The Exact Repair Problem

-Using Reed-Solomon Codes-

Master Thesis

Jonas Junker Jensen

Aalborg University
Department of Mathematics

Department of Mathematics
Aalborg University
Skjernvej 4, 9220 Aalborg Ø
http://www.aau.dk

AALBORG UNIVERSITY
STUDENT REPORT

Title:
The Exact Repair Problem

Theme:
Using Reed-Solomon Codes

Project Period:
Spring Semester 2019

Project Group:
10^{th} semester, 5.213b

Participant:
Jonas Junker Jensen

Supervisors:
Ignacio Cascudo Pueyo
Oliver Wilhelm Gnilke

Copies: 2

Page Numbers: 45

Date of Completion:
June 6, 2019

Abstract:

Technology has grown over the years, resulting in more stress on data storage servers. As the years goes by the data files size also grow, resulting in bigger files than a server can handle. In case the server crashes the data file is lost. Hence data distribution storage (DDS) systems are initiated, meaning the file is distributed across multiple servers. This indicated that error-correction codes could be used, as if one server crashes it can be reconstructed. However, for error-correction algorithms they reconstruct the entire codeword instead of the one broken entry. This is related to the exact repair problem (ERP) where this thesis will study the ERP for Reed-Solomon (RS) codes over extension fields. If the data is distributed as a RS codeword, then by solving the ERP then any data on a crashed server can be restored. This process is called a linear exact repair scheme. The approach in this project will combine Galois theory, coding theory and linear algebra to create an optimal repair scheme.

Contents

Preface

This master thesis is made during the 4^{th} semester of the master program in mathematics at Aalborg university spring 2019. The aim of this master thesis is to study the exact repair problem for Reed-Solomon codes. For this we need to understand the problem, and the problems which may occur in them. We need to understand how this is related to Distributed Storage Systems and how it is possible to restore any data on a failed server.
For this to make sense some basic knowledge about finite fields, coding theory, linear algebra and polynomial algebra is recommended. Important definitions will be stated in Chapter 1 for reference as well with some results needed later.

Reading Guide

In this project chapters are numbered x and sections are numbered x.x, where the first x is the number of the chapter. Subsections are numbered x.x.x, where the first and the second x indicates the chapter and the section, respectively.

Definitions, lemmas, propositions, remarks, examples and theorems are all numbered x.y where x indicate the chapter and y indicate the number in chapter x. Definitions, lemmas, proposition and theorems are written in light grey colored boxes with darker grey lines, which indicate the beginning and the ending of each. Definitions are also written with a heading of the content after the number of the definition. Proofs are written with a bold "Proof" in the beginning and end with a square. Sage code have a line at the beginning and the end, and the sage code has a 'sage:' in front of the code, and the output with nothing in front. The lines in the code are counted to the left.

Mathematical equations are centered and written in italic. If they are used later in the report, they are numbered (x.x) in the right margin, where the first x indicate the number of the chapter. Equations are referred to as (x.x).

The sources the chapter/sections/subsections are based on are written in the beginning of the chapter/section/subsection and are labeled [x] where the x refer to the x'th source in the Bibliography.

Aalborg University, June 6, 2019

Jonas Junker Jensen
<jjje14@student.aau.dk>

Introduction

As the world becomes more technologically advanced, we require more of our computers. A computer usually have one or two hard drives, which when burns out looses the data on it. What if both hard drives had half of every file stored? Then, in the case either of them crashing, only lose half of every file would be lost. This is the basic motivation behind the following idea to prevent complete loss of data.
Say a big file was distributed across multiple servers, this is called a distributed storage system. Instead of having one file on one server, which if crashes we lose the file, we now have the file split up into parts where multiple servers store a part of the file. If this is done in a smart way then from enough parts of the file, we can restore the original file. This way, even if one servers crashes we do not necessarily lose the file.
From coding theory the concept of MDS codes has been used to encode a message of length k into a codeword of length n, such that any k correct entries can reconstruct the entire codeword. However for certain situations when only one entry is broken, it seems to be wasteful to restore the entire codeword. This is where the title of the project comes in: "The Exact Repair Problem" ie. the problem of repairing a single entry from the remaining correct entries without reconstructing the entire codeword. This way when a server crashes, we do not have to reconstruct the entire codeword. Instead by a solution to the exact repair problem, we can reconstruct only the damaged entry.
When a server crashes that means there is the need of a new server, which we call a replacement server to the crashed server. Then if the file distributed to the servers by a MDS code, the exact repair problem then becomes to restore the part of the file that the crashed server stored, from the remaining working servers. Then the replacement server download it and then act as the crashed servers replacement.
Hence the replacement server will have to download information from the other servers, which also yields a new parameter: how much does the replacement server need to download from the other servers? This is called repair bandwidth. It turns out with Reed-Solomon codes over extensions fields it is enough to download some data from the subfield, where the server can compute the data itself under certain conditions on the set up.
Reed-Solomon codes are often used, but there was no working exact repair scheme. However [1] present a working scheme which we will look into. In [8] they show that there are a limit on how well you can do in the sense of how much each server has to storage and how much a replacement server has to download. [1] present an optimal scheme with minimal repair bandwidth which will be the focus of this thesis.

Chapter 1

Preliminaries

In this chapter, the theory which is required for the main subject, ie. the exact repair problem, will be explained and defined. Proofs will be given if it is not something we have done before.

1.1 Galois Theory

This section is based on [2] and [3].
The purpose of this section is to introduce some of the concept from Galois theory as they will come into play later.
The first definition we need is the definition of an automorphism.

Definition 1.1 (Automorphism)**.**
Let G be a group and let $\phi : G \to G$ be a group isomorphism. Then ϕ is a group automorphism.
Let $(R, +, *)$ be a ring and let $\mu : R \to R$ be a ring isomorphism. Then μ is a ring automorphism.

The usual definition of a field extension is used, where $\mathbb{E}$ is an extension field over $\mathbb{F}$ if $\mathbb{F} \subseteq \mathbb{E}$ (usually written $\mathbb{F} \leq \mathbb{E}$).

Definition 1.2 (Algebraic extension)**.**
Let $\mathbb{E}$ be an extension of $\mathbb{F}$. Then:

1) $\alpha \in \mathbb{E}$ is algebraic over $\mathbb{F}$ if α is root of some polynomial $f(x) \in \mathbb{F}[X]$.

2) $\mathbb{E}$ is an algebraic extension of $\mathbb{F}$ if all $\alpha \in \mathbb{E}$ is algebraic over $\mathbb{F}$.

Now we can define the Galois group which is a group of automorphisms.

Definition 1.3 (Galois group)**.**
Let $\mathbb{E}$ be an algebraic extension of $\mathbb{F}$. The Galois group, denoted as $(Gal(\mathbb{E}/\mathbb{F}), \circ)$ is the group of all automorphisms of $\mathbb{E}$ that restricts to the identity on $\mathbb{F}$:

$$Gal(\mathbb{E}/\mathbb{F}) = \{\sigma : \mathbb{E} \to \mathbb{E} | \sigma \text{ being an automorphism}, \sigma(a) = a, \ \forall a \in \mathbb{F}\}$$

where $\circ$ denote function composition.

$Gal(\mathbb{E}/\mathbb{F})$ is technically a set, but with function composition it can be shown to be a group. Let $\sigma_1, \sigma_2, \sigma_3 \in Gal(\mathbb{E}/\mathbb{F})$, and $\alpha, \beta \in \mathbb{E}$, then the following holds:

$$(\sigma_1 \circ \sigma_2)(\alpha\beta) = \sigma_1(\sigma_2(\alpha\beta)) = \sigma_1(\sigma_2(\alpha)\sigma_2(\beta))$$
$$= \sigma_1(\sigma_2(\alpha))\sigma_1(\sigma_2(\beta)) = (\sigma_1 \circ \sigma_2)(\alpha)(\sigma_1 \circ \sigma_2)(\beta)$$

since automorphism by definition is isomorphisms (which are homomorphism and bijective). This means that a composition of automorphisms is a group homomorphism. We also know that $\sigma_1 \circ \sigma_2$ has an inverse as σ_1, σ_2 does, meaning that a composition of automorphisms is an automorphism.
Now to show that $(Gal(\mathbb{E}/\mathbb{F}), \circ)$ is a group by showing it satisfy the 3 requirements:

- Associativity: this follows from functions composition:

  $$((\sigma_1 \circ \sigma_2) \circ \sigma_3)(\alpha) = (\sigma_1 \circ \sigma_2)(\sigma_3(\alpha)) = \sigma_1(\sigma_2(\sigma_3(\alpha)))$$

  and the other side:

  $$(\sigma_1 \circ (\sigma_2 \circ \sigma_3))(\alpha) = \sigma_1((\sigma_2 \circ \sigma_3)(\alpha)) = \sigma_1(\sigma_2(\sigma_3(\alpha)))$$

- Neutral element: let $\sigma_{id} : \mathbb{E} \to \mathbb{E}$ where it takes $\sigma_{id}(\alpha) = \alpha$ for all $\alpha \in \mathbb{E}$. Then $\sigma \circ \sigma_{id} = \sigma_{id} \circ \sigma$ for any $\sigma \in Gal(\mathbb{E}/\mathbb{F})$ satisfying the neutral element.

- Inverse element: as bijection means there exists an inverse function, all we have to show is that the inverse is an homomorphism. Let $\sigma \in Gal(\mathbb{E}/\mathbb{F})$ and let $\alpha, \beta \in \mathbb{E}$ where $\sigma(x) = \alpha, \sigma(y) = \beta$

  $$\sigma^{-1}(\alpha\beta) = \sigma^{-1}(\sigma(x)\sigma(y)) = \sigma^{-1}(\sigma(xy)) = xy$$

  and

  $$\sigma^{-1}(\alpha)\sigma^{-1}(\beta) = \sigma^{-1}(\sigma(x))\sigma^{-1}(\sigma(y)) = xy$$

  meaning $\sigma^{-1}(ab) = \sigma^{-1}(a)\sigma^{-1}(b)$ satisfying the homomorphism. And as it has an inverse it is an automorphism and $\sigma^{-1} \in Gal(\mathbb{E}/\mathbb{F})$ for any $\sigma \in Gal(\mathbb{E}/\mathbb{F})$.

This shows that $(Gal(\mathbb{E}/\mathbb{F}), \circ)$ is a group.

Definition 1.4 (Fixed field)**.**
Let $(G, \circ)$ be a group of automorphisms of a field $\mathbb{E}$. Then the fixed field:

$$Fix(G) = \{\alpha \in \mathbb{E} | \sigma(\alpha) = \alpha, \ \forall \sigma \in G\}.$$

The first observation will be that, by Definition 1.3 then $\mathbb{F} \subseteq Fix(Gal(\mathbb{E}/\mathbb{F}))$ as by Definition 1.3 $\sigma(a) = a$ for all $a \in \mathbb{F}$. The case where $\mathbb{F} = Fix(Gal(\mathbb{E}/\mathbb{F}))$ has a name of its own.

Definition 1.5 (Galois extension)**.**
Let $\mathbb{E}$ be an algebraic extension of $\mathbb{F}$. Then $\mathbb{E}$ is a Galois extension of $\mathbb{F}$ if $Fix(Gal(\mathbb{E}/\mathbb{F})) = \mathbb{F}$.

Definition 1.6 (Galois Conjugates)**.**
Let $\mathbb{E}$ be an algebraic extension of $\mathbb{F}$ and let $\alpha \in \mathbb{E}$. Then $\{\sigma(\alpha)|\sigma \in Gal(\mathbb{E}/\mathbb{F})\}$ is the set of Galois conjugates of α in $\mathbb{E}$.

The following definition is the reason why Galois theory was introduced.

Definition 1.7 (Trace)**.**
Let $\mathbb{E}$ be finite Galois extension of $\mathbb{F}$ with Galois group $G = Gal(\mathbb{E}/\mathbb{F})$. For $\alpha \in \mathbb{E}$ the trace ($\mathbb{E}$ to $\mathbb{F}$) of α is defined as

$$tr_{\mathbb{E}/\mathbb{F}}(\alpha) = \sum_{\sigma \in G} \sigma(\alpha).$$

The trace from $\mathbb{E}$ to $\mathbb{F}$ yields an element in $\mathbb{F}$ which the following proposition will prove.

Proposition 1.8.
The trace function (from Definition 1.7) yields elements in $\mathbb{F}$.

Proof. By assumption $\mathbb{E}/\mathbb{F}$ is a Galois extension therefore what we need to show is

$$tr_{\mathbb{E}/\mathbb{F}}(\alpha) \in \mathbb{F} = Fix(Gal(\mathbb{E}/\mathbb{F}))$$

for any $\alpha \in \mathbb{E}$. By Definition 1.3 and 1.4 we have

$$Fix(Gal(\mathbb{E}/\mathbb{F})) = \{x \in \mathbb{E} | \forall \mu \in Gal(\mathbb{E}/\mathbb{F}), \mu(x) = x\}$$

and hence we want to verify that for all $\mu \in Gal(\mathbb{E}/\mathbb{F}), \alpha \in \mathbb{E}$ that

$$\mu(tr_{\mathbb{E}/\mathbb{F}}(\alpha)) = tr_{\mathbb{E}/\mathbb{F}}(\alpha)$$

as then $tr_{\mathbb{E}/\mathbb{F}}(\alpha) \in \mathbb{F}$. By Definition 1.7 we have

$$tr_{\mathbb{E}/\mathbb{F}}(\alpha) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha)$$

meaning that we can look at the Galois conjugates instead. Therefore we want to verify that for all $\mu \in Gal(\mathbb{E}/\mathbb{F})$ that

$$\mu\left(\sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha)\right) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha)$$

By using that $Gal(\mathbb{E}/\mathbb{F})$ consists of automorphisms which are (ring) homomorphisms that preserves addition, which used repeatedly allows

$$\mu\left(tr_{\mathbb{E}/\mathbb{F}}(\alpha)\right) = \mu\left(\sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha)\right) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \mu\left(\sigma(\alpha)\right) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} (\mu \circ \sigma)(\alpha)$$

for some fixed $\mu \in Gal(\mathbb{E}/\mathbb{F})$. Then it reduces into showing that

$$\sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \mu(\sigma(\alpha)) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha)$$

for all $\sigma \in Gal(\mathbb{E}/\mathbb{F})$ and some fixed $\mu \in Gal(\mathbb{E}/\mathbb{F})$. Therefore the question is for a fixed μ if

$$\{\mu \circ \sigma | \sigma \in Gal(\mathbb{E}/\mathbb{F})\} = Gal(\mathbb{E}/\mathbb{F}).$$

as then by Definition 1.3 then $(\mu \circ \sigma) \in Gal(\mathbb{E}/\mathbb{F})$ and then it has to satisfy $(\mu \circ \sigma)(\alpha) = \alpha$. This is done by showing both sets are contained in each other:

- $\subseteq$: this is because $Gal(\mathbb{E}/\mathbb{F})$ is a group.
- $\supseteq$: let $\theta \in Gal(\mathbb{E}/\mathbb{F})$, then i can set $\sigma = \mu^{-1} \circ \theta$ as μ is an automorphism and therefore has an inverse. And this can be done for all $\mu \in Gal(\mathbb{E}/\mathbb{F})$.

