approach exploiting the Frobenius automorphism in a different way; although it will appear to be asymptotically less efficient than the above method, it may be useful in particular situations, as shown in Section 4.

Since the coefficients are in $\mathbb{F}_p$,

$$P(x) = P_{1,0}(x^p) + xP_{1,1}(x^p) \cdots + x^{p-1}P_{1,p-1}(x^p)$$

can be written as

$$P_{1,0}(x)^p + xP_{1,1}(x)^p \cdots + x^{p-1}P_{1,p-1}(x)^p \ ,$$

where $P_{1,i}(x)$, $i = 0, \ldots, p-1$, are polynomials of degree $\lfloor \frac{n}{p} \rfloor$ at most. Then we may evaluate these p polynomial in the same point α, and obtain $P(\alpha)$ as the linear combination

$$P_{1,0}(\alpha)^p + \alpha P_{1,1}(\alpha)^p \cdots + \alpha^{p-1}P_{1,p-1}(\alpha)^p \ .$$

A possible strategy is to evaluate recursively the powers α^j for $j = 2, \ldots, \lfloor \frac{n}{p} \rfloor$, compute the p numbers $P_{1,i}(\alpha)$, $i = 0, \ldots, p-1$, using sums and at most $\lfloor \frac{n}{p} \rfloor (p-2)$ products (the powers of α times their possible coefficients), and obtain $P(\alpha)$ with p p-th powers, $p-1$ products and $p-1$ additions. The total number of multiplications is $\lfloor \frac{n}{p} \rfloor - 1 + (p-1) + pc_p + \lfloor \frac{n}{p} \rfloor (p-2)$, where c_p denotes the number of products required by a p-th power (so $c_2 = 1$ and $c_p \leq 2\lfloor \log_2 p \rfloor$). The mechanism may be iterated: after L steps we need to evaluate p^L polynomials of degree nearly $\frac{n}{p^L}$, then $P(\alpha)$ is reconstructed performing back the linear combinations with the p-powers of the polynomials $P_{i,h}(x)$ substituted by the corresponding values $P_{i,h}(\alpha)$.

Theorem 2.2. *Let L_{opt} be the number of steps of this method yielding the minimum number of products, $G_2(p, n, L_{opt})$, required to evaluate a polynomial of degree n with coefficients in $\mathbb{F}_p$. Then L_{opt} lies in an interval around $\log_p \sqrt{\frac{n(p-1)^2}{pc_p+p-1}}$ of length at most 2, and asymptotically we have:*

$$G_2(p, n, L_{opt}) \approx 2\sqrt{n(pc_p + p - 1)} \ .$$

PROOF.

The total cost of the procedure, in terms of multiplications and additions, is composed of the following partial costs

- Evaluation of $\lfloor \frac{n}{p^L} \rfloor$ powers of α.
- Evaluation of p^L polynomials $P_{L,j}(x)$, of degree at most $\lfloor \frac{n}{p^L} \rfloor$, at the same point α, this step requires n additions and $\lfloor \frac{n}{p^L} \rfloor (p-2)$ products.
- Computation of $p + p^2 + \cdots + p^L = \frac{p^{L+1}-p}{p-1}$ p-th powers.
- Computation of $p - 1 + (p^2 - p) + \cdots + p^L - p^{L-1} = p^L - 1$ multiplications by powers of α.
- Computation of $p - 1 + (p^2 - p) + \cdots + p^L - p^{L-1} = p^L - 1$ additions.

Then the total number of products as a function of n, p and L is

$$G_2(p, n, L) = \lfloor \frac{n}{p^L} \rfloor - 1 + \frac{p^{L+1} - p}{p-1} c_p + (p^L - 1) + \lfloor \frac{n}{p^L} \rfloor (p-2) \ ,$$

which should be minimized with respect to L. The optimal value of L is obtained by conditions analogous to (2.2) and arguing as above we find that this optimal value must be included in a very small interval.

Setting $y = 4n(pc_p + p - 1)^{\frac{1}{p}}$, the optimal value for L turns out to be included into an interval around $L_1 = \log_p \sqrt{\frac{n(p-1)^2}{pc_p+p-1}}$ of extremes

$$L_1 - \frac{1}{2} - \log_p \left(\sqrt{1 + \frac{1}{y}} + \sqrt{\frac{1}{y}} \right) \quad \text{and} \quad L_1 + \frac{1}{2} + \log_p \left(\sqrt{1 + \frac{1}{y}} + \sqrt{\frac{1}{y}} \right) \ ,$$

which restricts the choice of L_{opt} to at most two values. Hence, we have the very good asymptotic estimation $L_{opt} \approx \log_p \sqrt{\frac{n(p-1)^2}{pc_p+p-1}}$, and correspondingly a very good asymptotic estimation for $G_2(p, n, L_{opt})$, that is

$$G_2(p, n, L_{opt}) \approx 2\sqrt{n(pc_p + p - 1)} \ . \tag{2.4}$$

□

2.2.1 $p = 2$

The prime 2 is particularly interesting because of its occurrence in many practical applications, for example in error correction coding. In this setting an important issue is the computation of syndromes for a binary code ([69]), where it is usually needed to evaluate a polynomial in several powers of a particular value, so that an additional advantage of the proposed method may be the possibility of precomputing the powers of α.
A polynomial $P(x)$ over the binary field is simply decomposed into a sum of two polynomials by collecting odd and even powers of x as

$$P(x) = P_{1,0}(x^2) + xP_{1,1}(x^2) = P_{1,0}(x)^2 + xP_{1,1}(x)^2 \ .$$

The mechanism is then the same as for odd p with a few simplifications. The main point is that we do not need to multiply with the coefficients, which are either 0 or 1, so only sums are finally involved when evaluating the polynomials.

And to evaluate 2^L polynomials at the same point α we would need to evaluate the powers α^j for $j = 2, \ldots, \lfloor \frac{n}{2^L} \rfloor$, and then obtain each $P_{Lj}(\alpha)$ by adding those powers corresponding to non-zero coefficients; the number of additions for each polynomial is nearly $\frac{n}{2^L}$, then the total number of additions is not more than n. But the actual number of additions is much smaller if sums of equal terms can be reused, and it is upper bounded by $O(\frac{n}{ln(n)})$. This bound is a consequence of the fact that in order to evaluate 2^L polynomials of degree $h = \lfloor \frac{n}{2^L} \rfloor$ at the same point α, we have to compute 2^L sums of the form

$$\alpha^{i_1} + \cdots + \alpha^{i_m}, \ \ m \leq h$$

having at disposal the h powers α^i. We can then think of a $2^L \times \lfloor \frac{n}{2^L} \rfloor$ binary matrix to be multiplied by a vector of powers of α, and assuming $2^L \approx \frac{n}{2^L}$ (as follows from the estimation of the minimum discussed above), we may consider the matrix to be square and apply [59, Theorem 2].

2.3 Automorphic evaluation of polynomials over extended fields

This section considers the evaluation in α, an element of $\mathbb{F}_{p^m}$, of polynomials $P(x)$ of degree n over $\mathbb{F}_{p^s}$, a subfield of $\mathbb{F}_{p^m}$ larger than $\mathbb{F}_p$, thus $s > 1$ and $s|m$. There are two ways to face the problem, one way is more direct, the second way exploits the Frobenius automorphism.

First method. Let β be a generator of a polynomial basis of $\mathbb{F}_{p^s}$, i.e. β is a root of an irreducible s-degree polynomial over $\mathbb{F}_p$, expressed as an element of $\mathbb{F}_{p^m}$, then $P(x)$ can be written as

$$P(x) = P_0(x) + \beta P_1(x) + \beta^2 P_2(x) + \cdots + \beta^{s-1} P_{s-1}(x) \ , \tag{2.5}$$

where $P_i(x)$, $i = 0, \ldots, s-1$, are polynomials over $\mathbb{F}_p$ (cf. also [101]). Then $P(\alpha)$ can be obtained as a linear combination of the s numbers $P_i(\alpha)$. Thus the problem of evaluating $P(\alpha)$ is reduced to the problem of evaluating s polynomials $P_i(x)$ with p-ary coefficients followed by the computation of $s-1$ products and $s-1$ sums in $\mathbb{F}_{p^m}$.

We can state then the following:

Theorem 2.3. *The minimum number of products required to evaluate a polynomial of degree n with coefficients in $\mathbb{F}_{p^s}$ is upper bounded by $2s(\sqrt{n(p-1)} + \frac{1}{2})$.*

PROOF. The upper bound is a consequence of Theorem 1 and the comments following equation (2.5).

□

The total complexity grows asymptotically with n as $2s\sqrt{n(p-1)}$, so that a general upper bound (possibly tight) for the number of multiplications that are sufficient to compute $P(\alpha)$, when $P(x)$ has coefficients in any subfield of $\mathbb{F}_{p^m}$, is then $2m\sqrt{n(p-1)}$.

Second method. This consists in generalizing the basic principle directly. We will show the following:

Theorem 2.4. $G_1(p^s, n, L_{opt}) \approx 2\sqrt{n(p^s-1)}$ *and* $G_2(p^s, n, L_{opt}) \approx 2\sqrt{n(p^s-1)}\sqrt{1 + c_{p^{s-1}} + c_p \frac{p}{p-1}}$.

PROOF.

As for the first description, the point now is that there are $p^s - 1$ possible coefficients to be multiplied, so that we get an asymptotic complexity of $G_1(p^s, n, L_{opt}) \approx 2\sqrt{n(p^s-1)}$.

Considering the second variant, $P(x) = P_{1,0}(x^p) + xP_{1,1}(x^p) \cdots + x^{p-1}P_{1,p-1}(x^p)$ is now not directly decomposable into a sum of powers of the polynomials $P_i(x)$ since the Frobenius automorphism σ alters their coefficients. However, we can write (2.1) as

$$P_{1,0}^{-1}(x)^p + xP_{1,1}^{-1}(x)^p \cdots + x^{p-1}P_{1,p-1}^{-1}(x)^p \ ,$$

where $P_{1,i}^{-1}(x)$ stands for the polynomial obtained from $P_{1,i}(x)$ by substituting its coefficients with their transforms through σ^{-1} (and if we iterate this for k times we would consider σ^{-k}). Notice that the polynomials $P_{1,i}^{-1}(x)$ have degree at most $n_i = \frac{n-i}{p}$, and are obtained by computing a total

of n automorphisms σ^{-1}. However, in order to compute the p numbers $P_{1,i}^{-1}(\alpha)$, $i = 0, \ldots, p-1$, it is not necessary to compute the total number of n inverse automorphisms observing that

$$P_{1,i}^{-1}(\alpha) = \sum_{j=0}^{n_i} \sigma^{-1}(a_j)\alpha^j = \sigma^{-1}(\sum_{j=0}^{n_i} a_j \sigma(\alpha^j)),$$

where a_j, $j = 1, \ldots, n_i$, are the coefficients of $P_{1,i}(x)$. It is then sufficient to first evaluate $\sigma(\alpha)$, compute then $P_{1,i}(\sigma(\alpha))$ and finally apply σ^{-1}. This procedure requires the application of only p automorphisms σ^{-1} instead of n.

If we perform L steps, we need to apply σ^{-L} a number of times not greater than p^L. Notice also that what interests us in σ^L is L modulo s because σ^s is the identity automorphism in $\mathbb{F}_{p^s}$, the field of the coefficients. The number of multiplications to be minimized becomes:

$$G_2(p^s, n, L) = c_p \frac{p^{L+1} - p}{p-1} + p^L - 1 + c_{p^{s-1}} p^L + \lfloor \frac{n}{p^L} \rfloor (p^s - 1) \ ,$$

where the automorphism σ^L counts like a power with exponent p^K, with $K = L \bmod s \leq s-1$. The optimal value of L is obtained by analogues of conditions (2.2) and arguing as above we find that this optimal value must be included in a very small interval.

Setting $y = \frac{4n(p-1)(pc_p+p-1+c_{p^{s-1}}(p-1))}{p(p^s-1)}$, the optimal value for L is included into an interval around $L_2 = \log_p \sqrt{\frac{n(p-1)(p^s-1)}{pc_p+p-1+c_{p^{s-1}}(p-1)}}$ of extremes

$$L_2 - \frac{1}{2} - \log_p \left(\sqrt{1 + \frac{1}{y}} + \sqrt{\frac{1}{y}} \right) \quad \text{and} \quad L_2 + \frac{1}{2} + \log_p \left(\sqrt{1 + \frac{1}{y}} + \sqrt{\frac{1}{y}} \right) \ , \tag{2.6}$$

which restricts the choice of L_{opt} to at most two values. Hence, we have the very good asymptotic estimation $L_{opt} \approx \log_p \sqrt{\frac{n(p-1)(p^s-1)}{pc_p+p-1+c_{p^{s-1}}(p-1)}}$, and correspondingly

$$G_2(p^s, n, L_{opt}) \approx 2\sqrt{n(p^s-1)} \sqrt{1 + c_{p^{s-1}} + c_p \frac{p}{p-1}} \ .$$

□

2.4 Examples and conclusions

In some circumstances, for example when $s \approx m \approx \log_p n$, the optimal L and the consequent estimated computational cost may obscure the advantages of the new approach, suggesting the practical use of standard techniques. However, this might not be always a good strategy, as shown by the following example borrowed from the error correcting codes.
Let us consider the Reed-Solomon codes that are used in any CD-ROM, or the famous Reed-Solomon code $[255, 223, 33]$ over $\mathbb{F}_{2^8}$ used by NASA ([123]): in such applications an efficient evaluation of polynomials over $\mathbb{F}_{2^m}$ in points of the same field is of the greatest interest (see also [106]).

What we now intend to show is that in particular scenarios the proposed methods allow additional cost reductions that can be obtained by a clever choice of the parameters, for example

choosing L as a factor of m that is close to the optimal value previously found and employing some other strategies as explained below.

The idea will be illustrated considering the computation of the syndromes needed in the decoding of the above mentioned Reed-Solomon code. We will only show how to obtain the 32 syndromes; from that point onwards decoding may employ the standard Berlekamp-Massey algorithm, the Chien search to locate errors, and the Forney algorithm to compute the error magnitudes ([18]).

Let $r(x) = \sum_{i=0}^{254} r_i x^i$, $r_i \in \mathbb{F}_{2^8}$, be a received code word of the Reed-Solomon code $[255, 223, 33]$ generated by the polynomial $g(x) = \prod_{i=1}^{32}(x - \alpha^i)$, with α a primitive element of $\mathbb{F}_{2^8}$, i.e. a root of $x^8 + x^5 + x^3 + x + 1$. The aim is to evaluate the syndromes $S_j = r(\alpha^j)$, $j = 1, \ldots, 32$.

A possible approach is as follows. The power $\beta = \alpha^{17}$ is a primitive element of the subfield $\mathbb{F}_{2^4}$, it is a root of the polynomial $x^4 + x^3 + 1$, and has trace 1 in $\mathbb{F}_{2^4}$. Therefore, a root γ of $z^2 + z + \beta$ is not in $\mathbb{F}_{2^4}$ (see [68, Corollary 3.79, p.118]), but it is an element of $\mathbb{F}_{2^8}$, and every element of $\mathbb{F}_{2^8}$ can be written as $a + b\gamma$ with $a, b \in \mathbb{F}_{2^4}$. Consequently, we can write $r(x) = r_1(x) + \gamma r_2(x)$ as a sum of two polynomials over $\mathbb{F}_{2^4}$, evaluate each $r_i(x)$ in the roots α^j of $g(x)$, and obtain each syndrome $S_j = r(\alpha^j) = r_1(\alpha^j) + \gamma r_2(\alpha^j)$ with 1 multiplication and 1 sum.

Now, we choose to adopt our second variant which turns out to be very well-suited since we will actually avoid to compute any automorphism. If $p(x)$ is either $r_1(x)$ or $r_2(x)$, in order to evaluate $p(\alpha^j)$ we must consider the decomposition

$$p(x) = (\sigma^{-1}(p_0) + \sigma^{-1}(p_2)x + \cdots + \sigma^{-1}(p_{254})x^{127})^2 + x(\sigma^{-1}(p_1) + \sigma^{-1}(p_3)x + \cdots + \sigma^{-1}(p_{253})x^{126})^2 \ .$$

Now, each of the two parts can be decomposed again into the sum of two polynomials of degree at most 63, for instance

$$\sigma^{-1}(p_0) + \sigma^{-1}(p_2)x + \cdots + \sigma^{-1}(p_{254})x^{127} = (\sigma^{-2}(p_0) + \sigma^{-2}(p_4)x + \cdots + \sigma^{-2}(p_{252})x^{63})^2 + x(\sigma^{-2}(p_2) + \sigma^{-2}(p_6)x + \cdots + \sigma^{-2}(p_{254})x^{63})^2$$

and at this stage we have four polynomials to be evaluated. The next two steps double the number of polynomials and halve their degree; one polynomial per each stage is given here as an example

$$\sigma^{-2}(p_0) + \sigma^{-2}(p_4)x + \cdots + \sigma^{-2}(p_{252})x^{63} = (\sigma^{-3}(p_0) + \sigma^{-3}(p_8)x + \cdots + \sigma^{-3}(p_{248})x^{31})^2 + x(\sigma^{-3}(p_4) + \sigma^{-3}(p_{12})x + \cdots + \sigma^{-3}(p_{252})x^{31})^2$$

$$\sigma^{-3}(p_0) + \sigma^{-3}(p_8)x + \cdots + \sigma^{-3}(p_{248})x^{31} = (\sigma^{-4}(p_0) + \sigma^{-4}(p_{16})x + \cdots + \sigma^{-4}(p_{240})x^{15})^2 + x(\sigma^{-4}(p_8) + \sigma^{-4}(p_{24})x + \cdots + \sigma^{-4}(p_{248})x^{15})^2$$

Since we choose to halt the decomposition at this stage (notice that $L = 4$ is a putative optimal value given by (2.6)), we must evaluate 16 polynomials of degree at most 15 with coefficients in $\mathbb{F}_{2^4}$. We do not need to compute σ^{-4} on the coefficients, as $\sigma^{-4}(p_i) = p_i$, since the coefficients are in $\mathbb{F}_{2^4}$ and any element β in this field satisfies the condition $\beta^{2^4} = \beta$.

We remark that up to know we have only indicated how to partition the original polynomial. This task does not require any computation, it just defines in which order to read the coefficients of the original polynomial.

Now, let K be the number of code words to be decoded. We compute only once the following field elements:

- α^i, $i = 2, \ldots, 254$ and this requires 253 multiplications;
- $\alpha^i \cdot \beta^j$ for $i = 0, \ldots, 254$ and $j = 1, \ldots, 14$, which requires $255 \cdot 14 = 3570$ multiplications.

Then only sums (that can be performed in parallel) are required to evaluate 16 polynomials of degree 15 for each α^j, $j = 1 \ldots, 32$. Once we have the values of these polynomials, in order to reconstruct each of $r_1(\alpha^j)$ and $r_2(\alpha^j)$, we need

- $16 + 8 + 4 + 2$ squares
- $8 + 4 + 2 + 1$ multiplications (and the same number of sums).

Summing up, every $r(\alpha^j) = r_1(\alpha^j) + \gamma r_2(\alpha^j)$ is obtained with $2 \cdot 45 + 1 = 91$ multiplications. Then the total cost of the computation of 32 syndromes drops down from $31 + 32 \cdot 254 = 8159$ with Horner's rule to $32 \cdot 91 + 3570 + 253 = 6735$. Since we have K code words the total cost drops from $31 + 8128 \cdot K$ to $3823 + 2912 \cdot K$, with two further advantages:

- many operations can be parallelized, further increasing the speed;
- the multiplications can be performed in $\mathbb{F}_{2^4}$ instead of $\mathbb{F}_{2^8}$, if we write $\alpha^j = a_j + \gamma b_j$; this might increase the number of multiplications, but they would be much faster.

As said, this example was meant to show that there are important applications of polynomial evaluation which can take advantage of a complexity reduction and that there are certainly many other possibilities to further reduce the costs, depending on the particular problem at hand, the model in consideration and the available technology (e.g. availability of storage, of pre-computed tables for finite field mutiplications, etc.). In particular, this paper has been mainly devoted to the single-point evaluation of polynomials, showing that it is possible to achieve significant complexity reduction with respect to Horner's rule even without any precomputation or storage, especially when the degree of the polynomial is large. In other models, it may be possible to have the powers of α as already given data and to store relatively large binary matrices in order to reduce the number of multiplications in a multi-point evaluation scenario or it may be possible to reduce them at the cost of a significant increase of the number of additions. For all these different models, we refer to the vast literature on multi-point evaluation, e.g. [18, 29, 101].

In conclusion, we have proposed some methods to evaluate polynomials in extensions of finite fields that have a multiplicative asymptotical complexity $O(\sqrt{n})$, much better than $O(n)$, the complexity of standard methods; the constant involved is a function of the field characteristic. We have proposed different variants and shown that the choice of an evaluation scheme that uses possibly the smallest number of multiplications follows from a careful analysis of the particular situation and might involve the adoption of special tricks dependent on the combination of parameters. It remains to ascertain whether there exists some evaluation algorithm doing asymptotically better, i.e. having a complexity $O(n^t)$ with $t < \frac{1}{2}$.

Chapter 3

On the Cantor-Zassenhaus polynomial factorization algorithm

Following [41] we revisit here the Cantor-Zassenhaus polynomial factorization algorithm, describing a new simplified version of it, which entails a lower computational cost. Moreover, we show that it can be used to find a factor of a fully splitting polynomial of degree t over $\mathbb{F}_{2^m}$ with $O(\frac{2^m}{3^t})$ attempts and over $\mathbb{F}_{p^m}$ for odd p with $O(\frac{p^m}{2^t})$ attempts.

We will use, among others, results on Gauss sums and additive decompositions by multiplicative characters, to which we have devoted a full-fledged analysis in Appendix.

3.1 Introduction

The Cantor-Zassenhaus polynomial factorization algorithm ([26]) is an efficient (polynomial-time) probabilistic algorithm for factoring polynomials over a finite field $\mathbb{F}_{p^m}$, that are the product of irreducible polynomials with a common degree s and multiplicity one. When the multiplicity is above 1, the factors can be separated by computing the greatest common divisor of the given polynomial and its formal derivative. If the irreducible polynomials have different degrees, the factors are separated by computing the greatest common divisors with polynomials of the form $x^{p^{mr}-1} - 1$, starting from $r = 1$, so as to obtain the product of all irreducible factors of degree $r = 1, 2, \ldots$ (see e.g. [49]). Thus standard methods can be used to reduce the problem to the above case.

We will now introduce the Cantor-Zassenhaus factorization algorithm, providing a non-standard explanation which will be the basis for the rest of the chapter: in the Sections below we will show how it can be improved, giving a new description with a more favorable estimate of its complexity and success rate. In fact this description leads us to consider a deterministic version of the algorithm, so that we will be concerned with the problem of establishing how many attempts are needed in the worst case to obtain a factor (with probability 1) and what is the least degree of the polynomial such that a factor is found with at most a fixed number of attempts.

Let $\sigma(z)$ be a polynomial of degree t over $\mathbb{F}_{p^m}$ which is a product of irreducible polynomials of degree s, i.e. $t = s \cdot d$.

Let us assume that $s = 1$ as a first instance and suppose that the trivial factor z does not divide $\sigma(z)$.

We first deal with the case $p = 2$, and following [26] we assume that m is even, otherwise we would consider a quadratic extension solely for the computations. If α is a known primitive element of $\mathbb{F}_{2^m}$, we define $\ell_m = \frac{2^m-1}{3}$ and $\rho = \alpha^{\ell_m}$, which is thus a primitive cubic root of unity in the field $\mathbb{F}_{2^m}$.

Let $c(z)$ be a non-constant polynomial over $\mathbb{F}_{2^m}$ of degree less than t, and let

$$a(z) = c(z)^{\ell_m} \bmod \sigma(z)$$

which is again a polynomial of degree at most $t-1$. Furthermore, we have

$$(c(z)^{\ell_m} + 1)(c(z)^{\ell_m} + \rho)(c(z)^{\ell_m} + \rho^2) = c(z)^{2^m-1} - 1 \ .$$

Now, either $\gcd(c(z), \sigma(z))$ is non-trivial (and thus we already have a factor of $\sigma(z)$) or else $c(z)^{2^m-1} - 1 = 0 \bmod \sigma(z)$. In this latter case, if we write $c(z)^{2^m-1} - 1 = Q(z)\sigma(z) + R(z)$ and specialize it in the roots $\{z_i\}$ of $\sigma(z)$, we see that $R(z)$, which is a polynomial of degree $t-1$, takes the value 0 for all t roots, as $\beta^{2^m-1} - 1 = 0$ for any $\beta \in \mathbb{F}_{2^m}^*$. This implies that $R(z)$ is identically 0. Thus we can write

$$(c(z)^{\ell_m} + 1)(c(z)^{\ell_m} + \rho)(c(z)^{\ell_m} + \rho^2) = (a(z) + 1)(a(z) + \rho)(a(z) + \rho^2) = 0 \bmod \sigma(z) \ .$$

Since every factor of the product $(a(z) + 1)(a(z) + \rho)(a(z) + \rho^2)$ has degree less than t, at least two of them must have a common non-trivial factor with $\sigma(z)$, unless $a(z) = 1, \rho, \rho^2$. In this latter case, the Cantor-Zassenhaus algorithm considers another random polynomial instead of $c(z)$, and reiterates the procedure until all factors have been found.

Notice that $a(z) \equiv 0$ never occurs, since $c(z)$ has degree less than $\sigma(z)$, so that at least one root of $\sigma(z)$, say β, is not a root of $c(z)$; then substituting β in the identity $c(z)^{\ell_m} = q(z)\sigma(z) + a(z)$, we get $a(\beta) \neq 0$, therefore $a(z)$ is not identically zero (this holds even if the roots of σ were not in the field of the coefficients, as in the original description of the algorithm).

For the case $p > 2$, the procedure is similar: we would consider $\ell_m = \frac{p^m-1}{2}$ and $\rho = \alpha^{\ell_m} = -1$, where α is a primitive element of $\mathbb{F}_{p^m}$. Here we would compute $a(z) = c(z)^{\ell_m} \bmod \sigma(z)$ and then factor as soon as $a(z) \neq \pm 1$.

Let us consider now the case $s > 1$. One option is to look at $\mathbb{F}_{p^{sm}}$, where the polynomial fully splits into linear factors: once a factor $z - \beta$ is found, it can be multiplied with the factors $z - \beta^{p^{mi}}$, with $1 \leq i \leq s-1$, to obtain an irreducible factor of degree s. A second option is the application of the algorithms over $\mathbb{F}_{p^m}$ ([12], [26]), to directly find the irreducible factors of degree s over $\mathbb{F}_{p^m}$. If $p = 2$, the argument follows as above: either $\gcd(c(z), \sigma(z))$ is non-trivial, or $\gcd(c(z), \sigma(z)) = 1$, in which case

$$(c(z)^{\ell_{sm}} + 1)(c(z)^{\ell_{sm}} + \rho)(c(z)^{\ell_{sm}} + \rho^2) = (a(z) + 1)(a(z) + \rho)(a(z) + \rho^2) = 0 \bmod \sigma(z) \ .$$

Since every factor of the product $(a(z) + 1)(a(z) + \rho)(a(z) + \rho^2)$ has degree less than t, at least two of them must have a common non-trivial factor with $\sigma(z)$ in $\mathbb{F}_{2^m}$, unless $a(z) = 1, \rho, \rho^2$. In this latter case, the Cantor-Zassenhaus algorithm considers another random polynomial $c(z)$, and reiterates the procedure until all factors have been found.

For the case $p > 2$, the procedure is similar: we would consider $\ell_{sm} = \frac{p^{sm}-1}{2}$ and compute $a(z) = c(z)^{\ell_{sm}} \bmod \sigma(z)$ and then factor as soon as $a(z) \neq \pm 1$.

In the next Section we will present a variant of the Cantor-Zassenhaus algorithm, according to the description given above, and then deal with probabilistic as well as deterministic considerations about its success rate.

3.2 An improved algorithm

We focus first on the case $s = 1$ and show that it is enough, and indeed convenient, to choose $c(z) = z$ as initial test polynomial and to choose $c(z) = z + \beta$, for some random $\beta \neq 0$, as further test polynomial, and continuing by choosing random βs different from the previous ones until a factor is found. A similar approach was already present in [96] for the case of odd characteristic (cf. also [5]).

We then consider the case $s > 1$, where polynomials of degree 1 or s will be involved as test polynomials in order to obtain bounds on the number of attempts to find a factor.

3.2.1 Case $s = 1$

Suppose $\sigma(z)$ is over $\mathbb{F}_{2^m}$ and $z^{\ell_m} = \rho^i \bmod \sigma(z)$, $i \in \{0, 1, 2\}$. Now, any element in $\mathbb{F}^*_{2^m}$ can be written as α^{k+3n}, with $k \in \{0, 1, 2\}$: we define $\mathcal{A}_0 = \{\alpha^{3i} : \quad i = 0, \ldots, \ell_m - 1\}$, that is the subgroup of the elements of $\mathbb{F}^*_{2^m}$ that are cubic powers, and let $\mathcal{A}_1 = \alpha\mathcal{A}_0$ and $\mathcal{A}_2 = \alpha^2\mathcal{A}_0$ be the two cosets that complete the coset partition of $\mathbb{F}^*_{2^m}$. If we substitute α^{k+3n} for any root z_i of $\sigma(z)$ in $z^{\ell_m} - \rho^i = Q(z)\sigma(z)$, we obtain $\rho^k - \rho^i = 0$, which implies $k = i$. This means that if $z^{\ell_m} = \rho^i \bmod \sigma(z)$, then all the roots of $\sigma(z)$ are of the form α^{i+3n}, that is they belong to the same coset. When this situation occurs, we consider another test polynomial $c(z) = z + \beta$, which is equivalent to testing $c(z) = z$ for the polynomial $\varsigma(z)$ whose set of roots is $\{z_i + \beta\}$. The test succeeds as soon as we find a β such that the roots $z_i + \beta$ do not all belong to the same coset.

The next step is to determine an upper bound to the number of attempts needed in the worst case scenario, or on average, until a factor is found.

Let us first consider the simple case $t = 2$: suppose that z_1 and z_2 belong to the same coset; then we look for a β such that $z_1 + \beta$ and $z_2 + \beta$ are in different cosets. For the worst case scenario, we need to know how many pairs $(z_1 + \beta, z_2 + \beta)$ have both elements in the same coset. This is equivalent to knowing the number of ways in which $z_1 - z_2 = z_1 + \beta - (z_2 + \beta)$ can be written as the sum of two elements in the same coset. This number is actually $\frac{2^m-1}{3} - 1$, as can be deduced from [126, Theorem 1] specialized with $i = 0$ and χ the cubic character. So at most with $\frac{2^m-1}{3}$ attempts we can factor a polynomial of degree 2. Clearly at each test we can factor with a probability of $\frac{2}{3}$, so that the expected number of attempts is 1.5.

If $\sigma(z)$ is a polynomial over $\mathbb{F}_{p^m}$, $p > 2$, then the maximum number of attempts is $\frac{p^m-1}{2}$, by similar reasoning: we again use some additive properties of residues ([80, 81, 98, 126]). At each test we can factor with a probability of $\frac{1}{2}$, so that the expected number of attempts is 2.

The remainder of this chapter will be devoted to establishing both probabilistic estimates and deterministic bounds on the number of attempts needed to successfully factor, for a generic t. A first deterministic, though very loose, bound is the following:

Proposition 3.1. *The maximum number of attempts needed to find a factor is upper bounded by ℓ_m (that is $\frac{2^m-1}{3}$ or $\frac{p^m-1}{2}$ for $p = 2$ or p odd, respectively). In particular, in the Cantor-Zassenhaus algorithm it is sufficient to consider only linear polynomials as test polynomials $c(z)$.*

PROOF. In characteristic 2, if a root z_i belongs to a given known coset, we can test all the ℓ_m elements of that coset, until we obtain z_i itself: $z_i + z_i$ adds to 0, which does not belong to any coset. Thus we will succeed with at most ℓ_m attempts. In characteristic p greater than 2, it is sufficient to add all the elements of the coset multiplied by $p - 1$.

That it is enough to consider all the p^m monic linear polynomials is anyway clear since computing $\gcd\{z-\beta, \sigma(z)\}$ for all β in $\mathbb{F}_{p^m}$ would be enough to find all the factors.

□

Remark 3.2. *The above argument implies that, if the first attempt fails, we know which coset the roots belong to, and can restrict our choice of β to that coset.*

Remark 3.3. *Alternatively, the upper bounds of the proposition follow from the above remarks about $t = 2$: clearly, if t is bigger than 2, then a degree-2 polynomial is anyway a factor of the t-degree polynomial, so that the maximum number of attempts cannot exceed the number needed to factor this degree-2 polynomial.*

Remark 3.4. *In the original version of the Cantor-Zassenhaus algorithm, $\gcd(a(z), \sigma(z))$ is computed when searching for a factor of $\sigma(z)$, corresponding to the case when $\gcd(c(z), \sigma(z))$ is non-trivial. Our version of the algorithm avoids this computation, since it is sufficient to evaluate $\sigma(z)$ in β with any efficient polynomial evaluation algorithm; this can be done before exponentiating to the power ℓ_m.*

Remark 3.5. *If q is a prime factor of $p^m - 1$, then we may consider the exponent $\ell_m = \frac{p^m-1}{q}$: in this case the probability of success is $\frac{q-1}{q}$ and the corresponding expected number of attempts is $\frac{q}{q-1}$, which is close to 1 already for small primes like 5 or 7; the drawback is that, if q is large, in the worst case we must check q greatest common divisors, namely $\gcd(a(z) + \zeta_q^j, \sigma(z))$, for $0 \leq j \leq q-1$, where ζ_q is a q-th primitive complex root of unity.*

3.2.2 Case $s > 1$

If $s > 1$, either we look for linear factors in $\mathbb{F}_{p^{ms}}$, and the analysis is the same as in the case $s = 1$, or we choose the direct method, as explained in the previous section. In this case, by a similar argument as above, the algorithm succeeds as soon as $c(z_i)$, z_i being the roots of $\sigma(z)$, are not all in the same coset. This is equivalent to ask that non conjugate roots are not all in the same coset, as

$$c(z_i^{p^m})^{\ell_{sm}} = ((c(z_i))^{p^m})^{\ell_{sm}} = ((c(z_i))^{\ell_{sm}})^{p^m} = (c(z_i))^{\ell_{sm}}$$

by the properties of the Frobenius automorphism.

Let us see this more precisely, describing in detail the case $p = 2$, while a similar argument applies in the case of odd primes. Let $\sigma(z)$ be, as above, a polynomial of degree t over $\mathbb{F}_{2^m}$, which is a product of d irreducible polynomials $\sigma_i(z)$ of degree s over the same field $\mathbb{F}_{2^m}$, where it is not restrictive to assume even m. According to Cantor-Zassenhaus algorithm, a polynomial $c(z)$ over $\mathbb{F}_{2^m}$, relatively prime with $\sigma(z)$, separates $\sigma(z)$ into two polynomials of smaller degree if $a(z) = c(z)^{\ell_{sm}} \bmod \sigma(z)$ is different from $1, \rho, \rho^2$: at least two factors $\sigma_i(z)$ are in two distinct greatest common divisors between $\sigma(z)$ and $a(z)+1$, $a(z)+\rho$, and $a(z)+\rho^2$, respectively.

Lemma 3.6. *With the above hypotheses and definitions, a polynomial $c(z)$ over $\mathbb{F}_{2^m}$ separates $\sigma(z)$ into two polynomials one containing the factor $\sigma_1(z)$, and a second one containing the factor $\sigma_2(z)$ if and only if $c(z)^{\ell_{sm}} \bmod \sigma_1(z) \neq c(z)^{\ell_{sm}} \bmod \sigma_2(z)$. Equivalently, $\sigma_1(z)$ and $\sigma_2(z)$ are separated if and only if $c(z_1)$ and $c(z_2)$ belong to different cosets $\mathcal{A}'_h$ of $\mathbb{F}^*_{2^{sm}}$, where z_1 and z_2 are roots of $\sigma_1(z)$ and $\sigma_2(z)$, respectively.*

PROOF. The polynomial $\sigma(z)$ can be written as a product of three polynomials, i.e. $\sigma_1(z)$, $\sigma_2(z)$, and $\sigma_r(z)$ which collects the remaining factors, thus $a(z)$ can be decomposed, using the Chinese Remainder Theorem (CRT), as

$$a(z) = a_1(z)\psi_1(z) + a_2(z)\psi_2(z) + a_r(z)\psi_r(z) \bmod \sigma(z) \ , \ \psi_1(z) + \psi_2(z) + \psi_r(z) = 1 \ ,$$

where $a_1(z) = c(z)^{\ell_{sm}} \bmod \sigma_1(z)$, $a_2(z) = c(z)^{\ell_{sm}} \bmod \sigma_2(z)$, and $a_r(z) = c(z)^{\ell_{sm}} \bmod \sigma_r(z)$.

If $a(z) = 1, \rho, \rho^2$, the uniqueness of the CRT decompositions implies that $a_1(z) = a_2(z) = a_r(z)$.

If $a(z) \neq 1, \rho, \rho^2$, then $c(z)$ separates $\sigma(z)$ into two polynomials of smaller degree, and we distinguish two cases:

1) $a_1(z) \neq a_2(z)$: the polynomials $\sigma_1(z)$ and $\sigma_2(z)$ are in different factors because, if both of them were in the same factor, they would both divide the same polynomial $a(z) + \rho^h$, thus $a_i(z) = a(z) = \rho^h$ modulo $\sigma_i(z)$, $i = 1, 2$, contrary to the assumption.

2) $a_1(z) = a_2(z)$: $\sigma_1(z)$ and $\sigma_2(z)$ are in the same factor; in fact, suppose they are not, then $a_1(z) = a(z) = \rho^{h_1} \bmod \sigma_1(z) \neq a_2(z) = a(z) = \rho^{h_2} \bmod \sigma_2(z)$, yielding a contradiction.

Also, since $a(z) = c(z)^{\ell_{sm}} \bmod \sigma(z)$ and $a(z) = a_i(z) = \rho^{h_i} \bmod \sigma_i(z)$, we have that $c(z_i)^{\ell_{sm}} = \rho^{h_i}$, $i = 1, 2$, which means that $c(z_i) \in \mathcal{A}'_{h_i}$, hence it follows from the first part of the lemma that $c(z)$ separates $\sigma_1(z)$ and $\sigma_2(z)$ if and only if $c(z_1) \neq c(z_2)$.

□

Now, as in the case $s = 1$, we are interested in upper bounds for the number of attempts and we can limit the choice of $c(z)$, according to our convenience. For example, if we know at least one primitive polynomial $m(z)$ of degree s, we can choose the polynomials $c(z)$ within the set of monic irreducible polynomials of degree s, so that we get directly $\frac{p^{ms}}{s}$ as an upper bound. If we do not have any primitive polynomial of degree s, that is no means to get and draw from the pool of irreducible polynomials of degree s, then we can choose the polynomials $c(z)$ within the larger set of monic polynomials of degree s, and we have the looser bound p^{ms}. Somehow surprisingly, we show next that usually it is actually sufficient to consider again linear polynomials.