Therefore as

$$\mu\left(tr_{\mathbb{E}/\mathbb{F}}(x)\right) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} (\mu \circ \sigma)(x) = \sum_{\sigma \in Gal(\mathbb{E}/\mathbb{F})} \sigma(\alpha) = tr_{\mathbb{E}/\mathbb{F}}(x)$$

then it must be an element of $\mathbb{F}$ as the assumption was that $\mathbb{E}$ was an Galois extension of $\mathbb{F}$. □

Later we shall use the trace function as a map from $\mathbb{E}$ to $\mathbb{F}$. However the Galois group depends on the specific fields and we need the case where $\mathbb{E} = \mathbb{F}_{q^t}$ and $\mathbb{F} = \mathbb{F}_q$ for q being a prime power and $t \in \mathbb{N}$. Then $\mathbb{F}_{q^t}$ has order q^t. Lagrange theorem then gives that all $\alpha \in \mathbb{F}_{q^t}$ satisfy $\alpha^{q^t} = \alpha$. As $\mathbb{F}_q \leq \mathbb{F}_{q^t}$ the elements of $\mathbb{F}_q$ must also satisfy this, however Lagrange gives that for $a \in \mathbb{F}_q$ must have a similar relation ie. $a^q = a$ and therefore $a^{q^m} = a$ for all $1 \leq m \leq t$.

Definition 1.9 (Frobenius map)**.**
Let $\mathbb{F}$ be a field of characteristic p. The Frobenius map:

$$\Phi : \mathbb{F}_q \to \mathbb{F}_q$$

is the map defined by $\phi(x) = x^p$ where $q = p^t$ for $t \in \mathbb{N}$ and p being prime.

The Frobenius map is a map from $\mathbb{F}_q \to \mathbb{F}_q$, hence all left for it to be an automorphism is that it is an isomorphism.

- Homomorphism: let $x, y \in \mathbb{F}_q$, then

$$\Phi(x)\Phi(y) = x^p y^p = (xy)^p = \Phi(xy).$$

- Inverse: by Lagrange's theorem $\forall \alpha \in \mathbb{F}_q,\ \alpha^q = \alpha$, meaning that by using the Frobenius map t times, we have $\Phi(x)^t = x^{p^t} = x^q = x$ which then is an inverse. Meaning that $\Phi(x)^{t-1}\Phi(x) = x$ and therefore $\Phi^{t-1} = \Phi^{-1}$.

Therefore the Frobenius map is an automorphism for finite fields. This is going to be used to describe the Galois group as that now can be written as:

$$Gal(\mathbb{F}_{q^t}/\mathbb{F}_q) = \{(\Phi(x))^m = (x^q)^m | 0 \leq m \leq t-1\}$$

as $\alpha^{q^t} = \alpha$ for any $\alpha \in \mathbb{F}_{q^t}$. Then the Galois conjugates of $\alpha \in \mathbb{F}_{q^t}$ is the following:

$$\{(\Phi(\alpha))^m | 0 \leq m \leq t-1\}.$$

All this can be summed up into the following field trace which is the function we will be using later, hence we give it in a definition.

Definition 1.10.
Let $\mathbb{F}_{q^t}$ be an extension field of degree t and let $\mathbb{F}_q$ be the base field. The field trace is from $\mathbb{F}_{q^t}$ to $\mathbb{F}_q$ is then defined as:

$$tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta) = \beta + \beta^q + \beta^{q^2} + \cdots + \beta^{q^{t-1}}$$

for $\beta \in \mathbb{F}_{q^t}$.

The following proposition both shows $\mathbb{F}_q$ linearity and that the trace from $\mathbb{F}_{q^t}$ to $\mathbb{F}_q$ gives an elements in $\mathbb{F}_q$. The second part is Proposition 1.8 with an simpler argument for this specific trace.

Lemma 1.11.
$tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}$ is a $\mathbb{F}_q$-linear map and maps any element from $\mathbb{F}_{q^t}$ to an element in $\mathbb{F}_q$.

Proof. The first statement is done straight forward for $a_1, a_2 \in \mathbb{F}_q, \alpha_1, \alpha_2 \in \mathbb{F}_{q^t}$ as

$$\begin{aligned}&tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(a_1\alpha_1 + a_2\alpha_2) = \\ &(a_1\alpha_1 + a_2\alpha_2) + (a_1\alpha_1 + a_2\alpha_2)^q + \cdots + (a_1\alpha_1 + a_2\alpha_2)^{q^{t-1}}\end{aligned}$$

by then using the theorem known as "Freshman's Dream" on every single term followed by rearranging and the fact that $x^q = x$ for any $x \in \mathbb{F}_q$:

$$\begin{aligned}&(a_1\alpha_1) + (a_1\alpha_1)^q + \cdots + (a_1\alpha_1)^{q^{t-1}} + (a_2\alpha_2) + (a_2\alpha_2)^q + \cdots + (a_2\alpha_2)^{q^{t-1}} \\ &= a_1\alpha_1 + a_1^q\alpha_1^q + \cdots + a_1^{q^{t-1}}\alpha_1^{q^{t-1}} + a_2\alpha_2 + a_2^q\alpha_2^q + \cdots + a_2^{q^{t-1}}\alpha_2^{q^{t-1}} \\ &= a_1(\alpha_1 + \alpha_1^q + \cdots + \alpha_1^{q^{t-1}}) + a_2(\alpha_2 + \alpha_2^q + \cdots + \alpha_2^{q^{t-1}}) \\ &= a_1 tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha_1) + a_2 tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha_2).\end{aligned}$$

For the second statement we know that for any $x \in \mathbb{F}_{q^t}$ it satisfy $x^{q^t} = x$. Hence we check:

$$\begin{aligned}&(tr_{\mathbb{F}_{q^n}/\mathbb{F}_q}(\alpha))^q = (\alpha + \alpha^q + \cdots + \alpha^{q^{t-1}})^q \\ &= \alpha^q + \alpha^{q^2} + \cdots + \alpha^{q^t} = \alpha + \alpha^q + \cdots + \alpha^{q^{t-1}} \\ &= tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha)\end{aligned}$$

meaning that for any $\alpha \in \mathbb{F}_{q^t}$ the same holds: $(tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha))^q = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha)$ therefore the trace yields an element in $\mathbb{F}_q$. □

1.2 Linear Algebra

This section is based on [2], [4] and [5].
First for summary the definition of a vector space over a field.

Definition 1.12 (Vector Space)**.**
A vector space V over a field $\mathbb{F}$ is an abelian group $(V, +)$ with a neutral element 0 and a scalar multiplication $\mathbb{F} \times V \to V$ denoted as $(a, v) \to av$, more precise for all $a, b \in \mathbb{F}$ and $v, w \in V$:

- $(ab)v = a(bv)$

- $1v = v$
- $(a + b)v = av + bv$
- $a(v + w) = av + aw$

In linear algebra, when working with a vector space, the basis that generates the vector space is used to describe the vector space. A set of vectors that generates the vector space will come into play later. To define this, the definition of linearly independent is required.

Definition 1.13 (Linearly Independence)**.**
Let V be a vector space over a field $\mathbb{F}$. A set of vectors $v_1, \ldots, v_t \in V$ is called linearly independent if

$$a_1v_1 + a_2v_2 + \cdots + a_tv_t = 0$$

where $a_1, \ldots, a_t \in \mathbb{F}$ implies that $a_1 = \cdots = a_t = 0$.

For later we need the following lemma.

Lemma 1.14 (Steinitz Exchange Lemma)**.**
Let V be a vector space over a field $\mathbb{F}$. Let $w_1, \ldots, w_m$ be a linearly independent set of vectors and let $v_1, \ldots, v_t$ be a generating set for V. Then $m \leq t$ and $w_1, \ldots, w_m, v'_{m+1}, \ldots v'_t$ is a generating set for V where $v'_{m+1}, \ldots, v'_t \in \{v_1, \ldots, v_t\}$

Proof. A proof can be found in [2]. □

A basis for a vector space has to be able to write every vector in the vector space as a linear combination of the basis.

Definition 1.15 (Basis for vector space over field)**.**
A basis for a vector space V is a linearly independent generating set for V.

The dimension of V as a vector-space over $\mathbb{F}$ is defined as follows:

Definition 1.16 (Dimension of vector-space over field)**.**
The dimension of V as a vector space over $\mathbb{F}$ is defined as the number of elements in a basis of V.

Earlier the trace function from $\mathbb{F}_{q^t}$ to $\mathbb{F}_q$ was defined, and this has an additional interesting property. Let

$$L_\gamma : \mathbb{F}_{q^t} \to \mathbb{F}_q$$

given by

$$L_\gamma(x) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\gamma x), \quad \gamma \in \mathbb{F}_{q^t}.$$

Then all linear maps from $\mathbb{F}_{q^t} \to \mathbb{F}_q$ can be written as $L_\gamma(x),\ \gamma \in \mathbb{F}_{q^t}$.

Theorem 1.17.
Let $\mathbb{F}_{q^t}$ be a finite extension of $\mathbb{F}_q$. Then:

1. $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}$ is surjective ($\mathbb{F}_{q^t}$ onto $\mathbb{F}_q$).

2. $L_\gamma \neq L_\beta$ whenever $\gamma \neq \beta$ for $\gamma, \beta \in \mathbb{F}_{q^t}$.

3. the linear transformations from $\mathbb{F}_{q^t}$ into $\mathbb{F}_q$ are exactly the mappings $L_\gamma(x),\ \gamma \in \mathbb{F}_{q^t}$.

Proof. 1. To show that $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}$ is surjective, we need to show that there exists an $\alpha \in \mathbb{F}_{q^t}$ such that $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha) \neq 0$ as then the rest follows from $\mathbb{F}_q$ linearity.

This is done by looking at Definition 1.10. we have that $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha) = 0$ if and only if $\alpha \in \mathbb{F}_{q^t}$ is a root in the polynomial

$$x + x^q + \cdots + x^{q^{t-1}} \in \mathbb{F}_{q^t}[x]$$

however this polynomial has at most q^{t-1} roots, and $|\mathbb{F}_{q^t}| = q^t$ meaning that there are elements that can not be a root of the polynomial. Then there exists at least $q^t - q^{t-1}$ elements in $\mathbb{F}_{q^t}$ that satisfy

$$tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha) = a \neq 0$$

Therefore the trace function is surjective.

2. Let $\alpha, \beta, \gamma \in \mathbb{F}_{q^t}$ and $\gamma \neq \beta$. Then

$$\begin{aligned} L_\gamma(\alpha) - L_\beta(\alpha) &= tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\gamma\alpha) - tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta\alpha) \\ &= tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\gamma\alpha - \beta\alpha) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\alpha(\gamma - \beta)) \neq 0 \end{aligned}$$

for some $\alpha \in \mathbb{F}_{q^t}$ as $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}$ is surjective, meaning that $L_\beta \neq L_\gamma$ for $\beta \neq \gamma$.

3. From Lemma 1.11 we have that $tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}$ is a linear over $\mathbb{F}_q$. So we only need to show that all the mappings from $\mathbb{F}_{q^t}$ to $\mathbb{F}_q$ is all of the possible ones.

We have already shown that for every $\beta \neq \gamma$ then the linear transformation is different. Therefore we know that for every $\beta \in \mathbb{F}_{q^t}$ the corresponding transformation L_β from $\mathbb{F}_{q^t}$ to $\mathbb{F}_q$ is different. This also means that there are q^t different transformations of L_β, one for every β. We only needs to show that there only are q^t different transformations, such that L_β is all of them.

So let $v_1, \ldots, v_t$ be a basis of $\mathbb{F}_{q^t}$ over $\mathbb{F}_q$. Then as for any arbitrary element $\alpha \in \mathbb{F}_{q^t}$ and for proper $a_1 \ldots, a_t \in \mathbb{F}_q$ we can have:

$$a_1 v_1 + \cdots + a_t v_t = \alpha \in \mathbb{F}_{q^t}.$$

Therefore from Lemma 1.11 we have the following equation for $\beta \in \mathbb{F}_{q^t}$:

$$\begin{aligned} &L_\beta(\alpha) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta\alpha) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta(a_1 v_1 + \cdots a_t v_t)) \\ &tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(a_1\beta v_1 + \cdots + a_t\beta v_t) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(a_1\beta v_1) + \cdots + tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(a_t\beta v_t) \\ &= a_1 tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta v_1) + \cdots + a_t tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\beta v_t) = \sum_{i=1}^{t} a_i L_\beta(v_i) \end{aligned}$$

As $|\mathbb{F}_q| = q$ each a_i can take q different values. And as there are t different a_i's then there are q^t different combinations. Which means that there are q^t different linear transformations, which as shown earlier can be written as $L_\beta,\ \beta \in \mathbb{F}_{q^t}$.

□

This gives the following corollary which is a consequence of Theorem 1.17.

Corollary 1.18.
If

$$L_{\beta_1}(\alpha) = L_{\beta_2}(\alpha)$$

for some $\beta_1, \beta_2 \in \mathbb{F}_{q^t}$ and all $\alpha \in \mathbb{F}_{q^n}$ then $\beta_1 = \beta_2$.

Proof. This is a proof of contradiction. Assume that

$$L_{\beta_1}(\alpha) = L_{\beta_2}(\alpha)$$

but $\beta_1 \neq \beta_2$. Then the contradiction comes directly from Theorem 1.17 as we have that $\beta_1 \neq \beta_2$ then

$$L_{\beta_1}(\alpha) \neq L_{\beta_2}(\alpha)$$

which is a contradiction meaning that the corollary is true. □

Later we need the concept of dual basis. For finite fields extensions the definition of dual basis is based on the trace.

Definition 1.19 (Dual basis)**.**
Let $\mathbb{F}_{q^t}$ be seen as a vector space over $\mathbb{F}_q$ and let $\{v_1, \ldots, v_t\}$ be a basis. Then let $\{w_1, \ldots, w_t\}$ also be a basis. It is called the dual basis if

$$tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(v_i w_j) = \begin{cases} 0, & \text{if } i \neq j, \\ 1, & \text{otherwise.} \end{cases}$$

It is not clear that a dual basis exists for any basis, and this of course has to be proven. However it turns out that there in fact exist a dual basis for any basis - and it is unique.

Theorem 1.20.
For any given basis $\bar{v} = \{v_1, \ldots, v_t\}$ for $\mathbb{F}_{q^t}$ over $\mathbb{F}_q$ there exists a unique dual basis.

Proof. Let $\alpha \in \mathbb{F}_{q^t}$ be an arbitrary element. Then by assumption $\bar{v}$ was a basis, meaning that α can be written as a linear combination of the basis:

$$\alpha = \sum_{i=1}^{t} a_i v_i \tag{1.1}$$

for some coefficients $a_i \in \mathbb{F}_q$. These coefficients can be written as a function:

$$c_i(\cdot) : \mathbb{F}_{q^t} \to \mathbb{F}_q, \ 1 \leq i \leq t,$$

satisfying $c_i(\alpha) = a_i$, however Theorem 1.17 yields that $c_i(\cdot)$ actually is of the form: L_{w_i} for some $w_i \in \mathbb{F}_{q^t}$, meaning that

$$c_i(\alpha) = L_{w_i}(\alpha) = tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i \alpha), \ 1 \leq i \leq t. \tag{1.2}$$

Combining (1.1) and (1.2) gives the equation

$$\alpha = \sum_{i=1}^{t} tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i \alpha) v_i \tag{1.3}$$

which is true for any $\alpha \in \mathbb{F}_{q^t}$. This also means that it is true for $\alpha = v_j$ where $v_j \in \bar{v}$, which changes the equation as follows:

$$v_j = \sum_{i=1}^{t} tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i v_j) v_i$$

however for this to be true, then

$$tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i v_j) = \begin{cases} 0, & \text{if } i \neq j \\ 1, & \text{otherwise} \end{cases}$$

as this is true for any $v_j \in \bar{v}$.
This proves that there exists the coefficients $w_1, \ldots, w_t$ satisfying the definition of dual basis. However we still need to show that they in fact is a basis and that the basis is unique.
By Definition 1.15 they need to be linearly independent and generate the entire vector space.