Let $\chi'_3(x)$ be a non-trivial cubic character over $\mathbb{F}_{2^{sm}}$, namely χ'_3 is a mapping from $\mathbb{F}^*_{2^{sm}}$ into the complex numbers defined as

$$\chi'_3(\alpha^h\theta) = \zeta_3^h \ , \ \theta \in \mathcal{A}'_0, \quad h = 0, 1, 2 \ ,$$

α being a primitive element of $\mathbb{F}^*_{2^{sm}}$, ζ_3 a primitive complex cubic root of unity, and $\mathcal{A}'_0$ the coset of cubes in $\mathbb{F}^*_{2^{sm}}$. Moreover, we set $\chi'_3(0) = 0$ by definition.

If z_1 and z_2 are roots of two distinct irreducible polynomials of degree s, we denote with $N_2^{(m)}(z_1, z_2)$ the number of monic polynomials $c(z) = z + \beta$ with $\beta \in \mathbb{F}_{2^m}$ such that $\chi'_3(c(z_1)) = \chi'_3(c(z_2))$.

Proposition 3.7. *The maximum number N_A of attempts needed to find an irreducible factor of degree s, using monic linear polynomials as test polynomials, is upper bounded by $\frac{2^m}{3}(1 + \frac{4s-2}{\sqrt{2^m}} + \frac{1}{2^m})$ if $p = 2$, or by $\frac{p^m}{2}(1 + \frac{2s-1}{\sqrt{p^m}})$ if p is odd. In particular linear polynomials are sufficient to find a factor if $\frac{4s-2}{\sqrt{2^m}} < 2$ or $\frac{2s-1}{\sqrt{p^m}} < 1$, respectively.*

PROOF. In the case of characteristic 2, N_A is upper bounded by the maximum of $N_2^{(m)}(z_1, z_2) + 1$ taken over all distinct pairs of roots z_1 and z_2 of distinct irreducible polynomials of degree s. Thus an upper bound for $N_2^{(m)}(z_1, z_2)$ independent of z_1 and z_2 is also an upper bound for $N_A - 1$.

Consider the indicator function

$$I_{\mathcal{A}'_h}(c(z_i)) = \frac{1 + \bar{\zeta}_3^h \chi'_3(c(z_i)) + \zeta_3^h \bar{\chi}'_3(c(z_i))}{3} \quad i = 1, 2 \ ,$$

which is 1 if the cubic character of $c(z_i)$ is ζ_3^h, and is 0 otherwise, if we suppose $c(z)$ relatively prime with $\sigma(z)$.

Therefore, for a given $c(z)$ we have a coincidence whenever the product $I_{\mathcal{A}'_h}(c(z_1))I_{\mathcal{A}'_h}(c(z_2))$ is 1. Thus,

$$\sum_{h=0}^{2} I_{\mathcal{A}'_h}(c(z_1))I_{\mathcal{A}'_h}(c(z_2)) = \frac{1}{3}\left(1 + \chi'_3(c(z_1))\bar{\chi}'_3(c(z_2)) + \bar{\chi}'_3(c(z_1))\chi'_3(c(z_2))\right)$$

is the coincidence indicator for a fixed polynomial $c(z)$. Summing over all monic linear polynomials $z + \beta$ over $\mathbb{F}_{2^m}$, we get the total number $N_2^{(m)}(z_1, z_2)$ of coincidences

$$N_2^{(m)}(z_1, z_2) = \frac{1}{3} \sum_{\beta \in \mathbb{F}_{2^m}} \left(1 + \chi'_3(z_1 + \beta)\bar{\chi}'_3(z_2 + \beta) + \bar{\chi}'_3(z_1 + \beta)\chi'_3(z_2 + \beta)\right) - \frac{2}{3} \ ,$$

where $-\frac{2}{3}$ comes from excluding those polynomials $z + \beta$ having z_1 or z_2 as root. We split the summation in three summations, the first summation is simply 2^m, and the second and third summations are complex conjugated, thus it is enough to evaluate only the summation

$$C = \sum_{\beta \in \mathbb{F}_{2^m}} \chi'_3(z_1 + \beta)\bar{\chi}'_3(z_2 + \beta) \ .$$

This summation is hard to evaluate in closed form, thus we content ourselves with a bound. Namely, as χ'_3 can be considered as the lifted character of a nontrivial character χ_3 over $\mathbb{F}_{2^m}$ [63], we can write

$$C = \sum_{\beta \in \mathbb{F}_{2^m}} \chi_3(N_{F_{2^{ms}}/F_{2^m}}(z_1 + \beta))\bar{\chi}_3(N_{F_{2^{ms}}/F_{2^m}}(z_2 + \beta)),$$

where $N_{F_{2^{ms}}/F_{2^m}}(x) = x \cdot x^{2^m} \cdots x^{2^{m(s-1)}}$ is the relative norm of x.

Since $N_{F_{2^{ms}}/F_{2^m}}(z_i + \beta)$, $i = 1, 2$, are polynomials of degree s in β, and $\bar{\chi}_3 = \chi_3^2$, we can then use the Weil bound ([108, Theorem 2C']; cf. also [120],[127, Lemma 2.2]) to obtain

$$C < (2s - 1)2^{m/2}.$$

In conclusion we obtain N_A bounded as $N_A < \frac{2^m}{3}(1 + \frac{4s-2}{\sqrt{2^m}} + \frac{1}{2^m})$. The same argument works similarly for p odd, and making the appropriate changes the conclusion is

$$N_A < \frac{p^m}{2}(1 + \frac{2s - 1}{\sqrt{p^m}}) \ .$$

□

In the following we analyse the algorithm more in detail both from a probabilistic and a deterministic point of view; in particular we will show that the maximum number of attempts to get a factor is usually very small, so that the algorithm, which is probabilistic in nature, can often be considered deterministic. In order to simplify the subsequent analysis, we will suppose that $s = 1$ from now on.

Remark 3.8. *Note that similar equalities of characters as those we use in the present paper to assess deterministic bounds have been already considered in [4, 12, 111], but the analysis was aimed at obtaining probabilistic estimates, and also other approaches are possible, such as algebraic geometric machinery [3].*

Remark 3.9. *The analysis concerns the scheme of Cantor-Zassenhaus where the test polynomials are chosen randomly. Therefore when considering linear polynomials $z + \beta$ we are always choosing different, but random βs. However this does not exclude that there are systematic choices for the βs that lead to a smaller maximum number of attempts, in particular, for prime fields we get a better estimate from [111, Lemma 3.3], but this might, on the other hand, affect negatively the average complexity [111].*

3.3 Probability of factoring

The Cantor-Zassenhaus algorithm is very efficient in factoring polynomials, but is not deterministic. We can show, however, that the maximum number of attempts, following the modified version above, decreases exponentially with the degree of the polynomial, so that the probability of factoring with one test is close to 1 when the degree is large enough.

Making the reasonably assumption that the set of $\{z_i + \beta\}$ for some β is made up of elements which belong to each coset $\mathcal{A}_i$ with probability $1/3$ (or $1/2$ in the case $p > 2$), independently of one another, then $3 \cdot \frac{1}{3^t}$ is the probability that they all belong to a common coset of the three cosets (and $2 \cdot \frac{1}{2^t}$ in case of the two cosets in $\mathbb{F}^*_{p^m}$, $p > 2$). Therefore the number of attempts to obtain a factor, in the worst case scenario, is roughly $\frac{2^m}{3^{t-1}}$ and $\frac{p^m}{2^{t-1}}$ respectively. And the expected number is $\frac{1}{1-\frac{1}{3^t-1}} = 1 + \frac{1}{3^{t-1}-1}$ or $1 + \frac{1}{2^{t-1}-1}$.

Furthermore, suppose we fail at the first attempt, then we can choose β within a certain coset, and the probability of failing at the next n attempts is only $\frac{1}{3^{tn}}$.

Clearly, once a factor is found, the polynomial splits into two parts to which we will re-apply the previous computation if we are interested in a complete factorization, untill all linear factors are obtained.

3.4 Deterministic splitting I: fixed t

If we use the proposed variant of the Cantor-Zassenhaus algorithm, the tightest upper bound to the number of attempts necessary to split a polynomial $\sigma(z)$ of degree t over F_{2^m} is equal to

$$1 + \max_{z_1 \neq z_2 \neq \cdots \neq z_t} N_2(t),$$

where $N_2(t)$ is the number of solutions β of a system of t equations in $\mathbb{F}_{2^m}$ of the form

$$\begin{cases} \alpha^j z_1^3 + \beta = \alpha^k y_1^3 \\ \alpha^j z_2^3 + \beta = \alpha^k y_2^3 \\ \quad \vdots \\ \alpha^j z_t^3 + \beta = \alpha^k y_t^3 \end{cases} \tag{3.1}$$

where $\alpha^j z_1^3, \alpha^j z_2^3, \cdots, \alpha^j z_t^3$ are given and distinct (i.e. they are the roots of $\sigma(z)$), whereas the y_is must be chosen in the field to satisfy the system, and the three values $\{0, 1, 2\}$ for k and j are all considered. However, we may assume $j = 0$, since dividing each equation by α^j, and setting $\beta' = \beta\alpha^{-j}$ and $k' = k - j \bmod 3$, we see that the number of solutions of the system is independent of j. If the system is unsolvable, then the number of attempts is 1.

To evaluate $N_2(t)$, we define an indicator function of the sets $\mathcal{A}_u$ using the cubic character, namely for every $x \neq 0$

$$I_{\mathcal{A}_u}(x) = \frac{1 + \zeta_3^{2u}\chi_3(x) + \zeta_3^u \bar{\chi}_3(x)}{3} = \begin{cases} 1 & \text{if } x \in \mathcal{A}_u \\ 0 & \text{otherwise} \end{cases} \quad u = 0, 1, 2 \ ,$$

(where the bar denotes complex conjugation). Then, given a z_i we can partition the elements $\beta \neq z_i^3$ in $\mathbb{F}_{2^m}$ into subsets depending on the $k \in \{0, 1, 2\}$ such that $\chi_3(\beta + z_i^3) = \zeta_3^k$. Therefore, a solution of (3.1) for a fixed k and $j = 0$ is singled out by the product

$$\prod_{i=1}^{t} I_{\mathcal{A}_k}(\beta + z_i^3) = \frac{1}{3^t}[1 + \sum_{i=1}^{t} \sigma_i^{(k)}] \ ,$$

where each $\sigma_i^{(k)}$ is a homogeneous sum of monomials which are products of i characters of the form $\chi_3(\beta + z_h^3)$ or $\bar{\chi}_3(\beta + z_h^3)$. Thus $N_2(t)$ is

$$N_2(t) = \sum_{\substack{\beta \in \mathbb{F}_{2^m} \\ \beta \notin \{z_i^3\}}} \left[\prod_{i=1}^{t} I_{\mathcal{A}_0}(\beta + z_i^3) + \prod_{i=1}^{t} I_{\mathcal{A}_1}(\beta + z_i^3) + \prod_{i=1}^{t} I_{\mathcal{A}_2}(\beta + z_i^3) \right] \ . \tag{3.2}$$

The roots z_i in the sum need not be considered, since in any case they are not solutions ($z_i^3 + z_i^3 = 0$ cannot be in the same coset as $z_i^3 + z_j^3$ if $i \neq j$).

Similarly, in characteristic greater than 2, the tightest upper bound to the number of attempts necessary to split a polynomial $\sigma(z)$ of degree t is equal to

$$1 + \max_{z_1 \neq z_2 \neq \cdots \neq z_t} N_p(t),$$

where $N_p(t)$ is the number of solutions β of a system of t equations in $\mathbb{F}_{p^m}$ of the form

$$\begin{cases} \alpha^j z_1^2 + \beta = \alpha^k y_1^2 \\ \alpha^j z_2^2 + \beta = \alpha^k y_2^2 \\ \quad \vdots \\ \alpha^j z_t^2 + \beta = \alpha^k y_t^2 \end{cases} \tag{3.3}$$

where $\alpha^j z_1^2, \alpha^j z_2^2, \cdots, \alpha^j z_t^2$ are given and distinct and the two values $\{0,1\}$ for k and j are considered. Again, we may assume $j=0$ and we can define an indicator function of the sets $\mathcal{B}_u$ using the quadratic character, where $\mathcal{B}_0$ is the set of squares and $\mathcal{B}_1$ the complementary set in $\mathbb{F}_{p^m}^*$: namely, let χ_2 be a mapping from $\mathbb{F}_{p^m}^*$ into the complex numbers defined as

$$\chi_2(\alpha^h\theta) = (-1)^h \ , \ \ \theta \in \mathcal{B}_0, \quad h = 0,1 \ .$$

Again, we set $\chi_2(0) = 0$.

The corresponding indicator function is thus

$$I_{\mathcal{B}_u}(x) = \frac{1+(-1)^u\chi_2(x)}{2} = \begin{cases} 1 & \text{if } \ x \in \mathcal{B}_u \\ 0 & \text{otherwise} \end{cases} \quad u = 0,1 \ .$$

Given a z_i we partition $\mathbb{F}_{p^m} \setminus \{z_i^2\}$ into subsets depending on the value of k, such that $\chi_2(\beta + z_i^2) = (-1)^k$. Therefore, a solution of (3.3) for a fixed k is given by the product

$$\prod_{i=1}^{t} I_{\mathcal{B}_k}(\beta + z_i^2) = \frac{1}{2^t}[1 + \sum_{i=1}^{t} \sigma_i^{(k)}] \ ,$$

where each $\sigma_i^{(k)}$ is a homogeneous sum of monomials which are product of i characters of the form $\chi_2(\beta + z_h^2)$. Thus $N_p(t)$ is

$$N_p(t) = \sum_{\substack{\beta \in \mathbb{F}_{p^m} \\ \beta \notin \{-z_i^2\}}} \left[\prod_{i=1}^{t} I_{\mathcal{B}_0}(\beta + z_i^2) + \prod_{i=1}^{t} I_{\mathcal{B}_1}(\beta + z_i^2) \right] \ . \tag{3.4}$$

The following subsections deal with computations of $N_p(t)$ for small values of t, then with general bounds on $N_p(t)$.

3.4.1 Computations for small t

In the following computations, we will use some properties of nontrivial characters that we briefly mention: $\sum_{x\in\mathbb{F}_q} \chi(x) = 0$; if $\beta \neq 0$, then $\sum_{x\in\mathbb{F}_q} \chi(x)\bar{\chi}(x+\beta) = -1$ ([102, 126]). Moreover,

$$\sum_{x\in\mathbb{F}_{2^m}} \chi_3(x)\chi_3(x+1) = G_m(1,\chi) = -(-2)^{m/2},$$

with $G_m(1,\chi)$ being the Gauss sum ([102]).

We will start with the case $p=2$. First we compute $N_2(2)$, already found above with another technique, then analogously $N_2(3)$.

$t=2$. Setting $x_i = \beta + z_i^3$, we have

$$\prod_{i=1}^{2} I_{\mathcal{A}_h}(x_i) = \frac{1}{9}\left(1 + \sigma_1^{(h)} + \sigma_2^{(h)}\right) \ \ h = 0,1,2 \ ,$$

where

$$\begin{aligned}\sigma_1^{(h)} &= \zeta_3^{2h}\chi_3(x_1)+\zeta_3^h\bar\chi_3(x_1)+\zeta_3^{2h}\chi_3(x_2)+\zeta_3^h\bar\chi_3(x_2)\\ \sigma_2^{(h)} &= \zeta_3^h\chi_3(x_1)\chi_3(x_2)+\chi_3(x_1)\bar\chi_3(x_2)+\bar\chi_3(x_1)\chi_3(x_2)+\zeta_3^{2h}\bar\chi_3(x_1)\bar\chi_3(x_2)\end{aligned}$$

Since $\sigma_1^{(0)}+\sigma_1^{(1)}+\sigma_1^{(2)}=0$ and $\sigma_2^{(0)}+\sigma_2^{(1)}+\sigma_2^{(2)}=3(\chi_3(x_1)\bar\chi_3(x_2)+\bar\chi_3(x_1)\chi_3(x_2))$, the sum of the three products $\prod_{i=1}^2 I_{\mathcal{A}_k}(x_i)$ is $\frac{1}{3}\left(1+\chi_3(x_1)\bar\chi_3(x_2)+\bar\chi_3(x_1)\chi_3(x_2)\right)$, and thus the sum over β in the whole field $\mathbb{F}_{2^m}$, with the exclusion of $\beta=z_1^3$ and $\beta=z_2^3$, is

$$N_2(2)=\frac{1}{3}\left(2^m-2+\sum_{\beta\neq z_1^3,z_2^3}\left(\chi_3(\beta+z_1^3)\bar\chi_3(\beta+z_2^3)+\bar\chi_3(\beta+z_1^3)\chi_3(\beta+z_2^3)\right)\right)\ .$$

Let S denote the above summation, then S can be evaluated in closed form: by the substitution $\beta=z_1^3+\eta$, since χ_3 is a nontrivial cubic character, we have

$$S=\sum_{\eta\neq 0,z_1^3+z_2^3}\left(\chi_3(\eta)\bar\chi_3(\eta+z_1^3+z_2^3)+\bar\chi_3(\eta)\chi_3(\eta+z_1^3+z_2^3)\right)=-2\ ,$$

as the summation of each of the two parts gives -1 ($z_1^3+z_2^3\neq 0$ by hypothesis). In conclusion, we have

$$N_2(2)=\frac{1}{3}\left(2^m-4\right)\ ,$$

so that

$$1+\max_{z_1\neq z_2}N_2(2)=\frac{1}{3}\left(2^m-1\right)\ .$$

$t=3$. In this case

$$\prod_{i=1}^3 I_{\mathcal{A}_h}(\beta+z_i^3)=\frac{1}{27}\left(1+\sigma_1^{(h)}+\sigma_2^{(h)}+\sigma_3^{(h)}\right)\ \ h=0,1,2\ ,$$

where

$$\begin{aligned}\sigma_1^{(h)} &= \zeta_3^{2h}\chi_3(x_1)+\zeta_3^h\bar\chi_3(x_1)+\zeta_3^{2h}\chi_3(x_2)+\zeta_3^h\bar\chi_3(x_2)+\zeta_3^{2h}\chi_3(x_3)+\zeta_3^h\bar\chi_3(x_3)\\ \sigma_2^{(h)} &= \zeta_3^h\chi_3(x_1)\chi_3(x_2)+\chi_3(x_1)\bar\chi_3(x_2)+\bar\chi_3(x_1)\chi_3(x_2)+\zeta_3^{2h}\bar\chi_3(x_1)\bar\chi_3(x_2)+\\ &\quad \zeta_3^h\chi_3(x_2)\chi_3(x_3)+\chi_3(x_2)\bar\chi_3(x_3)+\bar\chi_3(x_2)\chi_3(x_3)+\zeta_3^{2h}\bar\chi_3(x_2)\bar\chi_3(x_3)+\\ &\quad \zeta_3^h\chi_3(x_3)\chi_3(x_1)+\chi_3(x_3)\bar\chi_3(x_1)+\bar\chi_3(x_3)\chi_3(x_1)+\zeta_3^{2h}\bar\chi_3(x_3)\bar\chi_3(x_1)+\\ \sigma_3^{(h)} &= \chi_3(x_1)\chi_3(x_2)\chi_3(x_3)+\bar\chi_3(x_1)\bar\chi_3(x_2)\bar\chi_3(x_3)+\zeta_3^{2h}\bar\chi_3(x_1)\chi_3(x_2)\chi_3(x_3)+\\ &\quad \zeta_3^{2h}\chi_3(x_1)\bar\chi_3(x_2)\chi_3(x_3)+\zeta_3^{2h}\chi_3(x_1)\chi_3(x_2)\bar\chi_3(x_3)+\zeta_3^h\bar\chi_3(x_1)\bar\chi_3(x_2)\chi_3(x_3)+\\ &\quad \zeta_3^h\chi_3(x_1)\bar\chi_3(x_2)\bar\chi_3(x_3)+\zeta_3^h\bar\chi_3(x_1)\chi_3(x_2)\bar\chi_3(x_3)\end{aligned}$$

We thus have

$$\begin{aligned}\sigma_1^0+\sigma_1^1+\sigma_1^2 &= 0\\ \sigma_2^0+\sigma_2^1+\sigma_2^2 &= 3(\chi_3(x_1)\bar\chi_3(x_2)+\bar\chi_3(x_1)\chi_3(x_2)+\chi_3(x_2)\bar\chi_3(x_3)+\bar\chi_3(x_2)\chi_3(x_3)+\\ &\quad \chi_3(x_3)\bar\chi_3(x_1)+\bar\chi_3(x_3)\chi_3(x_1))\\ \sigma_3^0+\sigma_3^1+\sigma_3^2 &= 3(\chi_3(x_1)\chi_3(x_2)\chi_3(x_3)+\bar\chi_3(x_1)\bar\chi_3(x_2)\bar\chi_3(x_3))\end{aligned}$$

In the summation over β of the sum of the three products, the values of $\beta = z_1^3, z_2^3, z_3^3$ should be excluded. Thus we must compute

$$N_2(3) = \frac{1}{9}\left(2^m - 3 + \frac{1}{3}\sum_{\beta\neq z_1^3,z_2^3,z_3^3}\left[(\sigma_2^0+\sigma_2^1+\sigma_2^2)+(\sigma_3^0+\sigma_3^1+\sigma_3^2)\right]\right) \ .$$

Therefore, two types of summations must be evaluated, namely

$$S_2 = \sum_{\beta\neq z_1^3,z_2^3,z_3^3}\chi_3(\beta+z_1^3)\bar{\chi}_3(\beta+z_2^3) \quad \text{and} \quad S_3 = \sum_{\beta\neq z_1^3,z_2^3,z_3^3}\chi_3(\beta+z_1^3)\chi_3(\beta+z_2^3)\chi_3(\beta+z_2^3) \ ,$$

the remaining ones being obtained by symmetry or complex conjugation. Considering S_2, and defining for short $y_1 = z_2^3 + z_3^3$, $y_2 = z_1^3 + z_3^3$, and $y_3 = z_2^3 + z_1^3$, we have

$$S_2 = -\chi_3(y_2)\bar{\chi}_3(y_1) + \sum_{\beta\neq z_1^3,z_2^3}\chi_3(\beta+z_1^3)\bar{\chi}_3(\beta+z_2^3) = -\chi_3(y_2)\bar{\chi}_3(y_1) + \sum_{x\neq 0,y_3}\chi_3(x)\bar{\chi}_3(x+y_3) \ ,$$

thus $S_2 = -\chi_3(y_2)\bar{\chi}_3(y_1) - 1$. Considering S_3 we have

$$S_3 = \sum_{\beta\neq z_1^3,z_2^3,z_3^3}\chi_3(\beta+z_1^3)\chi_3(\beta+z_2^3)\chi_3(\beta+z_3^3) = \sum_{x\neq 0,y_2,y_3}\chi_3(x)\chi_3(x+y_3)\chi_3(x+y_2)$$

thus, with the change of variable $x = 1/z$, since the character is cubic we obtain

$$S_3 = \sum_{z\neq 0,1/y_2,1/y_3}\chi_3(1+zy_3)\chi_3(1+zy_2) = \sum_{X\neq 1,0,1+y_3/y_2}\chi_3(X)\chi_3(X\frac{y_2}{y_3}+1+\frac{y_2}{y_3})$$

$$\begin{aligned} S_3 &= \chi_3(y_2)\bar{\chi}_3(y_3)\sum_{X\neq 1,0,1+y_3/y_2}\chi_3(X)\chi_3(X+1+\frac{y_3}{y_2}) \\ &= -1+\chi_3(y_2)\bar{\chi}_3(y_3)\sum_{X\neq 0,1+y_3/y_2}\chi_3(X)\chi_3(X+1+\frac{y_3}{y_2}) \\ &= -1+\bar{\chi}_3(y_2)\bar{\chi}_3(y_3)\bar{\chi}_3(y_1)\sum_{x\in\mathbb{F}_{2^m}}\chi_3(x)\chi_3(x+1) \ . \end{aligned}$$

In conclusion, we obtain

$$\begin{aligned} N_2(3) &= \tfrac{1}{9}\Big[2^m - 11 - (-2)^{\frac{m}{2}}[\chi_3(y_1y_2y_3)+\bar{\chi}_3(y_1y_2y_3)] - \big(\chi_3(y_1y_2^2)+\chi_3(y_1^2y_2)+ \\ &\quad \chi_3(y_2y_3^2)+\chi_3(y_2^2y_3)+\chi_3(y_3y_1^2)+\chi_3(y_3^2y_1)\big)\Big] \end{aligned} \ .$$

Note that, if $z_1 = 0$ (which corresponds to choosing β in one particular coset), then y_2 and y_3 are cubes, and the number of solutions is

$$N_2(3) = \frac{1}{9}\left(2^m - 13 - [(-2)^{\frac{m}{2}}+2][\chi_3(y_1)+\bar{\chi}_3(y_1)]\right) \ .$$

Finally we focus our interest on the maximum over the z_i and obtain

$$1+\max_{z_1\neq z_2\neq z_3} N_2(3)=\begin{cases}\frac{1}{9}(2^m+2^{m/2}-2) & \text{for } m/2 \text{ even}\\ \frac{1}{9}(2^m+2^{m/2+1}+1) & \text{for } m/2 \text{ odd}\end{cases} .$$

Let us deal now with the case $p>2$:

$t=2$. In this case, we have

$$\prod_{i=1}^{2} I_{\mathcal{B}_h}(\beta+z_i^2)=\frac{1}{4}\left(1+\sigma_1^{(h)}+\sigma_2^{(h)}\right) \quad h=0,1 \ ,$$

where $\sigma_1^{(h)}=(-1)^h\chi_2(x_1)+(-1)^h\chi_2(x_2)$, and $\sigma_2^{(h)}=\chi_2(x_1)\chi_2(x_2)$.

Since $\sigma_1^{(0)}+\sigma_1^{(1)}=0$ and $\sigma_2^{(0)}+\sigma_2^{(1)}=2(\chi_2(x_1)\chi_2(x_2))$, the sum over β in the whole field $\mathbb{F}_{p^m}$ with the exclusion of $\beta=-z_1^2$ and $\beta=-z_2^2$ is

$$N_p(2)=\frac{1}{2}\left(p^m-2+\sum_{\beta\neq -z_1^2,-z_2^2}\left(\chi_2(\beta+z_1^2)\chi_2(\beta+z_2^2)\right)\right) .$$

Let S denote the above summation: we evaluate it in closed form by substituting $\beta=\eta-z_1^2$; since χ_2 is a nontrivial quadratic character, we have

$$S=\sum_{\eta\neq 0, z_1^2-z_2^2}\left(\chi_2(\eta)\chi_2(\eta+z_2^2-z_1^2)\right)=-1 \ ,$$

the summation being independent of the term $z_2^2-z_1^2$, which is non-zero by hypothesis. In conclusion we have

$$N_p(2)=\frac{1}{2}\left(p^m-3\right) \ ,$$

so that

$$1+\max_{z_1\neq z_2} N_p(2)=\frac{1}{2}\left(p^m-1\right) .$$

$t=3$. In this case

$$\prod_{i=1}^{3} I_{\mathcal{B}_h}(\beta+z_i^2)=\frac{1}{8}\left(1+\sigma_1^{(h)}+\sigma_2^{(h)}+\sigma_3^{(h)}\right) \quad h=0,1 \ ,$$

where $\sigma_1^{(h)}=(-1)^h\chi_2(x_1)+(-1)^h\chi_2(x_2)+(-1)^h\chi_2(x_3)$, $\sigma_2^{(h)}=\chi_2(x_1)\chi_2(x_2)+\chi_2(x_1)\chi_2(x_3)+\chi_2(x_2)\chi_2(x_3)$, and $\sigma_3^{(h)}=(-1)^h\chi_2(x_1)\chi_2(x_2)\chi_2(x_3)$.

Since $\sigma_1^0+\sigma_1^1=0$, $\sigma_2^0+\sigma_2^1=2(\chi_2(x_1)\chi_2(x_2)+\chi_2(x_1)\chi_2(x_3)+\chi_2(x_2)\chi_2(x_3))$, and $\sigma_3^0+\sigma_3^1=0$, the summation over β of the sum of the two products, where the values of β equal to $-z_1^2,-z_2^2$, and $-z_3^2$ are excluded, becomes

$$N_p(3)=\frac{1}{4}\left(p^m-3+\sum_{\beta\neq -z_1^2,-z_2^2,-z_3^2}\left[\chi_2(x_1)\chi_2(x_2)+\chi_2(x_1)\chi_2(x_3)+\chi_2(x_2)\chi_2(x_3)\right]\right) .$$

We thus need to evaluate only one type of summation, namely

$$S_2 = \sum_{\beta \neq -z_1^2, -z_2^2, -z_3^2} \chi_2(\beta+z_1^2)\chi_2(\beta+z_2^2) = \sum_{\substack{\eta \neq 0 \\ z_1^2-z_2^2, z_1^2-z_3^2}} \chi_2(\eta)\chi_2(\eta+z_2^2-z_1^2) = -1-\chi_2(z_1^2-z_3^2)\chi_2(z_2^2-z_3^2)\,,$$

the remainder being obtained by symmetry. In conclusion, we obtain

$$N_p(3) = \frac{1}{4}\left[p^m - 6 - (\chi_2(z_1^2-z_3^2)\chi_2(z_2^2-z_3^2) + \chi_2(z_1^2-z_2^2)\chi_2(z_3^2-z_2^2) + \chi_2(z_3^2-z_1^2)\chi_2(z_2^2-z_1^2))\right] \ .$$

And , if we consider the maximum, we have

$$1 + \max_{z_1 \neq z_2 \neq z_3} N_p(3) == \begin{cases} \frac{1}{4}(p^m-1) & p = 4k+1 \\ \frac{1}{4}(p^m+1) & p = 4k+3, \ \ m \text{ odd} \\ \frac{1}{4}(p^m-1) & p = 4k+3, \ \ m \text{ even} \end{cases}$$

3.4.2 Bounds

As the number of equations in system 3.1 or 3.3 becomes larger, exact computations become less meaningful for our purpose, as it would then be necessary to think about estimates and bounds on rather cumbersome expressions. We will thus shift our interest to a general upper bound for the function $N_p(r)$; we will first deal with the case $p = 2$, then the case $p > 2$.

Consider equation (3.2) written as

$$N_2(r) = \frac{1}{3^r} \sum_{\substack{\beta \in \mathbb{F}_{2^m} \\ \beta \notin \{z_i^3\}}} [\mathfrak{P}_0 + \mathfrak{P}_1 + \mathfrak{P}_2] \ \ , \tag{3.5}$$

where

$$\mathfrak{P}_k = 3^r \prod_{i=1}^{r} I_{\mathcal{A}_k}(x_i) = 1 + \sigma_1^{(k)} + \sigma_2^{(k)} + \cdots + \sigma_r^{(k)} \qquad k = 0, 1, 2 \ \ ,$$

x_i being $\beta + z_i^3$, and each $\sigma_j^{(k)}$ is a sum of monomials which are products of the same number j of distinct variables (characters) $\chi_3(x_i)$ or $\bar{\chi}_3(x_i)$, possibly times ζ_3 or ζ_3^2. In particular the number of addends in $\sigma_j^{(k)}$ is $2^j \begin{pmatrix} r \\ j \end{pmatrix}$.

Define $\sigma_j = \sigma_j^{(0)} + \sigma_j^{(1)} + \sigma_j^{(2)}$ for every $j = 1, \ldots, r$; then σ_j contains fewer addends than any $\sigma_j^{(k)}$, since all monomials multiplied by either ζ_3 or ζ_3^2 are canceled out with monomials multiplied by 1, and the surviving monomials are multiplied by 3 (see also the examples above). In particular, σ_1 is zero; σ_2 is a sum of monomials of the form $\chi_3(x_i)\bar{\chi}_3(x_l)$ (i, l distinct), whose total number is $2\begin{pmatrix} r \\ 2 \end{pmatrix}$; σ_3 is a sum of monomials of the form $\chi_3(x_i)\chi_3(x_l)\chi_3(x_m)$ (i, l, m all distinct), whose total number is $2\begin{pmatrix} r \\ 3 \end{pmatrix}$; and σ_4 is a sum of monomials of the form $\chi_3(x_i)\chi_3(x_l)\bar{\chi}_3(x_m)\bar{\chi}_3(x_s)$ (i, l, m, s all distinct), whose total number is $6\begin{pmatrix} r \\ 4 \end{pmatrix}$. In general, the number of surviving monomials of

degree j can be computed by considering that each monomial is a product of n_1 characters and n_2 complex conjugate characters; thus $n_1 + n_2 = j$. Supposing that $\chi_3(x_i)$ are multiplied by ζ_3 and $\bar{\chi}_3(x_h)$ are multiplied by ζ_3^2, the surviving monomial satisfies the condition $n_1 + 2n_2 = 0 \bmod 3$. Therefore, the admissible values of $0 \leq n_2 \leq j$ satisfy the condition $n_2 = 2j \bmod 3$: if $e = 2j \bmod 3$ and $e \in \{0, 1, 2\}$, the number of surviving monomials is $\left(\begin{array}{c} r \\ j \end{array} \right) a_j$, where $a_j = \sum_{h=0}^{\lfloor \frac{j-e}{3} \rfloor} \left(\begin{array}{c} j \\ e+3h \end{array} \right)$, with $\{a_j\}_{\mathbb{Z}_{>1}} = 2, 2, 6, 10, 22, 42, 86, 170, 342 \ldots$ matching the sequence A078008 in [113] with the first two terms disregarded. We observe now that the product of j characters, whose arguments are distinct linear functions of β, can be interpreted as a single character whose argument is a polynomial $f(\beta)$ with j distinct roots: by [108, Theorem 2C'], each sum of these characters is upper bounded by $(j-1)\sqrt{2^m}$, so that

$$N_2(r) \leq \frac{1}{3^{r-1}} \left[2^m - r + \sum_{j=2}^{r} a_j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) \sqrt{2^m} \right] \quad .$$

The summation above is evaluated as follows, using the expression $a_j = \frac{1}{3} \sum_{h=0}^{2} \zeta_3^{-he} (1 + \zeta_3^h)^j$ for the sequence a_j as can be found in [10, 11, 52]:

$$\sum_{j=2}^{r} a_j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) = \sum_{j=2}^{r} \frac{1}{3} \sum_{h=0}^{2} \zeta_3^{-he} (1+\zeta_3^h)^j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) = \frac{1}{3} \sum_{h=0}^{2} \sum_{j=2}^{r} \zeta_3^{-he} (1+\zeta_3^h)^j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right).$$

Now, observing that $e = -j \bmod 3$ and ζ_3 is a cubic root of the unity, we may substitute ζ_3^{hj} for ζ_3^{-he} and write $(\zeta_3^h + \zeta_3^{2h})^j$ for $\zeta_3^{hj}(1 + \zeta_3^h)^j$ in the last expression, which we then write as

$$\frac{1}{3} \sum_{h=0}^{2} \sum_{j=0}^{r} (\zeta_3^h + \zeta_3^{2h})^j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) + 1 = 1 + \frac{1}{3} \sum_{h=0}^{2} \left(\sum_{j=0}^{r} j (\zeta_3^h + \zeta_3^{2h})^j \left(\begin{array}{c} r \\ j \end{array} \right) - \sum_{j=0}^{r} (\zeta_3^h + \zeta_3^{2h})^j \left(\begin{array}{c} r \\ j \end{array} \right) \right) \quad .$$

Using the binomial sum and its derivative, we finally obtain

$$\sum_{j=2}^{r} a_j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) = 1 + \frac{1}{3} \sum_{h=0}^{2} \left(r(\zeta_3^h + \zeta_3^{2h})(1 + \zeta_3^h + \zeta_3^{2h})^{r-1} - (1 + \zeta_3^h + \zeta_3^{2h})^r \right) \quad ,$$

that is

$$\sum_{j=2}^{r} a_j (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) = 1 + \frac{1}{3} [2r 3^{r-1} - 3^r] \quad ,$$

because $(1 + \zeta_3^h + \zeta_3^{2h})$ is 3 when $h = 0$ and is 0 otherwise. In conclusion

$$N_2(r) \leq \frac{1}{3^{r-1}} \left[2^m + \sqrt{2^m} - r + 3^{r-2}(2r-3)\sqrt{2^m} \right] \quad ,$$

where we see that, when $3^{r-2}(2r-3)\sqrt{2^m} - r + \sqrt{2^m} << 2^m$, roughly $r << m/2$, then $N_2(r) \simeq \frac{2^m}{3^{r-1}}$, so that this deterministic bound supports the probabilistic estimate discussed above.

In the case $p > 2$, consider equation (3.4) written as

$$N_p(r) = \frac{1}{2^r} \sum_{\substack{\beta \in \mathbb{F}_{p^m} \\ \beta \notin \{-z_i^2\}}} [\mathfrak{Q}_0 + \mathfrak{Q}_1] \quad , \tag{3.6}$$

where

$$\mathfrak{Q}_k = 2^r \prod_{i=1}^{r} I_{\mathcal{B}_k}(x_i) = 1 + \sigma_1^{(k)} + \sigma_2^{(k)} + \cdots + \sigma_r^{(k)} \qquad k = 0, 1 \quad ,$$

x_i being $\beta + z_i^2$, and each $\sigma_j^{(k)}$ is a sum of monomials which are products of the same number j of distinct variables (characters) $\chi_2(x_i)$. In particular, only $\sigma_j^{(k)}$s with even subscripts occur, and clearly they are the elementary symmetric functions of r variables; thus the number of addends in $\sigma_j^{(k)}$ is $\left(\begin{array}{c} r \\ j \end{array} \right)$. The same argument used to upper bound $N_2(r)$ also applies here, in this case the sum of products of j characters is bounded as $(j-1)\sqrt{p^m}$ by [108, Theorem 2C'], so that

$$N_p(r) \leq \frac{1}{2^{r-1}} \left[p^m - r + \sum_{j=2}^{r} (j-1) \left(\begin{array}{c} r \\ j \end{array} \right) \sqrt{p^m} \right] \quad .$$

which, after some manipulation, can be written as

$$N_p(r) \leq \frac{1}{2^{r-1}} \left[p^m - r + [2^{r-1}(r-2) + 1] \sqrt{p^m} \right] \quad ,$$

and we see that, when $[2^{r-1}(r-2)+1]\sqrt{p^m} - r << p^m$, roughly $r << \frac{m}{2} \log_2 p$, then $N_p(r) \simeq \frac{p^m}{2^{r-1}}$ as in our probabilistic estimate.

3.5 Deterministic splitting II: fixed N

This section examines the smallest t such that the algorithm succeeds, in at most 1 or 2 attempts: we will call these $t_0(1)$ and $t_0(2)$, respectively.

Clearly, $t_0(1) = \ell_m + 1$, since there are exactly ℓ_m elements belonging to a given coset; then, if $t > \ell_m$, the algorithm succeeds at the first attempt.

To evaluate $t_0(2)$, we must examine the number of representations of a $\beta \neq 0$ in the field being the sum of an element in a given coset and an element in another (possibly the same) given coset (see also [81, 80, 98]). We then consider the maximum M, over $\beta \neq 0$ in the field and over all possible pairs of cosets, so that $t_0(2)$ is $1 + M$.

For the case of the cubic character, M can be calculated as follows:

$$M = \max_{i,j,\beta} \sum_{z \neq 0, \beta} \frac{1 + \zeta_3^{2j} \chi_3(z) + \zeta_3^{j} \bar{\chi}_3(z)}{3} \; \frac{1 + \zeta_3^{2i} \chi_3(\beta + z) + \zeta_3^{i} \bar{\chi}_3(\beta + z)}{3}$$

which is the maximum over i, j, β of the following expression:

$$\frac{1}{9} \left[2^m - 2 - \chi_3(\beta)(\zeta_3^{2i} + \zeta_3^{2j}) - \bar{\chi}_3(\beta)(\zeta_3^{i} + \zeta_3^{j}) - \zeta_3^{2i+j} - \zeta_3^{i+2j} - (-2)^{m/2}(\zeta_3^{2i+2j} \bar{\chi}_3(\beta) + \zeta_3^{i+j} \chi_3(\beta)) \right] ,$$

where we have again exploited the relations $\sum_{x\in\mathbb{F}_{2^m}}\chi_3(x)=0$, $\sum_{x\in\mathbb{F}_{2^m}}\chi_3(x)\bar{\chi}_3(x+\beta)=-1$ and $\sum_{x\in\mathbb{F}_{2^m}}\chi_3(x)\chi_3(x+1)=G_m(1,\chi_3)=-(-2)^{m/2}$ ([16, 102, 126]). Then we have

$$M=\left\{\begin{array}{l}\frac{1}{9}(2^m+2^{m/2}-2)\quad \text{for } m/2 \text{ even}\\ \frac{1}{9}(2^m+2^{m/2+1}+1)\quad \text{for } m/2 \text{ odd}\end{array}\right. .$$

For the case of the quadratic character, we consider similarly

$$\begin{aligned} M &= \max_{i,j,\beta}\sum_{z\neq 0,\beta}\frac{1+(-1)^j\chi_2(z)}{2}\,\frac{1+(-1)^i\chi_2(\beta-z)}{2}\\ &= \max_{i,j,\beta}\left\{\frac{1}{4}\left(p^m-2-\chi_2(\beta)(-1)^i-\chi_2(\beta)(-1)^j-(-1)^{i+j}\chi_2(-1)\right)\right\},\end{aligned}$$

therefore

$$M=\left\{\begin{array}{ll}\frac{1}{4}(p^m-1) & p=4k+1\\ \frac{1}{4}(p^m+1) & p=4k+3,\ m \text{ odd}\\ \frac{1}{4}(p^m-1) & p=4k+3,\ m \text{ even}\end{array}\right.$$

Remark 3.10. *It is interesting to notice that M, which is the maximum t such that it is still possible to fail splitting a polynomial of degree t with two attempts, is equal to the maximum number of attempts to split a polynomial of degree 3. Similarly, ℓ_m is at the same time the maximum t such that it is possible to fail splitting a polynomial of degree t at the first attempt and the maximum number of attempts to split a polynomial of degree 2.*

Chapter 4

On the decoding complexity of cyclic codes

The standard algebraic decoding algorithm of cyclic codes $[n, k, d]$ up to the BCH bound $\delta = 2t+1$ is very efficient and practical for relatively small n while it becomes unpractical for large n as its computational complexity is $O(nt)$.

Using the results of the previous chapters, we show here, following [106], how to make this algebraic decoding computationally more efficient: in the case of binary codes, for example, the complexity of the syndrome computation drops from $O(nt)$ to $O(t\sqrt{n})$, while the average complexity of the error location drops from $O(nt)$ to $\max\{O(t\sqrt{n}), O(t\log^2(t)\log\log(t)\log(n))\}$.

4.1 Introduction

The algebraic decoding of cyclic codes up to the BCH bound, as obtained early in the sixties with the contribution of many people, was considered very efficient for the needs of that time ([14, 19, 69, 72, 94, 95]). However, today we can and need to manage error correcting codes of sizes that require more efficient algorithms, possibly at the limit of their theoretical minimum complexity. We are proposing here an algorithm that goes in this direction.

Although we will focus as our main point of reference and comparison on the classical algebraic decoding, there are other decoding algorithms that have been recently proposed and that we limit ourselves to cite here as a reference, e.g. [43, 53, 82, 84].

Let us summarize now the standard algebraic decoding of cyclic codes: let $\mathcal{C}$ be an $[n, k, d]$ cyclic code over a finite field $\mathbb{F}_q$, $q = p^s$ for a prime p, with generator polynomial of minimal degree $r = n - k$

$$g(x) = x^r + g_1 x^{r-1} + \ldots + g_{r-1}x + g_r \ ,$$

$g(x)$ dividing $x^n - 1$, and let α be a primitive n-th root of unity lying in a finite field $\mathbb{F}_{p^m}$, where the extension degree is the minimum integer m such that n is a divisor of $p^m - 1$. Assuming that $\mathcal{C}$ has BCH bound $\delta = 2t + 1$ (if δ is even, we would just consider $\delta - 1$), then $g(x)$ has $2t$ roots with consecutive power exponents, so that the whole set of roots is

$$\mathfrak{R} = \{\alpha^{\ell+1}, \alpha^{\ell+2}, \ldots, \alpha^{\ell+2t}, \alpha^{s_{2t+1}}, \ldots, \alpha^{s_r}\} \ ,$$

where it is not restrictive to take $\ell = 0$ as it is usually done.
Let $R(x) = g(x)I(x) + e(x)$ be a received code word such that the error pattern $e(x)$ has no more than t nonzero coefficients. The Gorenstein-Peterson-Zierler decoding procedure ([69, 94]), which is a standard decoding procedure for every cyclic code up to the BCH bound, is made up of four steps:

- Computation of $2t$ syndromes: $S_j = R(\alpha^j)$, $j = 1, \ldots, 2t$.
- Computation of the error-locator polynomial $\sigma(z) = \sigma_t z^t + \sigma_{t-1} z^{t-1} + \cdots + \sigma_1 z + 1$ (we are assuming the case that exactly t errors occurred; if there are $t_e < t$ errors, this step would output a polynomial of degree t_e).
- Computation of the roots of $\sigma(z)$ in the form α^{-j_h}, $h = 1, \ldots, t$, yielding the error positions j_h.
- Computation of the error magnitudes.

Efficient implementations of this decoding algorithm combine the computation of $2t$ syndromes using Horner's rule, the Berlekamp-Massey algorithm to obtain the error-locator polynomial, the Chien search to locate the errors, and the evaluation of Forney's polynomial to estimate the error magnitudes.

The computation of the $2t$ syndromes using Horner's rule requires $2tn$ multiplications in $\mathbb{F}_{p^m}$, which may be prohibitive when n is large. The Berlekamp-Massey algorithm has multiplicative complexity $O(t^2)$ ([19, 57]), is very efficient and will not be discussed further later on. The Chien search requires again $O(tn)$ multiplications in $\mathbb{F}_{p^m}$ and Forney's algorithm $O(t^2)$ ([57]). Notice that this fourth step is not required if we deal with binary codes and that both the first and the fourth steps consist primarily in polynomial evaluations, so they can benefit from any efficient polynomial evaluation algorithm, as we will show.

The standard decoding procedure is satisfactory when the code length n is not too large (say $< 10^3$) and efficient implementations are set up taking advantage of the particular structure of the code. The situation changes dramatically when n is of the order of 10^6 or larger. In this case a complexity $O(tn)$, required by the syndrome evaluations and by the Chien search, is not acceptable anymore.

Here we describe some methods using the results of the previous chapters to make these steps more efficient and practical even for large n. We will follow the usual approach of focusing as above in computing the number of multiplications, as they are more expensive than sums (see also [39]).

The chapter is structured as follows: Section 4.2 concerns the computation of syndromes. Section 4.3 deals with the computation of the roots of the error-locator polynomial as well as the corresponding error positions; the error locator polynomial is supposed to be given (being computed by Berlekamp-Massey algorithm). Finally, Section 4.4 gives a numerical example illustrating the whole procedure.

4.2 Syndrome Evaluation

Let β be any element of $\mathfrak{R}$, the standard Horner's rule ([63],[67]) allows us to compute $R(\beta)$ in at most n products, thus for the computation of $2t$ syndromes we have the estimate $O(tn)$. However,

in Chapter 2 or in [40, 105] we showed that polynomials over a finite field of characteristic p can be evaluated more efficiently by exploiting the Frobenius automorphism with a significant computational cost reduction.

Briefly, to evaluate a polynomial $r(x)$ of degree n over $\mathbb{F}_{p^s}$, in β, an element of $\mathbb{F}_{p^m}$, one can achieve an overall complexity of approximately $2s\sqrt{n(p-1)}$. In the particular case of binary codes, the complexity is $2\sqrt{n}$.

It was also shown that in particular situations, like the example of the Reed-Solomon code $[255, 223, 33]$, a better cost reduction can be obtained by means of a different use of the Frobenius automorphism and a careful choice of the number of iterations.

We again point out that in hardware implementations, the proposed algorithm allows a strong parallelism, while Horner's rule is inherently serial. Moreover an additional gain may be given by the pre-computation of the powers of β, especially when the number of syndromes to be computed is big. Furthermore, like in Horner's rule, multiplication by β or its powers can be performed using Linear Feedback Shift Registers ([51, 69, 75]) with a further speed up at a very small cost, while the p-power operations would benefit from the use of a normal basis ([63, 68]).

4.3 Roots of the error-locator polynomial

Once the error locator polynomial $\sigma(z)$ is computed from the syndromes using the Berlekamp-Massey algorithm, its roots, represented in the form $\alpha^{-\ell_i}$, correspond to the error positions ℓ_i, $i = 1, \ldots, t$, which are generally found by testing $\sigma(\alpha^{-i})$ for all n possible powers α^{-i} with an algorithm usually referred to as the Chien search. In this approach, if $\sigma(\alpha^{-j}) = 0$ an error in position j is recognized, otherwise the position is correct. However, this simple mechanism can be unacceptably slow when n is large since its complexity is $O(tn)$: aim of this Section is to describe a less costly procedure.
The Cantor-Zassenhaus probabilistic factorization algorithm is very efficient in factoring a polynomial and consequently in computing the roots of a polynomial ([12, 49]). Since $\sigma(z)$ is the product of t linear factors $z + \rho_i$ over $\mathbb{F}_{p^m}$ (i.e. ρ_i is a p-ary polynomial in α of degree $m-1$), this factoring algorithm can be directly applied to separate these t factors. The error positions ℓ_i are then obtained by computing the discrete logarithm of $(\rho_i)^{-1} = \alpha^{\ell_i}$ to base α. This task can be performed by Shanks' algorithm ([109]), which we revisit below. The overall expected complexity of finding the error positions with this algorithm is $O(mt \log^2 t \log\log t)$ ([12]), plus $O(t\sqrt{n})$, where the second addend comes from Shanks' algorithm. Moreover, better computational estimates may be obtained taking into account the considerations and improvements highlighted in the previous chapter or in [41].

4.3.1 Cantor-Zassenhaus algorithm

The algorithm of Cantor-Zassenhaus is reported here again for easy reference as in [26] and only in the case of characteristic 2, which is by far the most common in practice. Assume that $p(z)$ is a polynomial over $\mathbb{F}_{2^m}$ that is a product of t polynomials of degree 1 over the same field $\mathbb{F}_{2^m}$, m even (when m is odd it is enough to consider a quadratic extension and proceed as in the case of even m). Suppose that α is a known primitive element in $\mathbb{F}_{2^m}$, and set $\ell_m = \frac{2^m-1}{3}$, then $\rho = \alpha^{\ell_m}$ is a primitive cubic root in $\mathbb{F}_{2^m}$, so that ρ is a root of $z^2 + z + 1$. The algorithm consists of the following steps:

1. Generate a random polynomial $b(z)$ of degree not greater than $t-1$ over $\mathbb{F}_{2^m}$.
2. Compute $a(z) = b(z)^{\ell_m} \bmod p(z)$.
3. IF $a(z) \neq 0, 1, \rho, \rho^2$, THEN at least a polynomial among
$$\gcd\{p(z), a(z)\}, \gcd\{p(z), a(z)+1\}, \gcd\{p(z), a(z)+\rho\}, \gcd\{p(z), a(z)+\rho^2\}$$
will be a non trivial factor of $p(z)$, ELSE repeat from point 1.
4. Iterate until all linear factors of $p(z)$ are found.

Remark 4.1. *As shown in the previous chapter, the polynomial $b(z)$ can be conveniently chosen of the form $z+\beta$, using $b(z) = z$ as initial choice.*

Success has been shown to happen probabilistically, and often deterministically, very soon, expecially when the degree of $\sigma(z)$ is high.

4.3.2 Shanks' algorithm

Shanks' algorithm can be applied to compute the discrete logarithm in a group of order n generated by the primitive element α. The exponent ℓ in the equality

$$\alpha^{\ell} = b_0 + b_1\alpha + \cdots + b_{s-1}\alpha^{s-1} \ .$$

is written in the form $\ell = \ell_0 + \ell_1\lceil\sqrt{n}\rceil$. A table $\mathcal{T}$ is constructed with $\lceil\sqrt{n}\rceil$ entries $\alpha^{\ell_1\lceil\sqrt{n}\rceil}$ which are sorted in some well defined order, then a cycle of length $\lceil\sqrt{n}\rceil$ is started computing

$$A_j = (b_0 + b_1\alpha + \cdots + b_{s-1}\alpha^{s-1})\alpha^{-j} \ \ j = 0, \ldots, \lceil\sqrt{n}\rceil - 1 \ ,$$

and looking for A_j in the Table; when a match is found with the κ-th entry, we set $\ell_0 = j$ and $\ell_1 = \kappa$, and the discrete logarithm ℓ is obtained as $j + \kappa\lceil\sqrt{n}\rceil$.
This algorithm can be performed with complexity $O(\sqrt{n})$ both in time and space (memory). In our scenario, since we need to compute t roots, the complexity is $O(t\sqrt{n})$.

Remark 4.2. *The Cantor-Zassenhaus algorithm finds the roots $X_j = \alpha^{\ell_j}$ of the reciprocal of the error locator polynomial, then the baby-step giant-step algorithm of Shanks' finds the error positions ℓ_js. As said in the introduction, this is the end of the decoding process for binary codes. For non-binary codes, Forney's polynomial $\Gamma(x) = \sigma(x)(S(x)+1) \bmod x^{2t+1}$, where $S(x) = \sum_{i=1}^{2t} S_i x^i$ ([122]), yields the error values*

$$Y_j = -X_j \frac{\Gamma(X_j^{-1})}{\sigma'(X_j^{-1})} \ .$$

Again we remark that this last step can benefit from an efficient polynomial evaluation algorithm, such as the one discussed in Chapter 2.

Remark 4.3. *We observe that the above procedure can be used to decode beyond the BCH bound, up to the minimum distance, whenever the error locator polynomial can be computed from a full set of syndromes ([38, 42, 99, 122]).*

4.4 A numerical example

In the previous sections we presented methods to compute syndromes and error locations in the GPZ decoding scheme of cyclic codes up to their BCH bound, which are asymptotically better than the classical algorithms. The following example illustrates the complete new procedure. Consider a binary BCH code $[63, 45, 7]$ with generator polynomial

$$g(x) = x^{18} + x^{17} + x^{14} + x^{13} + x^9 + x^7 + x^5 + x^3 + 1$$

whose roots are

$$\alpha, \alpha^2, \alpha^4, \alpha^8, \alpha^{16}, \alpha^{32}, \alpha^3, \alpha^6, \alpha^{12}, \alpha^{24}, \alpha^{48}, \alpha^{33}, \alpha^5, \alpha^{10}, \alpha^{20}, \alpha^{40}, \alpha^{17}, \alpha^{34},$$

thus the BCH bound is 7. Let $c(x) = g(x)I(x)$ be a transmitted code word, and the received word be

$$r(x) = x^{57}+x^{56}+x^{53}+x^{52}+x^{50}+x^{48}+x^{46}+x^{44}+x^{42}+x^{39}+x^{31}+x^{18}+x^{17}+x^{14}+x^{13}+x^7+x^5+x^3+1$$

where 3 errors occurred. The 6 syndromes are

$$\begin{cases} S_1 = \alpha^5 + \alpha^2 + \alpha \\ S_2 = S_1^2 \\ S_3 = \alpha^5 + \alpha^4 + \alpha^3 + \alpha^2 + \alpha \\ S_4 = S_1^4 \\ S_5 = \alpha^5 + \alpha^2 + 1 \\ S_6 = S_3^2 \end{cases}.$$

For example, S_1 has been computed considering $r(x)$ as

$$[r_{3,0} + zr_{3,1} + y(r_{3,2} + zr_{3,3})] + x[r_{3,4} + zr_{3,5} + y(r_{3,6} + zr_{3,7})],$$

with $y = x^2$, $z = x^4$, $w = x^8$ and

$$\begin{cases} r_{3,0} = w^7 + w^6 + 1 \\ r_{3,1} = w^6 + w^5 \\ r_{3,2} = w^6 + w^5 + w^2 \\ r_{3,3} = w^5 + w \\ r_{3,4} = w^7 + w^2 \\ r_{3,5} = w^6 + w + 1 \\ r_{3,6} = 1 \\ r_{3,7} = w^4 + w^3 + 1 \end{cases}$$

with only 16 products, namely 3 to compute α^2, α^4 and α^8, 6 for the powers of w up to w^7 and 7 multiplications by x, y and z.

The coefficients of the error locator polynomial turn out to be

$$\begin{cases} \sigma_1 = \alpha^5 + \alpha^2 + \alpha \\ \sigma_2 = \alpha^3 + \alpha^4 + \alpha \\ \sigma_3 = \alpha^4 + \alpha^5 + \alpha^2 \end{cases}.$$

The roots of $\sigma^*(z) = z^3\sigma(z^{-1}) = \prod_{i=1}^{3}(z - \alpha^{\ell_i})$ are computed as follows using the Cantor-Zassenhaus algorithm.

Let $\rho = \alpha^{21}$ be a cube root of the unity; consider a random polynomial, for instance $z + \rho$, of degree less than 3 and compute $a(z) = (z + \rho)^{21}$ modulo $\sigma^*(z)$ (the exponent of $z + \rho$ is $\frac{2^m-1}{3} = \frac{63}{3} = 21$):

$$(\alpha^5 + \alpha^4 + \alpha^2 + \alpha + 1)z^2 + (\alpha^3 + \alpha + 1)z + \alpha^5 + \alpha^4 + x^3 + 1 \ .$$

In this case $a(z)$ has no root in common with $\sigma^*(z)$, while

$$\gcd(a(z) + 1, \sigma^*(z)) = z + (\alpha^4 + \alpha^3 + 1) \quad (\ell_1 = 31),$$

$$\gcd(a(z) + \rho, \sigma^*(z)) = z + (\alpha^5 + \alpha^4 + \alpha^2 + 1) \quad (\ell_2 = 9),$$

$$\gcd(a(z) + \rho^2, \sigma^*(z)) = z + (\alpha^3 + \alpha) \quad (\ell_3 = 50).$$

The error positions have been obtained using Shanks' algorithm with a table of 8 entries, and a loop of length 8 for each root, for a total of 24 searches versus 63 searches of Chien's search.

4.5 Concluding remarks

A new decoding algorithm for cyclic codes has been presented having a very competitive complexity and targeting in particular those applications using error correcting codes with very large length.

Chapter 5

Enhanced public key security for the McEliece cryptosystem

This chapter presents a first application where an efficient decoding of cyclic codes would be highly desirable. We descrive a variant [7] of the McEliece cryptosystem able to ensure that the code used as the public key is no longer permutation-equivalent to the secret code. This increases the security level of the public key, thus opening the way for reconsidering the adoption of classical families of codes, like Reed-Solomon codes, that have been longly excluded from the McEliece cryptosystem for security reasons. It is well known that codes of these classes are able to yield a reduction in the key size or, equivalently, an increased level of security against information set decoding; so, these are the main advantages of the proposed solution. A drawback is on the other hand the increased decoding complexity, which, although not being the major concern in this context, has to be taken into account, possibly by means of more efficient algorithms like that of the previous chapter. We also describe possible vulnerabilities and attacks related to the considered system, and show which design choices are best suited to avoid them.

5.1 Introduction

The McEliece cryptosystem [74] is one of the most promising public-key cryptosystems able to resist attacks based on quantum computers. In fact, differently from cryptosystems exploiting integer factorization or discrete logarithms, it relies on the hardness of decoding a linear block code without any visible structure [15].

The original McEliece cryptosystem adopts the generator matrix of a binary Goppa code as the private key, and exploits a dense transformation matrix and a permutation matrix to disguise the secret key into the public one. It has resisted cryptanalysis for more than thirty years, since no polynomial-time attack to the system has been devised up to now; however, the increased computing power and the availability of optimized attack procedures have required to update its original parameters [17].

The main advantage of the McEliece cryptosystem consists in its fast encryption and decryption procedures, which require a significantly lower number of operations with respect to alternative solutions (like RSA). However, the original McEliece cryptosystem has two main disadvantages: low encryption rate and large key size, both due to the binary Goppa codes it is based

on. When adopting Goppa codes, a first improvement is obtained through the variant proposed by Niederreiter [83], which uses parity-check matrices instead of generator matrices.

A significant improvement in both the encryption rate and the key size would be obtained if other families of codes could be included in the system, allowing a more efficient code design and a more compact representation of their matrices. In particular, the use of Reed-Solomon (RS) codes could yield significant advantages. In fact, RS codes are maximum distance separable codes, which ensures they achieve maximum error correction capability under bounded-distance decoding. In the McEliece system, this translates into shorter keys for the same security level, or a higher security level for the same key size, with respect to binary Goppa codes (having the same code rate). In fact, Goppa codes are subfield subcodes of generalized RS codes and the subcoding procedure makes them less efficient than RS codes. However, this also makes them secure against key recovering attacks, while the algebraic structure of RS codes, when exposed in the public key (also in permuted form), makes them insecure against attacks aimed at recovering the secret code [112].

Many attempts of replacing Goppa codes with other families of codes have exposed the system to security threats [87], [124], and some recent proposals based on Quasi-Cyclic and Quasi-Dyadic codes have also been broken [119]. Low-Density Parity-Check (LDPC) codes, in principle, could offer high design flexibility and compact keys. However, also the use of LDPC codes may expose the system to severe flaws [79], [85]. Nevertheless, it is still possible to exploit Quasi-Cyclic LDPC codes to design a variant of the system that is immune to any known attack [6].

The idea in [6] is to replace the permutation matrix used in the original McEliece cryptosystem with a dense transformation matrix. The transformation matrix used in [6] is a sparse matrix and its density must be chosen as a trade-off between two opposite effects: i) increasing the density of the public code parity-check matrix so that it is too difficult to search for low weight codewords in its dual code and ii) limiting the propagation of the intentional errors so that they are still correctable by the legitimate receiver. The advantage of replacing the permutation with a more general transformation is that the code used as the public key is no longer permutation equivalent to the secret code. This increases the security of the public key, as it prevents an attacker from exploiting the permutation equivalence when trying to recover the secret code structure.

We elaborate on this approach by introducing a more effective class of transformation matrices and by generalizing their form also to the non-binary case. The new proposal is based on the fact that there exist some classes of dense transformation matrices that have a limited propagation effect on the intentional error vectors. The use of these matrices allows to better disguise the private key into the public one, with a controlled error propagation effect. So, we propose a modified cryptosystem that can restore the use of advantageous families of codes, as RS codes, by ensuring increased public key security.

5.2 Description of the cryptosystem

The proposed cryptosystem takes as its basis the classical McEliece cryptosystem, whose block scheme is reported in Figure 5.1, where $\mathbf{u}$ denotes a cleartext message and $\mathbf{x}$ its associated ciphertext. The main components of this system are:

- A private linear block code generator matrix $\mathbf{G}$
- A public linear block code generator matrix $\mathbf{G}'$

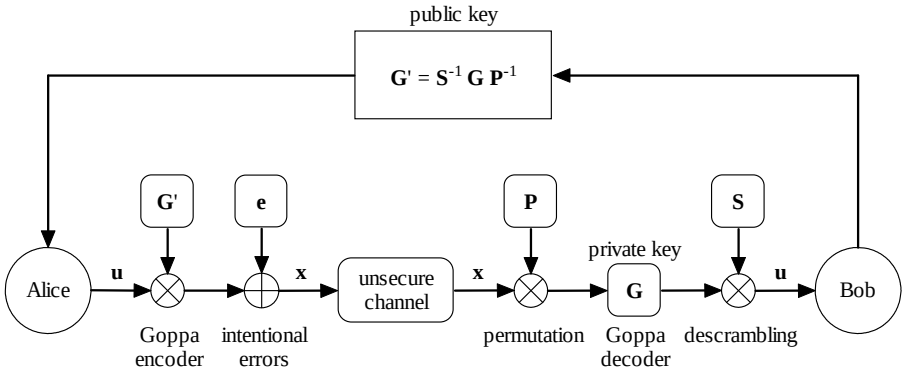


Figure 5.1: The McEliece cryptosystem.

- A secret scrambling matrix $\mathbf{S}$
- A secret permutation matrix $\mathbf{P}$
- A secret intentional error vector $\mathbf{e}$

As for the original system, the proposed cryptosystem can be implemented in the classical McEliece form or, alternatively, in the Niederreiter form. In both cases, the main element that differentiates the proposed solution from the original cryptosystem is the replacement of the permutation matrix $\mathbf{P}$ with a dense transformation matrix $\mathbf{Q}$, whose design is described next.

5.2.1 Matrix Q

The matrix $\mathbf{Q}$ is a non-singular $n \times n$ matrix having the form

$$\mathbf{Q} = \mathbf{R} + \mathbf{T}, \tag{5.1}$$

where $\mathbf{R}$ is a dense $n \times n$ matrix and $\mathbf{T}$ is a sparse $n \times n$ matrix. The matrices $\mathbf{R}$, $\mathbf{T}$ and $\mathbf{Q}$ have elements in $\mathbb{F}_q$, with $q \geq 2$.

The matrix $\mathbf{R}$ is obtained starting from two sets, $\mathcal{A}$ and $\mathcal{B}$, each containing w matrices having size $z \times n$, $z \leq n$, defined over $\mathbb{F}_q$: $\mathcal{A} = \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_w\}$, $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_w\}$. We also define $\mathbf{a} = \sum_{i=1}^{w} \mathbf{a}_i$. The matrices in $\mathcal{A}$ and $\mathcal{B}$ are secret and randomly chosen; then, $\mathbf{R}$ is obtained as:

$$\mathbf{R} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \vdots \\ \mathbf{a}_w \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_w \end{bmatrix}, \tag{5.2}$$

where ^{T} denotes transposition. Starting from (5.2), we make some simplifying assumptions, aimed at reducing the amount of secret data that is needed to be stored. In fact, for the instances of the proposed cryptosystem we consider, we will focus on two cases: i) $w = 1$, $\mathbf{a}_1 = \mathbf{a}$, any b_1 and ii) $w = 2$, $\mathbf{b}_2 = \mathbf{1} + \mathbf{b}_1$, any a_1, a_2 (we denote with $\mathbf{0}$ and $\mathbf{1}$, respectively, the all-zero and the all-one $z \times n$ matrix). In both these cases, there is no need to store nor choose the matrix $\mathbf{b}_2$. For this reason,

in order to simplify the notation, we will replace $\mathbf{b}_1$ with $\mathbf{b}$ in the following. This obviously does not limit the applicability of the general form (5.2) of the matrix $\mathbf{R}$.

Concerning the choice of the matrix $\mathbf{T}$, we denote by $\mathbf{\Pi}_i$ a generalized permutation matrix, that is, a matrix having only one non-zero element in each row and in each column, whose value is selected among the $q-1$ non-zero elements of $\mathbb{F}_q$. The matrix $\mathbf{T}$ is then obtained as the sum of $m \geq 1$ generalized permutation matrices, chosen at random:

$$\mathbf{T} = \mathbf{\Pi}_1 + \mathbf{\Pi}_2 + \ldots + \mathbf{\Pi}_m. \tag{5.3}$$

In the system we propose, the matrix $\mathbf{Q}$, having the form (5.1), replaces the permutation matrix $\mathbf{P}$ that is used in the original McEliece cryptosystem and in its Niederreiter version. As we will see in the following, both these versions exploit an intentional error vector $\mathbf{e} = [e_1, e_2, \ldots, e_n]$, randomly generated, having a predetermined weight t, like in the classical cryptosystem. Each error vector might then be subject or not to additional constraints, depending on the implementation we use, as shown later. Let us suppose now that a constraint is imposed to the vector $\mathbf{e}$ in the form:

$$\mathbf{a} \cdot \mathbf{e}^T = \mathbf{0}. \tag{5.4}$$

If we suppose that the matrix $\mathbf{a}$ is full rank, the number of constraints we impose on the intentional error vectors is equal to z. Obviously, in order to be implemented, this would require $\mathbf{a}$ to be disclosed as part of the public key, but, as we will see in the following, this, together with condition (5.4), may introduce a weakness in the system. This issue will be discussed next, together with the ways to avoid such a weakness.

For the moment, let us suppose that $\mathbf{a}$ is disclosed and that condition (5.4) is verified. As we will see in the following, for both versions of the cryptosystem it turns out that, during decryption, the matrix $\mathbf{Q}$ has a multiplicative effect on the intentional error vector $\mathbf{e}$. As a result, $\mathbf{e}$ is transformed into $\mathbf{e} \cdot \mathbf{Q} = \mathbf{e} \cdot (\mathbf{R} + \mathbf{T})$. If (5.4) holds, the contribution due to $\mathbf{R}$ becomes, for the two cases we focus on:

$$\mathbf{e} \cdot \mathbf{R} = \begin{cases} \mathbf{0}, & \text{if } \mathbf{a} = \mathbf{a}_1, \mathbf{a}_2 = \mathbf{0}, \\ \mathbf{e} \cdot \mathbf{a}_2^T \cdot \mathbf{1}, & \text{if } \mathbf{b}_2 = \mathbf{1} + \mathbf{b}. \end{cases} \tag{5.5}$$

So, in the former case, $\mathbf{e} \cdot \mathbf{Q}$ reduces to $\mathbf{e} \cdot \mathbf{T}$. In the latter case, instead, the legitimate receiver should know the value of $\mathbf{e} \cdot \mathbf{a}_2^T$ to remove the contribution due to $\mathbf{e} \cdot \mathbf{R}$. We will see in the following how this can be done.

When the result of $\mathbf{e} \cdot \mathbf{Q}$ can be reduced to $\mathbf{e} \cdot \mathbf{T}$, the use of the matrix $\mathbf{Q}$ as in (5.1) allows to amplify the number of intentional errors (at most) by a factor m. For $m = 1$, the required error correction capability is exactly the same as in the original McEliece and Niederreiter cryptosystems while, for $m > 1$, the limited error propagation effect can be compensated by using codes with a high error correction capability, as it occurs when adopting LDPC codes [6].

But the advantage of using the matrix $\mathbf{Q}$ is that it allows to disguise the private matrix of a code over $\mathbb{F}_q$ in a way that can be much stronger than what can be done by using a permutation matrix (as in the original McEliece system).

So, the proposed solution can help revitalizing previous attempts of using alternative families of codes in the McEliece system. A first idea is to reconsider the usage of RS codes over $\mathbb{F}_q$. In the following sections we will show that the attacks that have prevented their usage in the past cannot be directly applied to the new variant, so that it shall be considered secure against them.

5.2.2 McEliece version

In the McEliece version of the proposed system, Bob chooses his secret key as the $k \times n$ systematic generator matrix $\mathbf{G}$ of a linear block code over $\mathbb{F}_q$, able to correct t errors. He also chooses other two secret matrices: a $k \times k$ non-singular scrambling matrix $\mathbf{S}$ and the $n \times n$ non-singular transformation matrix $\mathbf{Q}$, defined as in (5.1). The public key is:

$$\mathbf{G}' = \mathbf{S}^{-1} \cdot \mathbf{G} \cdot \mathbf{Q}^{-1}. \tag{5.6}$$

So, in general, differently from the original McEliece cryptosystem, the public code is not permutation-equivalent to the private code.

Alice, after obtaining Bob's public key, applies the following encryption map:

$$\mathbf{x} = \mathbf{u} \cdot \mathbf{G}' + \mathbf{e}. \tag{5.7}$$

After receiving $\mathbf{x}$, Bob inverts the transformation as follows:

$$\mathbf{x}' = \mathbf{x} \cdot \mathbf{Q} = \mathbf{u} \cdot \mathbf{S}^{-1} \cdot \mathbf{G} + \mathbf{e} \cdot \mathbf{Q}, \tag{5.8}$$

thus obtaining a codeword of the secret code affected by the error vector $\mathbf{e} \cdot \mathbf{Q}$.

The special form we adopt for the matrix $\mathbf{Q}$ allows Bob to reduce $\mathbf{e} \cdot \mathbf{Q}$ to $\mathbf{e} \cdot \mathbf{T}$. Obviously, this is immediately verified when $\mathbf{e} \cdot \mathbf{R} = \mathbf{0}$, while it will be shown in Sections 5.3.2 and 5.3.3 how it can be achieved when $\mathbf{e} \cdot \mathbf{R} \neq \mathbf{0}$.

So, because of the limited error propagation effect that is due to $\mathbf{T}$, Bob is able to correct all the errors and get $\mathbf{u} \cdot \mathbf{S}^{-1}$, thanks to the systematic form of $\mathbf{G}$. He can then obtain $\mathbf{u}$ through multiplication by $\mathbf{S}$.

5.2.3 Niederreiter version

The Niederreiter version of the proposed cryptosystem works as follows. Bob chooses the secret linear block code over $\mathbb{F}_q$, able to correct t errors, by fixing its $r \times n$ parity-check matrix ($\mathbf{H}$), and obtains his public key as

$$\mathbf{H}' = \mathbf{S}^{-\mathbf{1}} \cdot \mathbf{H} \cdot \mathbf{Q}^T, \tag{5.9}$$

where the scrambling matrix $\mathbf{S}$ is a non-singular $r \times r$ matrix and the transformation matrix $\mathbf{Q}$ is defined as in (5.1).

Alice gets Bob's public key, she maps the cleartext vector into a weight t error vector $\mathbf{e}$ and calculates the ciphertext as the syndrome $\mathbf{x}$ of $\mathbf{e}$ through $\mathbf{H}'$, according to

$$\mathbf{x} = \mathbf{H}' \cdot \mathbf{e}^T. \tag{5.10}$$

In order to decrypt $\mathbf{x}$, Bob first calculates $\mathbf{x}' = \mathbf{S} \cdot \mathbf{x} = \mathbf{H} \cdot \mathbf{Q}^T \cdot \mathbf{e}^T = \mathbf{H} \cdot (\mathbf{e} \cdot \mathbf{Q})^T$. The special form of $\mathbf{Q}$ allows Bob to reduce $\mathbf{e} \cdot \mathbf{Q}$ to $\mathbf{e} \cdot \mathbf{T}$. Obviously, this is immediately verified when $\mathbf{e} \cdot \mathbf{R} = \mathbf{0}$, while it will be shown in Sections 5.3.2 and 5.3.3 how it can be achieved when $\mathbf{e} \cdot \mathbf{R} \neq \mathbf{0}$.

So, he gets $\mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T$ and, due to the limited error propagation effect of $\mathbf{T}$, he is able to obtain $\mathbf{T}^T \cdot \mathbf{e}^T$ by performing syndrome decoding through the private linear block code. Then, he multiplies the result by $(\mathbf{T}^T)^{-1}$ and finally demaps $\mathbf{e}$ into its associated cleartext vector $\mathbf{u}$.

In order to reduce the public key size, the matrix $\mathbf{H}'$, defined by (5.9), can be put in systematic form. Let us divide $\mathbf{H}'$ into a left $r \times r$ matrix $\mathbf{H}'_l$ and a right $r \times k$ matrix $\mathbf{H}'_r$, i.e. $\mathbf{H}' = [\mathbf{H}'_l | \mathbf{H}'_r]$. We can suppose, without loss of generality, that $\mathbf{H}'_l$ is full rank and obtain the systematic form of $\mathbf{H}'$ as:

$$\mathbf{H}'' = \left(\mathbf{H}'_l\right)^{-1} \cdot \mathbf{H}'. \tag{5.11}$$

If $\mathbf{H}''$ is used as the public key, only its rightmost k columns are needed to be stored. When Alice uses $\mathbf{H}''$ for encryption, she obtains a public message $\mathbf{x}'' = \mathbf{H}'' \cdot \mathbf{e}^T$. Then, Bob must compute $\mathbf{x} = \mathbf{H}'_l \cdot \mathbf{x}''$ in order to get $\mathbf{x}$ as expressed by (5.10).

5.3 System design

In this section, we describe some critical aspects and possible weaknesses that must be carefully considered in the design of the proposed system.

5.3.1 Subcode vulnerability

When $\mathbf{a} = \mathbf{a}_1$ and $\mathbf{a}_2 = \mathbf{0}$, a possible vulnerability results from condition (5.4), since, in such a case, a subcode of the public code is exposed, that is permutation-equivalent to a subcode of the private code. In fact, if we refer to the Niederreiter version of the system, an attacker could consider the subcode generated by the following parity-check matrix:

$$\mathbf{H}_S = \begin{bmatrix} \mathbf{H}' \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{S^{-1}} \cdot \mathbf{H} \cdot \mathbf{Q}^T \\ \mathbf{a} \end{bmatrix} = \begin{bmatrix} \mathbf{S^{-1}} \cdot \mathbf{H} \cdot \mathbf{R}^T + \mathbf{S^{-1}} \cdot \mathbf{H} \cdot \mathbf{T}^T \\ \mathbf{a} \end{bmatrix}. \tag{5.12}$$

Each codeword $\mathbf{c}$ in the code defined by $\mathbf{H}_S$ must verify $\mathbf{a} \cdot \mathbf{c}^T = \mathbf{0}$. Due to the form of $\mathbf{R}$, this also implies $\mathbf{R}^T \cdot \mathbf{c}^T = \mathbf{0}$, so $\mathbf{H}_S$ defines a subcode of $\mathbf{H}'$ in which all codewords satisfy $\mathbf{S^{-1}} \cdot \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{c}^T = \mathbf{0}$. Hence, the effect of the dense $\mathbf{R}$ is removed and, when $\mathbf{T}$ is a permutation matrix (that is, when $m = 1$), the subcode defined by $\mathbf{H}_S$ is permutation-equivalent to a subcode of the secret code.