- Linearly independent: By definition 1.13, we are going to assume that

$$\sum_{i=1}^{t} d_i w_i = 0 \tag{1.4}$$

  for some $d_i \in \mathbb{F}_q$. We want to show that for $1 \leq i \leq t$ that $d_i = 0$ for this to be linearly independent. Therefore i am going to rewrite (1.4) as follows:

$$\sum_{i=1}^{t} d_i w_i = 0 \Rightarrow (\sum_{i=1}^{t} d_i w_i) v_j = 0 \Rightarrow tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\sum_{i=1}^{t} d_i w_i v_j) = 0.$$

  Now we can use Lemma 1.11 as $d_i \in \mathbb{F}_q$ and the trace is linear over $\mathbb{F}_q$, meaning the equation is going to look like

$$\sum_{i=1}^{t} tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i v_j) d_i = 0 \tag{1.5}$$

  however (1.5) and (1.3) is of the same form, meaning that $d_i = 0$ for $1 \leq i \leq t$. This satisfies Definition 1.13.

- Spanning $\mathbb{F}_{q^t}$ over $\mathbb{F}_q$: As $\bar{w} = \{w_1 \ldots, w_t\}$ is linearly independent and we know that $\bar{v}$ is a basis (and therefore a generating set of $\mathbb{F}_{q^t}$) then we can use Lemma 1.14 to conclude that $\bar{w} = \{w_1 \ldots, w_t\}$ is a generating set.

Therefore $\bar{w}$ satisfies both linearly independent and spanning $\mathbb{F}_{q^t}$ making it a basis.
Last thing to show is that the basis is unique. For this assume that

$$\alpha = \sum_{i=1}^{t} tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(w_i \alpha) v_i = \sum_{i=1}^{t} tr_{\mathbb{F}_{q^t}/\mathbb{F}_q}(\omega_i \alpha) v_i$$

for another set $\bar{\omega} = \{\omega_1 \ldots, \omega_t\}$ satisfying the same as $\bar{w}$. Then by Corollary 1.18 as this is true for any $\alpha \in \mathbb{F}_{q^t}$ then we can conclude that $w_i = \omega_i$ for $1 \leq i \leq t$. □

1.3 Coding Theory

This section is based on [1], [6] and [7].
The exact repair problem is related to error-correction codes, in which the following definition well lead up to:

Definition 1.21 (Linear and non-linear codes)**.**
A block code C of length n over $\mathbb{F}_q$ is a nonempty set of $\mathbb{F}_q^n$. The elements of C are called codewords.
If for all codewords $c, c' \in C$, and constants $\lambda \in \mathbb{F}_q$ satisfy the following:

$$c + c' \in C$$
$$\lambda c \in C$$

then the code is called linear, and non-linear otherwise.

The following definition is the definition of error-correcting codes. This is an essential part of correcting codes.

Definition 1.22 (Error-Correction Codes)**.**
A code C is t-error correcting if for any two different codewords $c, c' \in C$ for any errors vectors e_1, e_2 with non-zero coordinates in at most t positions the following is satisfied:

$$c + e_1 \neq c' + e_2.$$

The following type of code is the main type of code this thesis will be looking at. The code is called Reed-Solomon codes.

Definition 1.23 (Reed-Solomon codes)**.**
Let $a_1, \ldots, a_n$ be pairwise different elements of a finite field $\mathbb{F}_q$ with $n \leq q$. Denote the set $\{a_1, \ldots, a_n\} = A$ as the evaluation points. A Reed-Solomon code $RS_q(n, k)$ is the code consisting of the codewords

$$(f(a_1), f(a_2), \ldots, f(a_n)),$$

where f is a polynomial in $\mathbb{F}_q[x]$ with degree less than $k < n$.

In the work of error-correction codes it has shown useful to have a measure for distance between codewords.

Definition 1.24 (Minimum distance)**.**
For $x = (x_1, \ldots, x_n), y = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$ the hamming distance for two vectors is defined as

$$d(x, y) = |\{i | x_i \neq y_i\}|$$

Let C be a linear code, then the minimum distance is

$$d = \min\{d(x, y) | x, y \in C, x \neq y\}.$$

It can be proven that a code C is t-error correcting if and only if $t < \frac{d}{2}$.
The following theorem is a famous bound, known as the Singleton bound, which gives an upper-bound on the minimum distance by their length n and dimension k.

Theorem 1.25 (Singleton bound)**.**
Let C be a linear code with minimum distance d, then

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

This leads up to maximum distance separable codes, which is codes that has an equality in the Singleton bound ie. that $d = n - k + 1$.

Definition 1.26 (Maximum distance separable codes)**.**
Let C be a linear code. Then the code C is a Maximum Distance Separable (MDS) code if

$$d = n - k + 1$$

The following proposition speaks for itself.

Proposition 1.27.
Reed-Solomon codes are MDS codes.

The idea with Reed-Solomon codes is that if k evaluations of the used polynomial is given with the corresponding evaluation points, the whole codeword can be reconstructed by Lagrange interpolation. But this also means that k evaluations/evaluation points are required to determine the polynomial.
Later the dual code will come into play, therefore we define it as the following.

Definition 1.28 (Dual code)**.**
The dual code $C^{\perp}$ is the code defined as

$$C^{\perp} = \{x \in \mathbb{F}_q^n | x \cdot c = 0, \ \forall c \in C\}$$

where $x \cdot c$ denotes the dot product.

A more general code than the Reed-Solomon code is going to be needed, as the dual of a Reed-Solomon code is a generalized Reed-Solomon code. For this definition we introduce $\mathbb{F}^*$ which is a notation for $\mathbb{F}^* = \mathbb{F} \backslash \{0\}$.

Definition 1.29 (Generalized Reed-Solomon codes)**.**
Let $a_1, \ldots, a_n$ be pairwise different elements of a finite field $\mathbb{F}_q$ with $n \leq q$. Denote the set $\{a_1, \ldots, a_n\} = A$ as the evaluation points and let $\lambda = (\lambda_1, \ldots, \lambda_n)$ be a set of elements, where $\lambda_i \in \mathbb{F}_q^*$ for $1 \leq i \leq n$. A Generalized Reed-Solomon code $GRS_q(n, k, \lambda)$ is the code consisting of the codewords

$$(\lambda_1 f(a_1), \lambda_2 f(a_2), \ldots, \lambda_n f(a_n)),$$

where f is a polynomial in $\mathbb{F}_q[x]$ with degree less than $k < n$.

A Reed-Solomon code defined as in Definition 1.23 is a generalized Reed-Solomon code where $\lambda = (1, \ldots, 1)$, ie.

$$RS_q(n, k) = GRS_q(n, k, (1, \ldots, 1)).$$

Proposition 1.30.
Let $0 \leq k < n \leq q+1$. Then $GRS_q(n,k,\lambda)$ is a $\mathbb{F}_q$-linear MDS code.

Proof. A proof can be found in [6]. □

For the following proposition we need some notation with respect to Lagrange interpolation. Let

$$L(x) = \prod_{i=1}^{n}(x - a_i)$$

where $A = \{a_1, \ldots, a_n\}$ is the mutually distinct evaluation points. Then let

$$L_i(x) = \frac{L(x)}{(x - a_i)} = \prod_{j \neq i}(x - a_j)$$

in which

$$l_i(x) = \frac{L(x)}{L_i(a_i)} = \prod_{1=i, i\neq j}^{n} \frac{(x - a_i)}{(a_j - a_i)}$$

is the Lagrange polynomials as they usually are seen. The Lagrange interpolation formula is then as follows:

$$f(x) = \sum_{i=1}^{n} l_i(x) f(a_i) \tag{1.6}$$

which can interpolate any polynomial f of degree at most $n-1$ given $(a_1, f(a_1)), \ldots, (a_n, f(a_n))$ where $a_i \neq a_j$ for $i \neq j$.

Proposition 1.31.
Let $a_1, \ldots, a_n$ be mutually distinct elements of $\mathbb{F}_q$ and the evaluation points of $GRS_q(n,k,\lambda)$, $GRS_q(n, n-k, \lambda^{\perp})$.
Let $\lambda = (\lambda_1, \ldots, \lambda_n)$ where $\lambda_i \in \mathbb{F}_q^*$ for $1 \leq i \leq n$ and $\lambda^{\perp} = (\lambda_1^{\perp}, \ldots, \lambda_n^{\perp})$ be the tuple with values

$$\lambda_i^{\perp} = \frac{1}{\lambda_i \prod_{i \neq j}(a_j - a_i)}$$

for $1 \leq i \leq n$. Then $GRS_q(n, n-k, \lambda^{\perp})$ is the dual of $GRS_q(n,k,\lambda)$.

Proof. This is proven by using Definition 1.28.
Let $c \in GRS_q(n,k,\lambda), \tilde{c} \in GRS_q(n, n-k, \lambda^{\perp})$ be any codeword in their respective code, such that

$$c = (\lambda_1 f(a_1), \ldots, \lambda_n f(a_n)),\ deg(f) < k$$
$$\tilde{c} = (\lambda_1^{\perp} g(a_1), \ldots, \lambda_n^{\perp} g(a_n)),\ deg(g) < n - k$$

where λ and $\lambda^{\perp}$ is specified as above.
We want to show that $c \cdot \tilde{c} = 0$ for all $c \in GRS_q(n,k,\lambda), \tilde{c} \in GRS_q(n, n-k, \lambda^{\perp})$.
As $deg(f(x)) \leq k-1, deg(g(x)) \leq n-k-1$

$$deg(f(x) \cdot g(x)) = deg(f(x)) + deg(g(x)) \leq (n-k-1) + (k-1) = n-2$$

meaning that Lagrange Interpolation can be used on their product. By (1.6) we have the following:

$$f(x)g(x) = \sum_{i=1}^{n} \frac{L_i(x)}{L_i(a_i)} f(a_i)g(a_i). \tag{1.7}$$

Then we notice that

$$\lambda_i^{\perp} = \frac{1}{\lambda_i L_i(a_i))}$$

by the notation of $L_i(a_i)$.
As the degree of $f(x)g(x)$ is at most $n-2$, then the coefficient in front of x^{n-1} must be 0. Therefore by setting the coefficient equal to itself we have the following:

$$\begin{aligned} 0 &= \sum_{i=1}^{n} \frac{1}{L_i(a_i)} f(a_i)g(a_i) = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda_i} \frac{1}{L_i(a_i)} f(a_i)g(a_i) \\ &= \sum_{i=1}^{n} (\lambda_i f(a_i)) \frac{g(a_i)}{\lambda_i L_i(a_i)} = \sum_{i=1}^{n} (\lambda_i f(a_i))(\lambda_i^{\perp} g(a_i)) = c \cdot \tilde{c} \end{aligned}$$

which proofs that $(GRS_q(n,k,\lambda))^{\perp} = GRS_q(n, n-k, \lambda^{\perp})$. □

Proposition 1.31 might generalize the dual code of GRS-codes, however later we only need the dual of a RS-code, where we already established $\lambda = (1, \ldots, 1)$ which makes

$$(RS_q(n,k))^{\perp} = GRS_q(n, n-k, \lambda^{\perp})$$

with $\lambda^{\perp} = (\lambda_1^{\perp}, \ldots, \lambda_n^{\perp})$ specified as

$$\lambda_i^{\perp} = \frac{1}{\prod_{i \neq j}(a_j - a_i)}.$$

Chapter 2

Repair Schemes

2.1 The Exact Repair Problem

This section is based on [1].

This section will explain what Distributed Storage Systems (DSS) are and how they are related to the Exact Repair Problem (ERP).
DDSs are used to store data on different servers, such that one server does not hold all of the data. This is (ideally) done such that when one server crashes, all data is not lost. If all data would be stored on one server the data would be lost if that server crashed. And if the data was large it would make sense to split it up such that one server does not overload.

The ERP is described as follows: Assume that a large file is distributed over n servers. Then one of these servers crashes, hence losing the data that server stored. The ERP is then to replace the data on the crashed server by using the correct data from the other servers without reconstructing the entire data and redistribute it again.

This indicated that for this problem error-correcting codes could be used, as for some codes they have been specifically designed to be good for error-correcting. As MDS codes by definition has to satisfy $d = n - k + 1$ and RS-codes are MDS, it is not difficult to construct a code with a 'big' minimum distance.
For example consider $d = 24, n = 35, k = 12$, which means that this is a 11-error correcting code for a field with more than 35 elements. By Reed-Solomon definition all there is required for one codeword is a polynomial in $\mathbb{F}_q[X]$ (q is a prime power and larger than 35) of degree at most $k - 1 = 11$. Any $k = 12$ correct entries in a codeword can then restore the entire codeword by Lagrange interpolation.
The problem is that even if this solves the problem (the problem of restoring a single entry in a codeword), it require that the entire codeword is reconstructed. That is a lot of wasted calculations, as the one entry which was incorrect will be replaced but the rest will be thrown away. And as explained in the ERP description above this is not the solution we are looking for.

Example 2.1. This example will show how a DSS could look like, not from a code-based perspective but from an intuitive perspective.
Assume that you have the data, which I will denote as A, B, C, D, E, F, G. This data will be

split up into servers which I demonstrate with the following figure:

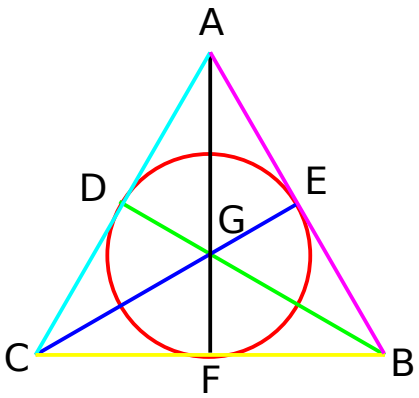

Figure 2.1: A graph to demonstrate how the data was split out to servers. Each different color symbolizes a different server.

The figure is a visualization of how the data was distributed among servers. On the figure each server is shown as a colored line where the data the server holds is the vertexes that colored line reaches. I will denote each server as the color given to them on Figure 2.1. Hence each server holds as the following table shows:

Teal	$\{A, D, C\}$
Yellow	$\{C, F, B\}$
Purple	$\{A, E, B\}$
Red	$\{D, E, F\}$
Green	$\{D, G, B\}$
Blue	$\{C, G, E\}$
Black	$\{A, G, F\}$

Table 2.1: The distribution of data for each server.

Now if any of the servers crashes, the data on that server can be repaired by the other servers as each of the data A, B, C, D, E, F, G is on 3 servers which can be seen on the graph in Figure 2.1.

Say for example that the server 'Green' crashes. Then it would be possible to make a replacement server, where it can receive data from the remaining servers to be a copy of the original. The servers 'Red', 'Blue' and 'Yellow' each have an element which was on 'Green'. 'Red' sends the data D, 'Blue' sends G and 'Yellow' sends B, which then combined yields the data that 'Green' held.

The above case is an example of where the ERP becomes relevant. As each data is on 3 different servers, there is still access to all the data if one server crashes. Instead of collecting all the data from the remaining servers and start all over, we just made the replacement server get the lost information from the remaining servers.

2.2 Linear Exact Repair Schemes

This section is based on [1].

RS-codes have the property that from k correct entries in a codeword, the whole codeword can be reconstructed. The problem is that k correct is necessary even if $n-1$ are correct and only one entry is incorrect. Using the methods known to reconstruct RS-codes (eg. Lagrange Interpolation) the whole codeword is reconstructed, even though only one entry was incorrect. That is a lot of wasted calculations, and this is why it is worth to study if it can be done better, in the sense of less wasted calculation. This is where linear exact repair schemes come in.

Recalling the fact that any element of a finite field can be written as an evaluation in a polynomial by using Lagrange interpolation, we can assume that entries in a codeword is given as polynomial evaluations. Therefore not only RS-codes can be expressed as polynomial evaluations, but all linear codes can if some set of evaluation points $\{a_1 \dots, a_n\} = A \subseteq \mathbb{F}_q$ is given, $a_i \neq a_j$ for $i \neq j$ and $n \leq q$.