The same vulnerability can also occur when $\mathbf{b}_2 = \mathbf{1} + \mathbf{b}$. In fact, in this case,

$$\mathbf{R} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \end{bmatrix}^T \cdot \begin{bmatrix} \mathbf{b} \\ \mathbf{1} + \mathbf{b} \end{bmatrix} = \mathbf{a}^T \cdot \mathbf{b} + \mathbf{a}_2^T \cdot \mathbf{1} \tag{5.13}$$

and

$$\mathbf{H} \cdot \mathbf{R}^T = \mathbf{H} \cdot \mathbf{b}^T \cdot \mathbf{a} + \mathbf{H} \cdot \mathbf{1}^T \cdot \mathbf{a}_2. \tag{5.14}$$

So, when the private code includes the all-one codeword, that is, $\mathbf{H} \cdot \mathbf{1}^T = \mathbf{0}$, it results $\mathbf{H} \cdot \mathbf{R}^T = \mathbf{H} \cdot \mathbf{b}^T \cdot \mathbf{a}$ and a vulnerable subcode is still defined by $\mathbf{H}_S$ as in (5.12). For this reason, when $\mathbf{R}$ is defined as in (5.13), codes including the all-one codeword cannot be used as secret codes. For example, when an RS code defined over $\mathbb{F}_q$ having length $n = q-1$ is used, the all-one codeword is always present. Shorter lengths should be considered in order to avoid the presence of the all-one codeword.

When an RS code is used and one of its subcodes is exposed (except for a permutation), an opponent could implement an attack of the type described in [124]. It is possible to verify that, for

practical choices of the system parameters, the subcode defined by $\mathbf{H}_S$ given by (5.12) is always weak against such an attack.

A similar situation occurs if LDPC codes are used as private codes, since low weight codewords could be searched in the dual of the subcode defined by $\mathbf{H}_S$, so revealing some rows of $\mathbf{H}$ (though permuted). Moreover, the existence of low weight codewords in the dual of a subcode of the public code could be dangerous for the system security even when $\mathbf{H}_S$ is not available to an attacker, since such codewords could still be searched in the dual of the public code. So, when dealing with LDPC codes, it is always recommended to define $\mathbf{T}$ as a sum of permutation matrices (that is, to fix $m > 1$) in order to avoid the existence of codewords with low weight in the dual of the public code [6].

In the following subsections we propose two implementations of the cryptosystem that avoid the subcode vulnerability. We describe them by making reference to the Niederreiter version of the cryptosystem, but they can also be applied to its McEliece version.

5.3.2 First implementation

A first solution to overcome the subcode vulnerability consists in maintaining $\mathbf{a}_1 = \mathbf{a}$ and $\mathbf{a}_2 = \mathbf{0}$, but hiding the constraint vector $\mathbf{a}$. This obviously would also eliminate the need of selecting the intentional error vectors according to condition (5.4).

We refer to the Niederreiter version of the cryptosystem and we fix, for simplicity, $z = 1$, but the same arguments can easily be extended to the general case $1 \leq z < n$. Let us suppose that $\mathbf{a}$ is private and that the error vector $\mathbf{e}$ generated by Alice is such that $\mathbf{a} \cdot \mathbf{e}^T = \gamma$, with $\gamma \in \mathbb{F}_q$. It follows that

$$\mathbf{R}^T \cdot \mathbf{e}^T = \gamma \mathbf{b}^T \tag{5.15}$$

and

$$\mathbf{x}' = \mathbf{S} \cdot \mathbf{x} = \gamma \mathbf{H} \cdot \mathbf{b}^T + \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T. \tag{5.16}$$

In this case, Bob can guess that the value of γ is γ_B and compute

$$\begin{array}{rcl} \mathbf{x}'' & = & \mathbf{x}' - \gamma_B \mathbf{H} \cdot \mathbf{b}^T \\ & = & (\gamma - \gamma_B)\mathbf{H} \cdot \mathbf{b}^T + \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T. \end{array} \tag{5.17}$$

So, if $\gamma_B = \gamma$, Bob obtains $\mathbf{x}'' = \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T$. In such a case, he can recover $\mathbf{e}$ through syndrome decoding, check its weight and verify that $\mathbf{a} \cdot \mathbf{e}^T = \gamma_B$. Otherwise, it is $\gamma_B \neq \gamma$ and, supposing that $\mathbf{b}$ is not a valid codeword, syndrome decoding fails or returns an error vector $\mathbf{e}' \neq \mathbf{e}$. This latter case is extremely rare, as shown below, and can also be identified by Bob by checking the weight of $\mathbf{e}'$ and the value of $\mathbf{a} \cdot \mathbf{e}'^T$. So, by iterating the procedure, that is, changing the value of γ_B, Bob is able to find the right γ.

The probability of finding a correctable syndrome $\mathbf{e}'$, for $\gamma_B \neq \gamma$, is very low. In fact, since $\mathbf{b}$ is randomly chosen, when $\gamma_B \neq \gamma$ we can suppose that the vector $(\gamma - \gamma_B)\mathbf{H} \cdot \mathbf{b}^T$ is a random $r \times 1$ vector over $\mathbb{F}_q$. The total number of correctable syndromes is $\sum_{i=1}^{t} \binom{n}{i} (q-1)^i$, while the total number of random $r \times 1$ vectors is q^r. So, the probability of obtaining a correctable syndrome is:

$$P_e = \frac{\sum_{i=1}^{t} \binom{n}{i} (q-1)^i}{q^r}. \tag{5.18}$$

The value of P_e, for practical choices of the system parameters, is very low. For example, by considering the set of parameters used in the original McEliece cryptosystem, that is, $q = 2$, $n = 1024$, $k = 524$, $t = 50$, it results $P_e \approx 10^{-65}$.

In concluding this subsection, we notice that, by using such an implementation, the complexity of the decryption stage is increased, on average, by a factor $\leq (q+1)/2$ with respect to the classical Niederreiter implementation. In fact, the average number of decryption attempts needed by Bob becomes $(q+1)/2$. However, some steps of the decryption procedure do not need to be repeated; so, an increase in the decryption complexity by a factor $(q+1)/2$ corresponds to a pessimistic estimate.

5.3.3 Second implementation

A second solution to the subcode vulnerability is to adopt the choice $\mathbf{a} = \mathbf{a}_1 + \mathbf{a}_2$, $\mathbf{b}_2 = \mathbf{1} + \mathbf{b}$ and to preserve condition (5.4), that implies, for Alice, the need to perform a selection of the error vectors. In this case, according to (5.5):

$$\mathbf{R}^T \cdot \mathbf{e}^T = \mathbf{1}^T \cdot \mathbf{a}_2 \cdot \mathbf{e}^T. \tag{5.19}$$

If we fix, for simplicity, $z = 1$ (but the same arguments can easily be extended to the general case $1 \leq z < n$) and suppose to work over $\mathbb{F}_q$, the possible values of $\alpha = \mathbf{a}_2 \cdot \mathbf{e}^T$ are, obviously, q. So, Bob needs to make up to q guesses on the value of α.

First, Bob computes $\mathbf{x}' = \mathbf{S} \cdot \mathbf{x} = \mathbf{H} \cdot (\mathbf{R} + \mathbf{T})^T \cdot \mathbf{e}^T$. By using (5.19), we have:

$$\mathbf{x}' = \mathbf{H} \cdot \mathbf{1}^T \cdot \alpha + \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T. \tag{5.20}$$

We observe that, if the secret code included the all-one codeword, then $\mathbf{H} \cdot \mathbf{1}^T = \mathbf{0}$ and Bob would not need to guess the value of α. However, in this version of the cryptosystem, the use of codes including the all-one codeword is prevented by the subcode vulnerability, as shown in Section 5.3.1, so this facilitation cannot be exploited. Instead, Bob needs to make a first guess by supposing $\alpha = \alpha_B$ and to calculate

$$\mathbf{x''}_{\alpha_B} = \mathbf{x}' - \mathbf{H} \cdot \mathbf{1}^T \cdot \alpha_B = \mathbf{H} \cdot \mathbf{1}^T \cdot (\alpha - \alpha_B) + \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T. \tag{5.21}$$

If $\alpha_B = \alpha$, then $\mathbf{x''}_{\alpha_B} = \mathbf{H} \cdot \mathbf{T}^T \cdot \mathbf{e}^T$; therefore, Bob can recover $\mathbf{e}$ through syndrome decoding, check its weight and verify that $\mathbf{a}_2 \cdot \mathbf{e}^T = \alpha_B$. Otherwise, the application of syndrome decoding on $\mathbf{x''}_{\alpha_B}$ results in a decoding failure or in obtaining $\mathbf{e}' \neq \mathbf{e}$, for $\alpha_B \neq \alpha$. As for the first implementation, in this case the probability of obtaining a correctable syndrome $\mathbf{e}'$ is very small; so, when $\alpha_B \neq \alpha$, the decoder will end up reporting failure in most cases.

Also in this case, the average number of decryption attempts needed by Bob is $(q+1)/2$, and the decryption complexity increases by a factor $\leq (q+1)/2$.

Concerning the subcode vulnerability, by using $\mathbf{a}_1 \neq \mathbf{a}$ and $\mathbf{a}_2 \neq \mathbf{a}$, the matrix $\mathbf{H}_S$ as in (5.12) no longer defines a subcode permutation-equivalent to a subcode of the secret code. So, provided that the private code does not include the all-one codeword (for the reasons explained in Section 5.3.1), the subcode vulnerability is eliminated.

Note that an attacker could try to sum two rows of $\mathbf{H}'$, hoping that one of them corresponds to a copy of the vector $\mathbf{a}_1$ in $\mathbf{R}$ and the other to a copy of the vector $\mathbf{a}_2$, so that the sum of the two rows might still contain the vector $\mathbf{a}$. If he were able to select only those sums of this type,

then he might be able to find a weak subcode. This appears to be a hard task for the following reasons. If he adds one row with all the other rows, he would get, on average, only $r/2 = (n-k)/2$ rows containing the vector $\mathbf{a}$, while the other sums would contain $2\mathbf{a}_1$ or $2\mathbf{a}_2$; even if he were able to select the rows corresponding to $\mathbf{a}$, the dimension of the subcode would not be large enough for a feasible attack [77], [124]. Furthermore, effectively obtaining $\mathbf{a}$ in the sum of two rows also depends on how $\mathbf{H}$ is built, i.e. it may occur only if some special relations between elements of $\mathbf{H}$ are satisfied. Lastly, to sum pairs of rows would also imply to sum pairs of rows of $\mathbf{T}^T$; so, their (very low) weight would be doubled with a very high probability, making decoding harder.

For these reasons, it seems not easy to devise a further vulnerability for the subcode that may allow to mount an attack against this implementation.

5.3.4 Choice of Q

Also the choice of the matrix $\mathbf{Q}$ can show some critical aspects. Let us focus on the binary case ($q = 2$) and consider a particular instance of the first implementation, in which the matrix $\mathbf{Q}$ is obtained as

$$\mathbf{Q}_1 = \mathbf{R} + \mathbf{P}_1, \tag{5.22}$$

with $\mathbf{P}_1$ being a permutation matrix and

$$\mathbf{R} = \mathbf{a}^T \cdot \mathbf{b} = \begin{bmatrix} a_1 & a_2 & \cdots & a_n \end{bmatrix}^T \cdot \begin{bmatrix} b_1 & b_2 & \cdots & b_n \end{bmatrix}, \tag{5.23}$$

where $\mathbf{a}$ and $\mathbf{b}$ are two random vectors over $\mathbb{F}_2$.

In the choice of $\mathbf{Q}_1$ it is important to avoid some special cases which could allow an attacker to derive a code that is permutation-equivalent to the secret one, thus bringing security back to that of the classical McEliece system.

Let us suppose that the j-th element of $\mathbf{b}$ is zero and that $\mathbf{P}_1$ has a symbol 1 at position (i, j). In this case, the j-th column of $\mathbf{Q}_1$ is null, except for its element at row i. Since $\mathbf{Q}_1^{-1} = \widehat{\mathbf{Q}}/|\mathbf{Q}|$, where $\widehat{\mathbf{Q}}$ is the adjoint matrix and $|\mathbf{Q}|$ is the determinant of $\mathbf{Q}_1$, it follows from the definition of $\widehat{\mathbf{Q}}$ that the i-th column of $\mathbf{Q}_1^{-1}$ is null, except for its element at row j. So, the i-th column of $\mathbf{Q}_1^{-1}$ has the effect of a column permutation, like in the original McEliece cryptosystem.

In order to avoid such a possible flaw, we impose that all the elements of $\mathbf{b}$ are non-zero. If we limit to the binary case, this imposes that $\mathbf{b}$ is the all-one vector. However, in such a case, further issues exist in the design of $\mathbf{Q}$. For example, let us consider $\mathbf{a}$ as an all-one vector too, so that $\mathbf{R} = \mathbf{1}$. A valid parity-check matrix for the public code is:

$$\mathbf{H}' = \mathbf{H} \cdot \mathbf{Q}^T, \tag{5.24}$$

where $\mathbf{H}$ is the parity-check matrix of the private code. In the special case of $\mathbf{Q}_1 = \mathbf{1} + \mathbf{P}_1$, we have $\mathbf{H}' = \mathbf{H} \cdot \mathbf{1} + \mathbf{H} \cdot \mathbf{P}_1^T$. By assuming a regular $\mathbf{H}$ (i.e. with constant row and column weights), two cases are possible:

- If the rows of $\mathbf{H}$ have even weight, $\mathbf{H} \cdot \mathbf{1} = \mathbf{0}$ and $\mathbf{H}' = \mathbf{H} \cdot \mathbf{P}_1^T$.
- If the rows of $\mathbf{H}$ have odd weight, $\mathbf{H} \cdot \mathbf{1} = \mathbf{1}$ and $\mathbf{H}' = \mathbf{1} + \mathbf{H} \cdot \mathbf{P}_1^T$.

In both cases, the public code has a parity-check matrix that is simply a permuted version of that of the secret code (or its complementary). This reduces the security to that of the original McEliece cryptosystem, that discloses a permuted version of the secret code. Such a security level is not sufficient when adopting, for example, LDPC codes, since the permuted version of the secret matrix $\mathbf{H}$ can be attacked by searching for low weight codewords in the dual of the secret code.

A more general formulation of the flaw follows from the consideration that $\mathbf{Q}_1 = \mathbf{1} + \mathbf{P}_1$ has a very special inverse. First of all, let us consider that $\mathbf{Q}_1$ is invertible only when it has even size. This is obvious since, for odd size, $\mathbf{Q}_1$ has even row/column weight; so, the sum of all its rows is the zero vector. If we restrict ourselves to even size $\mathbf{Q}_1$ matrices, it is easy to show that their inverse has the form $\mathbf{Q}_1^{-1} = \mathbf{1} + \mathbf{P}_1^T$, due to the property of permutation matrices (as orthogonal matrices) to have their inverse coincident with the transpose.

So, $\mathbf{Q}_1^{-1}$ has the same form of $\mathbf{Q}_1$ and, as in the case of $\mathbf{H}$, disclosing $\mathbf{G}' = \mathbf{S}^{-1}\mathbf{G}\mathbf{Q}_1^{-1}$ might imply disclosing a generator matrix of a permuted version of the secret code or its complementary (depending on the parity of its row weight). Therefore, the form $\mathbf{Q}_1 = \mathbf{1} + \mathbf{P}_1$ might reduce the security to that of the permutation used in the original McEliece cryptosystem.

Based on these considerations, one could think that adopting a vector $\mathbf{a}$ different from the all-one vector could avoid the flaw. However, by considering again that $\mathbf{Q}_1^{-1} = \widehat{\mathbf{Q}}/\left|\mathbf{Q}\right|$, it is easy to verify that a weight-1 row in $\mathbf{Q}_1$ produces a weight-1 row in $\mathbf{Q}_1^{-1}$ and a weight-$(n-1)$ row in $\mathbf{Q}_1$ produces a weight-$(n-1)$ row in $\mathbf{Q}_1^{-1}$. It follows that $\mathbf{Q}_1^{-1}$ contains couples of columns having Hamming distance 2. Since their sum is a weight-2 vector, the sum of the corresponding columns of the public matrix results in the sum of two columns of $\mathbf{S}^{-1}\mathbf{G}$. Starting from this fact, an attacker could try to solve a system of linear equations with the aim of obtaining a permutation-equivalent representation of the secret code, at least for the existing distance-2 column pairs.

So, our conclusion concerning the binary case is that the choice of $\mathbf{Q}$ as in (5.22) and (5.23) should be avoided. A safer $\mathbf{Q}$ is obtained by considering $z > 1$ and more than one permutation matrix (i.e. $m > 1$). This obviously has the drawback of requiring codes with increased error correction capability.

5.4 Comparison with other variants of the McEliece cryptosystem

The main difference between the proposed cryptosystem and many other variants of the McEliece cryptosystem consists in the way the secret generator matrix is disguised into the public one, that is, by using a more general transformation matrix in the place of the permutation matrix.

Other proposals for increasing key security have been made in the past, such as using a distortion matrix together with rank codes in the GPT cryptosystem [46] and exploiting the properties of subcodes in variants of the McEliece and the GPT cryptosystems [13]. Unfortunately, cryptanalysis has shown that such approaches exhibit security flaws [87], [124].

The idea of using a rank-1 matrix with the structure (5.23) can be found in [45]. However, such a matrix was added to the secret matrix (rather than multiplied by it) and no selection of the error vectors was performed, so that a completely different solution was implemented.

Instead, the idea of replacing the permutation in the McEliece cryptosystem with a more general transformation matrix is already present in the variant of the GPT cryptosystem adopting a column scrambler [86], [97] and in cryptosystems based on full decoding [64, sec. 8.3]. These proposals are shortly examined next.

5.4.1 Comparison with the modified GPT cryptosystem

The original GPT cryptosystem has been the object of Gibson's attack. To counter such an attack, in [86] a variant including a column scrambler in place of the permutation matrix has been proposed.

Apart from the code extension and the inclusion of an additive distortion matrix, in the modified GPT cryptosystem the public generator matrix is obtained through right-multiplication by a non-singular matrix that is not necessarily a permutation matrix. So, in principle, it is the same idea of using a more general transformation matrix as in the proposed cryptosystem. However, in order to preserve the ability to correct the intentional error vectors, the GPT cryptosystem works in the rank metric domain and adopts rank distance codes, as Gabidulin codes.

Unfortunately, the properties of Gabidulin codes make it possible to exploit the effect of the Frobenius automorphism on the public generator matrix in order to mount a polynomial-time attack [87]. Recently, it has been shown that this attack can be avoided [97], but the cryptosystem still needs to work with rank distance codes. Differently from the GPT cryptosystem, the proposed solution is able to exploit Hamming distance codes, that are more widespread than rank distance codes, can be chosen to have convenient properties or structure, like RS codes, and may take advantage of many efficient codec implementations that are already available.

5.4.2 Comparison with full-decoding cryptosystems

The main idea behind full-decoding cryptosystems in [64] is to let the intentional error vectors have any arbitrary weight. This way, an attacker would be forced to try full-decoding of the public code, that is known to be a NP-complete task. Obviously, the legitimate receiver must be able to decode any intentional error vector with reasonable complexity; so, the problem of full decoding must be transformed from a one-way function to a trapdoor function. For this purpose, the main idea is to use a transformation that maps a set of error vectors with weight $\leq t$ into a set of arbitrary weight intentional error vectors.

If this transformation is represented by the $n \times n$ matrix $\mathbf{M}$, the public code (as proposed first in [64]) would be $\mathbf{G}' = \mathbf{G} \cdot \mathbf{M}$. The basic point for obtaining a trapdoor function is to make Alice use only those error vectors that can be expressed as $\mathbf{e}' = \mathbf{e} \cdot \mathbf{M}$, where $\mathbf{e}$ is a weight-t error vector. This way, when Bob uses the inverse of the secret matrix $\mathbf{M}$ to invert the transformation, he re-maps each arbitrary weight error vector into a correctable error vector. Unauthorized users would instead be forced to try full-decoding over arbitrary weight error vectors; so, the trapdoor is obtained.

The set of intentional error vectors used in full-decoding cryptosystems is not the set (or a subset) of the correctable error vectors, as in the proposed cryptosystem, but a transformed version of it. In fact, the purpose of full-decoding cryptosystems is to increase the security level with respect to the McEliece cryptosystem by relying on a problem that is harder to solve. In order to exploit the full-decoding problem, Alice must use for encryption only those error vectors that can be anti-transformed into correctable error vectors. So, some information on the transformation used to originate them must be disclosed. A solution is that the first $p < n$ rows of $\mathbf{M}$ are made public [64]. However, it has been proved that, this way, the security reduces to that of the original McEliece cryptosystem, and an attacker does not have to attempt full-decoding, but only normal decoding.

Further variants aim at better hiding the secret transformation matrix in its disclosed version [64]. In the last variant, a generator matrix of a maximum distance-t anticode is used to hide the

secret transformation. This way, after inverting the secret transformation, the error vector remains correctable for the legitimate receiver. To our knowledge, the latter version has never been proved to be insecure nor to reduce to the same problem of the original McEliece cryptosystem. However, the construction based on anticodes seems unpractical.

Differently from full-decoding cryptosystems, our proposal still relies on the same problem as the original McEliece cryptosystem (that is, normal decoding); so, no transformation is performed over the correctable random error vectors, but we need, at most, only a selection of them. For this reason, the information leakage on the secret transformation matrix that is needed in the proposed cryptosystem is considerably smaller with respect to what happens in full-decoding cryptosystems.

5.5 Attacks against the proposed cryptosystem

A first concern about the proposed cryptosystem is to verify that it is actually able to provide increased key security, with respect to previous variants of the McEliece cryptosystem, in such a way as to allow the use of widespread families of codes (as RS and Generalized RS codes) without incurring in the attacks that have prevented their use up to now.

From the comparison with the variants described in Sections 5.4.1 and 5.4.2, we infer that previous attacks targeted to those cryptosystems do not succeed against the proposed one, due to the differences in the family of codes used and in the information leakage on the secret transformation. Concerning the latter point, we observe that, even if the whole matrix $\mathbf{R}$ (and not only the vector $\mathbf{a}$) was public, an attacker would not gain much information. In fact, in this case, he could compute $\mathbf{x} \cdot \mathbf{R} = \mathbf{u} \cdot \mathbf{G}' \cdot \mathbf{R}$. However, for the choices of the parameters we consider, $\mathbf{R}$ has rank $\ll n$, so $\mathbf{G}' \cdot \mathbf{R}$ is not invertible. Moreover, multiplication by $\mathbf{G}' \cdot \mathbf{R}$ only provides a small dimension syndrome of $\mathbf{u}$, whose decoding is known to be a hard problem [15].

The most powerful attack procedures against our proposed solution are those techniques that attempt information set decoding (ISD) on the public code; so we estimate the security level of the proposed cryptosystem against this kind of attacks. Actually, there is no guarantee that the public code, defined through the generator matrix (5.6) or, equivalently, the parity-check matrix (5.9), maintains the same minimum distance and error correction capability of the secret code. Since the private code is already a very good code, and the transformation matrix is randomly chosen, the public code will most probably be worse than the private one. So, in estimating the security level as the work factor of ISD attacks, we make the pessimistic assumption that the public code is still able to correct all intentional errors.

5.5.1 ISD attacks

In [17] the authors have proposed some smart speedup techniques to reduce the Stern algorithm work factor (WF) over the binary field, this way obtaining a theoretical WF close to 2^{60}. Their attack was implemented on a big cluster of computers that was able to break the McEliece cryptosystem with original parameters ($n = 1024$, $k = 524$, $t = 50$). As a consequence, the authors have proposed some new set of system parameters in order to increase the security level. The information set decoding attack is not polynomial in the code dimension, since it aims at decoding a random linear code without exploiting any structural property (even if present) and this task is notoriously non-polynomial. One of the biggest improvements presented in [17] is a smart way to

find k independent columns in the public generator matrix at each iteration without performing Gaussian reduction on all such columns. A further improvement consists in the pre-computation of the sum of some rows during the reduction.

In [93], Peters points out that these speedups are efficient on very small fields. As it results from the table available in [92], for $q > 16$ the maximum values of the speedup parameters are $c = 2$, $r = 1$, where c represents the number of columns to be changed in the case an iteration fails and r is the number of rows in a single pre-sum (1 means no speedup). So, for large fields, these speedups are not relevant and the algorithm is quite similar to Stern's one. The difference relies on guessing not only p error positions but also p error values in the k independent columns, due to the field cardinality. Finiasz and Sendrier have proposed a further improvement that could yield a slight modification in the WF, resulting in a maximum increase of 2^6 or a maximum decrease close to 2^3.

In Table 5.1 we report some values of the WF when using RS codes in the variant of the McEliece cryptosystem we propose. They were computed through the PARI/GP script available in [92], that allows the estimation of the security level, although it is not extremely accurate (it can be about 4-8 times higher than the actual value). The reported WF values are the lowest ones obtained for each set of parameters. We observe from Table 5.1 that, in order to reach a satisfactory level of security (that is, $\mathrm{WF} \geq 2^{80}$) we need to adopt RS codes defined on $\mathbb{F}_{256}$ or more. Based on Table 5.1, we can compare the proposed cryptosystem with the instances of the McEliece system presented in [17].

Example 1

To reach $\mathrm{WF} > 2^{80}$, the (1632, 1269) Goppa code is suggested, resulting in a public-key size of 460647 bits (obtained by storing the non-systematic part of $\mathbf{H}$, as in the Niederreiter cryptosystem).

With the new variant, we can consider the RS code with $n = 255$, $k = 195$, $t = 30$, having an estimated $\mathrm{WF} \approx 2^{86.06}$ and an actual $\mathrm{WF} \approx 2^{84.18}$ (found through the C program available in [92]). We can consider the Niederreiter version of the first implementation (see Section 5.3.2), and use $\mathbf{H}''$, defined by (5.11), having elements over $\mathbb{F}_{256}$, as the public key.

This way, we need to store only the last k columns of $\mathbf{H}''$, so obtaining a public key size of 93600 bits, that is about 80% less than in the revised McEliece cryptosystem [17]. If we instead adopt the second implementation (see Section 5.3.3), we also need to store the 1×255 vector $\mathbf{a}$, with elements over $\mathbb{F}_{256}$. This would increase the public key size by 2040 bits, that is not a significant change.

The security level of the two systems remains comparable when the constraint expressed by $\mathbf{a}$ is imposed on the intentional error vectors of the modified cryptosystem. In fact, as it will be shown in the next subsection, the introduction of each constraint results in a decreased WF for the ISD attack of 2^3 at most.

Example 2

As another example, we can consider the Goppa code suggested in [17] to achieve $\mathrm{WF} \geq 2^{128}$, which has $n = 2960$, $k = 2288$, yielding a key length of 1537536 bits.

An RS code with the same rate (0.77), defined over $\mathbb{F}_{512}$, is reported in Table 5.1 and has $n = 511$, $k = 395$. By considering this code in the Niederreiter version of the first implementation

(see Section 5.3.2), and storing the last k columns of $\mathbf{H}''$, defined by (5.11), we obtain a public key size of 412380 bits, that is about 73% less than in the revised McEliece cryptosystem.

Moreover, by using the new system, the security level grows up to $2^{158.67}$ (more precisely, it is estimated as $2^{155.89}$ with the C program from [92]). This value remains very high even when we consider the presence of the constraint expressed by $\mathbf{a}$ on the intentional error vectors.

Table 5.1: Work factor ($\log_2$) of ISD attacks on RS codes.

RS codes with $n = 127$ defined over $\mathbb{F}_{128}$															
Rate	0.75	0.73	0.72	0.70	0.69	0.67	0.65	0.64	0.62	0.61	0.59	0.57	0.56	0.54	0.53
t	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
WF	49.2	50.1	51.0	51.7	52.3	52.8	53.3	53.7	54.0	54.2	54.3	54.4	54.4	54.4	54.2
RS codes with $n = 255$ defined over $\mathbb{F}_{256}$															
Rate	0.81	0.80	0.78	0.76	0.75	0.73	0.72	0.70	0.69	0.67	0.65	0.64	0.62	0.61	0.59
t	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52
WF	79.0	81.6	83.9	86.1	87.9	89.6	91.1	92.4	93.5	94.4	95.2	95.8	96.2	96.5	96.7
RS codes with $n = 511$ defined over $\mathbb{F}_{512}$															
Rate	0.94	0.93	0.91	0.90	0.89	0.88	0.87	0.86	0.84	0.83	0.82	0.81	0.80	0.78	0.77
t	16	19	22	25	28	31	34	37	40	43	46	49	52	55	58
WF	81.3	90.1	98.1	105.6	112.4	118.8	124.7	130.2	135.3	140.0	144.3	148.4	152.1	155.5	158.7

5.5.2 Exploiting the knowledge on error vectors

It is important to assess whether the constraints that may be imposed on the intentional error vectors in the proposed cryptosystem have any consequences on its security.

For this purpose, a conservative approach consists in considering, in the WF computations, a reduced number of intentional errors, that is, $t' = t - z$. This approach is conservative in the sense that we assume that the attacker exactly knows both the position and the value of z errors, while he actually knows only their values. We can estimate the WF of an ISD attack in this scenario by using the same procedure as in Section 5.5.1. This has been done in Table 5.2. As we can observe from the values obtained (and their comparison with those reported in Table 5.1, corresponding to $z = 0$), we have a WF decrease close to 2^3 when z is increased by 1. So, the security level for the considered parameters does not vary significantly, on condition that the value of z is kept small.

Table 5.2: Work factor ($\log_2$) of ISD attacks on RS codes with $n = 255$, defined over $\mathbb{F}_{256}$, when $z = 1$ or $z = 2$ constraints are imposed on the error vectors.

Rate	0.81	0.80	0.78	0.76	0.75	0.73	0.72	0.70	0.69	0.67	0.65	0.64	0.62	0.61	0.59
t	24	26	28	30	32	34	36	38	40	42	44	46	48	50	52
WF ($z = 1$)	75.9	78.6	81.1	83.3	85.3	87.0	88.6	90.0	91.2	92.2	93.0	93.7	94.2	94.6	94.8
WF ($z = 2$)	72.8	75.6	78.2	80.5	82.6	84.5	86.1	87.6	88.9	89.9	90.9	91.6	92.2	92.6	92.9

5.6 Conclusion

We have introduced a variant of the McEliece cryptosystem that, by replacing the secret permutation matrix with a more general transformation matrix, is able to avoid that the public code is

permutation-equivalent to the secret code. This allows to prevent attacks against classical families of codes, as RS codes, and to reconsider them as possible good candidates in this framework.

We have proposed some practical implementations of the new cryptosystem, by considering both its McEliece and Niederreiter variants, and we have addressed some important issues that may influence their design.

We have also assessed the security level of the proposed cryptosystem by considering up-to-date attack procedures, and we have compared it with the classical McEliece cryptosystem and the Niederreiter variant. Our results show that the proposed solution, by exploiting RS codes, is able to guarantee an increased security level and, at the same time, a considerable reduction in the public key size.

Chapter 6

Coding solutions for a secure storage of biometric data

This chapter deals with the problem of securely storing biometric passwords, such as fingerprints and irises. The proposed solutions [8, 107] involve the use of error-correcting codes whose length can be very large, so that efficient decoding algorithms are required. We will discuss the use of both classic algebraic codes and iteratively decodable codes.

6.1 Introduction

The use of biometric passwords, such as fingerprints, irises, etc., has been an important issue in recent years, both because of the big advantages it may bring along and because of the clearly non negligible privacy concerns and implementation issues [117]. In fact, as far as privacy is concerned, the storage of raw biometric data is not an acceptable solution, but, on the other hand, a secure storage cannot be easily implemented, as for traditional passwords, by simply introducing an hash function. This is due to the fact that the binary strings derived from different acquisitions of the same biometric feature can slightly change from each other, and the biometric feature can slightly change itself. Therefore a certain threshold of tolerance is needed to be able to identify legitimate from non legitimate users, but this prevents the standard use of collision resistant hash functions [107].

This problem has prompted researchers to devise other solutions for the secure storage and use of biometric passwords (see [116] for a selected survey of the literature). The idea behind most of these methods is a combined use of error correcting codes and hash functions, whose model is the fuzzy commitment scheme [61], which we revisit below. This has been later generalized to other types of metrics, such as the set difference metric [62] and the edit distance metric [36].

In particular, the fuzzy vault [62] uses polynomial interpolation in order to allow authentication based on the matching of a sufficient number of features, while the fuzzy extractor [36] is a further generalization which combines the previous constructions with particular objects called random extractors. These make the previous schemes stronger with respect to information leakage, though they cannot prevent it [24, 37].

Briefly, privacy and implementation issues are still a concern and our aim is to give a further contribution concerning these issues. In particular we will show that a syndrome based fuzzy

hashing construction can be more convenient as far as information leakage is concerned. We will also address the design of codes and other implementation issues.

The chapter is structured as follows: in Section 6.2 we briefly review the fuzzy commitment scheme and its main issues. Section 6.3 is devoted to the syndrome based construction which we denote by fuzzy hashing.

6.2 The fuzzy commitment scheme

The fuzzy commitment scheme, proposed in [61], works as follows. Suppose we want to securely store a length n biometric vector $x \in \mathbb{F}_q^n$, where $\mathbb{F}_q$ is the Galois field of order q, and let e be the maximum number of different symbols with respect to the reference vector x that we can tolerate in any other acquisition of the same biometric feature.

According to the fuzzy commitment scheme, we choose a hash function H_a and an $[n, k]$-linear block code $C \subset \mathbb{F}_q^n$, able to correct e errors, and then store $(H_a(r_x), l)$, where r_x is a random codeword associated to x and $l = x - r_x$.

Given another biometric y, we compute the vector $z = y - l$ and apply the decoding algorithm of C. If decoding succeeds, this results in a codeword $c_z \in C$, and we compute $H_a(c_z)$. If $H_a(c_z)$ equals $H_a(r_x)$, i.e., the value previously stored, authentication succeeds, otherwise it fails.

In fact, if the hashes are the same, then $c_z = r_x$ (apart from a negligible probability of a hash collision), so $d(c_z, z) = d(r_x, z) \leq e$, where $d(\cdot)$ denotes the Hamming distance. Since $r_x = x - l$ and $z = y - l$, it results $d(r_x, z) = d(x, y) \leq e$.

Conversely, $d(x, y) \leq e$ implies $d(x - l, y - l) = d(r_x, z) \leq e$, so that decoding z results in $c_z = r_x$ and $H_a(c_z) = H_a(r_x)$.

Below we revisit the main problems concerning the use of this scheme, namely implementation issues and security issues [107].

6.2.1 Choice of the code

As a first instance, in order to prevent bruce force attacks, we require for the size of the code $|C| \geq 2^{80}$. This is due to the fact that it is generally accepted that a total search space of 2^{80} is beyond the capabilities of modern computers. As a result it is desirable that codes constructed over the binary field have dimension $k = \dim \mathcal{C} \geq 80$, possibly a bit smaller if one works over larger alphabets. In addition one wants to have a large relative minimum distance (because of the big error tolerance required) that only low rate codes can afford. Indeed because e.g. of the asymptotic Elias upper bound (see e.g. [19]) only very low rate binary codes can have relative distance larger than e.g. 0.4. Of course the code should come with efficient decoding algorithms even when the block length is about $n = 10^4$ or $n = 10^5$, depending on the type of biometric in use.

In [61] it is proposed that Reed-Solomon and BCH codes might provide useful results (see also [55]). For large code lengths, this might be a good option only if low-complexity decoding algorithms are used. Otherwise other types of codes have to be introduced, for example product codes or LDPC codes.

6.2.2 Distribution of biometric templates

Privacy concerns may arise if the biometric templates are not uniformly distributed in the ambient space, that is their entropy is not maximal. In that case, it may be feasible to infer from l some information about r_x and, therefore, endanger the system security. Depending on the size of $\mathcal{C}$ it might be possible to do a search among all codewords with a particular pattern and consequently break the system. A possible countermeasure would be to take a higher rate code, but at the expense of lowering the minimum distance.

6.2.3 Practical Implementation Issues

There are other practical difficulties that make the problem more complicated than how we stated it, for example the fact that the scheme requires that the two passwords to be compared are aligned, but one of the two passwords is not in the clear. In the literature [21, 22, 36, 44, 62, 65, 107, 118, 117, 128] we can find a deeper discussion of all these side problems together with proposals to attack some of them.

6.3 Syndrome fuzzy hashing

Starting from the fuzzy commitment principle, an alternative scheme (here fuzzy hashing) can be devised, in which syndromes are used in the place of codewords. Under a coding theory viewpoint, the two schemes are equivalent. Despite this, the use of syndromes has several advantages in the considered context. The idea of storing the syndrome of x, instead of a shift vector from a codeword, already appeared in [36], where it is considered as an example of a sketch construction. We will show that the use of fuzzy hashing is advantageous with respect to the classical fuzzy commitment scheme, also by considering the characteristics of typical biometric data.

In the fuzzy hashing scheme, an $[n, k]$-linear block code $C \subset \mathbb{F}_q^n$, able to correct e errors, is selected, and it is described through its $r \times n$ parity-check matrix H, with $r = n - k$. Given a biometric vector x to be stored, the pair $(H_a(x), Hx)$ is used to represent x, were H_a is a given hash function. When another biometric y is acquired and is compared with x, the value $Hx - Hy = H(x - y) = Hv$ is computed, that coincides with the syndrome associated to the difference vector $v = x - y$. Then, syndrome decoding is applied on Hv, according to the chosen code C. If y is taken from the same individual as x, then v has Hamming weight equal to $d(x, y) \leq e$ and it corresponds to a correctable error vector. So, syndrome decoding succeeds and correctly results in v. Then, starting from v and y, x can be computed, as well as $H_a(x)$. The latter coincides with the stored value, so authentication succeeds. Otherwise, syndrome decoding fails or reports $w \neq v$. In such case, $x' = w + y \neq x$ and $H_a(x') \neq H_a(x)$ is obtained, so authentication fails.

In the fuzzy commitment, the vector $l = x - r_x$ is stored. As some bits of the biometric x might be known with high probability, this reveals some information on the secret codeword r_x. The same may occur in fuzzy hashing, where the syndrome Hx is stored, but only under the condition $x = q + r_q$, where q is a correctable error vector and r_q is any codeword. In this case, syndrome decoding results in q; so, some bits of r_q can still be guessed, starting from the predictable bits of x. However, especially for very low rate codes, the probability that x is within the decoding radius of a codeword r_q is very low, so fuzzy hashing provides better security with respect to the classical fuzzy commitment.

6.3.1 Codes for fuzzy hashing

In order to design suitable codes to be included in the fuzzy hashing scheme, we must consider the features of the biometric vectors we work with. If we refer to fingerprints or irises, a common acquisition will consist of a vector of several thousands of bits. However, it would be unpractical to apply fuzzy hashing directly on the plain acquisition, since a number of impairments could jeopardize the identification process. In fact, small changes in the acquisition conditions (as ambient light or small movements of the subject) could result in significant differences between two images of the same biometric feature. So, a common procedure is to extract a set of representative features from the biometric data through algorithms aimed at making them invariant to some frequent acquisition impairments. An example of this kind of algorithms will be considered in Section 6.3.3. So, the code must be designed to work with the vectors produced as output by the feature extracting algorithm. Typically, such vectors have length of the order of 10k bits.

Another important aspect is the modeling of the errors affecting two vectors resulting from different biometric acquisitions from the same individual. In [115], the errors are modeled through a Binary Symmetric Channel (BSC) with transition probability p. We will adopt the same approach in this paper. As it will be shown in the next section, typical values of the percentage of different bits between the vectors representing two acquisitions of the same biometric range between 10% and 30%. So, we need codes with length about 10k bits that are able to correct such high fractions of errors and, hence, have very low rate (R). Just to give an idea, a BCH code with ($n = 2047$, $k = 100$), that is, rate $R \approx 0.05$, is able to correct 379 errors, which means it has a relative error correcting capability of about 19%. A BCH code with ($n = 4095$, $k = 110$), that is, rate $R \approx 0.03$, is able to correct 767 errors, that is almost the same percentage. A similar value is reached by the BCH code having ($n = 8191$, $k = 170$), hence rate $R \approx 0.02$, able to correct 1533 errors.

This evidences that, for classical algebraic codes, in order to maintain a given relative error correcting capability, the code rate must be decreased as the code length increases. Furthermore, due to the long code length, decoding may also yield complexity issues, although recent algorithms can reduce the decoding complexity, as we have seen in the previous chapters. So we have chosen to test the system with modern iteratively decoded error correcting codes, like Low-Density Parity-Check (LDPC) codes [100]. Actually, the use of LDPC codes in this context has already been proposed in [23, 107, 115], but the code design was not addressed in those works. In summary, fuzzy hashing with LDPC codes brings the following advantages: i) it reduces the amount of stored data, with respect to the fuzzy commitment, since $r < n$; ii) it reduces the predictability of the stored strings, as shown at the end of the previous section; iii) LDPC codes have greater error correction capabilities than classical algebraic codes; moreover, their relative error correcting capability, for a fixed rate, is almost constant as the code length increases; iv) LDPC codes allow to reduce the size of the code representation, by exploiting the sparse nature of H.

6.3.2 Code Design

We are interested in almost regular codes, since they allow an easier implementation with respect to irregular codes; so, we fix the column weight of the matrix H to be equal to an integer d_v. The row weight, for the code rate values here of interest, cannot be constant as well. However, it will be minimally dispersed around its mean $\langle d_c \rangle = d_v/(1 - R)$. If we suppose that (as it occurs for all

the codes we consider):

$$\frac{k}{n} = R < \frac{1}{d_v + 1}, \tag{6.1}$$

the matrix H can have rows with only the following two values of weight: d_v and $d_v + 1$. In this case, $r - k \cdot d_v$ rows have weight d_v and the other $k \cdot d_v$ rows have weight $d_v + 1$.

We can describe the column and row weight distributions of the matrix H through the polynomials $\lambda(x)$ and $\rho(x)$ representing, respectively, the variable node and check node degree distributions of the associated Tanner graph [100]. Since we adopt the edge perspective, λ_i (ρ_i) denotes the fraction of ones in the parity-check matrix H which are in columns (rows) of weight i. Based on the hypotheses above, for the considered ensemble of codes it results:

$$\begin{aligned} \lambda(x) &= x^{d_v - 1}, \\ \rho(x) &= [1 - R(1 + d_v)]x^{d_v - 1} + R(1 + d_v)x^{d_v}. \end{aligned} \tag{6.2}$$

Starting from (6.2), we can estimate the asymptotic performance (that is, for $n \to \infty$) of LDPC codes in this ensemble by applying the density evolution method [100].

Gallager's A algorithm [47] is an LDPC decoding algorithm for the BSC channel that permits an easy characterization through density evolution [9]. So, we have estimated its convergence threshold (that is the maximum channel error probability such that all the errors can be corrected using an infinite length code) for the variable and check node degree polynomials given by (6.2). Results are reported in Table 6.1, where the threshold values computed for $d_v = 3, 4, 5$ are provided, for code rates ranging between 0.1 and 0.01.

Table 6.1: Threshold values for the considered LDPC codes ensembles under Gallager's A decoding.

R	$d_v = 3$	$d_v = 4$	$d_v = 5$
0.1	0.159	0.078	0.045
0.09	0.163	0.079	0.045
0.08	0.166	0.08	0.046
0.07	0.169	0.081	0.046
0.06	0.173	0.083	0.047
0.05	0.177	0.084	0.048
0.04	0.18	0.085	0.048
0.03	0.184	0.087	0.049
0.02	0.188	0.088	0.05
0.01	0.192	0.089	0.051

As we observe from the table, the choice of a small value of d_v (like 3) should be preferred. On the other hand, the asymptotic performance under Gallager's A decoding is not very good. For example, to reach a relative error correcting capability of 19%, for $d_v = 3$, a code rate smaller than 0.02 is required, that is similar to that needed by a BCH code with $n = 8191$.

Gallager's A algorithm allows an easy density evolution analysis, that is useful to verify that LDPC codes can asymptotically reach the error correcting performance we need and for which values of code rate. On the other hand, when dealing with finite length codes, decoding algorithms with better performance can be used. In fact, Gallager's A algorithm is a majority-based algorithm exploiting a fixed decision threshold b, that is not the most effective choice. For example, adopting a variable b (as in Gallager's B algorithm) gives a first performance improvement. Furthermore, several improved versions of these algorithms have been proposed in the literature

[76], that are able to outperform Gallager's original algorithms. Finally, the classical Sum Product Algorithm (SPA) [54] can also be applied on the BSC, even though, in absence of soft-information from the channel, the initial likelihood associated to each bit can assume only two opposite values. Despite this, the SPA is able to significantly improve the error correction performance with respect to that predicted in Table 6.1, as we will show in the next section, by providing some examples of practical codes.

Examples

In this subsection we provide examples of LDPC codes having parameters of interest in the fuzzy hashing context. The codes have been designed through the Progressive Edge Growth (PEG) algorithm [58], by imposing almost constant column and row weights for their parity-check matrices. For given column and row weights, the PEG algorithm allows to design finite length LDPC codes with very good performance under belief propagation decoding.

In detail, we first fix the column weight d_v. Then, we impose the lower triangular form for the parity-check matrices, in such a way as to facilitate encoding, especially for very long codes. This introduces a last column having weight 1, and some columns having weight $< d_v$. However, their incidence with respect to the total number of columns is very small. Then, the PEG algorithm is used to optimize the length of the local cycles within the Tanner graph associated to each code, while keeping the row weight distribution as much concentrated as possible.

So, the characteristics of the codes we have designed are well overlaid with those fixed in the previous subsection. The codes mentioned above have been used to perform Montecarlo simulations over the BSC, based on SPA decoding.

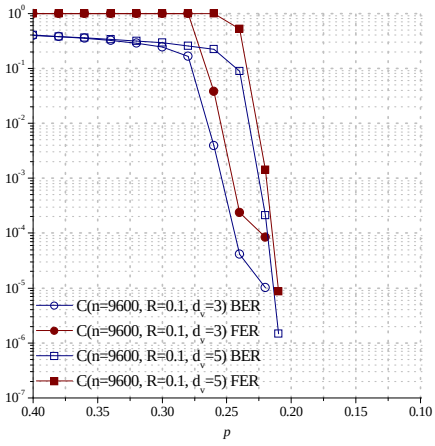


Figure 6.1: Performance of rate 0.1 LDPC codes with $d_v = 3$ and $d_v = 5$ over the BSC with SPA decoding.

A first set of results is reported in Fig. 6.1, where LDPC codes having $n = 9600$ and $k = 1000$ (hence, rate ≈ 0.1) have been considered. We have designed two codes with different column weights: $d_v = 3$ and $d_v = 5$. As we observe from the figure, the simulation confirms that the code with $d_v = 3$ has better performance, in the waterfall region, with respect to the code having $d_v = 5$. This was expected on the basis of the results of density evolution. However, we also observe that

the code with $d_v = 5$ has a better performance in the error floor region, so its Bit Error Rate (BER) and Frame Error Rate (FER) curves tend to intersect with those of the first code. So, the choice of $d_v = 3$ is suitable if a failure rate on the order of 10^{-4} or more is acceptable; otherwise, the choice of $d_v = 5$ should be preferred.

The performance improvement due to the SPA is evident: both codes are able to achieve a rather low error rate for a percentage of bit errors around 20%, or even more.

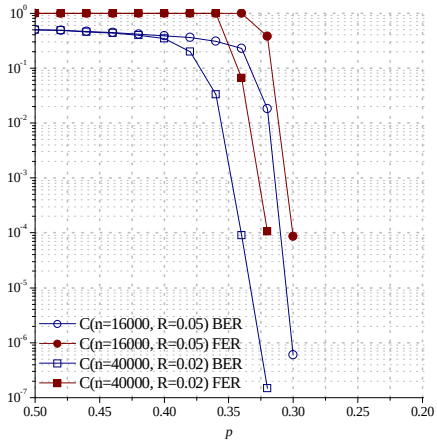


Figure 6.2: Performance of LDPC codes with $d_v = 3$ and rate 0.05 and 0.02 over the BSC with SPA decoding.

To further increase the error correcting capability of these codes, it is necessary to reduce their rate. To provide some examples in this sense, we have considered $k = 800$ and designed two other LDPC codes, having $d_v = 3$ and rate 0.05 and 0.02 (that is, $n = 16000$ and $n = 40000$), respectively.

As we observe from their simulated performance, reported in Fig. 6.2, by using the SPA, these codes are able to reach very low error rates for a percentage of bit errors around 30% and even more. Also in this case, the performance improvement due to the SPA with respect to the theoretical performance referred to Gallager's A algorithm is evident.

These results confirm that LDPC codes are well suited for the application in the considered context, in which a high correction capability is needed. Furthermore, we can observe that, in this study, we have limited ourselves to consider almost regular codes, in order to keep their implementation complexity low. The adoption of irregular LDPC codes can result in a further performance improvement.

6.3.3 Entropy analysis

In this section, we discuss the use of fuzzy hashing for iris recognition and we study how the adoption of syndromes affects some statistical properties of the biometric data. As a feature extractor, we use the algorithm described in [70] and available in [71], together with its associated matching algorithm. In our simulations, we refer to the iris pattern database known as CASIA V.1, provided by the Institute of Automation of the Chinese Academy of Sciences [27].

Since the bits in iris templates are mutually dependent [32], we should compute the entropy of a source with memory, but this is computationally unfeasible for the sizes we are dealing with.

So, according to [32, 30], we evaluate the discrimination entropy over both the sets of iris templates and of their fuzzy hashes. For this purpose, we first compute the distribution of the normalized Hamming distances between all the couples of patterns within the set (of images of the same iris or of images of different irises). Then, we compute the mean μ and the standard deviation σ of the normalized Hamming distance distribution. Finally, the discrimination entropy (also known as "Degrees of Freedom" or DOF) is obtained as $\mathrm{DOF} = \mu(1-\mu)/\sigma^2$.

Applying fuzzy hashing to an iris recognition framework is not straightforward, due to the high variability in the iris acquisition phase. In fact, we must try to avoid all the differences given not only by the measure variability (i.e., scale and rotation), but also by the eye variability, that can significantly change the amount of visible iris and its shape.

The standard way to take these issues into account is to compute a mask describing which bits in the iris template are free from such occlusions. The masks, in general, have a different number of set bits for each iris reading, resulting into information patterns having different lengths, both in the case they describe different irises and different readings of the same iris. This is not a problem in the case of the standard matching algorithms, since we can take, as inputs for the matching phase, the templates and masks of both the stored iris and the one we want to check. Then, we can just compute the union of the two masks and obtain the number of different bits between the two templates, limited to the region excluded from the union of the masks.

Instead, when we use syndromes, we cannot access the reference template in clear; so, we must cope with different lengths of the information patterns. One way is to treat the matching channel as an error-and-erasure channel [23], where erasures are given by the masks. However, in [23] the authors use a different algorithm, while, in our case, the large number of bits erased by the masks makes this approach unusable. In order to obtain a fixed length of the information patterns, we compute, for each template bit position i, the probability $m(i)$ that such bit is not erased by a mask. Then, we compute a pseudomask selecting the bit positions corresponding to a value of $m(i)$ lower than a threshold: $m(i) \leq m_{th}$. In our case, we fix $m_{th} = 2.4\%$.

We are aware that, with this approach, we may neglect some bits that were not erased by their associated masks, but we have verified that this has a very limited effect for the considered algorithm. In fact, using all the selected bits in each template, we obtain, between two different readings of the same iris, an average Hamming distance of 28.24%, a standard deviation $\sigma = 0.0435$ and a discrimination entropy equal to 107 bits. Instead, using only the bits selected by the pseudomask, we obtain an average Hamming distance of 26.2%, a standard deviation $\sigma = 0.0486$ and a discrimination entropy equal to 81 bit. The explanation for this moderate variation, in terms of discrimination entropy performance, is that the feature extraction algorithm does not compute the masks in the best possible way (for example the two eyelids are approximated with straight lines and not with curves); so, when we consider all the bits in each template, we introduce some errors that afflict the result, that is, we take some bits into account that we should actually erase.

We model the channel as BSC both in the case of intra-class and inter-class comparisons. The use of a BSC model is further justified by the fact that, by exploiting the considered feature extraction algorithm and pseudomask, we have an experimental transition probability $p_{1\to0} = 0.257$, $p_{0\to1} = 0.268$ for intra-class comparison and $p_{1\to0} = 0.480$, $p_{0\to1} = 0.501$ for the inter-class case, thus confirming the (almost) symmetric nature of the channel.

In order to show that the use of fuzzy hashing is able to increase the discrimination entropy, that is to provide a better protection against information leakage, we estimate the DOF on the set of plain templates, before and after the application of fuzzy hashing. The latter is performed through

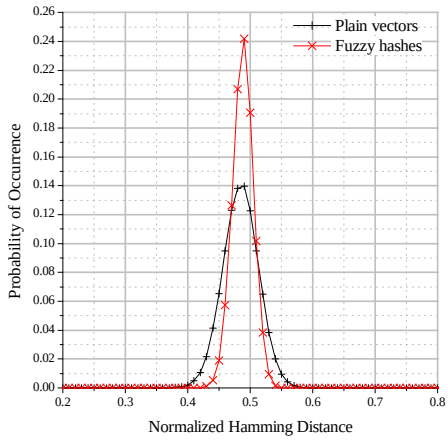


Figure 6.3: Inter-class analysis of the Hamming distance for the considered set of iris templates with and without fuzzy hashing.

the LDPC code having $n = 9600$, $R = 0.1$ and $d_v = 5$, described in the previous section. We compute the normalized Hamming distance between each pair of templates created from different irises and then estimate its probability density function. The results are reported in Fig. 6.3.

The set of plain template vectors has $\mu = 0.4897$, $\sigma = 0.0281$, hence $\mathrm{DOF} = 316.5$. After performing fuzzy hashing, the values become $\mu = 0.4932$, $\sigma = 0.0166$, $\mathrm{DOF} = 907.1$, thus confirming the positive effect of fuzzy hashing.

Chapter 7

Appendix I: Gauss Sums of the Cubic Character

We show here an elementary approach to derive the values of the Gauss sums of a cubic character over a finite field. We first deal with finite fields of characteristic 2 [102], then with fields with odd characteristic [103], which is the more difficult and interesting case.

New links between Gauss sums over different field extensions will be shown in terms of factorizations of the Gauss sums themselves, which then are revisited in terms of prime ideal decompositions. Interestingly, one of these results gives a representation of primes p of the form $6k+1$ by a binary quadratic form in integers of a subfield of the cyclotomic field of the p-th roots of unity.

Moreover a new relevant closed expression (equation (7.9)) is provided.

7.1 Characteristic 2

Let $\mathbb{F}_{2^s}$ be a Galois field over $\mathbb{F}_2$, with $\mathrm{Tr}_s(x) = \sum_{j=0}^{s-1} x^{2^j}$ being the trace function over $\mathbb{F}_{2^s}$, and $\mathrm{Tr}_{s/r}(x) = \sum_{j=0}^{s/r-1} x^{s^{rj}}$ the relative trace function over $\mathbb{F}_{2^s}$ relatively to $\mathbb{F}_{2^r}$, with $r|s$ [68].
Further let χ_m be a character of order m defined over $\mathbb{F}_{2^s}$ and taking values in $\mathbb{Q}(\zeta_m)$, where ζ_m denotes a primitive m-th root of unity and $\mathbb{Q}(\zeta_m)$ the corresponding cyclotomic field.
A Gauss sum of a character χ_m over $\mathbb{F}_{2^s}$ is defined as [16]

$$G_s(\beta, \chi_m) = \sum_{y \in \mathbb{F}_{2^s}} \chi_m(y) e^{\pi i \mathrm{Tr}_s(\beta y)} = \bar{\chi}_m(\beta) G_s(1, \chi_m) \qquad \forall \beta \in \mathbb{F}_{2^s} \ .$$

A cubic character χ_3 is a mapping from $\mathbb{F}_{2^s}^*$ into the complex numbers defined as

$$\chi_3(\alpha^{h+3j}) = \ \zeta_3^h \quad h = 0, 1, 2 \ \ j \in \mathbb{N} \ ,$$

where ζ_3 is a cubic root of unity, and α a primitive element in $\mathbb{F}_{2^s}^*$, furthermore we set by definition $\chi_3(0) = 0$.

The values of the Gauss sums of a cubic character over $\mathbb{F}_{2^s}$ can be found by computing the Gauss sum over $\mathbb{F}_4$ and applying Davenport-Hasse's theorem on the lifting of characters ([16, 63, 68]) for s even (and by computing the Gauss sum over $\mathbb{F}_2$ and then trivially lifting for s odd).

However it is possible to use a more elementary approach, and this is the subject of the present work.

If s is odd then the cubic character is trivial because every element β in $\mathbb{F}_{2^s}$ is a cube as the following chain of equalities shows

$$\beta \cdot 1 = \beta \cdot (\beta^{2^s-1})^2 = \beta\beta^{2^{s+1}-2} = \beta^{2^{s+1}-1} = (\beta^{\frac{2^{s+1}-1}{3}})^3 \ ,$$

since $\beta^{2^s-1} = 1$, and $s+1$ is even, so that $2^{s+1}-1$ is divisible by 3. In this case we have

$$G_s(1,\chi_3) = \sum_{y\in\mathbb{F}_{2^s}} \chi_3(y) e^{\pi i \mathrm{Tr}_s(y)} = \sum_{y\in\mathbb{F}^*_{2^s}} e^{\pi i \mathrm{Tr}_s(y)} = -1 \ ,$$

since the number of elements with trace 1 is equal to the number of elements with trace 0, ($\mathrm{Tr}_s(x) \in \mathbb{F}_2$; moreover $\mathrm{Tr}_s(x) = 1$ and $\mathrm{Tr}_s(x) = 0$ are two equations of degree 2^{s-1}), and $e^{\pi i \cdot 0} = 1$ while $e^{\pi i \cdot 1} = -1$.

If s is even, the cubic character is nontrivial, and the computation of the Gauss sums requires some more effort; before we show how they can be computed with an elementary approach, we need some preparatory lemmas.

7.1.1 Preliminary facts

First of all we recall that, for any nontrivial character χ_m over $\mathbb{F}_q$, $\sum_{x\in\mathbb{F}_q} \chi_m(x) = 0$. This is used to prove a property of a sum of characters, already known to Kummer [126], which can be formulated in the following form:

Lemma 7.1. *Let χ_m be a nontrivial character and β any element of $\mathbb{F}_q$; then*

$$\sum_{x\in\mathbb{F}_q} \chi_m(x)\bar{\chi}_m(x+\beta) = \begin{cases} q-1 & \text{if } \beta = 0 \\ -1 & \text{if } \beta \neq 0 \end{cases} \ .$$

PROOF. If $\beta = 0$, the summand is $\chi_m(x)\bar{\chi}_m(x) = 1$, unless $x = 0$ in which case it is 0, then the conclusion is immediate.
When $\beta \neq 0$, we can exclude again the term with $x = 0$, as $\chi_m(x) = 0$, so that x is invertible, and the summand can be written as

$$\chi_m(x)\bar{\chi}_m(x+\beta) = \chi_m(x)\bar{\chi}_m(x)\bar{\chi}_m(1+\beta x^{-1}) = \bar{\chi}_m(1+\beta x^{-1}) \ .$$

With the substitution $y = 1 + \beta x^{-1}$, the summation becomes

$$\sum_{\substack{y\in\mathbb{F}_{2^{2m}} \\ y\neq 1}} \chi_m(y) = -1 + \sum_{y\in\mathbb{F}_{2^{2m}}} \chi_m(y) = -1 \ ,$$

as $\chi_m(y) = 1$ for $y = 1$.

□

We are now interested in the sum $\sum_{x\in\mathbb{F}_q}\chi_m(x)\chi_m(x+1)$. Note that for the Gauss sums over $\mathbb{F}_{2^s}$ we have

$$G_s(1,\chi_m)=\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}_s(y)=0}}\chi_m(y)-\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}_s(y)=1}}\chi_m(y)\ . \tag{7.1}$$

It follows that, if χ_m is a nontrivial character, then the Gauss sum over $\mathbb{F}_{2^s}$ satisfies the following:

$$G_s(1,\chi_m)=2\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}_s(y)=0}}\chi_m(y).$$

In fact half of the field elements have trace 0 and the other half 1, so that

$$\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}(y)=0}}\chi_m(y)=-\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}(y)=1}}\chi_m(y)$$

as the sum over all field elements is zero, since χ_m is nontrivial.

Lemma 7.2. *If χ_m is a nontrivial character over $\mathbb{F}_{2^s}$, s even, then*

$$\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x)\chi_m(x+1)=G_s(1,\chi_m)\ .$$

PROOF. The sum $\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x)\chi_m(x+1)$ can be written as $\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x(x+1))$, since the character is a multiplicative function, now the function $f(x)=x(x+1)$ is a mapping from $\mathbb{F}_{2^s}$ onto the subset of elements with trace 0, as $\mathrm{Tr}_s(x)=\mathrm{Tr}_s(x^2)$ for any s, and each image comes exactly from two elements, x and $x+1$, that have the same trace, since $\mathrm{Tr}_s(1)=0$ for s even, which is our case. Therefore, half of the elements with trace 0 are images of elements with trace 0, and the remaining half are images of elements with trace 1. It follows that

$$\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x)\chi_m(x+1)=2\sum_{\substack{y\in\mathbb{F}_{2^s}\\ \mathrm{Tr}_s(y)=0}}\chi_m(y)=G_s(1,\chi_m)\ . \tag{7.2}$$

□

Lemma 7.3. *Let χ_m be a nontrivial character of order $m=2^r+1$. Then the Gauss sum $G_s(1,\chi_m)$, s even, is a real number, i.e. $G_s(1,\chi_m)=\pm 2^{s/2}$.*

PROOF. Using (7.2) we have

$$\bar{G}_s(1,\chi_m)=\sum_{x\in\mathbb{F}_{2^s}}\bar{\chi}_m(x)\bar{\chi}_m(x+1)=\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x^{2^r})\chi_m(x^{2^r}+1)=\sum_{x\in\mathbb{F}_{2^s}}\chi_m(x)\chi_m(x+1)=G_s(1,\chi_m)\ ,$$

as $\bar{\chi}_m(x)=\chi_m(x)^{2^r}=\chi_m(x^{2^r})$ and $x\to x^{2^r}$ is a field automorphism, so it just permutes the elements of the field. □

7.1.2 Main results

The absolute value of $G_s(1,\chi_m)$ can be evaluated using elementary standard techniques going back to Gauss (see e.g. [16]), while its argument requires a more subtle analysis. Our main theorems in this section derive in an elementary way the exact value of the Gauss sum for a cubic character χ_3 over $\mathbb{F}_{2^s}$, s even (the case of s odd is trivial, as shown above). Before we proceed, we show in a standard way what is its absolute value.

Since $G_s(\beta,\chi_3)=\bar\chi_3(\beta)G_s(1,\chi_3)$, on one hand, we have

$$\begin{aligned}\sum_{\beta\in\mathbb{F}_{2^s}} G_s(\beta,\chi_3)\bar G_s(\beta,\chi_3) &= \sum_{\beta\in\mathbb{F}_{2^s}} \bar\chi_3(\beta)\chi_3(\beta)G_s(1,\chi_3)\bar G_s(1,\chi_3)\\ &= \sum_{\beta\in\mathbb{F}_{2^s}^*} G_s(1,\chi_3)\bar G_s(1,\chi_3) = (2^s-1)G_s(1,\chi_3)\bar G_s(1,\chi_3)\ .\end{aligned} \tag{7.3}$$

On the other hand, by the definition of Gauss sum, we have

$$\sum_{\beta\in\mathbb{F}_{2^s}} G_s(\beta,\chi_3)\bar G_s(\beta,\chi_3) = \sum_{\beta\in\mathbb{F}_{2^s}}\sum_{\alpha\in\mathbb{F}_{2^s}}\sum_{\gamma\in\mathbb{F}_{2^s}} \bar\chi_3(\alpha)e^{\pi i\mathrm{Tr}_s(\beta\alpha)}\chi_3(\gamma)e^{-\pi i\mathrm{Tr}_s(\gamma\beta)}\ ,$$

and substituting $\alpha=\gamma+\theta$ in the last sum, we have

$$\sum_{\beta\in\mathbb{F}_{2^s}} G_s(\beta,\chi_3)\bar G_s(\beta,\chi_3) = \sum_{\gamma\in\mathbb{F}_{2^s}}\sum_{\theta\in\mathbb{F}_{2^s}} \bar\chi_3(\gamma+\theta)\chi_3(\gamma)\sum_{\beta\in\mathbb{F}_{2^s}} e^{\pi i\mathrm{Tr}_{2s}(\beta\theta)} = 2^s(2^s-1)\ , \tag{7.4}$$

as the sum on β is 2^s if $\theta=0$ and is 0 otherwise, since the values of the trace are equally distributed, as said above; consequently the sum over γ is 2^s-1 times 2^s, as $\chi_3(0)=0$. From the comparison of (7.3) with (7.4) we get $G_s(1,\chi_3)\bar G_s(1,\chi_3)=2^s$, then $|G_s(1,\chi_3)|=2^{s/2}$.

Few initial values are $G_2(1,\chi_3)=2$, $G_4(1,\chi_3)=-4$, $G_6(1,\chi_3)=8$, $G_8(1,\chi_3)=-16$, and $G_{10}(1,\chi_3)=32$, so a reasonable guess is $G_s(1,\chi_3)=-(-2)^{s/2}$. This guess is correct as proved by the following theorems.

Theorem 7.1. *If ℓ is odd, the value of the Gauss sum $G_{2\ell}(1,\chi_3)$ is 2^ℓ.*

PROOF. Let α a primitive cubic root of unity in $\mathbb{F}_{2^{2\ell}}$, then it is a root of x^2+x+1. In other words, a root α of x^2+x+1, which does not belong to $\mathbb{F}_{2^\ell}$, as ℓ is odd, can be used to define a quadratic extension of this field, i.e. $\mathbb{F}_{2^{2\ell}}$, and the elements of this extension can be represented in the form $x+\alpha y$, with $x,y\in\mathbb{F}_{2^\ell}$. Furthermore, the two roots α and $1+\alpha$ of x^2+x+1 are either fixed or exchanged by any Frobenius automorphism; in particular the automorphism $\sigma^\ell(x)=x^{2^\ell}$ necessarily exchanges the two roots as it fixes precisely all the elements of $\mathbb{F}_{2^\ell}$, while α does not belong to this field, so that $\sigma^\ell(\alpha)\neq\alpha$. Now, a Gauss sum $G_{2\ell}(1,\chi_3)$ can be written as

$$G_{2\ell}(1,\chi_3) = 2\sum_{\substack{z\in\mathbb{F}_{2^{2\ell}}\\ \mathrm{Tr}_{2\ell}(z)=0}}\chi_3(z) = 2\sum_{\substack{x,y\in\mathbb{F}_{2^\ell}\\ \mathrm{Tr}_{2\ell}(x+\alpha y)=0}}\chi_3(x+\alpha y) = 2\sum_{\substack{x,y\in\mathbb{F}_{2^\ell}\\ \mathrm{Tr}_{\ell}(y)=0}}\chi_3(x+\alpha y)\ , \tag{7.5}$$

where we used the trace property

$$\mathrm{Tr}_{2\ell}(x+\alpha y)=\mathrm{Tr}_{2\ell}(x)+\mathrm{Tr}_{2\ell}(\alpha y)=\mathrm{Tr}_\ell(x)+\mathrm{Tr}_\ell(x^{2^\ell})+\mathrm{Tr}_{2\ell}(\alpha y)=\mathrm{Tr}_{2\ell}(\alpha y),$$

and the fact that

$$\begin{aligned}\mathrm{Tr}_{2\ell}(\alpha y) &= \mathrm{Tr}_\ell(\alpha y) + \mathrm{Tr}_\ell(\alpha y)^{2^\ell} = \mathrm{Tr}_\ell(\alpha y) + \mathrm{Tr}_\ell((\alpha y)^{2^\ell}) \\ &= \mathrm{Tr}_\ell(\alpha y) + \mathrm{Tr}_\ell(\alpha^{2^\ell} y) = \mathrm{Tr}_\ell(\alpha y) + \mathrm{Tr}_\ell((\alpha+1)y) = \mathrm{Tr}_\ell(y) \ ,\end{aligned}$$

since $\alpha^{2^\ell} = \alpha + 1$ as previously shown. The last summation in (7.5) can be split into three sums by separating the cases $x = 0$ and $y = 0$

$$2 \sum_{\substack{x,y\in\mathbb{F}_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) = 2 \sum_{\substack{y\in\mathbb{F}_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(\alpha y) + 2 \sum_{x\in\mathbb{F}_{2^{2\ell}}} \chi_3(x) + 2 \sum_{\substack{x,y\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) \ .$$

Considering the three sums separately, we have:

$$\sum_{x\in\mathbb{F}_{2^{2\ell}}} \chi_3(x) = 2^\ell - 1 \ ,$$

as $\chi_3(x) = 1$ unless $x = 0$ since ℓ is odd;

$$\sum_{\substack{y\in\mathbb{F}_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(\alpha y) = \chi_3(\alpha)(2^{\ell-1} - 1) \ ,$$

as the character is multiplicative, $\chi_3(y) = 1$ unless $y = 0$, and only the 0-trace elements (which are $2^{\ell-1} - 1$) should be counted;

$$\sum_{\substack{x,y\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) = \sum_{\substack{x,y\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(y)\chi_3(xy^{-1}+\alpha) = \sum_{\substack{z,y\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(z+\alpha) = (2^{\ell-1}-1) \sum_{z\in\mathbb{F}^*_{2^{2\ell}}} \chi_3(z+\alpha) \ .$$

as y is invertible, $\chi_3(y) = 1$ since ℓ is odd, z has been substituted for xy^{-1}, and the sum we get in the end, being independent of y, is simply multiplied by the number of values assumed by y. Altogether we have

$$G_{2\ell}(1,\chi_3) = 2^{\ell+1} - 2 + \chi_3(\alpha)(2^\ell - 2) + (2^\ell - 2) \sum_{z\in\mathbb{F}^*_{2^\ell}} \chi_3(z+\alpha) = 2^{\ell+1} - 2 + (2^\ell - 2) \sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(z+\alpha) \ ,$$

and, for later use, we define $A(\alpha) = \sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(z+\alpha)$. In order to evaluate $A(\alpha)$, we consider the sum of $A(\beta)$, for every $\beta \in \mathbb{F}_{2^{2\ell}}$, and observe that $A(\beta) = 2^\ell - 1$ if $\beta \in \mathbb{F}_{2^\ell}$, while, if $\beta \notin \mathbb{F}_{2^\ell}$ all sums assume the same value $A(\alpha)$, which is shown as follows: set $\beta = u + \alpha v$ with $v \neq 0$, then

$$\sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(z+u+\alpha v) = \sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(v)\chi_3((z+u)v^{-1}+\alpha) = \sum_{z'\in\mathbb{F}_{2^\ell}} \chi_3(z'+\alpha) \ .$$

Therefore, the sum $\sum_{\beta\in\mathbb{F}_{2^{2\ell}}} A(\beta) = \sum_{\beta\in\mathbb{F}_{2^{2\ell}}} \sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(z+\beta) = \sum_{z\in\mathbb{F}_{2^\ell}} \sum_{\beta\in\mathbb{F}_{2^{2\ell}}} \chi_3(z+\beta) = 0$ yields

$$2^\ell(2^\ell - 1) + (2^{2\ell} - 2^\ell)A(\alpha) = 0$$

which implies $A(\alpha) = -1$, and finally

$$G_{2\ell}(1,\chi_3) = 2^{\ell+1} - 2 - (2^\ell - 2) = 2^\ell \ .$$

□

Remark 7.4. *The above theorem can also be proved using a theorem by Stickelberger ([68, Theorem 5.16])*

Theorem 7.2. *If ℓ is even, the Gauss sum $G_{2\ell}(1,\chi_3)$ is equal to $(-2)^{\ell/2}G_\ell(1,\chi_3)$.*

PROOF. The relative trace of the elements of $\mathbb{F}_{2^{2\ell}}$ over $\mathbb{F}_{2^\ell}$, which is

$$\mathrm{Tr}_{(2\ell/\ell)}(x) = x + x^{2^\ell} \ ,$$

introduces the polynomial $x + x^{2^\ell}$ which defines a mapping from $\mathbb{F}_{2^{2\ell}}$ onto $\mathbb{F}_{2^\ell}$ with kernel the subfield $\mathbb{F}_{2^\ell}$ ([68]). The equation $x^{2^\ell} + x = y$ has in fact exactly 2^ℓ roots in $\mathbb{F}_{2^{2\ell}}$ for every $y \in \mathbb{F}_{2^\ell}$. By definition we have

$$G_{2\ell}(1,\chi_3) = 2 \sum_{\substack{z\in\mathbb{F}_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell}(z)=0}} \chi_3(z) = 2 \sum_{\substack{x,y\in\mathbb{F}_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell}(x+\alpha y)=0}} \chi_3(x+\alpha y) \ ,$$

where α is a root of an irreducible quadratic polynomial $x^2 + x + b$ over $\mathbb{F}_{2^\ell}$, i.e. $\mathrm{Tr}_\ell(b) = 1$ ([68, Corollary 3.79]) and $\mathrm{Tr}_{(2\ell/\ell)}(\alpha) = 1$, which can be seen from the coefficient of x of the polynomial. Now

$$\mathrm{Tr}_{2\ell}(x+\alpha y) = \mathrm{Tr}_{2\ell}(x) + \mathrm{Tr}_{2\ell}(\alpha y) = \mathrm{Tr}_{2\ell}(\alpha y) = \mathrm{Tr}_\ell(\alpha y) + \mathrm{Tr}_\ell(\alpha^{2^\ell} y) \ ,$$

but $\alpha^{2^\ell} = 1 + \alpha$, so that $\mathrm{Tr}_{2\ell}(x + \alpha y) = \mathrm{Tr}_\ell(y)$, and we have

$$G_{2\ell}(1,\chi_3) = 2 \sum_{\substack{x,y\in\mathbb{F}_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) = 2 \sum_{x\in\mathbb{F}_{2^\ell}} \chi_3(x) + 2 \sum_{\substack{y\in\mathbb{F}^*_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(\alpha y) + 2 \sum_{\substack{x,y\in\mathbb{F}^*_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) \ ,$$

where the first summation has been split into the sum of three summations, by separating the cases $y = 0$ and $x = 0$. We observe that, since the character over $\mathbb{F}_{2^\ell}$ is not trivial, the first sum is 0 and the second is $\chi_3(\alpha)G_\ell(1,\chi_3)$, while the third sum can be written as follows

$$2 \sum_{\substack{x,y\in\mathbb{F}^*_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(x+\alpha y) = 2 \sum_{\substack{x,y\in\mathbb{F}^*_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(y)\chi_3(xy^{-1}+\alpha) = 2 \sum_{\substack{y\in\mathbb{F}^*_{2^\ell} \\ \mathrm{Tr}_\ell(y)=0}} \chi_3(y) \sum_{z\in\mathbb{F}^*_{2^\ell}} \chi_3(z+\alpha) \ .$$

Putting all together, we obtain

$$G_{2\ell}(1,\chi_3) = G_\ell(1,\chi_3) \sum_{z\in\mathbb{F}_{2^\ell}} \chi_3(z+\alpha) = G_\ell(1,\chi_3)A_\ell(\alpha) \ ,$$

which shows that $|A_\ell(\alpha)| = 2^{\ell/2}$ and that $A_\ell(\alpha)$ is real, as both $G_{2\ell}(1,\chi_3)$ and $G_\ell(1,\chi_3)$ are real. Note that this holds for any α with $\mathrm{Tr}_{(2\ell/\ell)}(\alpha) = 1$.
We will show now that $A_\ell(\alpha) = (-2)^{\ell/2}$. Consider the sum of $A_\ell(\gamma)$ over all γ with relative trace equal to 1, which is on one hand $2^\ell A_\ell(\alpha)$, as the polynomial $x^{2^\ell} + x = 1$ has exactly 2^ℓ roots in $\mathbb{F}_{2^{2\ell}}$ and on the other hand, explicitly we have

$$\sum_{\substack{\gamma\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell/\ell}(\gamma)=1}} A_\ell(\gamma) = \sum_{z\in\mathbb{F}_{2^\ell}} \sum_{\substack{\gamma\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell/\ell}(\gamma)=1}} \chi_3(z+\gamma) = \sum_{z\in\mathbb{F}_{2^\ell}} \sum_{\substack{\gamma'\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell/\ell}(\gamma')=1}} \chi_3(\gamma') = 2^\ell \sum_{\substack{\gamma'\in\mathbb{F}^*_{2^{2\ell}} \\ \mathrm{Tr}_{2\ell/\ell}(\gamma')=1}} \chi_3(\gamma') \ ,$$

where the summation order has been exchanged, and $\mathrm{Tr}_{2\ell/\ell}(\gamma) = \mathrm{Tr}_{2\ell/\ell}(\gamma')$ as $\mathrm{Tr}_{2\ell/\ell}(z) = 0$ for any $z \in \mathbb{F}_{2^\ell}$. Comparing the two results, we have

$$A_\ell(\alpha) = \sum_{\substack{\gamma' \in \mathbb{F}_{2^{2\ell}}^* \\ \mathrm{Tr}_{2\ell/\ell}(\gamma')=1}} \chi_3(\gamma') = M_0 + M_1\zeta_3 + M_2\zeta_3^2 \ ,$$

where M_0 is the number of γ' with $\mathrm{Tr}_{2\ell/\ell}(\gamma') = 1$ that are cubic residues, i.e. they have character $\chi_3(\gamma')$ equal to 1, M_1 is the number of γ' with $\mathrm{Tr}_{2\ell/\ell}(\gamma') = 1$ that have character ζ_3, and M_2 is the number of γ' with $\mathrm{Tr}_{2\ell/\ell}(\gamma') = 1$ that have character ζ_3^2, then $M_0 + M_1 + M_2 = 2^\ell$, and $M_1 = M_2$ since $A_\ell(\alpha)$ is real. Therefore,we have $A_\ell(\alpha) = M_0 - M_1$, and so we consider two equations for M_0 and M_1

$$\begin{cases} M_0 + 2M_1 = 2^\ell \\ M_0 - M_1 = \pm 2^{\ell/2} \end{cases}$$

solving for M_1 we have $M_1 = \frac{1}{3}(2^\ell \mp 2^{\ell/2})$. Since M_1 must be an integer, we have

$$\begin{cases} M_0 - M_1 = 2^{\ell/2} & \text{if } \ell/2 \text{ is even} \\ M_0 - M_1 = -2^{\ell/2} & \text{if } \ell/2 \text{ is odd.} \end{cases}$$

□

Corollary 7.5. *If ℓ is even, the value of the Gauss sum $G_{2\ell}(1, \chi_3)$ is -2^ℓ.*

PROOF. It is a direct consequence of the two theorems above.