With Section 1.3 in mind the problem is the following. Assume that this data is distributed using an MDS code over a field $\mathbb{F}_q$. Then the file, which is distributed, is some function $f \in \mathbb{F}_q[X]$. By the theory the whole file can be restored from k points on f. The distribution is made such that server i is associated with $a_i \in A$, and that server then holds $f(a_i)$. The Exact Repair Problem is then to restore the data on a random failed server a_j, where the data lost is $f(a_j)$, by restoring only the data $f(a_j)$ without recreating the entire codeword. The information available to assist in this is the remaining servers and the data stored on them.

In Example 2.1, to restore a failed server the other servers did not send all their information. Only the data required was send. Instead of using all the data, all the remaining servers has available, it can be done by only using a part of it (depending on the server). In math terms the server only has to send some amount of sub-symbols, which will be viewed as an element of a sub field $\mathbb{B} \leq \mathbb{F}$. I will focus on linear repair schemes, which makes this the case of $\mathbb{F}$ as a vector space over $\mathbb{B}$.
The following is what we define a linear exact repair scheme to be. There is a required that the data is split such that the data from different servers is associated with each other and can help reconstruct each other, and then that there exist a formula to reconstruct $f(a'), \forall a' \in A$ by a linear combination.

Definition 2.2 (Linear Exact Repair Scheme)**.**
Let C be a linear code over $\mathbb{F}$ of length n and dimension k given by a set of functions $\mathbf{F}$ and a set of evaluation points A and let $f \in \mathbf{F}$. Denote a codeword in C as

$$(f(a_1), \dots, f(a_n)) \in C.$$

Then a Linear Exact Repair Scheme(LERS) for C over a subfield $\mathbb{B} \leq \mathbb{F}$ consists of the following:

- For all $a' \in A$ and for all $a \in A \backslash \{a'\}$, there is a set of queries $Q_a(a') \subseteq \mathbb{F}$ that is a set of information that server a holds to help to reconstruct $f(a')$.

- For every $a' \in A$ there is a linear reconstruction algorithm that computes:

$$f(a') = \sum_i \lambda_i v_i$$

where $\lambda_i \in \mathbb{B}$ and $v_1, \ldots, v_t$ is a basis for $\mathbb{F}$ over $\mathbb{B}$ so that the coefficients λ_i are $\mathbb{B}$-linear combination of the queries

$$\bigcup_{a \in A \setminus \{a'\}} \{tr_{\mathbb{F}/\mathbb{B}}(\gamma f(a)) | \gamma \in Q_a(a')\}.$$

In Definition 2.2 it is required that the failed coordinate a' can be 'repaired' by computing of a linear combination of the basis and some coefficient defined by the field trace of the queries.
The idea is to only take some number of sub-symbols from each (correct) server to restore the failed server. Ideally the amount of sub-symbols required should be as small as possible. This is expressed as repair bandwidth which is defined as follows.

Definition 2.3 (Repair Bandwith)**.**
The repair bandwith b of the exact repair scheme is the total number of sub-symbols in B returned by each a:

$$b = \max_{a' \in A \setminus \{a'\}} \sum_{a \in A \setminus \{a'\}} |Q_a(a')|.$$

Repair bandwidth is the worst case scenario of sub-symbols that can be needed to repair an arbitrary failed server. So by assuming, one by one, that all node eventually fail, what is the worst case scenario for repairing a failed server.
The following definition is a measure for how many (non-failed) servers at most can be required to fix any failed server. As before, the logic behind is to find the worst case scenario and then say that the LERS has Repair Locality d, which is defined as follows.

Definition 2.4 (Repair Locality)**.**
The repair locality of the exact repair problem is the maximum amount of servers which are assigned to help reconstruct an arbitrary failed server:

$$d = \max_{a' \in A} \sum_{a \in A \setminus \{a'\}} 1_{Q_a(a') \neq \emptyset}$$

where $1_{Q_a(a')}$ is an indicator function that is 1 if $Q_a(a') \neq \emptyset$ and is 0 otherwise.

To make these definitions easier to understand I give the following example.

Example 2.5. To get an idea of the different definitions I will continue from Example 2.1. In Example 2.1 there was not used a code for distribution of the data, meaning that not all of the definitions makes sense, however some of them does.
The table below is same distribution table as in Example 2.1.

Teal	$\{A, D, C\}$
Yellow	$\{C, F, B\}$
Purple	$\{A, E, B\}$
Red	$\{D, E, F\}$
Green	$\{D, G, B\}$
Blue	$\{C, G, E\}$
Black	$\{A, G, F\}$

Table 2.2: The distribution of data for each server.

In Definition 2.2 a set defined as queries $Q_a(a')$ was the set of information $f(a)$ holds that can help $f(a')$ to reconstruct the data on a'. In Example 2.1 there were seven servers denoted with colors. However the queries has to be pre determined in the LERS, and one example could be seen in the following table:

a'	Teal	Yellow	Purple	Red	Green	Blue	Black
$Q_{\text{Teal}}(a')$	{∅}	{C}	{A}	{D}	{∅}	{C}	{∅}
$Q_{\text{Yellow}}(a')$	{C}	{∅}	{B}	{F}	{∅}	{∅}	{F}
$Q_{\text{Purple}}(a')$	{A}	{B}	{∅}	{E}	{B}	{∅}	{A}
$Q_{\text{Red}}(a')$	{D}	{F}	{E}	{∅}	{D}	{E}	{∅}
$Q_{\text{Green}}(a')$	{∅}	{∅}	{∅}	{∅}	{∅}	{G}	{∅}
$Q_{\text{Blue}}(a')$	{∅}	{∅}	{∅}	{∅}	{G}	{∅}	{G}
$Q_{\text{Black}}(a')$	{∅}	{∅}	{∅}	{∅}	{∅}	{∅}	{∅}

Table 2.3: A possible set of queries for each server.

Table 2.3 is one possibility of queries. This is by no mean the unique possible one. All the data are on 3 different servers hence more combinations are possible. Table 2.3 is an 'optimal' set of queries in the sense that a server crashes, it does not download an element two times. For example if Black crashed, it downloads F from Yellow, A from Purple, G from Black, but additionally we could have set $Q_{\text{Green}}(\text{Black}) = \{\text{G}\}$ as well - then Black just had to download the same element twice.
Let $T = \{\text{Teal, Yellow, Purple, Red, Green, Blue, Black}\}$ and denote an arbitrary crashed server as t'. In this case the repair bandwidth is

$$\max_{t' \in T} \sum_{t \in T \setminus \{t'\}} |Q_t(t')| = 3.$$

Because no set of queries contain more than one element, the repair locality is the same.
This means that the repair bandwidth is 3 which means that 3 elements has to be downloaded to repair an arbitrary crashed server. And that 3 servers are required to respond to repair an arbitrary crashed server.
This example is not an specific example of Definition 2.2, 2.3 and 2.4, but should give an idea of what the different terms means.

2.3 Construction of a LERS for MDS Codes

This section is based on [1].
The purpose of this section is to explain the general MDS LERS algorithm by explaining the steps and sum it up in an algorithm after.
Let $C \subseteq \mathbb{F}^n$ be an MDS code of length n and dimension k and let $\mathbb{F}$ be a field with subfield $\mathbb{B}$. Let the degree of $\mathbb{F}$ over $\mathbb{B}$ be t. Let $c \in C$ be a codeword which as described earlier can be written as polynomial evaluations

$$(f(a_1), \dots, f(a_n)) = c \in C,$$

with evaluation points $\{a_1, \dots, a_n\} = A \subseteq \mathbb{F}$ and $deg(f) < k$. Then assume that an arbitrary server (as it was called in Section 2.2) $a' \in A$ crashes and therefore we lose $f(a')$. Then the idea to reconstruct $f(a')$ is the following.
First find and fix a basis Z of $\mathbb{F}$ over $\mathbb{B}$. Let this basis be denoted as $Z = \{\zeta_1, \dots, \zeta_t\}$.
The next step is then to find coefficients $\mu_{\zeta_i,a}(a') \in \mathbb{F}$ for $a \in A \backslash \{a'\}$ and $\zeta_i \in Z$ that satisfies

$$\zeta_i f(a') = \sum_{a \in A \backslash \{a'\}} \mu_{\zeta_i,a}(a') f(a). \tag{2.1}$$

That these coefficients exists or how to find them is not clear. For now we only prove that they exists, and later we will show how they can be chosen. For the existing part we present the following proposition.

Proposition 2.6.
Assume there is a linear repair scheme for an $[n, k]$ MDS code $C \subseteq \mathbb{F}^n$ and let $\mathbb{B} \leq \mathbb{F}$ given by query sets $Q_a(a')$ and a linear repair algorithm (as in Definition 2.2). Then there is a basis Z for $\mathbb{F}$ over $\mathbb{B}$ such that the following is true.
For each $a' \in A$ and $a \in A \backslash \{a'\}$, $\zeta_i \in Z$ there exists coefficients $\mu_{\zeta_i,a}(a')$ satisfying

$$\zeta_i f(a') = \sum_{a \in A \backslash \{a'\}} \mu_{\zeta_i,a}(a') f(a)$$

for all $c \in C$ where $c = (f(a_1) \dots, f(a_n))$ with $deg(f) < k$. Also that for all $a \neq a' \in A$:

$$\{\mu_{\zeta_i,a}(a')\} \subseteq span_B(Q_a(a')).$$

Proof. By Definition 2.2 and by assumptions of the proposition there is a linear reconstruction algorithm given as:

$$f(a') = \sum_{i=1}^{t} \lambda_i v_i \tag{2.2}$$

where $v_1 \dots, v_t$ is a basis of $\mathbb{F}$ over $\mathbb{B}$ and λ_i are $\mathbb{B}$-linear functions of the queries which means there are on the form:

$$\lambda_i = \sum_{a \in A \backslash \{a'\}} \sum_{\gamma \in Q_a(a')} \beta_{a,\gamma,i} \cdot tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot f(a)) \tag{2.3}$$

for some $\beta_{a,\gamma,i} \in \mathbb{B}$.
As $v_1, \dots, v_t$ is a basis, then let $\zeta_1, \dots, \zeta_t$ be unique dual basis guaranteed by Theorem 1.20. Thus by Definition 1.19:

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_l v_i) = 1 \tag{2.4}$$

for $i = l$ by definition and 0 for $i \neq l$. Then by using (2.2) we can rewrite the following for $1 \leq i \leq t$:

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_l f(a')) = tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_l \sum_{i=1}^{t} \lambda_i v_i\right) = tr_{\mathbb{F}/\mathbb{B}}\left(\sum_{i=1}^{t} \lambda_i \zeta_l v_i\right) = \sum_{i=1}^{t} \lambda_i tr_{\mathbb{F}/\mathbb{B}}(\zeta_l v_i) = \lambda_l$$

which follows from (2.4) and the linearity over $\mathbb{B}$. Therefore we have the following from (2.3) by rewriting it:

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_l f(a')) = \lambda_l = \sum_{a \in A \setminus \{a'\}} tr_{\mathbb{F}/\mathbb{B}}\left(\underbrace{\sum_{\gamma \in Q_a(a')} \beta_{a,\gamma,l} \cdot \gamma}_{\mu_{\zeta_l,a}(a')} \cdot f(a)\right) \tag{2.5}$$

and where the coefficients $\mu_{\zeta_i,a}(a')$ are defined. Then (2.5) satisfy

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_l f(a')) = \sum_{a \in A \setminus \{a'\}} tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_l,a}(a') \cdot f(a)\right)$$

which it does for all $c \in C$ which was assumed MDS (and therefore linear) which means that this must also be true for $\gamma \cdot c$ for $c \in C$ and all $\gamma \in \mathbb{F}$ and therefore:

$$tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot \zeta_l f(a')) = tr_{\mathbb{F}/\mathbb{B}}\left(\gamma \cdot \sum_{a \in A \setminus \{a'\}} \mu_{\zeta_l,a}(a') \cdot f(a)\right)$$

and as this is true for all $\gamma \in \mathbb{F}$ which combined with Corollary 1.18 yields

$$\zeta_i f(a') = \sum_{a \in A \setminus \{a'\}} \mu_{\zeta_i,a}(a') f(a)$$

which proves that the coefficients exists for every $\zeta_i \in Z$.
Lastly we have to show that the coefficients $\{\mu_{\zeta_i,a}(a')\} \subseteq span_{\mathbb{B}}(Q_a(a'))$. As defined

$$\mu_{\zeta_i,a}(a') = \sum_{\gamma \in Q_a(a')} \beta_{a,\gamma,i} \cdot \gamma$$

where $\beta_{a,\gamma,i} \in \mathbb{B}$ and $\gamma \in Q_a(a')$ from the summation. Therefore all the elements are $\mathbb{B}$-linear combinations of elements in $Q_a(a')$. Therefore

$$\{\mu_{\zeta_i,a}(a') | 1 \leq i \leq t\} \subseteq span_B(Q_a(a')).$$

□

The coefficients $\mu_{\zeta_i,a}(a')$ both depends on the evaluation points $a \in A \setminus \{a'\}$ and $\zeta_i \in Z$. Therefore in total there are $n \cdot t$ $\mu_{\zeta_i,a}(a')$ to account for. For every $a \in A \setminus \{a'\}$ there are t (one for every $\zeta_i \in Z$) which means for every $\zeta_i \in Z$ we can find a spanning set for $\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\}$ over $\mathbb{B}$, and this set we denote with $\tilde{Q}_a(a') \subseteq \mathbb{F}$.
For every $\gamma \in \tilde{Q}_a(a')$ we have the corresponding query:

$$tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot f(a)) \quad \gamma \in \tilde{Q}_a(a').$$

Then from Lemma 1.11 we showed that the trace function is $\mathbb{B}$-linear meaning by a linear combination of $\gamma_1, \ldots, \gamma_m$ with proper scalars $\beta_1, \ldots, \beta_m \in \mathbb{B}$ we can express

$$\mu_{\zeta_i,a}(a') = \beta_1\gamma_1 + \cdots + \beta_m\gamma_m$$

where $dim_{\mathbb{B}}(\tilde{Q}_a(a')) = m \leq t$ as $\tilde{Q}_a(a')$ was a spanning set.
Then we compute

$$\begin{aligned}
&\beta_1 tr_{\mathbb{F}/\mathbb{B}}\left(\gamma_1 f(a)\right) + \cdots + \beta_m tr_{\mathbb{F}/\mathbb{B}}\left(\gamma_m f(a)\right) \\
&= tr_{\mathbb{F}/\mathbb{B}}\left(\beta_1\gamma_1 f(a)\right) + \cdots + tr_{\mathbb{F}/\mathbb{B}}\left(\beta_m\gamma_m f(a)\right) \\
&= tr_{\mathbb{F}/\mathbb{B}}\left(\beta_1\gamma_1 f(a) + \cdots + \beta_m\gamma_m f(a)\right) \\
&= tr_{\mathbb{F}/\mathbb{B}}\left((\beta_1\gamma_1 + \cdots + \beta_m\gamma_m) f(a)\right) \\
&= tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a') f(a)\right)
\end{aligned} \tag{2.6}$$

for every $a \in A \backslash \{a'\}$.
Then by taking the trace of each side of (2.1) we can compute

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a')) = \sum_{a \in A\backslash\{a'\}} tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a') f(a)\right) \tag{2.7}$$

as all of terms of the sum was computed in (2.6).
When the above is done for all $\zeta_i \in Z$ we have the following t values:

$$\{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a')) | \zeta_i \in Z\}.$$

These values are now used to reconstruct the failed server. As Z was a basis for $\mathbb{F}$ over $\mathbb{B}$, we denote the dual basis as $V = \{v_1 \ldots v_t\}$ which Theorem 1.20 ensures exists.
As $f(a') \in \mathbb{F}$ it can be written as a linear combination of the basis V as:

$$f(a') = \sum_{i=1}^{t} b_i v_i \tag{2.8}$$

for some coefficients $b_i \in \mathbb{B}$. The basis V is known, however the coefficients b_i is not, and therefore we need a way to determine them. We are going to assume they exists, and then find them by:

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_j f(a')) = tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_j \sum_{i=1}^{t} b_i v_i\right) = tr_{\mathbb{F}/\mathbb{B}}\left(\sum_{i=1}^{t} b_i \zeta_j v_i\right) = \sum_{i=1}^{t} b_i tr_{\mathbb{F}/\mathbb{B}}(\zeta_j v_i) = b_j \tag{2.9}$$

from the definition of dual basis, meaning that combining (2.9) and (2.8) yields:

$$f(a') = \sum_{i=1}^{t} tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a')) v_i. \tag{2.10}$$

All of the above is now summarized in the following algorithm.