□

7.2 Odd characteristic

Let $\mathbb{F}_{p^r}$ be a Galois field of order p^r, with $\mathrm{Tr}_r(x) = \sum_{j=0}^{r-1} x^{p^j}$ being the trace function over $\mathbb{F}_{p^r}$, and $\mathrm{Tr}_{r/d}(x) = \sum_{j=0}^{r/d-1} x^{p^{dj}}$ the relative trace function over $\mathbb{F}_{p^r}$ relatively to $\mathbb{F}_{p^d}$, with $d|r$ [68].
Further let χ_m be a character of order m defined over $\mathbb{F}_{p^r}$ and taking values in the cyclotomic field $\mathbb{Q}(\zeta_m)$, where ζ_m denotes a primitive m-th complex root of unity.
The Gauss sum of χ_m over $\mathbb{F}_{p^r}$ is defined, [16, 60], for any $\beta \in \mathbb{F}_{p^r}$ as

$$G_r(\beta, \chi_m) = \sum_{y \in \mathbb{F}_{p^r}} \chi_m(y)\zeta_p^{\mathrm{Tr}_r(\beta y)} = \bar{\chi}_m(\beta)G_r(1, \chi_m) \ .$$

We will focus our interest on cubic characters χ_3 for odd primes p, while the case $p = 2$ is dealt with in [102]. A cubic character χ_3 can be either the principal character, i.e. $\chi_3(\beta) = 1$ for all $\beta \in \mathbb{F}_{p^r}^*$, or a non-principal character (if $p^r \equiv 1 \bmod 6$)

$$\chi_3(\alpha^{h+3j}) = \zeta_3^h \quad h = 0, 1, 2 \ , \ j \in \mathbb{N} \ ,$$

where α is a generator of $\mathbb{F}_{p^r}^*$. In addition, $\chi_3(0) = 0$ by definition.
By the above assumptions, the values of the Gauss sums of a cubic character over $\mathbb{F}_{p^r}$ are in general algebraic integers in the field $\mathbb{Q}(\zeta_3, \zeta_p) = \mathbb{Q}(\zeta_{3p})$, $p \neq 3$ and $\mathbb{Q}(\zeta_3)$, if $p = 3$. Our aim is to give

a thorough overview and derive more precise statements about these values with elementary techniques. In particular, we have obtained an interesting new result in the vein of Gauss' closed expression for $G_1(1,\chi_2)$ in terms of a fourth root of unity and a root of an integral quadratic polynomial. Specifically, our equation (7.9) expresses $G_1(1,\chi_3)$ in terms of a cubic root of unity and a root of an integral cubic polynomial (see Equation (7.6)), whose coefficients are given explicit functions of p. Gauss sums over extended fields are obtained from the expression of Gauss sums over smaller fields, either using Davenport-Hasse's theorem or with new methods developed here to link values over different field extensions.

7.2.1 Lemmas

For the considerations below, let

$$A_s(\alpha) = \sum_{y\in\mathbb{F}_{p^s}} \chi_m(y+\alpha) \quad \text{and} \quad B_{r,s}(\alpha) = \sum_{z_1\ldots,z_{r-1}\in\mathbb{F}_{p^s}} \chi_m(1+\sum_{i=1}^{r-1} z_i\alpha^i) \ ,$$

where α may be in an extension of $\mathbb{F}_{p^s}$, and χ_m defined in the same extension.

Lemma 7.6. *Let χ_m be a nontrivial character of order m over $\mathbb{F}_{p^{rs}}$, p prime, whose restriction to $\mathbb{F}_{p^s}$ is also nontrivial, and assume that there exists an irreducible polynomial $X^r-\beta$, for a suitable $\beta\in\mathbb{F}_{p^s}$. Then*

$$G_{rs}(1,\chi_m) = \bar{\chi}_m(r)G_s(1,\chi_m)B_{r,s}(\alpha),$$

where α is a root of $X^r-\beta$ (thus with relative trace $\mathrm{Tr}_{rs/s}(\alpha)=0$).

PROOF. Since $\mathbb{F}_{p^{rs}}$ is an extension of order r of $\mathbb{F}_{p^s}$, its elements can be written in the form $\sum_{i=0}^{r-1} x_i\alpha^i$ with $x_0,\ldots,x_{r-1}\in\mathbb{F}_{p^s}$. We thus have

$$\begin{aligned} G_{rs}(1,\chi_m) &= \sum_{z\in\mathbb{F}_{p^{rs}}} \chi_m(z)\zeta_p^{\mathrm{Tr}_{rs}(z)} = \sum_{x_0,\ldots,x_{r-1}\in\mathbb{F}_{p^s}} \chi_m(\sum_{i=0}^{r-1} x_i\alpha^i)\zeta_p^{\mathrm{Tr}_{rs}(\sum_{i=0}^{r-1} x_i\alpha^i)} \\ &= \sum_{x_0,\ldots,x_{r-1}\in\mathbb{F}_{p^s}} \chi_m(\sum_{i=0}^{r-1} x_i\alpha^i)\zeta_p^{\mathrm{Tr}_s(rx_0)} = \sum_{\substack{x_0\in F_{p^s}^* \\ x_1\ldots,x_{r-1}\in\mathbb{F}_{p^s}}} \chi_m(\sum_{i=0}^{r-1} x_i\alpha^i)\zeta_p^{\mathrm{Tr}_s(rx_0)}, \end{aligned}$$

where we used that

$$\mathrm{Tr}_{rs}(\sum_{i=0}^{r-1} x_i\alpha^i) = \mathrm{Tr}_s(\mathrm{Tr}_{rs/s}(\sum_{i=0}^{r-1} x_i\alpha^i)) = \sum_{i=0}^{r-1} \mathrm{Tr}_s(x_i\mathrm{Tr}_{rs/s}(\alpha^i)) = \mathrm{Tr}_s(rx_0) \ ,$$

as $\mathrm{Tr}_{rs/s}(x_i\alpha^i) = x_i\mathrm{Tr}_{rs/s}(\alpha^i)$ because $x_i^{p^s}=x_i$, further $\mathrm{Tr}_{rs/s}(1)=r$ and $\mathrm{Tr}_{rs/s}(\alpha^i)=0$ if $i>0$ because α is a root of the polynomial $X^r-\beta$ which has every coefficient (that is an elementary symmetric function of the roots) equal to zero with the exclusion of the constant coefficient, then every symmetric function of the powers of the roots $s_i=\sum_{j=0}^{r-1}(\alpha^{p^{js}})^i = \mathrm{Tr}_{rs/s}(\alpha^i)$ with $i=1,\ldots,r-1$ is zero as a consequence of the Newton formulas [25, vol. I, pg. 166], we thus have $\mathrm{Tr}_{rs}(\sum_{i=0}^{r-1} x_i\alpha^i) = \mathrm{Tr}_s(rx_0)$; the sum $\sum_{x_1\ldots,x_{r-1}\in\mathbb{F}_{p^s}} \chi_m(\sum_{i=1}^{r-1} x_i\alpha^i)$ is zero, since with the

change of variable $x_i = x_i'\lambda$, where λ is an element of $\mathbb{F}_{p^s}^*$ with $\chi_m(\lambda) \neq 1$, the sum becomes $\chi_m(\lambda)\sum_{x_1\ldots,x_{r-1}\in\mathbb{F}_{p^s}} \chi_m(\sum_{i=1}^{r-1} x_i\alpha^i)$.
Now, as $x_0 \neq 0$, we may perform the change of variables $x_i = x_0 z_i$, $i = 1, \ldots, r-1$ and write

$$\begin{aligned} G_{rs}(1,\chi_m) &= \sum_{x_0\in F_{p^s}^*} \zeta_p^{\mathrm{Tr}_s(rx_0)} \sum_{z_1\ldots,z_{r-1}\in\mathbb{F}_{p^s}} \chi_m(x_0 + x_0\sum_{i=1}^{r-1} z_i\alpha^i) \\ &= \sum_{x_0\in F_{p^s}^*} \zeta_p^{\mathrm{Tr}_s(rx_0)}\chi_m(x_0) \sum_{z_1\ldots,z_{r-1}\in\mathbb{F}_{p^s}} \chi_m(1 + \sum_{i=1}^{r-1} z_i\alpha^i) \ . \end{aligned}$$

The conclusion is immediate, noting that the first summation is simply $\bar\chi_m(r)G_s(1,\chi_m)$. □

The following Lemma is a corollary of the previous one, specialized to the case $r = 2$. However, we present another proof, whose structure and running results are instrumental to proofs of further theorems.

Lemma 7.7. *Let χ_m be a nontrivial character of order m over $\mathbb{F}_{p^{2s}}$, p odd, whose restriction to $\mathbb{F}_{p^s}$ is also nontrivial; then*

$$G_{2s}(1,\chi_m) = \chi_m(\alpha)G_s(1,\chi_m)A_s(\frac{1}{2\alpha}) \ ,$$

where α is a root of an irreducible polynomial $X^2 - \beta$ for a suitable $\beta \in \mathbb{F}_{p^s}$.

We note that from the definition of α and β it follows that $\mathrm{Tr}_{2s/s}(\alpha) = 0$ and that $\chi_2(\beta) = -1$ for the nontrivial quadratic character over $\mathbb{F}_{p^s}$.
PROOF. Since $\mathbb{F}_{p^{2s}}$ is a quadratic extension of $\mathbb{F}_{p^s}$, its elements can be written in the form $x + \alpha y$ with $x, y, \in \mathbb{F}_{p^s}$. We thus have

$$G_{2s}(1,\chi_m) = \sum_{z\in\mathbb{F}_{p^{2s}}} \chi_m(z)\zeta_p^{\mathrm{Tr}_{2s}(z)} = \sum_{x,y\in\mathbb{F}_{p^s}} \chi_m(x+\alpha y)\zeta_p^{\mathrm{Tr}_{2s}(x+\alpha y)} = \sum_{x,y\in\mathbb{F}_{p^s}} \chi_m(x+\alpha y)\zeta_p^{\mathrm{Tr}_s(2x)} \ ,$$

where we have used the equality $\mathrm{Tr}_{2s}(x + \alpha y) = 2\mathrm{Tr}_s(x) = \mathrm{Tr}_s(2x)$. Multiplying the last sum by $\bar\chi_m(2)\chi_m(2) = 1$, we can write

$$G_{2s}(1,\chi_m) = \bar\chi_m(2) \sum_{x',y\in\mathbb{F}_{p^s}} \chi_m(x' + 2\alpha y)\zeta_p^{\mathrm{Tr}_s(x')} \ ,$$

and split the summation into three sums

$$\bar\chi_m(2) \sum_{y\in\mathbb{F}_{p^s}} \chi_m(2\alpha y) \ , \ \bar\chi_m(2) \sum_{x'\in\mathbb{F}_{p^s}^*} \chi_m(x')\zeta_p^{\mathrm{Tr}_s(x')} \ , \ \bar\chi_m(2) \sum_{x',y\in\mathbb{F}_{p^s}^*} \chi_m(x' + 2\alpha y)\zeta_p^{\mathrm{Tr}_s(x')} \ .$$

The first summation is 0, the second summation is $\bar\chi_m(2)G_s(1,\chi_m)$; the third summation can be written as follows: the substitution $y = zx'$ yields

$$\bar\chi_m(2) \sum_{x',z\in\mathbb{F}_{p^s}^*} \chi_m(x' + 2\alpha zx')\zeta_p^{\mathrm{Tr}_s(x')} = \bar\chi_m(2) \sum_{x'\in\mathbb{F}_{p^s}^*} \chi_m(x')\zeta_p^{\mathrm{Tr}_s(x')} \sum_{z\in\mathbb{F}_{p^s}^*} \chi_m(1 + 2\alpha z) =$$

$$\bar{\chi}_m(2)G_s(1,\chi_m)\chi_m(2\alpha)\sum_{z\in\mathbb{F}_{p^s}^*}\chi_m(z+\frac{1}{2\alpha}) = \bar{\chi}_m(2)G_s(1,\chi_m)\chi_m(2\alpha)[A_s(\frac{1}{2\alpha})-\chi_m(\frac{1}{2\alpha})] \ .$$

In conclusion, by combining the above summations, we have $G_{2s}(1,\chi_m) = \chi_m(\alpha)G_s(1,\chi_m)A_s(\frac{1}{2\alpha})$.

□

Corollary 7.8. *Suppose p is odd and $t = 2^k s$, with $k \geq 1$, and let χ_m be a nontrivial character over $\mathbb{F}_{p^t}$, whose restriction to $\mathbb{F}_{p^s}$ is also nontrivial. Then*

$$G_t(1,\chi_m) = G_s(1,\chi_m)\prod_{i=1}^{k}\chi_m(\alpha_i)A_{2^{i-1}s}(\frac{1}{2\alpha_i}),$$

where α_i is a root of an irreducible polynomial $X^2-\beta_i$ over $\mathbb{F}_{p^{2^{i-1}s}}$, $i=1,\ldots,k$.

Lemma 7.9. *Let χ_m be a character over $\mathbb{F}_{p^s}$, $p \equiv -1$ mod m and m odd. Then $G_s(1,\chi_m)$ is real.*

PROOF. We can write

$$G_s(1,\chi_m) = G_0 + \zeta_m G_1 + \zeta_m^2 G_2 + \cdots + \zeta_m^{m-1}G_{m-1} \ ,$$

where ζ_m is a primitive m-th root of unity and

$$G_j = \sum_{\chi_m(x)=\zeta_m^j}\zeta_p^{\mathrm{Tr}_s(x)} \ , \ 0 \leq j \leq m-1 \ ,$$

known as Gauss periods [50], are real numbers, since $\chi_m(x) = \chi_m(-x)$, as $-1 = (-1)^m$ is an m-th power. Thus, in each sum the exponentials occur in complex conjugated pairs. Furthermore, $G_j = G_{m-j}$ as proved by the following chain of equalities:

$$G_j = \sum_{\chi_m(x)=\zeta_m^j}\zeta_p^{\mathrm{Tr}_s(x)} = \sum_{\chi_m(x)=\zeta_m^j}\zeta_p^{\mathrm{Tr}_s(x^p)} = \sum_{\chi_m(x^p)=\zeta_m^{pj}}\zeta_p^{\mathrm{Tr}_s(x^p)} = \sum_{\chi_m(y)=\zeta_m^{m-j}}\zeta_p^{\mathrm{Tr}_s(y)} = G_{m-j} \ .$$

In fact raising the trace argument to the power p leaves the trace invariant; $\zeta_m^{pj} = \zeta_m^{-j}$ as p is congruent to -1 modulo m; lastly, the automorphism $\sigma(x) = x^p$ simply permutes the elements of the field. Then, for any j, $\zeta_m^j G_j$ and $\zeta_m^{m-j}G_{m-j}$ sum to give a real number, hence $G_s(1,\chi_m)$ is also real.

□

Corollary 7.10. *Let χ_3 be a cubic character, $p \equiv 2$ modulo 3 ($p=2$ or $p=6k+5$). Then $G_s(1,\chi_3)$ is real.*

Remark 7.11. *For the case $p=2$, see an alternative proof in the previous section.*

Lemma 7.12. *Let χ_m be a nontrivial character over $\mathbb{F}_{p^{2s}}$, with p and m odd. If $(p^s-1,m)=1$ (in particular the restriction of χ_m to $\mathbb{F}_{p^s}$ is trivial), then $G_{2s}(1,\chi_m) = p^s$.*

PROOF. As in Lemma 7.7, let α be defined as a root of an irreducible polynomial $X^2 - \beta$, with a suitable $\beta \in \mathbb{F}_{p^s}$. Then

$$G_{2s}(1, \chi_m) = \sum_{x,y \in \mathbb{F}_{p^s}} \chi_m(x + \alpha y) \zeta_p^{\mathrm{Tr}_s(2x)} \ .$$

We split the summation into three: $S_1 = \bar{\chi}_m(2) \sum_{y \in \mathbb{F}_{p^s}} \chi_m(2\alpha y)$,

$$S_2 = \bar{\chi}_m(2) \sum_{x' \in \mathbb{F}_{p^s}^*} \chi_m(x') \zeta_p^{\mathrm{Tr}_s(x')}, \text{ and } S_3 = \bar{\chi}_m(2) \sum_{x', y \in \mathbb{F}_{p^s}^*} \chi_m(x' + 2\alpha y) \zeta_p^{\mathrm{Tr}_s(x')} \ .$$

The first summation is $S_1 = \chi_m(\alpha)(p^s - 1)$, since the character is trivial over $\mathbb{F}_{p^s}$, the second summation is $S_2 = -\bar{\chi}_m(2)$, and the third summation, after the substitution $y = zx'$, gives

$$S_3 = \bar{\chi}_m(2) \sum_{x' \in \mathbb{F}_{p^s}^*} \chi_m(x') \zeta_p^{\mathrm{Tr}_s(x')} \sum_{z \in \mathbb{F}_{p^s}^*} \chi_m(1 + 2\alpha z) = -\bar{\chi}_m(2) [\chi_m(2\alpha) \sum_{z \in \mathbb{F}_{p^s}} \chi_m(\frac{1}{2\alpha} + z) - 1] \ .$$

In order to evaluate $A_s(\frac{1}{2\alpha}) = \sum_{z \in \mathbb{F}_{p^s}} \chi_m(\frac{1}{2\alpha} + z)$, we consider the sum of $A_s(\beta)$, for every $\beta \in \mathbb{F}_{p^{2s}}$, and observe that $A_s(\beta) = p^s - 1$ if $\beta \in \mathbb{F}_{p^s}$, since all elements in this field are m-th powers, while, if $\beta \notin \mathbb{F}_{p^s}$ all sums assume the same value $A_s(\alpha)$, which is shown as follows: set $\beta = u + \alpha v$ with $v \neq 0$, then

$$\sum_{z \in \mathbb{F}_{p^s}} \chi_m(z + u + \alpha v) = \sum_{z \in \mathbb{F}_{p^s}} \chi_m(v) \chi_m((z + u)v^{-1} + \alpha) = \sum_{z' \in \mathbb{F}_{p^s}} \chi_m(z' + \alpha) = A_s(\alpha) \ .$$

Therefore, the sum $\sum_{\beta \in \mathbb{F}_{p^{2s}}} A(\beta) = \sum_{\beta \in \mathbb{F}_{p^{2s}}} \sum_{z \in \mathbb{F}_{p^s}} \chi_m(z + \beta) = \sum_{z \in \mathbb{F}_{p^s}} \sum_{\beta \in \mathbb{F}_{p^{2s}}} \chi_m(z + \beta) = 0$ yields

$$p^s(p^s - 1) + (p^{2s} - p^s) A(\alpha) = 0$$

which implies $A(\alpha) = -1 = A(\frac{1}{2\alpha})$. Finally, by combining the above,

$$G_{2s}(1, \chi_m) = \chi_m(\alpha)(p^s - 1) + \chi_m(\alpha) = \chi_m(\alpha) p^s = p^s \ ,$$

because α, a root of $X^2 - \beta$, is an m-th power, since every $\beta \in \mathbb{F}_{p^s}$ is an m-th power.

□

Remark 7.13. *The above lemma can also be proved using a theorem by Stickelberger, [68, Theorem 5.16] or [114].*

7.2.2 Results

Trivial character. Let χ_3 be trivial, then

$$G_r(1, \chi_3) = \sum_{y \in \mathbb{F}_{p^r}} \chi_3(y) \zeta_p^{\mathrm{Tr}_r(y)} = \sum_{y \in \mathbb{F}_{p^r}} \zeta_p^{\mathrm{Tr}_r(y)} - 1 = p^{r-1} \sum_{a \in \mathbb{F}_p} \zeta_p^a - 1 = -1 \ ,$$

since the number of elements with the same trace $a \in \mathbb{F}_p$ (0 included) is equal to p^{r-1}, i.e. the number of roots in $\mathbb{F}_{p^r}$ of the equation $\mathrm{Tr}_r(x) = a$. This result settles in particular all the cases of the fields $\mathbb{F}_{3^r}$, or $\mathbb{F}_{p^r}$ with $p \equiv 5 \bmod 6$ and odd r, where there is only the principal character, because every field element is a cube.

Nontrivial character: case p=6k+5. If $p = 6k + 5$ and r is even, a nontrivial cubic character exists and it will be shown that $G_r(1, \chi_3) = -(-p)^{r/2}$, without recurring to Davenport-Hasse's theorem.

Theorem 7.3. *If $p = 6k + 5$ and s is odd, then $G_{2s}(1, \chi_3) = p^s$.*

PROOF. Since $p^s = -1 \bmod 3$, the conclusion is a consequence of Lemma 7.12.

□

Theorem 7.4. *If $p = 6k + 5$ and s is even, then $G_{2s}(1, \chi_3) = (-p)^{s/2} G_s(1, \chi_3)$.*

PROOF. Let $\alpha \in \mathbb{F}_{p^{2s}}$ be a cube and root of an irreducible polynomial $X^2 - \beta$ over $\mathbb{F}_{p^s}$ (clearly such an α exists, since if γ is a root of $X^2 - \beta$, with $\chi_2(\beta) = -1$, then γ^3, a cube, is a root of $X^2 - \beta^3$ and $\chi_2(\beta^3) = \chi_2(\beta)^3 = -1$). Then by Lemma 7.7

$$G_{2s}(1, \chi_3) = G_s(1, \chi_3) A_s(\frac{1}{2\alpha}) \ ,$$

where $A_s(\frac{1}{2\alpha}) = \sum_{z \in \mathbb{F}_{p^s}} \chi_3(\frac{1}{2\alpha} + z)$ is an algebraic integer in the cyclotomic field $\mathbb{Q}(\zeta_3)$ which can be written as $A_0 + \zeta_3 A_1 + \zeta_3^2 A_2$, where A_0, A_1, and A_2 are the numbers of z for which $\chi_3(\frac{1}{2\alpha} + z)$ is equal to 1, ζ_3 or ζ_3^2, respectively, and $A_0 + A_1 + A_2 = p^s$.

Now, by Lemma 7.9, both $G_{2s}(1, \chi_3)$ and $G_s(1, \chi_3)$ are real, which implies that $A_s(\frac{1}{2\alpha})$ is also real, so that $A_1 = A_2$. We also know that $A_0 + A_1 + A_2 = A_0 + 2A_1 = p^s$, so we consider two equations for A_0 and A_1:

$$\begin{cases} A_0 + 2A_1 = p^s \\ A_0 - A_1 = \pm p^{s/2} \end{cases}$$

obtained from the fact that we know the absolute values of $G_s(1, \chi_3)$ and $G_{2s}(1, \chi_3)$, [16, Theorem 1.1.4, pg. 10], [102].
Solving for A_1 we have $A_1 = \frac{1}{3}(p^s \mp p^{s/2})$. As A_1 must be an integer, we have

$$A_s(\frac{1}{2\alpha}) = A_0 - A_1 = \begin{cases} p^{s/2} & \text{if } s/2 \text{ is even} \\ -p^{s/2} & \text{if } s/2 \text{ is odd.} \end{cases}$$

□

Corollary 7.14. *If $p = 6k + 5$ and s is even, then $G_{2s}(1, \chi_3) = -p^s$.*

Nontrivial character: case p=6k+1. If $p = 6k+1$, $p^r - 1$ is divisible by 3, so there exists a nontrivial cubic character in $\mathbb{F}_{p^r}$ for every $r \geq 1$: we know that the Gauss sum over $\mathbb{F}_p$ of a nontrivial cubic character is an algebraic integer in $\mathbb{Q}(\zeta_{3p})$ of absolute value $\sqrt{p}$. Specifically we have

Theorem 7.5. *If $p = 6k + 1$, then $G_1(1, \chi_3)$ is an element of $\mathbb{Q}(\zeta_3, \eta)$, a subfield of $\mathbb{Q}(\zeta_{3p})$ with degree 6 over $\mathbb{Q}$, where η is a root of a cubic polynomial with rational integer coefficients and cyclic Galois group over $\mathbb{Q}$.*

PROOF. As in the proof of Lemma 7.9, for a cubic character we can write

$$G_1(1, \chi_3) = G_0 + \zeta_3 G_1 + \zeta_3^2 G_2 \ ,$$

where ζ_3 is a primitive cube root of unity and, for $0 \le j \le 2$,

$$G_j = \sum_{\chi_3(x)=\zeta_3^j} \zeta_p^x \ ,$$

which are real numbers since $\chi_3(x) = \chi_3(-x)$, as -1 is a cubic power. Then, to evaluate the Gauss sum $G_1(1, \chi_3)$ is tantamount to computing the Gauss periods G_0, G_1, and G_2. The following derivation can be found partly, in different form, in Gauss [50, art. 350-352].

Let a be any positive integer less than p and $\sigma_a \in \mathfrak{G}(\mathbb{Q}(\zeta_p)/\mathbb{Q})$ be the element of the Galois group of $\mathbb{Q}(\zeta_p)$ whose action on ζ_p is defined as $\sigma_a(\zeta_p) = \zeta_p^a$, [121], then

$$\sigma(G_j) = \sum_{\chi_3(x)=\zeta_3^j} \zeta_p^{ax} = \sum_{\chi_3(x'a^{-1})=\zeta_3^j} \zeta_p^{x'} = \sum_{\chi_3(x')=\zeta_3^j\chi_3(a)} \zeta_p^{x'} \ .$$

This implies that any of these automorphisms induces a permutation of G_0, G_1, and G_2, and therefore leaves their symmetric functions invariant, which thus belong to $\mathbb{Q}$. In particular, the three elementary symmetric functions

$$s_1 = G_0 + G_1 + G_2 \ , \ \ s_2 = G_0G_1 + G_1G_2 + G_2G_1 \ , \ \ s_3 = G_0G_1G_2 \ ,$$

are rational integers; it follows that G_0, G_1, and G_2 are the roots of a cubic polynomial with rational coefficients $q(z) = z^3 - s_1z^2 + s_2z - s_3$, which has a cyclic Galois group of order 3 (since $\frac{p-1}{3}$ values of a give the same permutation of its roots). Thus $q(z)$ is irreducible over $\mathbb{Q}$, and denoting one root with η, the other roots can be expressed as polynomials with integer coefficients $r_1(\eta)$ and $r_2(\eta)$ of degree 2 in η.

□

Gauss computed the coefficients of the cubic polynomial $q(z)$ by a clever manipulation of the periods, a task that generally has non-polynomial-time complexity in p. The following theorem proves that these coefficients can be computed, with deterministic polynomial-time complexity, exploiting pure arithmetic features of p without dealing with Gauss periods.

Theorem 7.6. *Let $q(z)$ be the monic polynomial whose roots are G_0, G_1 and G_2. Then*

$$q(z) = z^3 + z^2 - \frac{p-1}{3}z - \frac{(3+u)p-1}{27}, \tag{7.6}$$

where u is obtained from the representation $4p = u^2 + 27v^2$ and taken with the sign making the constant term an integer.

PROOF. Let $q(z)$ be as above $z^3 - s_1z^2 + s_2z - s_3$. It is immediately seen that $s_1 = -1$, as $\sum_{x=0}^{p-1} \zeta_p^x = 0$. Using the structure constants of the integral algebra generated by G_0, G_1, and G_2, we will show that $s_2 = -\frac{p-1}{3}$, while s_3 ultimately depends on the representation (u, v) of $4p$ by the quadratic form $u^2 + 27v^2$.

Let σ be a generator of the cyclic Galois group of $\mathbb{Q}(\eta)$ (σ is also a generator of the Galois group of $q(z)$), then $G_0 = \eta$, $G_1 = \sigma(\eta)$, and $G_2 = \sigma^2(\eta)$ are $\mathbb{Q}$-linearly independent [2] and generate an algebra, [81, Lemma 2.2], whose constants of multiplication [35] are rational integers, [81, Remark 2.3], so that G_0G_1, G_1G_2, and G_2G_0 are linear combinations of G_0, G_1, and G_2 with integer

coefficients. Furthermore, since G_0, G_1, and G_2 are cyclically permuted by the action of σ, we can write

$$\begin{cases} G_0G_1 = aG_0 + bG_1 + cG_2 \\ G_1G_2 = aG_1 + bG_2 + cG_0 \\ G_2G_0 = aG_2 + bG_0 + cG_1 \end{cases} \tag{7.7}$$

where a, b, c are integers whose sum is $\frac{p-1}{3}$, since each G_j contains $\frac{p-1}{3}$ powers of ζ_p and each G_iG_j, $i \neq j$, expands into $\frac{(p-1)^2}{9}$ terms which are powers of ζ_p whose exponents, reduced modulo p, are never 0. Then, summing the three equations, we get

$$s_2 = (a+b+c)(G_0+G_1+G_2) = -\frac{p-1}{3} \ .$$

The evaluation of s_3 requires the explicit knowledge of c: by summing the three equations, multiplied by G_2, G_0, and G_1 respectively, we obtain the relation

$$3G_0G_1G_2 = (a+b)s_2 + c(G_0^2+G_1^2+G_2^2) = (\frac{p-1}{3} - c)s_2 + c(1-2s_2) \ ,$$

which yields $s_3 = \frac{1}{3}[cp - \frac{(p-1)^2}{9}]$. Now, the value of c is specified as follows.
Since the Galois group of $q(z)$, which is a polynomial with integer coefficients, is cyclic of order 3, its discriminant Δ is the square of an integer [31, Proposition 7.4.2]. The direct computation yields

$$\Delta = -\frac{p^2(p^2 - 2p + 1 - 18cp - 18c + 81c^2)}{27} \ ,$$

then Δ must be of the form v^2p^2, whence c is obtained from the equation

$$0 = p^2 - 2p + 1 - 18cp - 18c + 81c^2 + 27u^2 = -4p + (9c - p - 1)^2 + 27v^2 \ . \tag{7.8}$$

It is known [28, 50, 56, 60] that primes of the form $6k+1$ are essentially (up to signs) represented in a unique way by the quadratic form $x^2 + 3y^2$ (note that this representation may be computed in deterministic polynomial time using the Schoof algorithm [110] and the Gauss reduction algorithm of quadratic forms [73]). Further, $4 = 1 + 3$ is represented by the same form, then $4p$ has essentially three different representations, namely

$$4p = (2x)^2 + 3(2y)^2 \quad , \quad 4p = (x-3y)^2 + 3(x+y)^2 \quad , \quad 4p = (x+3y)^2 + 3(x-y)^2 \quad .$$

Since x is relatively prime with 3, necessarily exactly one of $2y$, or $x+y$ or $x-y$ is divisible by 3 and allows us to write $4p = u^2 + 27v^2$. By comparison with (7.8) we have $9c - p - 1 = u$, and u should be taken with the sign that makes the expression $p + 1 + u$ divisible by 9, in order to have an integer c, that is u should be taken with the sign that makes it congruent to 1 modulo 3: since $p+1+u = 0 \bmod 9$ implies that the same expression is 0 modulo 3 and p is congruent to 1 modulo 3, then also u must be congruent 1 modulo 3. □

Corollary 7.15. *The multiplicative constants of the algebra generated by the periods (equation (7.7)) are obtained from the representation $4p = u^2 + 27v^2$ as*

$$a = \frac{2p - u + 9v - 4}{18} \quad , \quad b = \frac{2p - u - 9v - 4}{18} \quad , \quad c = \frac{p + 1 + u}{9},$$

where the sign of u is specified in Theorem 7.6, and the sign of v is such that a and b are compliant with the chosen definitions of G_1 and G_2.

PROOF. The value $c = \frac{p+1+u}{9}$ has been found in the proof of Theorem 7.6, where it was also remarked that $a + b + c = \frac{p-1}{3}$, thus for computing a and b, we only need a further independent relation.
Adding member by member the three equations in (7.7) after their orderly multiplication by G_0, G_1, G_2, or by G_1, G_2, G_0, respectively, we obtain

$$r_1 = G_0^2G_1 + G_1^2G_2 + G_2^2G_0 = a(G_0^2 + G_1^2 + G_2^2) + (b + c)(G_0G_1 + G_1G_2 + G_2G_0) \ ,$$

$$r_2 = G_1^2G_0 + G_0^2G_2 + G_2^2G_1 = b(G_0^2 + G_1^2 + G_2^2) + (a + c)(G_0G_1 + G_1G_2 + G_2G_0) \ .$$

If we know either r_1, or r_2, then we have a second linear equation for a and b. To compute r_1 and r_2, we observe that they are exchanged by permuting, for example, G_0 and G_1, and are invariant under a cyclic permutation of G_0, G_1, and G_2. Then, their sum $r_1 + r_2$ and product r_1r_2 are symmetric functions of the roots of $q(z)$ and a theorem of Lagrange's [25] assures that they can be expressed by means of the elementary symmetric functions s_1, s_2, and s_3 (the coefficients of $q(z)$). Thus, using properties of the symmetric functions [25], we have

$$r_1 + r_2 = s_1s_2 - 3s_3 \quad , \quad r_1r_2 = s_2^3 - 6s_1s_2s_3 - 9s_3^2 - s_3s_1^3 \ ,$$

and substituting the explicit values of s_1, s_2, and s_3 given in Theorem 7.6, we obtain

$$r_1 + r_2 = \frac{3p - 1 - p(u + 3)}{9} \quad \text{and} \quad r_1r_2 = \frac{1 + pu + p^2u^2 - 3p^3}{81} \ .$$

Solving a second degree equation, we find $r_1 = \frac{1}{2}(\frac{3p-1-p(u+3)}{9} + pv)$ and $r_2 = \frac{1}{2}(\frac{3p-1-p(u+3)}{9} - pv)$, where the sign of v should be properly chosen to match the values of r_1 and r_2 obtained from the definition of the Gauss periods. In conclusion, from the system

$$\begin{cases} a + b & = & \dfrac{2p - 4 - u}{9} \\ a^{\frac{2p+1}{3}} - b^{\frac{p-1}{3}} & = & \dfrac{2p^2 + 27pv - pu - 2u - 8}{54} \end{cases}$$

we obtain $a = \frac{2p-u+9v-4}{18}$ and $b = \frac{2p-u-9v-4}{18}$.

□

It is possible to obtain a representation of the Gauss sum $G_1(1, \chi_3)$ in terms of a single root η_p of $q(z)$ by expressing G_0, G_1, and G_2 in terms of η_p, because $\mathbb{Q}(\eta_p)$ is the splitting field of $q(z)$, as seen in Theorem 3 (cf. also [2]). For example, we may set $G_0 = \eta_p$, thus the other roots, that is, Gauss periods, are

$$G_1 = \frac{G_0^2}{v} + \frac{4 - u - 3v}{6v}G_0 + \frac{2 - u - 9v - 4p}{18v} \ ,$$

$$G_2 = -\frac{G_0^2}{v} - \frac{3v - u + 4}{6v}G_0 - \frac{9v - u - 4p + 2}{18v} \ .$$

We note that changing the sign of v is equivalent to exchanging the values of G_1 and G_2. These equations establish a correspondence between G_0, and G_1 and G_2, thus the coefficients a, b, and c

in equation (7.7) can be uniquely specified, for instance the equation $G_0G_1 = aG_0 + bG_1 + cG_2$ is satisfied choosing

$$a = \frac{2p - u - 9v - 4}{18} \quad \text{and} \quad b = \frac{2p - u + 9v - 4}{18} \ ,$$

whatever be the sign of v; c is specified in any case as shown in Theorem 7.6.

These observations yield the following representation:

Theorem 7.7. *The Gauss sum $G_1(1, \chi_3)$ is uniquely characterized in terms of a root η_p of $q(z)$, with u, v obtained from the representation $u^2 + 27v^2$ of $4p$, as*

$$\eta_p + \zeta_3(\frac{\eta_p^2}{v} + \frac{4 - u - 3v}{6v}\eta_p + \frac{2 - u - 9v - 4p}{18v}) + \zeta_3^2(-\frac{\eta_p^2}{v} - \frac{3v - u + 4}{6v}\eta_p - \frac{9v - u - 4p + 2}{18v}) \ . \quad (7.9)$$

Remark 7.16. *Since $\zeta_3^2 = -1 - \zeta_3$, we can write*

$$G_1(1, \chi_3) = G_0 - G_2 + \zeta_3(G_1 - G_2) \ ,$$

thus the relation $G_1(1, \chi_3)\bar{G}_1(1, \chi_3) = p$ yields

$$p = (G_0 - G_2)^2 - (G_0 - G_2)(G_1 - G_2) + (G_1 - G_2)^2$$

which shows that the equation $x^2 - xy + y^2 = p$ has further solutions in the maximal order of $\mathbb{Q}(\zeta_p)$ besides the solutions in rational integers, for example $x = r_1(\eta) - \eta$ and $y = r_2(\eta) - \eta$.

Example Consider $p = 7$, then the Gauss sum has the form

$$G_1(1, \chi) = (\zeta_7 + \zeta_7^6) + \zeta_3(\zeta_7^2 + \zeta_7^5) + \zeta_3^2(\zeta_7^3 + \zeta_7^4) \ ,$$

the coefficients G_j of the powers of ζ_3 are real, and are roots of the cubic polynomial $z^3 + z^2 - 2z - 1$, which has a cyclic Galois group of order 3 over $\mathbb{Q}$. Let η_7 be a root of this polynomial. The other roots are $-2 + \eta_7^2$ and $1 - \eta_7 - \eta_7^2$, thus if we choose the roots $\eta_7 = \zeta_7 + \zeta_7^6$, and the other two roots equal to $\zeta_7^2 + \zeta_7^5$ and $\zeta_7^3 + \zeta_7^4$, respectively, we obtain the expression $\eta_7 + \zeta_3(-2 + \eta_7^2) + \zeta_3^2(1 - \eta_7 - \eta_7^2)$, which coincides with the expression obtained specializing (7.9) with $p = 7$, $u = 1$, and $v = 1$. Furthermore, it is direct to check that $x = -2 + \eta_7^2 - \eta_7$ and $y = 1 - \eta_7 - \eta_7^2 - \eta_7$ give a representation of 7 through the quadratic form $x^2 - xy + y^2$ in integers of $\mathbb{Q}(\eta_7)$, which may be of interest besides the 12 representations in rational integers (see e.g. [60, Proposition 8.3.1], [78]), namely $x = 2$ and $y = -1$ and those obtained through associates and conjugates of $2 - \zeta_3$ in $\mathbb{Q}(\zeta_3)$.

A Gauss sum over $\mathbb{F}_{p^r}$ is in general also not rational, as can be found using Davenport-Hasse's theorem, [16, 63], by lifting the case over $\mathbb{F}_p$. If there exists an irreducible polynomial $X^r - \beta$ over $\mathbb{F}_p$ we can use Lemma 7.6 to obtain the following theorem:

Theorem 7.8. *If $p = 6k + 1$, and if there exists an irreducible polynomial $X^r - \beta$ over $\mathbb{F}_p$, then $G_r(1, \chi_3)$ is again an element of the subfield $\mathbb{Q}(\zeta_3, \eta)$ of degree 6 of $\mathbb{Q}(\zeta_{3p})$, and in particular it can be written in the form*

$$G_r(1, \chi_3) = \bar{\chi}_3(r)G_1(1, \chi_3)B_{r,1}(\alpha) \ ,$$

where α is a root of $X^r - \beta$. The factor $B_{r,1}(\alpha)$ has the form $B_0 + B_1\zeta_3 + B_2\zeta_3^2$ where B_0, B_1, and B_2 are positive rational integers, that can be computed, up to a permutation, from the solutions of a quadratic Diophantine equation.