Algorithm 2.7. Framework for a Linear Exact Repair Scheme for MDS codes C.
Input: A MDS code $C \subseteq \mathbb{F}^n$. Let $\mathbb{F}$ be an extention field over $\mathbb{B}$ where the degree of $\mathbb{F}$ over $\mathbb{B}$ is t. Let C be described by a set of polynomials $\mathbf{F}$ with degree less than k with corresponding evaluation points A. Let a' be a crashed server, and assume there is access to linear queries on the form $tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot f(a))$ for $a \in A\backslash\{a'\}$ where $f \in \mathbf{F}$.
Output: The value of $f(a')$.

1. Choose a basis Z of $\mathbb{F}$ over $\mathbb{B}$.

2. Choose coefficients $\mu_{\zeta_i,a}(a')$ for $a \in A\backslash\{a'\}, \zeta_i \in Z$ satisfying

$$\zeta_i f(a') = \sum_{a \in A\backslash\{a'\}} \mu_{\zeta_i,a}(a') f(a)$$

3. For every $\zeta_i \in Z$ do the following:

   - Find a spanning set for $\{\mu_{\zeta_i,a}(a') \mid \zeta_i \in Z\}$ over $\mathbb{B}$, denoted by $\tilde{Q}_a(a') \subseteq \mathbb{F}$
   - Compute the queries:
   $$tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot f(a)) \quad \gamma \in \tilde{Q}_a(a')$$
   - By using $\mathbb{B}$-linearity of the trace function compute:
   $$tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta_i,a}(a') \cdot f(a)) \quad a \in A\backslash\{a'\}$$
   - Then take the trace of each
   $$\sum_{a \in A\backslash\{a'\}} tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a') f(a)\right) = tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a'))$$
   and save the value.

4. Compute $f(a')$ by
$$f(a') = \sum_{i}^{t} tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a)) v_i$$
where $V = \{v_1, \ldots, v_t\}$ is the dual basis of Z.

The following proposition is the first observation about Algorithm 2.7.

Proposition 2.8.
Algorithm 2.7 is a linear repair scheme for C over $\mathbb{B}$ with repair bandwidth

$$b = \max_{a' \in A} \sum_{a \in A\backslash\{a'\}} dim_{\mathbb{B}}\left(\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\}\right).$$

Proof.

- Linear repair scheme: this follows by Definition 2.2 as there is both a set of queries $Q_a(a') \subseteq \mathbb{F}$ and a linear reconstruction scheme to reconstruct the failed server.
- Repair bandwidth: by Definition 2.3 it is just a matter of rewriting $|Q_a(a')|$. In Algorithm 2.7 we have the $\tilde{Q}_a(a')$ which were a spanning set of the $\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\}$ over $\mathbb{B}$ meaning that:
$$|\tilde{Q}_a(a')| = dim_{\mathbb{B}}\left(\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\}\right)$$
which proves the proposition.

□

The following corollary tells us that any linear repair scheme can be on the same form as Algorithm 2.7.

Corollary 2.9.
Let $\mathbb{B} \leq \mathbb{F}$ be a subfield and let $A \subseteq \mathbb{F}$ be any set of evaluation points. Let $k \leq |A|$ be an integer. Let $C \subseteq \mathbb{F}^n$ be an MDS code with evaluations A and dimension k. Then the following is equivalent:

1. There is a linear repair scheme for C over $\mathbb{B} \leq \mathbb{F}$ with bandwidth b.
2. There is a linear repair scheme for C over $\mathbb{B}$ of the form of Algorithm 2.7, with bandwidth at most b.

Proof.

- (1) $\Rightarrow$ (2): Suppose there is a linear repair scheme for an MDS code C with query sets $Q_a(a')$ as Definition 2.2 gives by assumption. Then by Proposition 2.6 we can choose a basis Z and coefficients $\mu_{\zeta_i,a}(a')$. Then by Proposition 2.8 we have that the bandwidth of Algorithm 2.7 is given by

$$b = \max_{a' \in A} \sum_{a \in A \backslash \{a'\}} dim_{\mathbb{B}} \left(\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\} \right).$$

  Then by Proposition 2.6 we have that

$$\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\} \subseteq span_{\mathbb{B}}(Q_a(a'))$$

  meaning that for all $a' \in A$:

$$dim_{\mathbb{B}} \left(\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\} \right) \leq |Q_a(a')|$$

  which proves the wanted.

- (2) $\Rightarrow$ (1): By assumptions this follows directly from Proposition 2.8.

□

Hence by Corollary 2.9 any linear repair scheme can have the form of Algorithm 2.7. However to come up with an algorithm of the same form as Algorithm 2.7 we need a way to come up with the coefficients $\mu_{\zeta_i,a}(a')$. The following lemma will tell us that $\mu_{\zeta_i,a}(a')$ and $C^{\perp}$ is connected.

Lemma 2.10.
Fix a set $A \subseteq \mathbb{F}$ with $|A| = n$, and $\mathbb{B} \leq \mathbb{F}$ where $\mathbb{F}$ has degree t over $\mathbb{B}$ and let $k < n$. Fix $a' \in A$ and numbers $d_a \leq t$ for every $a \in A \backslash \{a'\}$. Let $C \subseteq \mathbb{F}^n$ be an MDS code with evaluation points A. Then the following is equivalent:

1. There is a basis Z for $\mathbb{F}$ over $\mathbb{B}$ and coefficients

$$\{\mu_{\zeta_i,a}(a') | a \in A \backslash \{a'\}, \zeta_i \in Z\}$$

   such that for all $f \in C$ and all $\zeta_i \in Z$

$$\zeta_i f(a') = \sum_{a \in A \backslash \{a'\}} \mu_{\zeta_i,a}(a') f(a) \tag{2.11}$$

   and for all $a \in A \backslash \{a'\}$,

$$dim_{\mathbb{B}} \left(\{\mu_{\zeta_i,a}(a') | \zeta_i \in Z\} \right) = d_a \tag{2.12}$$

2. There is a set $P(a') \subseteq C^\perp$ of size t such that

$$dim_{\mathbb{B}}\Big(p(a')|p \in P(a')\Big) = t$$

and for all $a \in A\backslash\{a'\}$,

$$dim_{\mathbb{B}}\Big(\{p(a)|p \in P(a')\}\Big) = d_a$$

Proof. Both cases follows from Definition 1.28 as by rewriting (2.11) we can get:

$$\left(\sum_{a\in A\backslash\{a'\}} \mu_{\zeta_i,a}(a')f(a)\right) - \zeta_i f(a') = 0. \tag{2.13}$$

1. (1) $\Rightarrow$ (2): Define $p_{\zeta_i} : \mathbb{F} \to \mathbb{F}$ as the following:

$$p_{\zeta_i}(x) = \begin{cases} -\zeta_i & \text{for } x = a' \\ \mu_{\zeta_i,a}(a') & \text{for } x = a \in A\backslash\{a'\} \end{cases}$$

and then let $P(a') = \{p_{\zeta_i}|\zeta_i \in Z\}$. Then as the $\mathbb{F}$ has degree t over $\mathbb{B}$ then

$$dim_{\mathbb{B}}\Big(\{p_{\zeta_i}(a')|p_{\zeta_i} \in P(a')\}\Big) = t$$

as $\zeta_i \in Z$ and Z is a basis of size t. And by assumption from (2.12)

$$dim_{\mathbb{B}}\Big(\{p_{\zeta_i}(a)|p_{\zeta_i} \in P(a')\}\Big) = d_a$$

follows.

2. (2) $\Rightarrow$ (1): Let $P(a') = \{p_1, \ldots, p_t\} \subseteq C^\perp$ where p_i is the codeword $(p_i(a_1), \ldots, p_i(a_n))$ with $deg(p_i) < n - k$ for $1 \leq i \leq n$. Then define $\zeta_i = p_i(a')$ and let $Z = \{\zeta_1, \ldots, \zeta_t\}$ and let $\mu_{\zeta_i,a}(a') = p_i(a)$. Then the rest follows from (2.13) as by assumption $P(a')$ is in the dual code of C.

□

We summarize Corollary 2.9 and Lemma 2.10 in the following theorem.

Theorem 2.11.
Let $\mathbb{B} \leq \mathbb{F}$ be a subfield where $\mathbb{F}$ over $\mathbb{B}$ has degree t and let $A \subseteq \mathbb{F}$ be a set of evaluation points. Let $C \subseteq \mathbb{F}$ be an MDS code of dimension $k < |A|$ with evaluation points A. Then the following are equivalent:

- There is a linear repair scheme for C over $\mathbb{B}$ with bandwidth b.
- For every $a' \in A$ there is a $P(a') \subseteq C^\perp$ of size t so that

$$dim_{\mathbb{B}}\Big(\{p(a')|p \in P(a')\}\Big) = t$$

and the sets $\{p(a)|p \in P(a')\}$ for $a' \neq a$ satisfy

$$b \geq \max_{a'\in A} \sum_{a\in A\backslash\{a'\}} dim_{\mathbb{B}}\Big(\{p(a)|p \in P(a')\}\Big)$$

2.4 A LERS for Reed-Solomon Codes

This section is based on [1].
In Section 2.3 the theory for general MDS LERS was explained, however for the main difficulty was to find the coefficients $\mu_{\zeta,a}(a')$ in Algorithm 2.7 and it still is. It turns out when we limit the search within a specific type of code, in this case Reed-Solomon codes it simplifies.
Proposition 1.31 yields that for a Reed-Solomon code, the dual of the code is a generalized Reed-Solomon code, with coefficients

$$(RS_q(n,k))^\perp = (GRS_q(n,k,(1,\dots,1)))^\perp = GRS_q(n,n-k,(\lambda_1^\perp,\dots,\lambda_n^\perp))$$

where

$$\lambda_j^\perp = \frac{1}{\prod_{i\neq j} a_j - a_i} \tag{2.14}$$

for $\{a_1,\dots,a_n\} = A \subseteq \mathbb{F}$ being the evaluation points of the code. Reed-Solomon codes are polynomial evaluations of the elements in A. However the dual of a RS-code is polynomial evaluations as well, where the polynomial has degree less than $n-k$. The point is that the only requirement on the polynomial is the degree.
The dual of a RS-code is however a GRS-code, where the evaluations has some multipliers. However by using the notation from Lemma 2.10 we have

$$dim_{\mathbb{B}}\left(\lambda_i^\perp p(a_i) | p \in P(a')\right) = dim_{\mathbb{B}}\left(p(a_i) | p \in P(a')\right) = d_{a_i}$$

for $a_i \in A\backslash\{a'\}$, and $\lambda_i^\perp$ is of the form of (2.14), meaning that we can leave them out.
Theorem 2.11 is for general MDS codes, however this specified for Reed-Solomon codes is described by the following Corollary.

Corollary 2.12.
Let $\mathbb{B} \leq \mathbb{F}$ be a sub field, where $\mathbb{F}$ has degree t over $\mathbb{B}$. Let $A \subseteq \mathbb{F}$ be any set of evaluation points. Then the following is equivalent.

1. There is a linear repair scheme for $RS_q(n,k)$ over $\mathbb{B}$ with bandwidth b.
2. For every $a' \in A$ there is a set $P(a') \subseteq \mathbb{F}[X]$ be a set of t polynomials of degree less than $n-k$ so that
$$dim_{\mathbb{B}}\left(\{p(a') | p \in P(a')\}\right) = t$$
and the sets $\{p(a) | p \in P(a')\}$ for $a' = a$ satisfy
$$b \geq \max_{a' \in A} \sum_{a \in A\backslash\{a'\}} dim_{\mathbb{B}}\left(\{p(a) | p \in P(a')\}\right).$$

Moreover, suppose that 2. holds, then the linear repair scheme in 1. is given by Algorithm 2.7, with coefficients:

$$\mu_{\zeta,a}(a') = p(a) \cdot \frac{\prod_{\alpha\in A\backslash\{a'\}}(a'-\alpha)}{\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)}$$

and the basis Z given by

$$Z = \{p(a')|p \in P(a')\} = \{\zeta_1, \ldots, \zeta_t\}$$

Proof. The 1. and 2. is directly specified from Theorem 2.11, and the only thing that is left is the last part.
What we want to verify is that these coefficients satisfy

$$\zeta f(a') = \sum_{\alpha \in A \setminus \{\alpha'\}} \mu_{\zeta,a}(a') f(a).$$

By plugging the formula in, and rearranging we get

$$\sum_{\alpha \in A \setminus \{\alpha'\}} p(a) \cdot f(a) \cdot \frac{\prod_{\alpha \in A \setminus \{a'\}} (a' - \alpha)}{\prod_{\alpha \in A \setminus \{a\}} (a - \alpha)} = \sum_{\alpha \in A \setminus \{\alpha'\}} (pf)(a) \cdot \frac{\prod_{\alpha \in A \setminus \{a'\}} (a' - \alpha)}{\prod_{\alpha \in A \setminus \{a\}} (a - \alpha)} = (pf)(a') = p(a')f(a')$$

and as in Lemma 2.10 we set $p(a') = \zeta$ which by doing this makes the equation true. This makes sense as pf has degree at most $(n - k - 1) + (k - 1) = n - 2$ which makes the Lagrange interpolation doable. □

This does not solve the problem, but it limits the problem to finding a set of polynomials $P(a') \subseteq C^{\perp}$, which can be done different ways.
The following theorem introduces a set of polynomials for when we set $A = \mathbb{F}$ with repair bandwidth $n - 1$ for the LERS scheme given by Corollary 2.9.

Theorem 2.13.
Let $\mathbb{B} \leq \mathbb{F}$ be any subfield of $\mathbb{F}$ with degree t. Let $A = \mathbb{F}$ where $|\mathbb{F}| = n$. Choose $k \leq n(1 - \frac{1}{|\mathbb{B}|})$. Then there is a LERS for $RS_q(n, k)$ with repair bandwidth $n - 1$ over $\mathbb{B}$.

Proof. Before the proof starts we want to choose a set of polynomials $P(a')$ such that the Corollary 2.12 holds where the polynomials in $P(a')$ has degree less than $\frac{n}{|\mathbb{B}|}$ for every $a' \in A$. Fix a basis Z for $\mathbb{F}$ over $\mathbb{B}$ and let t be the degree of $\mathbb{F}$ over $\mathbb{B}$. Without loss of generality assume that $\mathbb{F} = \mathbb{F}_{q^t}$ and $\mathbb{B} = \mathbb{F}_q$ for q being a prime power.
Then we choose

$$P(a') = \left\{ \frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(X - a'))}{X - a'} \mid \zeta_i \in Z \right\}.$$

The polynomials in $P(a')$ has degree at most

$$deg\left(\frac{(\zeta_i(X - a'))^{q^{t-1}}}{X - a'} \right) = |\mathbb{B}|^{t-1} - 1 = \frac{|\mathbb{F}|}{|\mathbb{B}|} - 1 = \frac{n}{|\mathbb{B}|} - 1 < \frac{n}{|\mathbb{B}|}$$

as $|\mathbb{B}|^t = |\mathbb{F}|$. Then for all $a \neq a'$ we have

$$\left\{ p(a) | p \in P(a') \right\} = \left\{ \frac{\beta}{a - a'} | \beta \in \mathbb{B} \right\}$$

as the trace gives an element in $\mathbb{B}$. And in fact

$$dim_{\mathbb{B}} \left(\left\{ \frac{\beta}{a - a'} | \beta \in \mathbb{B} \right\} \right) = 1$$

for all $a \neq a'$.
For $p(a')$ let us expand the trace from $\mathbb{F}$ to $\mathbb{B}$ as follows:

$$tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(X-a')) = (\zeta_i(X-a')) + (\zeta_i(X-a'))^{q^1} + \cdots + (\zeta_i(X-a'))^{q^{t-1}}$$

meaning that

$$\begin{aligned} p(a') &= \frac{(\zeta_i(X-a')) + (\zeta_i^{q^1}(X-a')^{q^1}) + \cdots + (\zeta_i^{q^{t-1}}(X-a')^{q^{t-1}})}{(X-a')}\Big|_{X=a'} \\ &= \zeta_i + (\zeta_i(X-a'))^{q^1-1} + \cdots + (\zeta_i(X-a'))^{q^{t-1}-1}\Big|_{X=a'} = \zeta_i \end{aligned}$$

Thus the following must hold

$$\{p(a')|p \in P(a'))\} = \{\zeta_1, \ldots, \zeta_t\} = Z$$

which is a basis and therefore has full rank. Then the conditions of Corollary 2.12 are satisfied. Therefore we have

$$b = \sum_{a \neq a'} dim_{\mathbb{B}}\{p(a)|p \in P(a')\} = \sum_{a \neq a'} 1 = n-1,$$

where Corollary 2.12 then states there is a LERS for $RS_q(n,k)$ over $\mathbb{B}$ with bandwidth $n-1$. □

Theorem 2.13 gives a set of polynomials $P(a')$ for a LERS with repair bandwidth $n-1$. The Repair Bandwidth from Theorem 2.13 does not depend on k. This means that for maximum encoding, you would want to chose the biggest k possible, ie. $k = n(1-\frac{1}{|\mathbb{B}|})$. For smaller k the repair bandwidth remains the same. For the usual Lagrange Interpolation the k matters, as that is how many points of the polynomial is required to construct the entire polynomial. Therefore for usual RS error correction a smaller k can have an advantage, but for Theorem 2.13 you lose nothing by choosing the biggest k.
We still have yet to give an example of a LERS where a failed server is repaired, and so far we have had trouble defining the coefficients $\mu_{\zeta,a}(a')$ (from Proposition 2.6) and $P(a') \subseteq C^{\perp}$ but with Theorem 2.13 and Corollary 2.12 we actually have both for a $RS_q(n,k)$-code where $A = \mathbb{F}$.

Example 2.14. This example is only for showing that the calculations works, and to show the mathematical steps. How the servers communicate with each other will be shown later.
Let the fields we will be working over be $\mathbb{F} = \mathbb{F}_{2^2}$ and $\mathbb{B} = \mathbb{F}_2$. We will use the irreducible polynomial $X^2 + X + 1$, and let α be a root of it, meaning that $\alpha^2 = \alpha + 1$.
From this we can determine a basis $Z = \{\zeta_1, \zeta_2\} = \{1, \alpha\}$ and the (unique) dual basis $V = \{v_1, v_2\} = \{a+1, 1\}$ (the dual basis was found by Sagemath).
All the elements in $\mathbb{F}$ are the following:

$$\mathbb{F} = \{0, \alpha, \alpha+1, 1\}$$

where we now set $A = \mathbb{F}$ and denote each entry of A as the following:

$$(a_1, a_2, a_3, a_4) = (0, \alpha, \alpha+1, 1).$$

From Theorem 2.13 we have a bound for k where we want to use the largest k possible:

$$k = \lfloor n \cdot \left(1 - \frac{1}{|\mathbb{B}|}\right) \rfloor = \lfloor 4 \cdot \left(1 - \frac{1}{2}\right) \rfloor = 2.$$

Then we want a codeword $c \in RS_{2^2}(4,2)$. It can be checked that

$$c = (c_1, c_2, c_3, c_4) = (1, \alpha, 0, \alpha + 1) \in RS_{2^2}(4,2)$$

with the polynomial $f(X) = \alpha X + 1$. Then c_i is the data for the corresponding a_i for $1 \leq i \leq 4$. We now 'corrupt' the server a_1 such that c_1 is lost. We now want to repair c_1. You could use usual RS code error-correction (Lagrange interpolation), but we want to use the methods for the ERP.
From Theorem 2.13 we have the set of polynomials $P(a')$ we now determine:

$$P(a') = \left\{ \frac{tr_{\mathbb{F}/\mathbb{B}}(1 \cdot (X - a'))}{X - a'}, \frac{tr_{\mathbb{F}/\mathbb{B}}(\alpha \cdot (X - a'))}{X - a'} \right\} = \{p_1, p_2\},$$

where all the $\mu_{\zeta_i,a}(a') = \mu_{\zeta_i,a}(0)$ can be computed for $\zeta_i \in Z, a \in A \backslash \{0\}$.
We will show the computations for $\mu_{\zeta_2,a_3}(a') = \mu_{\alpha,\alpha+1}(0)$ and then give the rest:

$$\begin{aligned}
\mu_{\alpha,\alpha+1}(0) &= p_2(\alpha+1) \cdot \frac{\prod_{\beta \in A \backslash \{a'\}}(0 - \beta)}{\prod_{\beta \in A \backslash \{a\}}(\alpha + 1 - \beta)} \\
&= p_2(\alpha+1) \cdot \frac{(\alpha)(\alpha+1)(1)}{(\alpha+1-0)(\alpha+1-\alpha)(\alpha+1-1)} \\
&= p_2(\alpha+1) = \frac{tr_{\mathbb{F}/\mathbb{B}}(\alpha \cdot (\alpha+1-0))}{\alpha+1-0} \\
&= \frac{tr_{\mathbb{F}/\mathbb{B}}(1)}{\alpha+1} = \frac{1^{2^0} + 1^{2^1}}{\alpha+1} = \frac{0}{\alpha+1} = 0.
\end{aligned}$$

Then all the $\mu_{\zeta,a}(0)$ are the following:

$\mu_{\zeta_i,a_i}(0)$	$\zeta_1 = 1$	$\zeta_2 = \alpha$
$a_1 = 0$	1	α
$a_2 = \alpha$	$\alpha + 1$	$\alpha + 1$
$a_3 = \alpha + 1$	α	0
$a_4 = 1$	0	1

Table 2.4: The $\mu_{\zeta,a}(0)$ for every server.

Then from (2.7) we compute the following sums:

$$\sum_{i=1, i \neq 1}^{4} tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta_1,a_i}(0) \cdot c_i) = tr_{\mathbb{F}/\mathbb{B}}((\alpha+1) \cdot \alpha) + tr_{\mathbb{F}/\mathbb{B}}(\alpha \cdot (\alpha+1)) + tr_{\mathbb{F}/\mathbb{B}}(0 \cdot 1) = 0, \quad (2.15)$$

$$\sum_{i=1, i \neq 1}^{4} tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta_2,a_i}(0) \cdot c_i) = tr_{\mathbb{F}/\mathbb{B}}((\alpha+1) \cdot \alpha) + tr_{\mathbb{F}/\mathbb{B}}(0 \cdot (\alpha+1)) + tr_{\mathbb{F}/\mathbb{B}}(1 \cdot 1) = 1. \quad (2.16)$$

These values are $tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot c_1)$ for $i = 1, 2$, which is exactly what we need for (2.10). Using (2.10) we have the following computation for $f(a')$:

$$c_1 = 0 \cdot v_1 + 1 \cdot v_2 = 0 + 1 = 1$$

which restores c_1 correctly.

In this example all the calculations were open, where everyone could see anything. Normally this is not the case, and computations are done locally. Ideally you want to send the minimum amount of informations around such that the computations remains computable.
They all had to communicate for both (2.15) and (2.16) and then send those values to a replacement server. That is 6 elements that was send, which is more than $n-1=4-1=3$ that Theorem 2.13 guarantees. To reach 3, we have to be smart.

2.4.1 The Reconstruction Procedure with Repair Bandwidth $n-1$ for RS Codes with $A=\mathbb{F}$

This section is based on [1].
This procedure is based on Theorem 2.13. The objective is to construct a LERS to reconstruct any failed server with repair bandwidth $n-1$. For RS-codes and $A=\mathbb{F}$ with the set of polynomials that Theorem 2.13 yields ie. the following:

$$P(a')=\left\{\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(X-a'))}{X-a'} \mid \zeta_i \in Z\right\}.$$

These were proven to satisfy (for a fixed $a' \in A$)

$$dim_{\mathbb{B}}\{p(a')|p \in P(a')\}=t \tag{2.17}$$
$$dim_{\mathbb{B}}\{p(a)|p \in P(a')\}=1, \quad a \in A\backslash\{a'\} \tag{2.18}$$

where Corollary 2.9 yields a formula to describe the coefficients $\mu_{\zeta_i,a}(a')$

$$\mu_{\zeta,a}(a')=p(a)\cdot\frac{\prod_{\alpha\in A\backslash\{a'\}}(a'-\alpha)}{\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)}.$$

In the case for $A=\mathbb{F}$ we have that the product:

$$\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)=\prod_{\beta\in\mathbb{F}^*}\beta$$

is the product of all non-zero elements in the field $\mathbb{F}$ for all $a \in A$. This means that

$$\frac{\prod_{\alpha\in A\backslash\{a'\}}(a'-\alpha)}{\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)}=\frac{\prod_{\beta\in\mathbb{F}^*}\beta}{\prod_{\beta\in\mathbb{F}^*}\beta}=1.$$

This is only the case when $A=\mathbb{F}$.
From (2.18) as the dimension for $a \in A\backslash\{a'\}$ is 1 this means that they only differ by a constant scaling $b \in \mathbb{B}$. This means that by only knowing one $\mu_{\zeta_i,a}(a')$ then the $\mu_{\zeta_j,a}(a')$ can be computed as follows:

$$\mu_{\zeta_j,a}(a')=\mu_{\zeta_i,a}(a')\cdot b \Rightarrow \frac{\mu_{\zeta_j,a}(a')}{\mu_{\zeta_i,a}(a')}=b$$
$$\Rightarrow \frac{\mu_{\zeta_j,a}(a')}{\mu_{\zeta_i,a}(a')}=\frac{\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_j(a-a'))}{a-a'}\cdot\frac{\prod_{\alpha\in A\backslash\{a'\}}(a'-\alpha)}{\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)}}{\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(a-a'))}{a-a'}\cdot\frac{\prod_{\alpha\in A\backslash\{a'\}}(a'-\alpha)}{\prod_{\alpha\in A\backslash\{a\}}(a-\alpha)}}=\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_j(a-a'))}{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(a-a'))}$$
$$\Rightarrow \mu_{\zeta_j,a}(a')=\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_j(a-a'))}{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(a-a'))}\cdot\mu_{\zeta_i,a}(a'). \tag{2.19}$$

As the trace goes from $\mathbb{F} \to \mathbb{B}$ then $\frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_j(a-a'))}{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(a-a'))} \in \mathbb{B}$. However one problem that can occur is if $\mu_{\zeta_i,a}(a') = 0$ but this we can come around by simply requiring each server to sends a query computed by a non-zero $\mu_{\zeta_i,a}(a')$. The above is all the necessary theory we need to start explaining the DDS and how they can reconstruct the data on exactly one server.
As explained in Section 2.2 the situation is as follows: let $\mathbb{F} = \mathbb{F}_{q^t}$ be an extension field over $\mathbb{B} = \mathbb{F}_q$ where the degree of $\mathbb{F}$ over $\mathbb{B}$ is t for q being a prime power and $t \in \mathbb{N}$. Let Z be a basis for $\mathbb{F}$ over $\mathbb{B}$ and let V denote the unique dual basis. Then set $A = \mathbb{F}$ and let there be $n = |A|$ servers, where each server is associated with an element in A. The code we will use is $RS_{q^t}(n,k)$ codes, for some k satisfying Theorem 2.13. Every server has its own evaluation point $a \in A$ for the RS codewords which means they are an evaluation of some polynomial of degree of at most k. We now have to set up the LERS (from definition 2.2) such that we can reconstruct any failed server.
Now let $a_1, \ldots, a_n$ denote the n servers. Then let

$$c = (c_1, \ldots, c_n) = (f(a_1), \ldots, f(a_n))$$

be a codeword in $RS_{q^t}(n,k)$ (meaning $\deg(f) < k$) which we now distribute across all the servers, such that a_i holds $f(a_i) = c_i$ for $1 \leq i \leq n$.
From Definition 2.2 each server has to hold some queries $Q_a(a')$ for any possible a' such that when any server crashes then a reconstruction protocol can be initiated. These queries for this specific case are as follows:

$$Q_a(a') = \begin{cases} \{\mu_{\zeta_j,a}(a') | j = \min_{1\leq i\leq t} \mu_{\zeta_i,a}(a') \neq 0\} & \text{if one } \mu_{\zeta,a}(a') \neq 0 \\ \{0\} & \text{if } \mu_{\zeta_1,a}(a') = \cdots = \mu_{\zeta_t,a}(a') = 0 \end{cases}$$

Since the queries $Q_a(a')$ does not depends on the codeword, only on A and Z all the servers can compute these. Then we let the queries be the first nonzero $\mu_{\zeta_i,a}(a')$ when we take $i = 1, 2 \ldots, t$. There is only the need of one as by (2.19) the rest can be computed if $\mu_{\zeta,a}(a') \neq 0$.
For the linear reconstruction algorithm that is also required for a LERS, we need to calculate t sums

$$\sum_{a\in A\setminus\{a'\}} tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a')f(a)\right) = tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_i f(a')\right) \tag{2.20}$$

for all $\zeta_i \in Z$ where from Algorithm 2.7 where we then use (2.20) as coefficients with $v_i \in V$ to reconstruct $f(a')$.
These two points is enough to define a LERS scheme for $RS_{q^t}(n,k)$. So for the reconstruction to begin we do the following:

1. Let
$$c = (c_1, \ldots, c_n) = (f(a_1), \ldots, (f(a_n))) \in RS_{q^t}(n,k)$$
where each server a_i holds $f(a_i)$ and the queries with both Z and V as the following table shows,

Servers	Data	Additional information stored
a_1	$\{f(a_1)\}$	$Q_{a_1}(a'), \forall a' \in A\backslash\{a_1\}, V, Z, A$
a_2	$\{f(a_2)\}$	$Q_{a_2}(a'), \forall a' \in A\backslash\{a_2\}, V, Z, A$
$\vdots$	$\vdots$	$\vdots$
a_n	$\{f(a_n)\}$	$Q_{a_n}(a'), \forall a' \in A\backslash\{a_n\}, V, Z, A$

Table 2.5: The distribution of data for each server.

2. Now let an arbitrary server $a_k \in A$ crash and initiate the LERS to reconstruct a_k's data which then the replacement server will store.

3. Every server $a \in A\backslash\{a_k\}$ then computes the following:
$$tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a_k)f(a)\right) \tag{2.21}$$
where $\mu_{\zeta_i,a}(a_k) \in Q_{a_i}(a_k)$.

4. Install the basic information on a' which is Z, V, A and then a' downloads (2.21) for $a \in A\backslash\{a_k\}$ (a' keeps track on what was downloaded from every server).