PROOF. By Lemma 7.6, we have

$$G_r(1,\chi_3) = \bar{\chi}_3(r) G_1(1,\chi_3) B_{r,1}(\alpha) \ ,$$

where $B_{r,1}(\alpha)$ is an element of $\mathbb{Q}(\zeta_3)$ with absolute value $\sqrt{p^{r-1}}$ (by taking absolute values of both sides). This expression shows that $G_r(1,\chi_3)$ belongs to $\mathbb{Q}(\eta,\zeta_3)$ as $G_1(1,\chi_3)$, since $B_{r,1}(\alpha)$ belongs to $\mathbb{Q}(\zeta_3)$. The factor $B_{r,1}(\alpha) \in \mathbb{Q}(\zeta_3)$ can be written as

$$B_{r,1}(\alpha) = B_0 + B_1\zeta_3 + B_2\zeta_3^2$$

where B_0, B_1, and B_2 are positive integers whose sum is p^{r-1}. Since the square of the norm of $B_{r,1}(\alpha)$ is p^{r-1}, we have the Diophantine equation

$$p^{r-1} = B_0^2 + B_1^2 + B_2^2 - B_0B_1 - B_1B_2 - B_2B_0 \ .$$

Using the relation $B_0 + B_1 + B_2 = p^{r-1}$, we eliminate B_2 and obtain a quadratic Diophantine equation that can be solved for B_0 and B_1:

$$3B_0^2 + 3B_1^2 + 3B_0B_1 - 3p^{r-1}B_0 - 3p^{r-1}B_1 + 3(p^{2r-1} - p^{r-1}) = 0 \ .$$

With the substitution

$$B_0 = \frac{X + p^{r-1}}{3} \ , \ B_1 = \frac{Y + p^{r-1}}{3} \ ,$$

we obtain the equation

$$X^2 + XY + Y^2 - 3p^{r-1} = 0 \ ,$$

whose solutions can be obtained from the solution of

$$u^2 + uv + v^2 = p$$

as coming from

$$X - \zeta_3 Y = (1 - \zeta_3)(u - v\zeta_3)^{r-1}$$

by composition of quadratic forms. The ultimate assignment of the solutions to the B_i depends on the choice of the primitive roots in the definition of the Gauss sums.

□

In the following we focus on the special case $r = 2$ to enlighten some properties and relations of the Gauss sums seen from different perspectives.

Theorem 7.9. *If $p = 6k + 1$, then $G_2(1,\chi_3)$ is again an element of the subfield $\mathbb{Q}(\zeta_3,\eta)$ of degree 6 of $\mathbb{Q}(\zeta_{3p})$, and in particular it can be written in the form*

$$G_2(1,\chi_3) = G_1(1,\chi_3) A_1(\frac{1}{2\alpha}) \ ,$$

where α is a root of $x^2 - \beta$ and $\beta \in \mathbb{F}_p$ is a cube and quadratic non-residue.

PROOF. As in Theorem 7.4, we can find such an α and then use Lemma 7.7 to deduce

$$G_2(1,\chi_3) = G_1(1,\chi_3)\sum_{z\in\mathbb{F}_p}\chi_3(\frac{1}{2\alpha}+z) = G_1(1,\chi_3)A_1(\frac{1}{2\alpha}) \ ,$$

where $A_1(\frac{1}{2\alpha})$ is an element of $\mathbb{Q}(\zeta_3)$ with absolute value $\sqrt{p}$ and $\chi_3(\alpha)=1$.
In conclusion $G_2(1,\chi_3) = G_1(1,\chi_3)A_1(\frac{1}{2\alpha})$ shows that $G_2(1,\chi_3)$ belongs to $\mathbb{Q}(\eta,\zeta_3)$.

□

Remark 7.17. *Theorem 7.9 states that the Gauss sum $G_2(1,\chi_3)$ is the product of $G_1(1,\chi_3)$ and $A_1(\frac{1}{2\alpha})$, a result that is slightly different from that obtained using Davenport-Hasse's theorem [60], which states that $G_2(1,\chi_3') = -G_1(1,\chi_3)^2$, where χ_3' is defined by extending the nontrivial character over $\mathbb{F}_p$ to a character over $\mathbb{F}_{p^m}$, using the extension rule*

$$\chi_3'(x) = \chi_3(N_{\mathbb{F}_{p^m}}(x)),$$

where $N_{\mathbb{F}_{p^m}}(x) = x\cdot x^p\cdots x^{p^{m-1}}$ is the norm of x. In our case we have $\chi_3'(x) = \chi_3(N_{\mathbb{F}_{p^2}}(x))$, and whenever χ_3' is restricted to $\mathbb{F}_p$, we specifically have

$$\chi_3'(x) = \chi_3(N_{\mathbb{F}_{p^2}}(x)) = \chi_3(x^2) = \bar{\chi}_3(x) \quad \forall x\in\mathbb{F}_p \ .$$

Therefore, since $G_2(1,\bar{\chi}_3') = \chi_3'(-1)\bar{G}_2(1,\chi_3') = \bar{G}_2(1,\chi_3')$, the equation given by Davenport-Hasse can be read as

$$G_2(1,\chi_3') = -\bar{G}_1(1,\chi_3')^2 \ ,$$

where χ_3' is a cubic character defined in $\mathbb{F}_{p^2}$, and $\bar{G}_1(1,\chi_3')$ is evaluated on the subset $\mathbb{F}_p$.

In the following proposition we show how this relation may also be derived elementarily. First we need a well-known lemma (see also [60, Proposition 8.3.3] or [1]), for which we present an alternative proof:

Lemma 7.18. *If $p=6k+1$, then $G_1(1,\chi_3)^3 = p\sum_{x\in\mathbb{F}_p}\chi_3(x(x-1))$.*

PROOF. The proof is straightforward from the computation of the cube

$$G_1(1,\chi_3)^3 = \sum_{x,y,z\in\mathbb{F}_p}\chi_3(xyz)\zeta_p^{x+y+z} = \sum_{x,y,u\in\mathbb{F}_p}\chi_3(xy(u-x-y))\zeta_p^u \ ,$$

in which the substitution $u=x+y+z$ has been performed. The summation over x can be split into two summations S_1 and S_2, depending on whether $u=y$ or $u\neq y$. The first summation turns out to be 0, since

$$S_1 = \sum_{y\in\mathbb{F}_p}\sum_{x\in\mathbb{F}_p}\chi_3(xy\ \ (x))\zeta_p^y = \sum_{y\in\mathbb{F}_p}\chi_3(y)\zeta_p^y\sum_{x\in\mathbb{F}_p}\chi_3(x^2) = \sum_{y\in\mathbb{F}_p}\chi_3(y)\zeta_p^y\sum_{x\in\mathbb{F}_p}\chi_3^2(x) = 0 \ .$$

The second summation, with the substitution $x=x'(u-y)$, becomes

$$S_2 = \sum_{\substack{y,u\in\mathbb{F}_p\\ y\neq u}}\zeta_p^u\sum_{x\in\mathbb{F}_p}\chi_3(xy(u-x-y)) = \sum_{\substack{y,u\in\mathbb{F}_p\\ y\neq u}}\chi_3(y)\zeta_p^u\sum_{x'\in\mathbb{F}_p}\bar{\chi}_3(u-y)\chi_3(x'(1-x')) \ .$$

Defining $A = \sum_{x' \in \mathbb{F}_p} \chi_3(x'(1-x'))$, a constant that does not depend on u or y, we may write

$$S_2 = A \sum_{u \in \mathbb{F}_p} \zeta_p^u \sum_{y \neq u} \chi_3(y)\bar{\chi}_3(u-y) = A[\sum_{y \in \mathbb{F}_p} \chi_3(y)\bar{\chi}_3(0-y) + \sum_{u \neq 0} \zeta_p^u \sum_{y \in \mathbb{F}_p} \chi_3(y)\bar{\chi}_3(u-y)] \ .$$

In conclusion, we have $S_2 = Ap$, since the first summation over y is $p-1$, the second summation over y is -1 independently of u, [102, 126]; finally, the summation over u is -1, so that $p-1+(-1)(-1) = p$.

□

Since $G_1(1,\chi_3)\bar{G}_1(1,\chi_3) = p$, the above result gives

$$G_1(1,\chi_3)^3 = G_1(1,\chi_3)\bar{G}_1(1,\chi_3)A$$

which implies that $G_1(1,\chi_3)^2 = \bar{G}_1(1,\chi_3)A$. On the other hand, Theorem 7.9 gives

$$G_2(1,\chi_3) = G_1(1,\chi_3)A_1(\frac{1}{2\alpha}) \ ,$$

thus we can prove the identity $G_2(1,\chi_3) = -\bar{G}_1(1,\chi_3)^2$, implied by Davenport-Hasse's theorem, if we can prove that $A = -\bar{A}_1(\frac{1}{2\alpha})$. It is in fact immediately seen that both A and $A_1(\frac{1}{2\alpha})$ are primes of the form $a + b\zeta_3$ and field norm p in $\mathbb{Z}(\zeta_3)$. Less direct is the exact relation between them, which we establish in the following proposition making use of the function defined as

$$F(d,i) = \sum_{\substack{y \in \mathbb{F}_p^* \\ \chi_3(y)=1}} \left(\frac{g^i y + d}{p}\right) \ ,$$

where g is a primitive element in $\mathbb{F}_p$.

Proposition 7.19.

$$A = -\bar{A}_1(\frac{1}{2\alpha}) \ .$$

PROOF. We can write A in the following form

$$A = \sum_{x \in \mathbb{F}_p} \chi_3(x(x-1)) = \sum_{z \in \mathbb{F}_p} \chi_3(z^2 - \frac{1}{4}) \ , \tag{7.10}$$

where the last expression was obtained by making the substitution $x = z + \frac{1}{2}$. Furthermore, $A_1(\frac{1}{2\alpha})$ can be written in a similar form, arguing as follows:

$$\bar{A}_1(\frac{1}{2\alpha}) = \sum_{x \in \mathbb{F}_p} \bar{\chi}_3(x + \frac{1}{2\alpha}) = \sum_{x \in \mathbb{F}_p} \chi_3(x + \frac{1}{2\alpha})^2 \ .$$

Furthermore, the identity $\chi_3(y) = \chi_3(y)^p = \chi_3(y^p)$, which is true since p is congruent to 1 modulo 3 and χ_3 is a multiplicative character, implies

$$\chi_3(x + \frac{1}{2\alpha}) = \chi_3(x + \frac{1}{2\alpha})^p = \chi_3(x^p + \frac{1}{(2\alpha)^p}) = \chi_3(x - \frac{1}{(2\alpha)}) \ ,$$

as x and 2 belong to $\mathbb{F}_p$, α is a root of $x^2-\beta$ and the Frobenius automorphism exchanges the roots. Then

$$\bar{A}_1(\frac{1}{2\alpha}) = \sum_{x\in\mathbb{F}_p} \chi_3(x+\frac{1}{2\alpha})\chi_3(x+\frac{1}{2\alpha}) = \sum_{x\in\mathbb{F}_p} \chi_3(x^2-\frac{1}{4\beta}) \ . \tag{7.11}$$

We notice now that, by definition, the value of any summation $\sum_{z\in\mathbb{F}_p}\chi_3(z^2-d)$ can be written in the form $a_0+a_1\zeta_3+a_2\zeta_3^2$, where a_0, a_1 and a_2 are the numbers of $z\in\mathbb{F}_p$ such that the value of $\chi_3(z^2-d)$ is either 1, or ζ_3, or ζ_3^2. Therefore, writing $z^2-d=g^iy$, with $\chi_3(y)=1$, we have

$$a_i = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1+\left(\frac{g^iy+d}{p}\right)] = \frac{p-1}{3}+F(d,i) \quad i=0,1,2 \ ,$$

since $[1+\left(\frac{g^iy+d}{p}\right)]$ is equal to 0, if g^iy+d is not a square; it is equal to 1 if $g^iy+d=0$; and it is equal to 2 if g^iy+d is a square.
Then, setting $A=b_0+b_1\zeta_3+b_2\zeta_3^2$ and $\bar{A}_1(\frac{1}{2\alpha})=c_0+c_1\zeta_3+c_2\zeta_3^2$, and using the expressions for A and $\bar{A}_1(\frac{1}{2\alpha})$ given in (7.10) and (7.11), we obtain

$$b_i = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1+\left(\frac{g^iy+\frac{1}{4}}{p}\right)] \text{ and } c_i = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1+\left(\frac{g^iy+\frac{1}{4\beta}}{p}\right)] \quad i=0,1,2 \ .$$

The numbers c_i can be written as follows

$$c_i = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1+\left(\frac{g^iy+\frac{1}{4\beta}}{p}\right)] = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1-\left(\frac{\beta}{p}\right)\left(\frac{g^iy+\frac{1}{4\beta}}{p}\right)] = \sum_{\substack{y\in\mathbb{F}_p^* \\ \chi_3(y)=1}} [1-\left(\frac{g^i\beta y+\frac{1}{4}}{p}\right)]$$

because β is a quadratic non-residue; furthermore, since β is a cube, setting $w=y\beta$, we deduce that

$$c_i = \sum_{\substack{w\in\mathbb{F}_p^* \\ \chi_3(w)=1}} [1-\left(\frac{g^iw+\frac{1}{4}}{p}\right)] = \frac{p-1}{3}-F(\frac{1}{4},i) \ ,$$

which only differs in sign from $b_i=\frac{p-1}{3}+F(\frac{1}{4},i)$. The proposition follows from the fact that

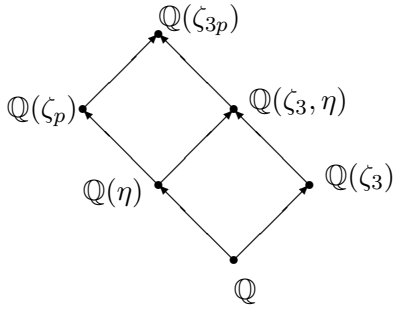
$$A=b_0+b_1\zeta_3+b_2\zeta_3^2=(b_0-b_2)+(b_1-b_2)\zeta_3$$

and

$$\bar{A}_1(\frac{1}{2\alpha}) = (c_0-c_2)+(c_1-c_2)\zeta_3 = (b_2-b_0)+(b_2-b_1)\zeta_3 \ .$$

□

Remark 7.20. *As has been said, Gauss sums are algebraic integers that belong to a subfield of a cyclotomic field, and in the above theorems we found some factorizations of Gauss sums into elements that may belong to different subfields. For example Theorem 4 shows that $G_2(1,\chi_3)\in\mathbb{Q}(\zeta_3,\eta)$ can be expressed as a product of $G_1(1,\chi_3)\in\mathbb{Q}(\zeta_3,\eta)$ and $A_1(\frac{1}{2\alpha})\in\mathbb{Q}(\zeta_3)$. The general picture of the fields involved in these factorizations is shown in the following figure*



Every extension is Galois, in particular $\mathbb{Q}(\eta)$, $\mathbb{Q}(\zeta_3)$ and $\mathbb{Q}(\zeta_3,\eta)$ have Galois groups $\mathfrak{G}(\mathbb{Q}(\eta)/\mathbb{Q})$, $\mathfrak{G}(\mathbb{Q}(\zeta_3)/\mathbb{Q})$, and $\mathfrak{G}(\mathbb{Q}(\zeta_3,\eta)/\mathbb{Q})$, which are cyclic groups of order 3, 2, and 6, respectively; moreover, the third group $\mathfrak{G}(\mathbb{Q}(\zeta_3,\eta)/\mathbb{Q}) = \mathfrak{G}(\mathbb{Q}(\zeta_3)/\mathbb{Q}) \times \mathfrak{G}(\mathbb{Q}(\eta)/\mathbb{Q})$ is a direct product of the other two (see also [121]). In these fields, every rational prime p of the form $6k+1$ splits into prime ideals as follows:

$(p) = \mathfrak{p}^3$ in $\mathbb{Q}(\eta)$, i.e. the ideal (p) fully ramifies;

$(p) = (\pi_1)(\pi_2)$ in $\mathbb{Q}(\zeta_3)$, i.e. the ideal (p) fully splits into principal ideals;

$(p) = \mathfrak{P}_1^3\mathfrak{P}_2^3$ in $\mathbb{Q}(\zeta_3,\eta)$, i.e. the ideal (p) fully splits into ramified ideals;

$(\pi_1) = \mathfrak{P}_1^3$ and $(\pi_2) = \mathfrak{P}_2^3$, i.e. the principal ideals of $\mathbb{Q}(\zeta_3)$ fully ramify in $\mathbb{Q}(\zeta_3,\eta)$;

$\mathfrak{p} = \mathfrak{P}_1\mathfrak{P}_2$ in $\mathbb{Q}(\zeta_3,\eta)$.

These factorizations can be established by the properties given in [33, pg. 137-138], that is Dedekind's formulation in terms of ideals of a theorem of Kummer's, or in [121, pg. 15].
Let τ_2 denote the automorphism of order 2 in $\mathfrak{G}(\mathbb{Q}(\zeta_3)/\mathbb{Q})$, which leaves the elements of $\mathbb{Q}(\eta)$ invariant when considered as elements of $\mathfrak{G}(\mathbb{Q}(\zeta_3,\eta)/\mathbb{Q})$, then $\tau_2(\mathfrak{P}_1) = \mathfrak{P}_2$.
Now, the Gauss sum $G_1(1,\chi_3)$ is an element of $\mathbb{Q}(\zeta_3,\eta)$ that divides p, as $G_1(1,\chi_3)\bar{G}_1(1,\chi_3) = p$, [16], so that $(G_1(1,\chi_3))(\tau_2(G_1(1,\chi_3))) = (G_1(1,\chi_3))(\bar{G}_1(1,\chi_3)) = (p)$.Therefore the principal ideal $(G_1(1,\chi_3))$ will be a product of powers of the two primes $\mathfrak{P}_1$ and $\mathfrak{P}_2$, i.e $(G_1(1,\chi_3)) = \mathfrak{P}_1^a\mathfrak{P}_2^b$, where $a+b=3$ by the unique factorization in prime ideals, since the previous relation gives $(p) = \mathfrak{P}_1^a\mathfrak{P}_2^b\tau_2(\mathfrak{P}_1^a\mathfrak{P}_2^b) = \mathfrak{P}_1^a\mathfrak{P}_2^b\mathfrak{P}_2^a\mathfrak{P}_1^b = \mathfrak{P}_1^{a+b}\mathfrak{P}_2^{a+b}$.
Thus, we may assume that $(G_1(1,\chi_3)) = \mathfrak{P}_1\mathfrak{P}_2^2$, as $G_1(1,\chi_3)$ belongs properly to $\mathbb{Q}(\zeta_3,\eta)$, whence Theorem 7.9 and Proposition 7.19 show that $(G_2(1,\chi_3)) = \mathfrak{P}_1^4\mathfrak{P}_2^2 = (\pi_1)\mathfrak{P}_1\mathfrak{P}_2^2$.
In this framework, if the character χ_3' is used, the role of the two prime ideals is simply exchanged, i.e. $(G_2(1,\chi_3')) = \mathfrak{P}_2^4\mathfrak{P}_1^2 = (\pi_2)\mathfrak{P}_2\mathfrak{P}_1^2$, which is the expression defined by Davenport-Hasse's theorem written in terms of ideals.
In general, the Gauss sums $G_s(1,\chi_3)$ for any s can be expressed in terms of ideals as follows: $(G_s(1,\chi_3)) = \mathfrak{P}_2^s\mathfrak{P}_1^{2s}$.
However, these formulations in terms of ideals (see also [34, 66, 114]) conceal the information about which units are involved. In this sense, the elementary direct approach can be more informative, although it may require different approaches for different situations. Considering for example the Gauss sum mentioned above, $G_1(1,\chi)$ for $p=7$ (see also [48]), setting $\eta_7 = \zeta_7 + \zeta_7^6$, we

can explicitly write the expression (which can also be obtained specializing (7.9) with $p = 7, u = 1, v = 1$)

$$G_1(1,\chi) = \eta_7 + \zeta_3(-2 + \eta_7^2) + \zeta_3^2(1 - \eta_7 - \eta_7^2) \ ,$$

whereas, choosing the ideals $\mathfrak{P}_1 = (\zeta_3 - \eta_7)$, $\mathfrak{P}_2 = (\zeta_3^2 - \eta_7)$, we must find a unit in order to obtain a complete factorization: $G_1(1,\chi) = (4 - \eta_7 - 2\eta_7^2)(\zeta_3 - \eta_7)(\zeta_3^2 - \eta_7)^2$, where $4 - \eta_7 - 2\eta_7^2$ is a unit.

Chapter 8

Appendix II: Additive decompositions by multiplicative characters

In 1952, Perron showed that quadratic residues in a field of prime order satisfy certain additive properties. This result has been generalized in different directions, and our contribution [104] is to provide a further generalization concerning multiplicative quadratic and cubic characters over any finite field. In particular, recalling that a character partitions the multiplicative group of the field into cosets with respect to its kernel, we will derive the number of representations of an element as a sum of two elements belonging to two given cosets. These numbers are then related to the equations satisfied by the polynomial characteristic functions of the cosets.

Further, we show a connection, a quasi-duality, with the problem of determining how many elements can be added to each element of a subset of a coset in such a way as to obtain elements still belonging to a subset of a coset.

8.1 Introduction

Back in 1952, Perron [91] proved that every quadratic residue in $\mathbb{F}_p$, with p an odd prime, can be written as the sum of two quadratic residues in exactly $\lfloor \frac{p+1}{4} \rfloor - 1$ ways, and as the sum of two quadratic non-residues in exactly $\lfloor \frac{p+1}{4} \rfloor$ ways, a symmetric statement also holding for quadratic non-residues.

Winterhof [126] generalized this result, proving that if x_j is a nonzero element in a finite field $\mathbb{F}_q$, χ a nontrivial multiplicative character of order n and ω a primitive n-th root of unity, then $\sigma_0 + 1 = \sigma_1 = \cdots = \sigma_{n-1} = \frac{q-1}{n}$, where σ_i, $i = 0, \ldots, n-1$, stands for the number of field elements x such that $\chi(x)\bar{\chi}(x + x_j) = \omega^i$.

A generalization in a different direction was provided by Monico and Elia [80, 81]: they proved that the only partition of the field satisfying the additive properties found by Perron is that of quadratic residues and non-residues, providing a sort of converse and a way to define residues additively; moreover, they generalized Perron's result for any multiplicative character defined over a prime field.

Lastly, in his dissertation [98] Raymond showed how to generalize the case of the quadratic character to any finite field.

We provide here further generalizations considering quadratic and cubic characters over any finite field. Section 8.2 provides notations and preliminaries for the rest of the chapter. In Section 8.3 we derive expressions for the number of (ordered) representations of an element as a sum of two elements belonging to two given cosets of the character partition. Furthermore, these expressions are shown to be related to the equations satisfied by the polynomial characteristic functions of the cosets. In Section 8.4 we point out the mentioned quasi-duality relationship with the other problem, and recall from chapter 3 how the two problems are involved in polynomial factorization.

8.2 Preliminaries

Throughout the chapter we will deal with multiplicative characters, which are supposed to be non-trivial.

Let $\mathbb{F}_{p^m}$, p an odd prime, be a finite field with polynomial basis $\{1, \eta, \eta^2, \ldots, \eta^{m-1}\}$ where η is a root of an irreducible polynomial of degree m over $\mathbb{F}_p$, and let $\mathcal{B}_0$ be the set of squares (excluding 0) and $\mathcal{B}_1$ the complementary set in $\mathbb{F}^*_{p^m}$. We recall that the quadratic character is a mapping from $\mathbb{F}^*_{p^m}$ into the complex numbers defined as

$$\chi_2(\alpha^h \theta) = (-1)^h \ , \ \theta \in \mathcal{B}_0, \quad h = 0, 1 \ ,$$

where α is a primitive element in $\mathbb{F}^*_{p^m}$. Furthermore, we set $\chi_2(0) = 0$.

We can define an indicator function of the sets $\mathcal{B}_j$ using the quadratic character, namely, for every $\gamma \neq 0$,

$$I_{\mathcal{B}_j}(\gamma) = \frac{1 + (-1)^j \chi_2(\gamma)}{2} = \begin{cases} 1 & \text{if } \gamma \in \mathcal{B}_j \\ 0 & \text{otherwise} \end{cases} \quad j = 0, 1 \ .$$

Also, writing $\gamma = \gamma_0 + \gamma_1 \eta + \cdots + \gamma_{m-1} \eta^{m-1}$, a polynomial characteristic function that identifies the set $\mathcal{B}_j$ can be defined as

$$f_{\mathcal{B}_j}(\mathbf{X}) = \sum_{\gamma \in \mathbb{F}^*_{p^m}} I_{\mathcal{B}_j}(\gamma) \mathbf{X}^\gamma \ ,$$

where $\mathbf{X}^\gamma$ is a short notation standing for the monomial $x_0^{\gamma_0} x_1^{\gamma_1} \cdots x_{m-1}^{\gamma_{m-1}}$. Note that this notation allows us to formally write

$$\mathbf{X}^\gamma \mathbf{X}^\delta = \mathbf{X}^{\gamma+\delta} \mod \mathfrak{I}_{\mathbf{X}} \ ,$$

where $\mathfrak{I}_{\mathbf{X}} = \langle (x_1^p - 1), \ldots, (x_m^p - 1) \rangle$ is an ideal in $\mathbb{Q}[\mathbf{X}]$.

It is immediate to see that, if $\Phi_p(x)$ is the p-th cyclotomic polynomial,

$$f_{\mathcal{B}_0}(\mathbf{X}) + f_{\mathcal{B}_1}(\mathbf{X}) + 1 = \sum_{\gamma \in \mathbb{F}^*_{p^m}} \mathbf{X}^\gamma + 1 = \prod_{i=0}^{m-1} \Phi_p(x_i) \ . \tag{8.1}$$

In the following we will indicate the last product with the notation $\Phi(\mathbf{X})$.

In the case of cubic characters, non-trivial characters exist only if p is 2 with even exponent m, or p is an odd prime congruent to 1 modulo 6, or p is an odd prime congruent to 5 modulo 6 with even exponent m.

Let us write any nonzero element of the field as α^{k+3n}, with $k \in \{0,1,2\}$ and α a primitive element: we define $\mathcal{A}_0 = \{\alpha^{3i} : \ i = 0, \ldots, \frac{p^m-1}{3} - 1\}$, that is the subgroup of the cubic powers in $\mathbb{F}^*_{p^m}$, and let $\mathcal{A}_1 = \alpha\mathcal{A}_0$ and $\mathcal{A}_2 = \alpha^2\mathcal{A}_0$ be the two cosets that complete the coset partition of the set of nonzero elements of $\mathbb{F}_{p^m}$.

Similarly to the case of the quadratic character, we define an indicator function of the sets $\mathcal{A}_j$ using a cubic character, that is a mapping of the type

$$\chi_3(\alpha^h\theta) = \omega^h \ , \ \ \theta \in \mathcal{A}_0, \quad h = 0,1,2 \ ,$$

with ω a primitive cubic root of unity in $\mathbb{C}$ (and $\chi_3(0) = 0$ by definition).

The indicator function, for every $x \neq 0$, is then

$$I_{\mathcal{A}_j}(x) = \frac{1 + \omega^{2j}\chi_3(x) + \omega^j\bar{\chi}_3(x)}{3} = \begin{cases} 1 & \text{if } \ x \in \mathcal{A}_j \\ 0 & \text{otherwise} \end{cases} \quad j = 0,1,2 \ ,$$

(where the bar denotes complex conjugation), and a characteristic function that identifies the set $\mathcal{A}_j$ can be defined in the same way as above:

$$f_{\mathcal{A}_j}(\mathbf{X}) = \sum_{\gamma \in \mathbb{F}^*_{p^m}} I_{\mathcal{A}_j}(\gamma)\mathbf{X}^\gamma \quad j = 0,1,2 \ .$$

Again it is immediate to see that

$$f_{\mathcal{A}_0}(\mathbf{X}) + f_{\mathcal{A}_1}(\mathbf{X}) + f_{\mathcal{A}_2}(\mathbf{X}) + 1 = \sum_{\gamma \in \mathbb{F}^*_{p^m}} \mathbf{X}^\gamma + 1 = \Phi(\mathbf{X}) \ . \tag{8.2}$$

Remark 8.1. *It is worthwhile to note that the use of the indicator functions provides a sort of bridge between our problems about the sums of residues and the classical problem of counting the number of solutions of certain diagonal equations over finite fields [16, 68]. The special diagonal equations we refer to are equations of the form $a_1x_1^k + a_2x_2^k + \cdots + a_nx_n^k = \alpha$, where $a_1, a_2, \ldots, a_n, \alpha$ are fixed elements of a finite field $\mathbb{F}_q$, the exponents k is a positive integer, and the unknowns $x_1, \ldots x_n$ may assume any value in the same field. A difference between the two problems is evident when the exponent k is a factor of $q-1$, for example counting the number of solutions (x, y) of $x^3 + y^3 = \alpha$ with $q - 1$ divisible by 3, is not the same as counting the number of representations of a given element α as a sum of cubic residues, that is counting the number of solutions of the equation $x + y = \alpha$ where x and y are constrained to be cubic residues. The count of the number of solutions of diagonal equations includes all x's whose power x^k has the same value, while the sum of k-th residues does not consider this multiplicity, and also excludes the zero value. Therefore, the two problems are clearly related, but a considerable effort is needed to obtain the solution of one of them from the solution of the other.*

8.3 Results

To count the number of representations of a $\beta \neq 0$ in the field $\mathbb{F}_{p^m}$ as the sum of an element in $\mathcal{B}_j$ and an element in $\mathcal{B}_i$ (i not necessarily different from j), it suffices to compute

$$\sum_{z \neq 0,\beta} \frac{1+(-1)^j \chi_2(z)}{2} \ \frac{1+(-1)^i \chi_2(\beta - z)}{2} \tag{8.3}$$

Analogously, when we have a cubic character, to count the number of representations of a $\beta \neq 0$ in the field $\mathbb{F}_{p^m}$ as the sum of an element in $\mathcal{A}_j$ and an element in $\mathcal{A}_i$ (i not necessarily different from j), it suffices to compute

$$\sum_{z \neq 0,\beta} \frac{1+\omega^{2j}\chi_3(z)+\omega^j \bar{\chi}_3(z)}{3} \ \frac{1+\omega^{2i}\chi_3(\beta - z)+\omega^i \bar{\chi}_3(\beta - z)}{3}. \tag{8.4}$$

We summarize the conclusions in the next three theorems.

Theorem 8.1. *The number of representations $R^{(2)}_{p^m}(\beta, i, j)$ of a $\beta \neq 0$ in the field $\mathbb{F}_{p^m}$, p an odd prime, as the sum of an element in $\mathcal{B}_j$ and an element in $\mathcal{B}_i$ is*

$$R^{(2)}_{p^m}(\beta, i, j) = \frac{1}{4}\left(p^m - 2 - \chi_2(\beta)(-1)^i - \chi_2(\beta)(-1)^j - (-1)^{i+j}\chi_2(-1)\right) \ , \tag{8.5}$$

and depends only on the quadratic residuacity of β.

PROOF. The proof is immediate from (8.3), since, for any nontrivial character χ, $\sum_{x \in \mathbb{F}_{p^m}} \chi(x) = 0$ and $\sum_{x \in \mathbb{F}_{p^m}} \chi(x)\bar{\chi}(x+\gamma) = -1$ ([16, 88, 102, 126]).

□

Remark 8.2. *From (8.5) the desired values can be easily read, depending on whether β is in $\mathcal{B}_0$ or $\mathcal{B}_1$ and whether p^m is congruent to 1 or 3 modulo 4 (which determines $\chi_2(-1)$ by the properties of the Jacobi symbol and the quadratic reciprocity law); in particular, we can also read the values for which $i \neq j$, that are not usually explicitly included in the literature; in this case equation (8.5) becomes*

$$\frac{1}{4}\left(p^m - 2 + \chi_2(-1)\right) \ . \tag{8.6}$$

Theorem 8.2. *The number of representations $R^{(3)}_{p^m}(\beta, i, j)$ of a $\beta \neq 0$ in the field $\mathbb{F}_{p^m}$ as the sum of an element in $\mathcal{A}_j$ and an element in $\mathcal{A}_i$ is*

$$R^{(3)}_{p^m}(\beta, i, j) = \frac{1}{9}(p^m - 2 - K - \bar{K}) \tag{8.7}$$

with

$$K = \chi_3(\beta)(\omega^{2i} + \omega^{2j}) + \omega^{2i+j} - \omega^{2i+2j}\bar{\chi}_3(\beta)J(\chi_3, \chi_3),$$

$J(\chi_3, \chi_3)$ being a Jacobi sum ($\sum_{c_1+c_2=1} \chi_3(c_1)\chi_3(c_2)$), and depends only on the cubic residuacity of β.

PROOF. Expanding equation (8.4), using $\sum_{x\in\mathbb{F}_{p^m}}\chi_3(x)=0$ and $\sum_{x\in\mathbb{F}_{p^m}}\chi_3(x)\bar{\chi}_3(x+\gamma)=-1$, and given that $\chi_3(-1)=1$, we get

$$\frac{1}{9}\left[p^m-2-\chi_3(\beta)(\omega^{2i}+\omega^{2j})-\bar{\chi}_3(\beta)(\omega^i+\omega^j)-(\omega^{2i+j}+\omega^{i+2j})+\omega^{2i+2j}A(\beta)+\omega^{i+j}\bar{A}(\beta)\right]$$

where $A(\beta)=\sum_{x\in\mathbb{F}_{p^m}}\chi_3(x)\chi_3(\beta-x)$; this summation can be manipulated as

$$\sum_{x\in\mathbb{F}_{p^m}}\chi_3(x)\chi_3(\beta-x)=\chi_3(\beta^2)\sum_{x\in\mathbb{F}_{p^m}}\chi_3(\beta^{-1}x)\chi_3(1-\beta^{-1}x)=\bar{\chi}(\beta)A(1)=\bar{\chi}(\beta)J(\chi_3,\chi_3),$$

whence the conclusion follows.

Remark 8.3. *The last expression can be further simplified taking into account that $J(\chi_3,\chi_3)$ can be computed [68, Th. 5.21] with the Gauss sums of cubic characters (cf. also [103, 102]), as*

$$J(\chi_3,\chi_3)=\frac{G_m^2(1,\chi_3)}{G_m(1,\bar{\chi}_3)}\ .$$

In particular, if $p=2$,

$$\sum_{x\in\mathbb{F}_{2^m}}\chi_3(x)\chi_3(x+1)=J(\chi_3,\chi_3)=G_m(1,\chi_3)=-(-2)^{m/2}\ .$$

Theorem 8.3. *If $p\equiv 1$ mod 4, then $R^{(2)}_{p^m}(0,i,j)$ is $\frac{p-1}{2}$ for $i=j$ or 0 if $i\neq j$; if $p\equiv 3$ mod 4, then $R^{(2)}_{p^m}(0,i,j)$ is $\frac{p-1}{2}$ for $i\neq j$ or 0 if $i=j$. $R^{(3)}_{p^m}(0,i,j)$ is $\frac{p-1}{3}$ for $i=j$ or 0 if $i\neq j$.*

PROOF. The proof is immediate, taking into account that an element α is in the same coset as $-\alpha$ exactly when $\chi_2(-1)=1$ (resp. $\chi_3(-1)=1$).

□

Working with the characteristic functions, the counterpart of the above theorems would be to multiply $f_{\mathcal{B}_i}(\mathbf{X})$ and $f_{\mathcal{B}_j}(\mathbf{X})$ (or $f_{\mathcal{A}_i}(\mathbf{X})$ and $f_{\mathcal{A}_j}(\mathbf{X})$) modulo $\mathfrak{I}_{\mathbf{X}}$, and then read the coefficients in the output, which involves exactly the same computations as above. But there is more: the characteristic functions $f_{\mathcal{B}_i}(\mathbf{X})$ and $f_{\mathcal{A}_j}(\mathbf{X})$ satisfy equations of second and third degree, respectively, whose coefficients are intrinsically related to the number of representations of the field elements as sums of elements with given quadratic or cubic residuacity.