5. Then for every downloaded information from $a \in A\backslash\{a_k\}$ a' computes $tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_j,a}(a')f(a)\right)$ for $i \neq j$ by the following:
$$b_{i,j} = \frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_j(a-a'))}{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(a-a'))} \in \mathbb{B}$$
where ζ_i is the ζ used in (2.21) where $b_{i,j} \in \mathbb{B}$. The coefficients $b_{i,j}$ denotes the transformation from $\mu_{\zeta_i,a}(a')$ to $\mu_{\zeta_j,a}(a')$. As the trace from $\mathbb{F}$ to $\mathbb{B}$ is $\mathbb{B}$ linear then a' can compute the following:
$$b_{i,j} \cdot tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a_k)f(a)\right) = tr_{\mathbb{F}/\mathbb{B}}\left(b_{i,j} \cdot \mu_{\zeta_i,a}(a_k)f(a)\right) = tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_j,a}(a_k)f(a)\right) \tag{2.22}$$
for all $a \in A\backslash\{a'\}$ and $\zeta_j \in Z$.

6. By construction of $\mu_{\zeta,a}(a_k)$ they satisfy
$$\sum_{a\in A\backslash\{a_k\}} tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_i,a}(a_k)f(a)\right) = tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_i f(a_k)\right) \tag{2.23}$$
which the server a' can compute for all $\zeta_i \in Z$.

7. After the previous step then a' holds all necessary information for the linear reconstruction algorithm:
$$\sum_{i}^{t} tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_i f(a_k)\right) v_i = f(a_k) \tag{2.24}$$
where $v_i \in V$ which reconstructs the data correctly.

The key point here is that a' only had to download $tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta,a_i}(a_k) \cdot f(a)\right) \in \mathbb{B}$ and from there could compute the remaining itself. That is one element from every $a \in A\backslash\{a_k\}$ (which Theorem 2.13 also states). This is the procedure to reconstruct any failed server correctly for RS codes with $A = \mathbb{F}$. This will be shown in detail in the following example.

Example 2.15. This will be a full example with distrubution, what each server holds etc. We will do Example 2.14 with proper step this time. To recap we have the following:

$$\mathbb{F} = \mathbb{F}_{2^2} \qquad \mathbb{B} = \mathbb{F}_2 \qquad \alpha^2 = \alpha + 1$$
$$Z = \{1, \alpha\} \qquad V = \{a + 1, 1\} \qquad A = \{0, \alpha, \alpha + 1, 1\}$$

where we have the following codeword:

$$c = (1, \alpha, 0, \alpha + 1) \in RS_{2^2}(4, 2).$$

This codeword is now distributed to the servers along with the additional information:

Servers	Data	Additional information stored
$a_1 = 0$	$\{1\}$	$Q_{a_1}(a'), \forall a' \in A \backslash \{a_1\}, V, Z, A$
$a_2 = \alpha$	$\{\alpha\}$	$Q_{a_2}(a'), \forall a' \in A \backslash \{a_2\}, V, Z, A$
$a_3 = \alpha + 1$	$\{0\}$	$Q_{a_3}(a'), \forall a' \in A \backslash \{a_3\}, V, Z, A$
$a_4 = 1$	$\{\alpha + 1\}$	$Q_{a_4}(a'), \forall a' \in A \backslash \{a_4\}, V, Z, A$

Table 2.6: The information that each server holds.

where we have to calculate all the queries. We computed all the queries for when a_1 crashed in Example 2.14 which can be found in Table 2.4. The procedure for the others are the same, but the calculations will be skipped. All the $\mu_{\zeta,a}(a')$ for all $\zeta \in Z, a, a' \in A$ can be seen in the following table.

$a' = a_1$	$\zeta_1 = 1$	$\zeta_2 = \alpha$	$a' = a_2$	$\zeta_1 = 1$	$\zeta_2 = \alpha$
$a_1 = 0$	1	α	a_1	$\alpha + 1$	$\alpha + 1$
$a_2 = \alpha$	$\alpha + 1$	$\alpha + 1$	a_2	1	α
$a_3 = \alpha + 1$	α	0	a_3	0	1
$a_4 = 1$	0	1	a_4	α	0
$a' = a_3$	$\zeta_1 = 1$	$\zeta_2 = \alpha$	$a' = a_4$	$\zeta_1 = 1$	$\zeta_2 = \alpha$
a_1	α	0	a_1	0	1
a_2	0	1	a_2	α	0
a_3	1	α	a_3	$\alpha + 1$	$\alpha + 1$
a_4	$\alpha + 1$	$\alpha + 1$	a_4	1	α

Table 2.7: All the $\mu_{\zeta,a}(a')$ for all $\zeta \in Z, a, a' \in A$.

We then pick the first nonzero entry for all cases of $a' = a_1, a_2, a_3, a_4$ from the left to get the following queries:

$Q_a(a')$	$a' = a_1$	$a' = a_2$	$a' = a_3$	$a' = a_4$
$a = a_1$	$\emptyset$	$\alpha + 1$	α	1
$a = a_2$	$\alpha + 1$	$\emptyset$	1	α
$a = a_3$	α	1	$\emptyset$	$\alpha + 1$
$a = a_4$	1	α	$\alpha + 1$	$\emptyset$

Table 2.8: The queries $Q_a(a')$ for $a' \in A, a \in A \backslash \{a'\}$.

Every row in Table 2.8 is now the queries every $a \in A$ holds.
As in Example 2.14 server a_1 now crashes and the LERS reconstruction now begins.
Let a' denote the replacement server for a_1. Every server now computes the following:

Server	Computations
a_2	$tr_{\mathbb{F}/\mathbb{B}}\left((\alpha+1)\cdot\alpha\right) = tr_{\mathbb{F}/\mathbb{B}}\left(1\right) = 1^{2^0} + 1^{2^1} = 0$
a_3	$tr_{\mathbb{F}/\mathbb{B}}\left(\alpha\cdot 0\right) = tr_{\mathbb{F}/\mathbb{B}}(0) = 0$
a_4	$tr_{\mathbb{F}/\mathbb{B}}\left(1\cdot(\alpha+1)\right) = (\alpha+1)^{2^0} + (\alpha+1)^{2^1} = 1$

Table 2.9: The elements in $\mathbb{B}$ which a' downloads from a_2, a_3, a_4.

These values a' now downloads. Now the job of a_2, a_3, a_4 is done and the rest can be computed by a'. By (2.22) a' can compute the rest. Since the queries does not depend on the codeword c, a' can compute Table 2.7 himself. Then by the method of choosing the queries a' knows what ζ was used and can then compute the remaining $tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta,a}(a_1)f(a))$. Notice in this case both a_2, a_3 sends a zero, so this will always remain a 0, hence we will only show the calculations for a_4.

$$tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_1,a_4}(a_1)f(a_4)\right) = \frac{tr_{\mathbb{F}/\mathbb{B}}(1\cdot(1-0))}{tr_{\mathbb{F}/\mathbb{B}}(\alpha\cdot(1-0))}\cdot 1 = \frac{tr_{\mathbb{F}/\mathbb{B}}(1)}{tr_{\mathbb{F}/\mathbb{B}}(\alpha)} = \frac{0}{1} = 0.$$

Hence by (2.23) a' can compute $tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_i f(a_1)\right)$ for $i = 1, 2$ using the following values:

Server	$tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_1,a}(a_1)\cdot f(a)\right)$	$tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta_2,a}(a_1)\cdot f(a)\right)$
a_2	0	0
a_3	0	0
a_4	0	1

Table 2.10: The $tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_i \cdot f(a)\right)$ for $i = 1, 2$. The blue values is the received ones and the green is the computed ones.

Hence by (2.23):

$$tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_1 f(a_1)\right) = 0 + 0 + 0 = 0$$
$$tr_{\mathbb{F}/\mathbb{B}}\left(\zeta_2 f(a_1)\right) = 0 + 0 + 1 = 1$$

which by (2.24) then can reconstruct $f(a_1)$ by

$$f(a_1) = 0 \cdot v_1 + 1 \cdot v_2 = 1 \cdot 1 = 1$$

which is the correct reconstruction.

All the servers has access to Z, A and therefore $P(a')$, which means they all can compute the

$\mu_{\zeta,a}(a')$ for all $a \in A, \zeta \in Z$ by the polynomials in $P(a')$. The codeword that is distributed between the servers is not mentioned anywhere. This indicated that the queries is not dependent on the codeword which allows the queries can be reused for different codeword.
As servers initiate a reconstruction procedure, each server computes the following

$$tr_{\mathbb{F}/\mathbb{B}}\left(\mu_{\zeta,a}(a')f(a)\right)$$

which the replacement server downloads. That value depends on the codeword, but the queries does not. This is a good thing, as that allows multiple codewords to be distributed between servers, and they can use their queries for all of them.
One thing that we have not mentioned in a while is Definition 2.4 ie. Repair Locality. However it is redundant in the sense that since $|Q_a(a')| = 1$ the Repair Locality is equal the Repair Bandwidth. This means that all the servers are required to respond, but as all servers has to send 1 element, that is expected.

2.5 Lower Bound for Repair Bandwidth for MDS codes

This section is based on [1].
In Theorem 2.13 we proved that for certain conditions we could make a LERS for $RS_q(n,k)$ codes with bandwidth $n-1$. This seems good, as to repair a server, the new server only had to download one element from all servers.
The question is, is that the best you can do? The following theorem will state a lower bound on the repair bandwidth, and it turns out that Theorem 2.13 is good!

Theorem 2.16.
Let C be a $[n,k]$ MDS-code with evaluation points $A = \{a_1, \ldots, a_n\}$ over a field $\mathbb{F}$. Let $\mathbb{B} \leq \mathbb{F}$ be a subfield, where the degree of $\mathbb{F}$ over $\mathbb{B}$ is t. Any LERS for C over $\mathbb{B}$ must have bandwidth (measures in subsymbols of $\mathbb{B}$) at least

$$b \geq (n-1)\log_{|\mathbb{B}|}\left(\frac{n-1}{n-k}\right)$$

Proof. Fix an arbitrary $a' \in A$. Then we consider the LERS which can repair $f(a')$ for

$$(f(a_1), \ldots, f(a'), \ldots, f(a_n)) = c \in C, deg(f) < k.$$

Then from Theorem 2.11 we know that there exists a set $P \subseteq C^{\perp}$ of size t with

$$dim_{\mathbb{B}}\left(\{p(a')|p \in P\}\right) = t, \tag{2.25}$$

$$dim_{\mathbb{B}}\left(\{p(a)|p \in P\}\right) = d_a \tag{2.26}$$

with repair bandwidth

$$b = \sum_{a \in A \setminus \{a'\}} d_a \tag{2.27}$$

as a' is fixed, so there is no need to take the max of all $a' \in A$.
Then let $x \in \mathbb{B}^t$ and let $p_x : \mathbb{F} \to \mathbb{F}$ be the function given as

$$p_x(X) = \sum_{p \in P} x_p \cdot p(X)$$

where x_p denotes the entry corresponding to $p \in P$ ie. every function $p \in P$ has a index in x. Then we introduce the following set

$$S_a = \{x \in \mathbb{B}^t | p_x(a) = 0\}$$

which by (2.26) is a vector space over $\mathbb{B}$ of dimension $t - d_a$ by the dimension theorem. Consider the average amount of non-zero vectors in S_a as follows:

$$\frac{1}{|\mathbb{B}|^t - 1} \sum_{x \neq 0, x \in \mathbb{B}} \Big| \{a \in A \backslash \{a'\} | x \in S_a\} \Big| = \frac{1}{|\mathbb{B}|^t - 1} \sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{t - d_a}$$
$$= \frac{1}{|\mathbb{B}|^t - 1} \sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^t |\mathbb{B}|^{-d_a} = \frac{|\mathbb{B}|^t}{|\mathbb{B}|^t - 1} \sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{-d_a} = \frac{|\mathbb{F}|}{|\mathbb{F}| - 1} \sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{-d_a} := r.$$

As r is the average amount of vectors in S_a for $a \in A \backslash \{a'\}$ then that also means that there exists a $\bar{x} \in \mathbb{B}^t$ such that

$$|\{a | \bar{x} \in S_a\}| \geq r$$

from where we now define

$$\bar{p}(X) = p_{\bar{x}}(X) = \sum_{p \in P} \bar{x}_p \cdot p(X). \tag{2.28}$$

From the way we chose $\bar{x}$ then $\bar{p}$ has to have at least r zeros of $a \in A \backslash \{a'\}$, however $\bar{p}$ also has to be non-zero as otherwise

$$\bar{p}(a') = \sum_{p \in P} \bar{x}_p \cdot p(a') = 0$$

which contradicts (2.25) as $p(a')$ has full rank (more specifically the set (2.25) is linearly independent).
Since $\bar{p} \in C^\perp$ and C is a MDS code, then so is $C^\perp$. As $S_a \subseteq (C^\perp)^\perp = C$ meaning that

$$r \leq d = n - k + 1 \Rightarrow r < n - k$$

which implies that

$$\sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{-d_a} < n - k. \tag{2.29}$$

Combining (2.29) with (the minimum of) (2.27) we want to minimize:

$$b \geq \min_{d_a \in [0,t]} \sum_{a \in A \backslash \{a'\}} d_a \tag{2.30}$$
$$\sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{-d_a} < n - k \tag{2.31}$$

To minimize this problem, we assume now that $d_a \in \mathbb{R}$. However we only need an approximate solution to this, therefore instead of (2.31) we will assume that

$$\sum_{a \in A \backslash \{a'\}} |\mathbb{B}|^{-d_a} \leq n - k.$$

For this we need Jensen's inequality:

$$\phi \left(\frac{\sum_{i=1}^n x_i}{n} \right) \leq \frac{\sum_{i=1}^n \phi(x_i)}{n} \tag{2.32}$$

for ϕ being a convex function. For this inequality we will use the following function

$$\phi(x) = |\mathbb{B}|^{-x}$$

as by differentiating $\phi(x)$ twice we have

$$\phi''(x) = \frac{(\ln(|\mathbb{B}|)^2)}{|\mathbb{B}|^x}$$

where $\phi(x) \geq 0$ for all x, which shows that $\phi(x)$ is convex.
Therefore consider

$$|\mathbb{B}|^{-\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1}} = \phi\left(\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1}\right) \leq \frac{\sum_{a\in A\setminus\{a'\}} \phi(d_a)}{n-1} = \frac{\sum_{a\in A\setminus\{a'\}} |\mathbb{B}|^{-d_a}}{n-1}$$

which by multiplying with $(n-1)$ we have the following

$$(n-1)|\mathbb{B}|^{-\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1}} \leq \sum_{a\in A\setminus\{a'\}} |\mathbb{B}|^{-d_a} \leq n-k. \tag{2.33}$$

By then using the left and right part of (2.33) and divide by $(n-1)$ again we get:

$$|\mathbb{B}|^{-\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1}} \leq \frac{n-k}{n-1}.$$

By then using $\log_{|\mathbb{B}|}$ on both sides and then multiplying with $-(n-1)$ we get:

$$\log_{|\mathbb{B}|}\left(|\mathbb{B}|^{-\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1}}\right) = -\sum_{a\in A\setminus\{a'\}} \frac{d_a}{n-1} \leq \log_{|\mathbb{B}|}\left(\frac{n-k}{n-1}\right)$$
$$\Rightarrow \sum_{a\in A\setminus\{a'\}} d_a \geq (n-1)\log_{|\mathbb{B}|}\left(\frac{n-1}{n-k}\right).$$