Theorem 8.4. *The characteristic functions $f_{\mathcal{B}_0}(\mathbf{X})$ and $f_{\mathcal{B}_1}(\mathbf{X})$ are roots of a quadratic equation*

$$y^2-\sigma_1 y+\sigma_2=0 \bmod \mathfrak{I}_{\mathbf{X}} \tag{8.8}$$

in the residue ring of multivariate polynomials $\mathbb{Z}[\mathbf{X}]/\mathfrak{I}_{\mathbf{X}}$. The sum and the product of the roots (polynomials) are

$$\begin{cases}\sigma_1=-1+\Phi(\mathbf{X}) \bmod \mathfrak{I}_{\mathbf{X}}\\ \sigma_2=-\frac{1}{4}\left[p^m\chi_2(-1)-1-\Phi(\mathbf{X})(p^m-2+\chi_2(-1))\right] \bmod \mathfrak{I}_{\mathbf{X}}\end{cases}$$

PROOF. Throughout the proof all the multivariate polynomials should be intended modulo the ideal $\mathfrak{I}_{\mathbf{X}}$. The coefficient σ_1 is directly obtained from equation (8.1), that is

$$\sigma_1 = f_{\mathcal{B}_0}(\mathbf{X}) + f_{\mathcal{B}_1}(\mathbf{X}) = \sum_{\gamma \in \mathbb{F}_{p^m}^*} (I_{\mathcal{B}_0}(\gamma) + I_{\mathcal{B}_1}(\gamma))\mathbf{X}^\gamma = -1 + \Phi(\mathbf{X}) \ .$$

The coefficient σ_2 is computed using the symbols $R^{(2)}_{p^m}(\beta, i, j)$ as follows. Starting from

$$\sigma_2 = f_{\mathcal{B}_0}(\mathbf{X}) f_{\mathcal{B}_1}(\mathbf{X}) = \sum_{\gamma \in \mathbb{F}_{p^m}^*} \sum_{\kappa \in \mathbb{F}_{p^m}^*} I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(\kappa) \mathbf{X}^{\gamma+\kappa} \ ,$$

and observing that exchanging the summation indices (variables) leaves the result invariant, we may consider the symmetric summation

$$\sigma_2 = \frac{1}{2} \sum_{\gamma \in \mathbb{F}_{p^m}^*} \sum_{\kappa \in \mathbb{F}_{p^m}^*} \left[I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(\kappa) + I_{\mathcal{B}_0}(\kappa) I_{\mathcal{B}_1}(\gamma) \right] \mathbf{X}^{\gamma+\kappa} \ ,$$

and perform the index substitution $\kappa = \beta - \gamma$; the index β may be 0 but it cannot assume the value γ; thus we may write

$$\sigma_2 = \frac{1}{2} \sum_{\gamma \in \mathbb{F}_{p^m}^*} \sum_{\substack{\beta \in \mathbb{F}_{p^m} \\ \beta \neq \gamma}} \left[I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(\beta - \gamma) + I_{\mathcal{B}_0}(\beta - \gamma) I_{\mathcal{B}_1}(\gamma) \right] \mathbf{X}^{\beta} \ .$$

Now, in the summation over β we separate the term with $\beta = 0$ and write

$$\sigma_2 = \frac{1}{2} \sum_{\gamma \in \mathbb{F}_{p^m}^*} \left\{ \sum_{\substack{\beta \in \mathbb{F}_{p^m}^* \\ \beta \neq \gamma}} \left[I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(\beta - \gamma) + I_{\mathcal{B}_0}(\beta - \gamma) I_{\mathcal{B}_1}(\gamma) \right] \mathbf{X}^{\beta} + C \right\} \ .$$

where $C = [I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(-\gamma) + I_{\mathcal{B}_0}(-\gamma) I_{\mathcal{B}_1}(\gamma)] = \frac{1-\chi_2(-1)}{2}$, thus exchanging the two summations and, noting that $\sum_{\gamma \in \mathbb{F}_{p^m}^*} C = (p^m - 1)C$, we have

$$\sigma_2 = \frac{p^m - 1}{2} C + \frac{1}{2} \sum_{\beta \in \mathbb{F}_{p^m}^*} \mathbf{X}^{\beta} \left\{ \sum_{\substack{\gamma \in \mathbb{F}_{p^m}^* \\ \gamma \neq \beta}} \left[I_{\mathcal{B}_0}(\gamma) I_{\mathcal{B}_1}(\beta - \gamma) + I_{\mathcal{B}_0}(\beta - \gamma) I_{\mathcal{B}_1}(\gamma) \right] \right\} \ .$$

Recalling the definition of $R^{(2)}_{p^m}(\beta, i, j)$, we may write the summation over γ as

$$\sigma_2 = \frac{p^m - 1}{2} C + \frac{1}{2} \sum_{\beta \in \mathbb{F}_{p^m}^*} \mathbf{X}^{\beta} \left\{ R^{(2)}_{p^m}(\beta, 0, 1) + R^{(2)}_{p^m}(\beta, 1, 0) \right\} \ .$$

In conclusion, since $R^{(2)}_{p^m}(\beta, 0, 1) = R^{(2)}_{p^m}(\beta, 1, 0)$ does not depend on β, we obtain

$$\sigma_2 = \frac{p^m - 1}{2} \frac{1 - \chi_2(-1)}{2} + R^{(2)}_{p^m}(\beta, 0, 1)(\Phi(\mathbf{X}) - 1) \ ,$$

and, using (8.6), we finally obtain

$$\sigma_2 = -\frac{1}{4}\left[p^m\chi_2(-1) - 1 - \Phi(\mathbf{X})(p^m - 2 + \chi_2(-1))\right] \ .$$

□

Theorem 8.5. *The characteristic functions $f_{\mathcal{A}_0}(\mathbf{X})$, $f_{\mathcal{A}_1}(\mathbf{X})$, and $f_{\mathcal{A}_2}(\mathbf{X})$ are roots of a cubic equation*

$$y^3 - \sigma_1 y^2 + \sigma_2 y - \sigma_3 = 0 \mod \mathfrak{I}_{\mathbf{X}} \ , \tag{8.9}$$

where

$$\begin{cases} \sigma_1 &= \Phi(\mathbf{X}) - 1 \bmod \mathfrak{I}_{\mathbf{X}} \\ \sigma_2 &= \frac{1}{3}(p^m - 1)(\Phi(\mathbf{X}) - 1) \bmod \mathfrak{I}_{\mathbf{X}} \\ \sigma_3 &= \frac{1}{27}\left[(\Phi(\mathbf{X}) - 1)^3 + (3 - 3\Phi(\mathbf{X}) + J(\chi_3, \chi_3) + \bar{J}(\chi_3, \chi_3))(p^m - \Phi(\mathbf{X}))\right] \bmod \mathfrak{I}_{\mathbf{X}} \end{cases}$$

PROOF. Throughout the proof all the multivariate polynomials should be intended modulo the ideal $\mathfrak{I}_{\mathbf{X}}$. The coefficient σ_1 is easily computed as

$$f_{\mathcal{A}_0}(\mathbf{X}) + f_{\mathcal{A}_1}(\mathbf{X}) + f_{\mathcal{A}_2}(\mathbf{X}) = \sum_{\gamma \in \mathbb{F}_{p^m}^*} (I_{\mathcal{A}_0}(\gamma) + I_{\mathcal{A}_1}(\gamma) + I_{\mathcal{A}_2}(\gamma))\mathbf{X}^\gamma = -1 + \Phi(\mathbf{X}) \ ,$$

because $I_{\mathcal{A}_0}(\gamma) + I_{\mathcal{A}_1}(\gamma) + I_{\mathcal{A}_2}(\gamma) = 1$ and equation (8.1) is used. The elementary symmetric function σ_2 is the sum

$$f_{\mathcal{A}_0}(\mathbf{X})f_{\mathcal{A}_1}(\mathbf{X}) + f_{\mathcal{A}_1}(\mathbf{X})f_{\mathcal{A}_2}(\mathbf{X}) + f_{\mathcal{A}_2}(\mathbf{X})f_{\mathcal{A}_0}(\mathbf{X}) \ ,$$

then we need to compute the summation

$$\sigma_2 = \sum_{\gamma \in \mathbb{F}_{p^m}^*} \sum_{\eta \in \mathbb{F}_{p^m}^*} (I_{\mathcal{A}_0}(\gamma)I_{\mathcal{A}_1}(\theta) + I_{\mathcal{A}_1}(\gamma)I_{\mathcal{A}_2}(\theta) + I_{\mathcal{A}_2}(\gamma)I_{\mathcal{A}_0}(\theta))\mathbf{X}^{\gamma+\theta} \ .$$

Using (8.7), we get

$$\sigma_2 = -\frac{1}{3}(p^m - 1) + \frac{1}{3}(p^m - 1)\Phi(\mathbf{X}) \ .$$

Lastly, the elementary symmetric function $\sigma_3 = f_{\mathcal{A}_0}(\mathbf{X})f_{\mathcal{A}_1}(\mathbf{X})f_{\mathcal{A}_2}(\mathbf{X})$ is given by the summation

$$\sigma_3 = \sum_{\gamma \in \mathbb{F}_{p^m}^*} \sum_{\theta \in \mathbb{F}_{p^m}^*} \sum_{\kappa \in \mathbb{F}_{p^m}^*} I_{\mathcal{A}_0}(\gamma)I_{\mathcal{A}_1}(\theta)I_{\mathcal{A}_2}(\kappa)\mathbf{X}^{\gamma+\theta+\kappa} \ .$$

Expanding the product of the indicator functions and performing the summations, most of the 27 sums are canceled, and it remains to compute the following:

$$\frac{1}{27} \sum_{\gamma \in \mathbb{F}_q^*} \sum_{\theta \in \mathbb{F}_{p^m}^*} \sum_{\kappa \in \mathbb{F}_{p^m}^*} \left(1 - 3\chi_3(\gamma)\bar{\chi}_3(\theta) + \chi_3(\gamma)\chi_3(\theta)\chi_3(\kappa) + \bar{\chi}_3(\gamma)\bar{\chi}_3(\theta)\bar{\chi}_3(\kappa)\right) \mathbf{X}^{\gamma+\theta+\kappa} \ .$$

To complete the task we then need to compute only three summations.

1. The summation
$$\sum_{\gamma\in\mathbb{F}_{p^m}^*}\sum_{\theta\in\mathbb{F}_{p^m}^*}\sum_{\kappa\in\mathbb{F}_{p^m}^*}\mathbf{X}^{\gamma+\theta+\kappa}=(\Phi(\mathbf{X})-1)^3\ ;$$
is easily obtained, because the three summations on γ, θ, and κ can be performed independently.

2. The summation $\sum_{\gamma\in\mathbb{F}_{p^m}^*}\sum_{\theta\in\mathbb{F}_{p^m}^*}\sum_{\kappa\in\mathbb{F}_{p^m}^*}\chi_3(\gamma)\bar{\chi}_3(\theta)\mathbf{X}^{\gamma+\theta+\kappa}$ is computed by extending the sum range to include 0; this is done using the function $\delta(\kappa)$ which is 1 if $\kappa=0$, and is 0 otherwise, thus the summation is $\sum_{\gamma\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\sum_{\kappa\in\mathbb{F}_{p^m}}\chi_3(\gamma)\bar{\chi}_3(\theta)(1-\delta(\kappa))\mathbf{X}^{\gamma+\theta+\kappa}$ which splits into two summations
$$\sum_{\gamma\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\sum_{\kappa\in\mathbb{F}_{p^m}}\chi_3(\gamma)\bar{\chi}_3(\theta)\mathbf{X}^{\gamma+\theta+\kappa}-\sum_{\gamma\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\sum_{\kappa\in\mathbb{F}_{p^m}}\chi_3(\gamma)\bar{\chi}_3(\theta)\delta(\kappa)\mathbf{X}^{\gamma+\theta+\kappa}\ .$$
In the triple summations, the sum over κ can be performed independently and gives $\Phi(\mathbf{X})$ for the first, and simply 1 for the second. Thus we have
$$(\Phi(\mathbf{X})-1)\sum_{\gamma\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\chi_3(\gamma)\bar{\chi}_3(\theta)\mathbf{X}^{\gamma+\theta}\ .$$
The double summation can be easily evaluated with the substitution $\theta=\beta-\gamma$
$$\sum_{\beta\in\mathbb{F}_{p^m}}\sum_{\gamma\in\mathbb{F}_{p^m}}\chi_3(\gamma)\bar{\chi}_3(\beta-\gamma)\mathbf{X}^{\beta}=p^m-1-\sum_{\beta\in\mathbb{F}_{p^m}^*}\mathbf{X}^{\beta}=p^m-\Phi(\mathbf{X})\ ,$$
because the sum over γ assumes only two values, namely -1 if $\beta\neq 0$ and p^m-1 if $\beta=0$. In conclusion we obtain
$$(\Phi(\mathbf{X})-1)(p^m-\Phi(\mathbf{X}))\ .$$

3. The sums in the triple summation
$$\sum_{\gamma\in\mathbb{F}_{p^m}^*}\sum_{\theta\in\mathbb{F}_{p^m}^*}\sum_{\kappa\in\mathbb{F}_{p^m}^*}\chi_3(\gamma)\chi_3(\theta)\chi_3(\kappa)\mathbf{X}^{\gamma+\theta+\kappa}=\sum_{\beta\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\sum_{\gamma\in\mathbb{F}_{p^m}}\chi_3(\gamma)\chi_3(\theta)\chi_3(\beta-(\gamma+\theta))\mathbf{X}^{\beta}\ ,$$
have been extended throughout $\mathbb{F}_{p^m}$ as $\chi_3(0)=0$, together with the substitution $\kappa=\beta-(\gamma+\theta)$. Now, the summation over γ has two values, namely 0 if $\beta=\theta$, and $\bar{\chi}_3(\beta-\theta)A(1)$ if $\beta\neq\theta$, where as above $A(1)=\sum_{x\in\mathbb{F}_{p^m}}\chi_3(x)\chi_3(1-x)=J(\chi_3,\chi_3)$, therefore we obtain
$$\sum_{\beta\in\mathbb{F}_{p^m}}\sum_{\substack{\theta\in\mathbb{F}_{p^m}\\ \theta\neq\beta}}\chi_3(\theta)\bar{\chi}_3(\beta-\theta)A(1)\mathbf{X}^{\beta}=\sum_{\beta\in\mathbb{F}_{p^m}}\sum_{\theta\in\mathbb{F}_{p^m}}\chi_3(\theta)\bar{\chi}_3(\beta-\theta)A(1)\mathbf{X}^{\beta}=A(1)(p^m-\Phi(\mathbf{X}))$$
since the restriction $\theta\neq\beta$ can be removed and the summation over θ is -1 if $\beta\neq 0$ or p^m-1 if $\beta=0$.

In conclusion, collecting the results we obtain

$$\sigma_3 = \frac{1}{27}\left((\Phi(\mathbf{X})-1)^3 - 3(\Phi(\mathbf{X})-1)(p^m-\Phi(\mathbf{X})) + [J(\chi_3,\chi_3)+\bar{J}(\chi_3,\chi_3)](p^m-\Phi(\mathbf{X}))\right)$$

□

Remark 8.4. *Note that $J(\chi_3,\chi_3)+\bar{J}(\chi_3,\chi_3)$ is always an integer, being twice the sum of real parts of cubic roots of unity.*

Remark 8.5. *Even though its derivation involved handling products of three characters, the expression of σ_3 only involves the Jacobi sum $A(1)$, i.e. fundamentally only the number of representations as the sum of two elements of two given cosets.*

8.4 Connections with other problems

In this section we point out a sort of duality relationship with the following problem.

Suppose that we have t elements of a finite field $\mathbb{F}_{p^m}$ all belonging to one of the cosets determined by the character partition. We would like to know how many βs there are in the field such that, adding β to all the t elements, we get t elements still belonging to a common coset. If the character has order n, we let $N_{p^m}^{(n)}(t)$ be the number of βs; i.e. it is the number of solutions β of a system of t equations in $\mathbb{F}_{p^m}$ of the form

$$\begin{cases} \alpha^j z_1^n + \beta = \alpha^k y_1^n \\ \alpha^j z_2^n + \beta = \alpha^k y_2^n \\ \quad\vdots \\ \alpha^j z_t^n + \beta = \alpha^k y_t^n \end{cases} \tag{8.10}$$

where $\alpha^j z_1^n, \alpha^j z_2^n, \cdots, \alpha^j z_t^n$ are given and distinct, α being a primitive element, whereas the elements y_is must be chosen in the field to satisfy the system, and the n values $\{0,1,\ldots,n-1\}$ for k and j are all considered. However, we may assume $j=0$, since dividing each equation by α^j, and setting $\beta' = \beta\alpha^{-j}$ and $k' = k-j \bmod n$, we see that the number of solutions of the system is independent of j.

An explicit solution when the character is quadratic or cubic can be obtained, again by means of the indicator functions. For example, if we have a cubic character over $\mathbb{F}_{2^m}$, given a z_i we can partition the elements $\beta \neq z_i^3$ in $\mathbb{F}_{2^m}$ into subsets depending on the $k \in \{0,1,2\}$ such that $\chi(\beta+z_i^3)=\omega^k$. Therefore, a solution of (8.10) for a fixed k and $j=0$ is singled out by the product

$$\prod_{i=1}^{t} I_{\mathcal{A}_k}(\beta+z_i^3) = \frac{1}{3^t}[1+\sum_{i=1}^{t}\sigma_i^{(k)}] \ ,$$

where each $\sigma_i^{(k)}$ is a homogeneous sum of monomials which are products of i characters of the form $\chi(\beta+z_h^3)$ or $\bar{\chi}(\beta+z_h^3)$. Thus $N_{2^m}^{(3)}(t)$ is

$$N_{2^m}^{(3)}(t) = \sum_{\substack{\beta\in\mathbb{F}_{2^m} \\ \beta\notin\{z_i^3\}}} \left[\prod_{i=1}^{t} I_{\mathcal{A}_0}(\beta+z_i^3) + \prod_{i=1}^{t} I_{\mathcal{A}_1}(\beta+z_i^3) + \prod_{i=1}^{t} I_{\mathcal{A}_2}(\beta+z_i^3)\right] \ . \tag{8.11}$$

The z_i are excluded from the sum, since $z_i^3 + z_i^3 = 0$ does not belong to any coset.

In Chapter 3 or in [41], which deal with this problem to analyse the success rate of the Cantor-Zassenhaus polynomial factorization algorithm, we computed exactly some of the above expressions for small values of t, and gave bounds for more general cases. In particular, we found that

$$1 + \max_{z_1 \neq z_2 \neq z_3} N_{2^m}^{(3)}(3) = \begin{cases} \frac{1}{9}(2^m + 2^{m/2} - 2) & \text{for } m/2 \text{ even} \\ \frac{1}{9}(2^m + 2^{m/2+1} + 1) & \text{for } m/2 \text{ odd} \end{cases}$$

and

$$1 + \max_{z_1 \neq z_2 \neq z_3} N_{p^m}^{(2)}(3) = \begin{cases} \frac{1}{4}(p^m - 1) & \text{for } p = 4k+1 \\ \frac{1}{4}(p^m + 1) & \text{for } p = 4k+3, \ m \text{ odd} \\ \frac{1}{4}(p^m - 1) & \text{for } p = 4k+3, \ m \text{ even} \end{cases} .$$

Recalling that $R_{p^m}^{(n)}(\beta, i, j)$, $n = 2, 3$, denotes the number of representations of a $\beta \neq 0$ in a finite field $\mathbb{F}_{p^m}$ as the sum of two element belonging to two cosets indexed by i and j in the partition given by a character of order n, we find the remarkable identities:

$$\max_{i,j,\beta} R_{2^m}^{(3)}(\beta, i, j) = 1 + \max_{z_1 \neq z_2 \neq z_3} N_{2^m}^{(3)}(3)$$

and

$$\max_{i,j,\beta} R_{p^m}^{(2)}(\beta, i, j) = 1 + \max_{z_1 \neq z_2 \neq z_3} N_{p^m}^{(2)}(3) \ .$$

In the polynomial factorization context of Chapter 3 this quasi-duality had the following interesting interpretation: the maximum t such that it is still possible to fail to split a polynomial of degree t with two attempts is equal to the maximum number of attempts to split a polynomial of degree 3.

Bibliography

[1] S.D. Adhikari, The early reciprocity laws: from Gauss to Eisenstein, *Cyclotomic fields and related topics*, Bhaskaracharya Pratishthana, 2000, pp. 55-74.

[2] E. Artin, *Galois Theory*, Notre Dame University, 1959.

[3] E. Bach, Realistic analysis of some randomized algorithms, *J. Comput. System Sci.*, Vol. 42, 1991, pp. 30-53.

[4] E. Bach, J. Shallit, *Algorithmic Number Theory*, Vol. 1, M.I.T. Press, 1996.

[5] E. Bach, V. Shoup, Factoring polynomials using fewer random bits, *J. Symbolic Comput.*, Vol. 9, 1990, pp.229-239.

[6] M. Baldi, M. Bodrato, F. Chiaraluce, A new analysis of the McEliece cryptosystem based on QC-LDPC codes, in: *Security and Cryptography for Networks*, LNCS, Springer, Vol. 5229, 2008, pp. 246-262.

[7] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, Enhanced public key security for the McEliece cryptosystem, *arXiv:math.IT/1108.2462*, 2011.

[8] M. Baldi, M. Bianchi, F. Chiaraluce, J. Rosenthal, D. Schipani, On fuzzy syndrome hashing with LDPC coding. Proceedings 4th International Symposium on Applied Sciences in Biomedical and Communication Technologies (ISABEL), ACM 2011, pp. 1-5.

[9] L. Bazzi, T.J. Richardson, R.L. Urbanke, Exact thresholds and optimal codes for the binary-symmetric channel and Gallager's decoding algorithm A, *IEEE Trans. Inform. Theory*, Vol. 50(9), 2004, pp. 2010-2021.

[10] A.T. Benjamin, J.N. Scott, Third and Fourth Binomial Coefficients, *Fibonacci Quart.*, Vol. 49(2), 2011, pp. 99-101.

[11] A.T. Benjamin, B. Chen, K. Tucker, Sums of evenly spaced binomial coefficients, *Math. Mag.*, Vol. 83, 2010, pp. 370-373.

[12] M. Ben-Or, Probabilistic Algorithms in Finite Fields, *Proc. 22nd Annual IEEE Symp. Foundations of Computer Science (FOCS'1981)*, 1981, pp. 394-398.

[13] T.P. Berger, P. Loidreau, How to mask the structure of codes for a cryptographic use. *Designs, Codes and Cryptography*, Vol. 35, 2005, pp. 63-79.

[14] E. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, 1968.

[15] E. Berlekamp, R. McEliece, H. van Tilborg, On the inherent intractability of certain coding problems. *IEEE Trans. on Inform. Theory*, Vol. 24(3), 1978, pp. 384-386.

[16] B. Berndt, R.J. Evans, H. Williams, *Gauss and Jacobi Sums*, Wiley, 1998.

[17] D.J. Bernstein, T. Lange, C. Peters, Attacking and defending the McEliece cryptosystem, in: *Post-Quantum Cryptography*, LNCS, Springer, Vol. 5299, 2008, pp. 31-46.

[18] R.E. Blahut, *Theory and Practice of Error Control Codes*, Addison-Wesley, 1983.

[19] R.E. Blahut, *Algebraic Codes for Data Transmission*, Cambridge Univ. Press, Cambridge, 2003.

[20] A. Borodin, I. Munro, *The computational complexity of algebraic and numeric problems*, Elsevier, 1975.

[21] X. Boyen, Reusable cryptographic fuzzy extractors, *ACM Conference on Computer and Communications Security*, 2004, pp. 82-91.

[22] X. Boyen, Y. Dodis, J. Katz, R. Ostrovsky, A. Smith, Secure remote authentication using biometric data, in: *Advances in Cryptology-EUROCRYPT 2005*, LNCS, Springer, 2004, pp. 147-163.

[23] J. Bringer, H. Chabanne, G. Cohen, B. Kindarji, G. Zemor, Optimal iris fuzzy sketches, *Proc. IEEE BTAS*, 2007.

[24] I. Buhan, J. Doumen, P. Hartel, Controlling leakage of biometric information using dithering, *Proc. EUSIPCO*, 2008.

[25] W.S. Burnside, A.W. Panton, *The theory of equations with an introduction to the theory of binary quadratic forms*, Dover, 1960.

[26] D.G. Cantor, H. Zassenhaus, A new Algorithm for Factoring Polynomials over Finite Fields, *Math. Comp.*, Vol. 36, N. 154, April 1981, pp. 587-592.

[27] Center for Biometrics and Security Research, Institute of Automation, Chinese Academy of Sciences, CASIA V.1 Iris Database, 2011, Available: *http://www.cbsr.ia.ac.cn/english/IrisDatabase.asp*.

[28] H.H. Chan, L. Long, Y.F. Yang, A Cubic Analogue of the Jacobsthal Identity, *Amer. Math. Monthly*, Vol. 116(4), 2011, pp. 316-326.

[29] E. Costa, S.V. Fedorenko, P.V. Trifonov, On computing the syndrome polynomial in Reed-Solomon decoder, *European Trans. on Telecommunications*, vol. 15(4), 2004, pp. 337-342.

[30] T. Cover, J. Thomas, *Elements of information theory*, Wiley, 1991.

[31] D.A. Cox, *Galois Theory*, Wiley, 2004.

[32] J. Daugmann, How iris recognition works, *IEEE Trans. Circuits Syst. Video Technol.*, Vol. 14(1), 2004, pp. 21-30.

[33] R. Dedekind, *Theory of Algebraic Numbers*, Cambridge, 1996.

[34] R. Denomme, A History of Stickelberger's Theorem, *Senior Honors Thesis*, The Ohio State University, 2009.

[35] L.E. Dickson, *Algebras and their Arithmetics*, Dover, 1960.

[36] Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, *SIAM J. Comp.*, Vol. 38(1), 2008, pp. 97-139.

[37] Y. Dodis, A. Smith, Correcting errors without leaking partial information, *Proc. ACM STOC 2005*, 2005, pp. 654-663.

[38] M. Elia, Algebraic Decoding of the $(23, 12, 7)$ Golay Code *IEEE Trans. on Inform. Theory*, Vol. IT-33(1), 1981, pp. 150-151.

[39] M. Elia, M. Leone, On the Inherent Space Complexity of Fast Parallel Multipliers for $GF(2^m)$, *IEEE Trans. on Computer*, Vol. 51(3), March 2002, pp. 346-351.

[40] M. Elia, J. Rosenthal, D. Schipani, Polynomial evaluation over finite fields: new algorithms and complexity bounds, *Applicable Algebra in Engineering, Communication and Computing*, 2011, to appear (already available online).

[41] M. Elia, D. Schipani, Improvements on Cantor-Zassenhaus Factorization Algorithm, *arXiv:math.NT/1012.5322*, 2011.

[42] G.-L. Feng, K.K. Tzeng, Decoding cyclic and BCH codes up to actual minimum distance using nonrecurrent syndrome dependence relations, *IEEE Trans. on Inform. Theory*, IT-37(6), 1991, pp. 1716-1723.

[43] P. Fitzpatrick, On the key equation, *IEEE Trans. on Inform. Theory*, Vol. IT-41(5), September 1995, pp. 1290-1302.

[44] N. Frykholm, A. Juels. Error-tolerant password recovery, *Proceedings of the 8th ACM conference on Computer and Communications Security*, 2001, pp 1-9.

[45] E.M. Gabidulin, O. Kjelsen, How to avoid the Sidel'nikov-Shestakov attack, in: *Error Control, Cryptology, and Speech Compression*, LNCS, Springer, Vol. 829, 1994, pp. 25-32.

[46] E.M. Gabidulin, A.V. Paramonov, O.V. Tretjakov, Ideals over a non-commutative ring and their application in cryptography, in: *D.W. Davies, Ed., Advances in Cryptology - EUROCRYPT 91*, LNCS, Springer, Vol. 547, 1991.

[47] R.G. Gallager. *Low-density parity-check codes*, M.I.T. Press, 1963.

[48] P. Garrett, *Kummer, Eisenstein, computing Gauss sums as Lagrange resolvents*, http://www.math.umn.edu/~garrett/m/v/kummer_eis.pdf, 2010.

[49] J. von zur Gathen, J. Gerhard, *Modern Computer Algebra*, Cambridge Univ. Press, 1999.

[50] C.F. Gauss, *Disquisitiones Arithmeticae*, Springer, 1966.

[51] S.W. Golomb, *Shift Register Sequences*, Aegean Park Press, 1982.

[52] H.W. Gould, *Combinatorial Identities*, Morgantown Printing and Binding Co., 1972.

[53] E. Guerrini, A. Rimoldi, FGLM-like decoding: from Fitzpatrick's approach to recent developments, in: *M. Sala et al. (eds.) Gröbner Bases, Coding, and Cryptography*, RISC, Springer, 2009, pp. 197-218.

[54] J. Hagenauer, E. Offer, L. Papke, Iterative decoding of binary block and convolutional codes, *IEEE Trans. Inform. Theory*, Vol. 42(2), 1996, pp. 429-445.

[55] F. Hao, R. Anderson, J. Daugman, Combining cryptography with biometrics effectively, *Technical Report 640*, University of Cambridge-Computer Laboratory, 2005.

[56] G.H. Hardy, E.M. Wright, *An introduction to the theory of numbers*, Oxford University Press, 2008.

[57] J. Hong, M. Vetterli, Simple Algorithms for BCH Decoding, *IEEE Trans. on Communications*, Vol. 43(8), 1995, pp. 2324-2333.

[58] X. Y. Hu, E. Eleftheriou, Progressive edge-growth Tanner graphs, *Proc. IEEE GLOBECOM'01*, 2001, pp. 995-1001.

[59] J.C. Interlando, E. Byrne, J. Rosenthal, The Gate Complexity of Syndrome Decoding of Hamming Codes, *Proc. 10th Int. Conf. on Applications of Computer Algebra*, 2004, pp. 33-37.

[60] K. Ireland, M. Rosen, *A Classical Introduction to Modern Number Theory*, Springer, 1990.

[61] A. Juels, M. Wattenberg, A fuzzy commitment scheme, *Proc. 6th ACM Conference on Computer and Communications Security*, 1999, pp. 28-36.

[62] A. Juels, M. Sudan, A fuzzy vault scheme, *Designs, Codes and Cryptography*, Vol. 38(2), 2006, pp. 237-257.

[63] D. Jungnickel, *Finite Fields, Structure and Arithmetics*, Wissenshaftsverlag, 1993.

[64] G. Kabatiansky, E. Krouk, S. Semenov, *Error Correcting Coding and Security for Data Networks: Analysis of the Superchannel Concept*, John Wiley & Sons, 2005.

[65] S. Kamara, B. Medeiros, S. Wetzel, Secret locking: Exploring new approaches to biometric key encapsulation, *Proceedings of the 2nd International Conference on e-Business and Telecommunications*, 2005.

[66] S.A. Katre, Gauss-Jacobi sums and Stickelberger's theorem, *Cyclotomic fields and related topics*, Bhaskaracharya Pratishthana, 2000, pp. 75-92.

[67] D.E. Knuth, *The Art of Computer Programming*, Seminumerical algorithms, Vol. II, Addison-Wesley, 1981.

[68] R. Lidl, H. Niederreiter, *Finite Fields*, Cambridge Univ. Press, 1997.

[69] F.J. MacWilliams, N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North Holland, 1977.

[70] L. Masek, *Recognition of human iris patterns for biometric identification*, School of Computer and Software Engineering, University of Western Australia, 2003.

[71] L. Masek and P. Kovesi, *Matlab source code for a biometric identification system based on iris patterns*, School of Computer Science and Software Engineering, University of Western Australia, 2003. Available: *http://www.csse.uwa.edu.au/ pk/studentprojects/libor/sourcecode.html*.

[72] J.L. Massey, Shift-Register Synthesis and BCH decoding, *IEEE Trans. on Inform. Theory*, IT-15, 1969, pp. 122-127.

[73] G.B. Mathews, *Theory of numbers*, Chelsea Pub. Co., 1980.

[74] R.J. McEliece, A public-key cryptosystem based on algebraic coding theory, *DSN Progress Report*, 1978, pp. 114-116.

[75] R.J. McEliece, *Finite Fields for Computer Scientists and Engineers*, Kluwer Academic Press, 1987.

[76] N. Miladinovic, M. P. C. Fossorier, Improved bit-flipping decoding of low-density parity-check codes, *IEEE Trans. Inform. Theory*, Vol. 51(4), 2005, pp. 1594-1606.

[77] L. Minder, *Cryptography based on error correcting codes*, Ph.D. thesis, École Polytechnique Fédérale de Lausanne, 2007.

[78] R.A. Mollin, *Advanced Number Theory with Applications*, CRC Press, 2010.

[79] C. Monico, J. Rosenthal, A. Shokrollahi, Using low density parity check codes in the McEliece cryptosystem, in: *Proc. IEEE Int. Symp. on Inform. Theory*, 2000, p. 215.

[80] C. Monico, M. Elia, Note on an Additive Characterization of Quadratic Residues Modulo p, *J. Comb. Inf. Syst. Sci.*, Vol. 31, 2006, pp. 209-215.

[81] C. Monico, M. Elia, An Additive Characterization of Fibers of Characters on $\mathbb{F}_p^*$, *Int. J. Algebra*, Vol. 4(3), 2010, pp. 109-117.

[82] T. Mora, E. Orsini, Decoding cyclic codes: the Cooper philosophy, in: *M. Sala et al. (eds) Gröbner Bases, Coding, and Cryptography*, , RISC, Springer, 2009, pp. 69-91.

[83] H. Niederreiter, Knapsack-type cryptosystems and algebraic coding theory. *Probl. Contr. and Inform. Theory*, Vol. 15, 1986, pp. 159-166.

[84] E. Orsini, M. Sala, Correcting errors and erasures via the syndrome variety, *J. Pure Appl. Algebra*, vol. 200, 2005, pp. 191-226.

[85] A. Otmani, J.P. Tillich, L. Dallot, Cryptanalysis of two McEliece cryptosystems based on quasi-cyclic codes, in: *Proc. First International Conference on Symbolic Computation and Cryptography*, 2008.

[86] A. Ourivski, E.M. Gabidulin, Column scrambler for the GPT cryptosystem. *Discrete Applied Mathematics*. Vol. 128, 2003, pp. 207-221.

[87] R. Overbeck, Structural attacks for public key cryptosystems based on Gabidulin codes, *Journal of Cryptology*, Vol. 21(2), 2008, pp. 280-301.

[88] R.E.A.C. Paley, "On orthogonal matrices". J. Math. Phys. 12, 1933, pp. 311-320.

[89] V.Y. Pan, Methods of computing values of polynomials, *Uspekhi Mat. Nauk*, Vol. 21, 1966, pp. 103-134.

[90] M. Paterson, L. Stockmeyer, On the number of nonscalar multiplications necessary to evaluate polynomials, *SIAM J. Computing*, Vol. 2, 1973, pp. 60-66.

[91] O. Perron, Bemerkungen uber die Verteilung der quadratischen Reste, *Mathematische Zeitschrift*, Vol. 56, 1952, pp. 122-130.

[92] C. Peters, *http://www.win.tue.nl/ cpeters/isdfq.html*, 2010.

[93] C. Peters, Information-set decoding for linear codes over $\mathbb{F}_q$, in: *Sendrier, N. (ed.) Post-Quantum Cryptography*, LNCS, Springer, vol. 6061, 2010, pp. 81-94.

[94] W.W. Peterson, E.J. Weldon, *Error-Correcting Codes*, M.I.T Press, 1981.

[95] V.S. Pless, W.C. Huffman, *Handbook of Coding Theory*, vol. I and II, Noth-Holland, 1998.

[96] M.O. Rabin, Probabilistic algorithms in finite fields, *SIAM J. Comput.*, Vol. 9, 1980, pp. 273-280.

[97] H. Rashwan, E.M. Gabidulin, B. Honary, Security of the GPT cryptosystem and its applications to cryptography, *Security Comm. Networks*, 2010.

[98] D. Raymond, *An Additive Characterization of Quadratic Residues*, Master Degree thesis, Texas Tech University, 2009.

[99] Reed, I.S., Truong, T.K., Chen, X., Yin, The algebraic decoding of the $(41, 21, 9)$ quadratic residue code *IEEE Trans. on Information Theory*, IT-38(3), 1992, pp. 974-986.

[100] T.J. Richardson, R.L. Urbanke, The capacity of low-density parity-check codes under message-passing decoding, *IEEE Trans. Inform. Theory*, Vol. 47, 2001, pp. 599-618.

[101] D.V. Sarwate, Semi-Fast Fourier Transforms over $GF(2^m)$, *IEEE Transactions on Computers*, Vol. C-27(3), 1978, pp. 283-285.

[102] D. Schipani, M. Elia, Gauss Sums of the Cubic Character over $GF(2^m)$: an elementary derivation, *Bull. Pol. Acad. Sci. Math.*, Vol. 59(1), 2011, pp. 11-18.

[103] D. Schipani, M. Elia, Gauss sums of cubic characters over $\mathbb{F}_{p^r}$, p odd, *Bull. Polish Acad. Sci. Math.*, Vol. 60(1), 2012, pp. 1-19.

[104] D. Schipani, M. Elia, Additive decompositions induced by multiplicative characters over finite fields, *arXiv:math.NT/1107.1364*, 2011.

[105] D. Schipani, M. Elia, J. Rosenthal, Efficient evaluation of polynomials over finite fields, *Proc. 2011 Australian Communications Theory Workshop*, 2011, pp. 154-157.

[106] D. Schipani, M. Elia, J. Rosenthal, On the Decoding Complexity of Cyclic Codes up to the BCH Bound, *Proc. Int. Symp. on Information Theory*, 2011, pp.835-839.

[107] D. Schipani, J. Rosenthal, Coding solutions for the secure biometric storage problem, *Proc. of IEEE Inform. Theory Workshop*, 2010.

[108] W.M. Schmidt, *Equations over Finite Fields: An Elementary Approach* , LNM, Springer, Vol. 536, 1975.

[109] D. Shanks, Class number, a theory of factorization and genera, *Proc. Symp. Pure Math.*, AMS, N.20, 1971, pp.415-440.

[110] R. Schoof, Elliptic curves over finite fields and the computation of square roots mod p, *Math. Comp.*, Vol. 44, N. 170, 1985, pp. 483-494.

[111] V. Shoup, On the deterministic complexity of factoring polynomials over finite fields, *Inform. Process. Lett.*, Vol. 33, 1990, pp. 261-267.

[112] V. M. Sidel'nikov, S. O. Shestakov, On insecurity of cryptosystems based on generalized Reed-Solomon codes, *Discrete Mathematics and Applications* Vol. 2(4), 1992, pp. 439-444.

[113] N.J.A. Sloane, *The On-Line Encyclopedia of Integer Sequences^{TM} (OEIS^{TM})*.

[114] L. Stickelberger, Ueber eine Verallgemeinerung der Kreistheilung, *Math. Ann.*, XXXVII Band, 3 Heft, 1890, pp. 321-367.

[115] Y. Sutcu, S. Rane, J.S. Yedidia, S.C. Draper, A. Vetro, Feature extraction for a Slepian-Wolf biometric system using LDPC codes. *Proc. IEEE Int. Symp. on Inform. Theory*, 2008, pp. 2297-2301.

[116] P. Tuyls, B. Skoric,T. Kevenaar (eds.), *Security with noisy data*, Springer, 2007.

[117] U. Uludag, S. Pankanti, S. Prabhakar, A.K. Jain, Biometric cryptosystems: Issues and challenges, in: *Proceedings of the IEEE*, Vol. 92(6), 2004, pp. 948-960.

[118] U. Uludag, A.K. Jain, Securing fingerprint template: Fuzzy vault with helper data, *Computer Vision and Pattern Recognition Workshop*, 2006, p. 163.

[119] V.G. Umana, G. Leander, Practical key recovery attacks on two McEliece variants, in: *C. Cid, J.C. Faugere (eds.) Proc. 2nd Int. Conf. on Symbolic Computation and Cryptography*, 2010, pp. 27-44.

[120] D. Wan, Genarators and irreducible polynomials over finite fields, *Math. Comp.*, Vol. 66, N. 219, 1997, pp. 1195-1212.

[121] L.C. Washington, *Introduction to Cyclotomic Fields*, Springer, 1997.

[122] S.B. Wicker, *Error control systems for Digital Communication and Storage*, Prentice-Hall, 1995.

[123] S.B. Wicker, V.K. Bhargava, eds. *Reed-Solomon Codes and their Applications*, IEEE Press, 1994.

[124] C. Wieschebrink, Cryptanalysis of the Niederreiter public key scheme based on GRS subcodes, in: *Sendrier, N. (ed.) Post-Quantum Cryptography: PQCrypto 2010*, LNCS, Springer, Vol. 6061, 2010, pp. 61-72.

[125] S. Winograd, On the number of multiplications required to compute certain functions, *Proc. Natl. Acad. Sci. U.S.A.*, Vol. 58(5), 1967, pp. 1840-1842.

[126] A. Winterhof, On the Distribution of Powers in Finite Fields, *Finite Fields Appl.*, Vol. 4, 1998, pp. 43-54.

[127] A. Winterhof, Character sums, primitive elements, and powers in finite fields, *J. Number Theory*, Vol. 91, 2001, pp. 153-163.

[128] S. Yang, I.M. Verbauwhede, Secure fuzzy vault based fingerprint verification system, *Conference Record of the Thirty-Eighth Asilomar Conference on Signals, Systems and Computers*, Vol. 1, 2004, pp. 577-581.