As this is true for all $d_a \in \mathbb{R}$, especially also for $d_a \in [0,t]$, then it must be a solution to (2.30) which yields:

$$b \geq (n-1)log_{|\mathbb{B}|}\left(\frac{n-1}{n-k}\right)$$

as we wanted to show. □

With a lower bound given, we now have a measure of how good our LERS are with respect to repair bandwidth.
With Theorem 2.13 in mind, let us assume $\mathbb{B} = \mathbb{F}_p, \mathbb{F} = \mathbb{F}_{p^t}$ and $n = |\mathbb{F}| = p^t$. Then from Theorem 2.13 assume that $k = n(1-\frac{1}{p}) = n - \frac{n}{p}$ from which we can rewrite the lower bound from Theorem 2.16 as

$$(n-1)\cdot\log_p\left(\frac{n-1}{n-n+\frac{n}{p}}\right) = (n-1)\cdot\log_p\left(\frac{n-1}{\frac{n}{p}}\right)$$
$$= (n-1)\cdot\log_p\left(p\cdot\frac{(n-1)}{n}\right) = (n-1)\left(\log_p(p) + \log_p\left(\frac{n-1}{n}\right)\right)$$
$$= (n-1)\left(1+\log_p\left(\frac{n-1}{n}\right)\right)$$

We know that as n grows, then

$$\lim_{n\to\infty} \log_p\left(\frac{n-1}{n}\right) = 0$$

which means for a big n we are actually approaching repair bandwidth $n-1$ that Theorem 2.13 states. Recall that $n = p^t$ where p is a prime number (that is a positive integer) and $t \in \mathbb{N}$ meaning that as either of them grows the repair bandwidth from Theorem 2.13 is approached.
In Example 2.14, 2.15 we had the following values $p = 2, t = 2, n = 4, k = 2$ in which the lower bound for repair bandwidth yields:

$$\lceil (4-1)\cdot \log_2\left(\frac{4-1}{4-2}\right)\rceil = \lceil (3)\cdot \log_2\left(\frac{3}{2}\right)\rceil \approx \lceil 1.75\rceil = 2$$

which is the promised $n-1 = 4-1 = 3$ that Theorem 2.13 states. A decimal repair bandwidth does not make sense, hence we are rounding up to the nearest integer as rounding down would contradict it being a lower bound.
On one hand a difference between the lower bound and what Theorem 2.13 yields is 1 which is not to bad. On the other hands its $3/2 = 1.5$ which means that repair bandwidth that Theorem 2.13 gives is 1.5 times bigger than what could be done. However as shown this is better for bigger p or t, or at least approach the lower bound.
If we then increased t to $t = 10$ such that $n = 2^{10} = 1024, k = 2^{10} - \frac{2^{10}}{2} = 512$ we get the following:

$$\lceil (1024-1)\cdot \log_2\left(\frac{1024-1}{1024-512}\right)\rceil = \lceil 1023\cdot \log_2\left(\frac{1023}{512}\right)\rceil \approx \lceil 1021.56\rceil = 1022$$

where again the difference is 1, but this time we also have

$$\frac{1023}{1022} \approx 1.00098$$

which is a lot better.
With sage, we can easily print out some tables to compare which is done with the following code:

```
sage: def LBlist(p,t):
sage:     q=p^t
sage:     n=q
sage:     k=floor(n*(1-1/p))
sage:     LB=ceil((n-1)*log((n-1)/(n-k),p).numerical_approx())
sage:     return([p,q,n,k,n-1,LB,(n-1)-LB, (n-1)/LB])

sage: OP=[['p','q','n', 'k', 'b', 'LB', 'LB diff','Ratio']]
sage: for i in range(2,10):
sage:     OP.append(LBlist(2,i))

sage: table(OP)
p   q     n     k     b     LB    LB diff  Ratio
2   4     4     2     3     2     1        3/2
2   8     8     4     7     6     1        7/6
2   16    16    8     15    14    1        15/14
2   32    32    16    31    30    1        31/30
2   64    64    32    63    62    1        63/62
2   128   128   64    127   126   1        127/126
```

```
2   256   256   128   255   254   1        255/254
2   512   512   256   511   510   1        511/510
```

This tells us that for $p = 2$, $t = 2, \ldots, 9$ we will continue to have a difference of 1 but the Ratio will go towards 1.
The same can be done of other prime numbers p, which is the following is for $p = 5, t = 2, \ldots, 9$:

```
sage: OP=[['p','q','n', 'k', 'b', 'LB', 'LB diff','Ratio']]
sage: for i in range(2,10):
sage:   OP.append(LBlist(5,i))

sage: table(OP)

p   q         n         k         b         LB        LB diff  Ratio
5   25        25        20        24        24        0        1
5   125       125       100       124       124       0        1
5   625       625       500       624       624       0        1
5   3125      3125      2500      3124      3124      0        1
5   15625     15625     12500     15624     15624     0        1
5   78125     78125     62500     78124     78124     0        1
5   390625    390625    312500    390624    390624    0        1
5   1953125   1953125   1562500   1953124   1953124   0        1
```

This actually hits the lower bound for the repair bandwidth. We checked for other primes numbers as well with the following code.

```
sage: P=Primes()
sage: m=0; l=0
sage: for j in range(2,6):
sage:   t=j
sage:   for i in range(100):
sage:     p=P.unrank(i)
sage:     q=p^t; n=q; k=n*(1-1/p)
sage:     LB=ceil((n-1)*log((n-1)/(n-k),p).numerical_approx())
sage:     if (n-1)/LB!=1:
sage:       print 'Prime',p,'is not optimal with t=',j,'as the fraction is',(n-1)/LB
sage:       l=l+1
sage:     else:
sage:       m=m+1
sage: print 'Nonoptimal schemes:',l
sage: print 'Optimal schemes:',m
Prime 2 is not optimal with t= 2 as the fraction is 3/2
Prime 2 is not optimal with t= 3 as the fraction is 7/6
Prime 2 is not optimal with t= 4 as the fraction is 15/14
Prime 2 is not optimal with t= 5 as the fraction is 31/30
Nonoptimal schemes: 4
Optimal schemes: 396
```

This code tests the first 100 prime numbers for $t = 2, 3, 4, 5$ and compare the repair bandwidth given by Theorem 2.13 with the lower bound from Theorem 2.16. We also tried for larger t and more prime numbers, but after this point (almost) every case was non-optimal. We think it is

an approximation error from Sagemath.
This indicated that Theorem 2.13 gives an LERS with the optimal repair bandwidth for other prime values than $p = 2$.

Chapter 3

Conclusion

3.1 Conclusion

The goal of this project was to examine the exact repair problem and how it was applied in data storage systems. It turned out that it was relevant when you had a data file distributed across multiple servers, also called a data distribution storage system. We used linear codes as some codes is specifically constructed for error-correction, and most type of codes already has error-correction algorithms. However the exact repair problem was to only reconstruct one entry of a damaged codeword (assuming only one entry was damaged). Therefore the usual error-corrections algorithms no longer apply, as those reconstruct the entire codeword. By combining the field trace, and a dual basis we were able to create such a scheme to repair a single entry of a codeword by using information contained in the remaining correct ones.
With Theorem 2.16 we proved that a LERS for a MDS code at least had repair bandwidth

$$b \geq (n-1)\log_{|\mathbb{B}|}\left(\frac{n-1}{n-k}\right),$$

and with Theorem 2.13 we could make a LERS with bandwidth $n-1$. By examining the lower bound we found out:

$$b \geq (n-1)\left(1+\log_p\left(\frac{n-1}{n}\right)\right)$$

which with for n growing goes toward $n-1$. This means that Theorem 2.13 is an optimal LERS in the sense that its repair bandwidth is the minimum.
Theorem 2.13 showed that for RS-codes we could create a LERS when $A = \mathbb{F}$, by determining a set of polynomials which satisfied Corollary 2.9. This set of polynomials were defined using the trace from $\mathbb{F} \to \mathbb{B}$ as

$$P(a') = \left\{ \frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(X-a'))}{X-a'} \mid \zeta_i \in Z \right\}.$$

for $Z = \{\zeta_1, \ldots, \zeta_t\}$.
This was all we needed, as for $p \in P(a')$ we could define the coefficients $\mu_{\zeta,a}(a')$ for RS-codes as the following:

$$\mu_{\zeta,a}(a') = p(a) \cdot \frac{\prod_{\alpha \in A\setminus\{a'\}}(a'-\alpha)}{\prod_{\alpha \in A\setminus\{a\}}(a-\alpha)}.$$

Then in Theorem 2.13 the key point was that $\mu_{\zeta_i,a}(a')$ had dimension 1, which meant that $\mu_{\zeta_j,a}(a') = \mu_{\zeta_i,a}(a') \cdot b, b \in \mathbb{B}$. Finding this b was something all servers could do, as this only

depended on the basis Z and the evaluation points A. Hence by downloading one $\mu_{\zeta,a}(a')$ they could compute the rest. This was ideal, as that meant by using the property that $tr_{\mathbb{F}/\mathbb{B}}$ is $\mathbb{B}$-linear then every server could compute

$$\sum_{a \in A \setminus \{a'\}} tr_{\mathbb{F}/\mathbb{B}} \left(\mu_{\zeta_i,a}(a') f(a) \right) = tr_{\mathbb{F}/\mathbb{B}} \left(\zeta_i f(a') \right)$$

for all $\zeta_i \in Z$. Then they could use the unique dual basis $V = \{v_1, \ldots, v_t\}$ to

$$\sum_{i}^{t} tr_{\mathbb{F}/\mathbb{B}} \left(\zeta_i f(a') \right) v_i = f(a') \tag{3.1}$$

which reconstructed the lost entry. All of this, by downloading one element in the subfield $\mathbb{B}$ from the remaining servers.

3.2 Danish Resumé

Målet i dette projekt var at undersøge "The Exact Repair Problem" (ERP), og hvordan det er relateret til data opbevaring. Dette bliver brugt i "Data Distribution Storage" (DDS) systemer, som er et opbevarings system, hvor en stor data fil bliver fordelt ud på flere servere. Dette sikrer at hvis en server går ned, og deraf mister alt data på serveren, mister man hele sit data. Derimod kun en del af sin data. Hvis man havde gemt alt sit data på en server, og den server går ned så ville man miste alt sit data. Så ideen med at fordele sit data ud er kan sikre at man ikke mister alt, i tilfældet af en server går ned.
I [1] præsenteres en "Linear Exact Repair Scheme" (LERS) for Reed-Solomon (RS) koder, med længde og dimension på henholdsvis n og k, som kan med "Repair Bandwidth" $n-1$ reparer en vilkårlig indgang i et kodeord i en RS kode.
Før alt dette giver mening var vi nød til at undersøge Galois teori for at få definitionen af trace mellem endelige legemer $tr_{\mathbb{F}/\mathbb{B}}$ hvor $\mathbb{F}$ er en udvidelse af $\mathbb{B}$, altså $\mathbb{B} \leq \mathbb{F}$. $tr_{\mathbb{F}/\mathbb{B}}$ er en afbildning fra $\mathbb{F} \to \mathbb{B}$ som vi skulle bruge den til senere.
Siden $\mathbb{F}$ er en udvidelse af $\mathbb{B}$ kunne vi se $\mathbb{F}$ som et vektorrum over $\mathbb{B}$ (og deraf kunne vi se lineær koder over $\mathbb{F}$ som et vektorrum over $\mathbb{B}$).
Det viste sig at alle de lineære transformationer fra $\mathbb{F}_{q^t}$ til $\mathbb{F}_q$, hvor $q = p^r$ og p er et primtal og $r, t \in \mathbb{N}$, kunne skrives som $L_\gamma(x) = tr_{\mathbb{F}/\mathbb{B}}(\gamma \cdot x)$ og at $L_\gamma \neq L_\beta$ hvis $\gamma \neq \beta$ for $\gamma, \beta \in \mathbb{F}$.
Fra kodnings teori ved vi at Reed-Solomon koder allerede har algoritmer, så som Lagrange Interpolation, til at reparer kodeord, men problemet her ligger i at hele kodeordet bliver genskabt. Også selvom det kun er en indgang i kodeordet som skal rekonstrueres. Og for at Lagrange Interpolation skal bruges, skal man bruge k korrekte evalueringer og deres evaluerings punkter for polynomiet brugt i RS koden. Målet er så at lave en metode til at rekonstruerer en koordinat ved at sende minimalt mængder data.
I [1] viser de et LERS, hvor hvis man bruger alle evaluerings punkter mulige (altså hele $\mathbb{F}_{q^t}$) punkter, så kan alle servere nøjes med at sende et element fra $\mathbb{F}_q$ til en server for at rekonstruerer en indgang i et RS kodeord. Dette kalder vi for "Repair Bandwidth", som er med til at beskrive hvor meget data LERS skal sende i værste tilfælde.
Dette gjorde vi ved kalde evaluerings punkterne brugt i RS koden som A og lade alle servere være forbundet med et evaluerings punkt i A. Så ved finde en fast basis for $\mathbb{F}_{q^t}$ over $\mathbb{F}_q$ kaldt Z, kunne vi betegne nogle koefficienter $\mu_{\zeta,a}(a')$, hvor a' betegner serveren som er gået ned, og $\zeta \in Z, a \in A$. Disse $\mu_{\zeta,a}(a')$ overholder

$$\zeta_i f(a') = \sum_{a \in A \setminus \{a'\}} \mu_{\zeta_i,a}(a') f(a)$$

hvor $f(a)$ er indgange i et kodeord. Dette var godt fordi vi kunne reconstruer en data på en server som følgende

$$f(a') = \sum_{i=1}^{t} tr_{\mathbb{F}/\mathbb{B}}(\zeta_i \cdot f(a'))v_i.$$

hvor $v_i \in V$ som er den unikke dual basis til Z. Problemet indtil videre er dog at $\mu_{\zeta,a}(a')$ var meget besværlige at finde, men det kunne vises at problemet reduceret til at finde en mængde af polynomier $P(a')$ i den duale RS kode da

$$\mu_{a,\zeta}(a') = p(a) \cdot \frac{\prod_{\alpha \in A \setminus \{a'\}}(a' - \alpha)}{\prod_{\alpha \in A \setminus \{a\}}(a - \alpha)}.$$

for $p \in P(a')$. I [1] gav de

$$P(a') = \left\{ \frac{tr_{\mathbb{F}/\mathbb{B}}(\zeta_i(X - a'))}{X - a'} \mid \zeta_i \in Z \right\}.$$

som opfyldte:

$$\{p(a') | p \in P(a'))\} = \{\zeta_1, \ldots, \zeta_t\} = Z$$
$$\left\{p(a) | p \in P(a')\right\} = \left\{ \frac{\beta}{a - a'} | \beta \in \mathbb{B} \right\}$$

hvor $\left\{ \frac{\beta}{a-a'} | \beta \in \mathbb{B} \right\}$ havde dimension 1, som betød at ved at finde den rigtige skalar kunne man udregne $\mu_{\zeta_j,a}(a')$ fra $\mu_{\zeta_i,a}(a')$ hvor $j \neq i$, så længde $\mu_{\zeta_i,a}(a') \neq 0$. Så ved kun at sende et $tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta_i,a}(a')f(a))$ kan man udregne $tr_{\mathbb{F}/\mathbb{B}}(\mu_{\zeta_j,a}(a')f(a))$ ved

$$b_{i,j} \cdot tr_{\mathbb{F}/\mathbb{B}} \left(\mu_{\zeta_i,a_i}(a_k) f(a_i) \right) = tr_{\mathbb{F}/\mathbb{B}} \left(b_{i,j} \cdot \mu_{\zeta_i,a_i}(a_k) f(a_i) \right) = tr_{\mathbb{F}/\mathbb{B}} \left(\mu_{\zeta_j,a_i}(a_k) f(a_i) \right)$$

da $tr_{\mathbb{F}/\mathbb{B}}$ er lineær over $\mathbb{B}$.
Alt ovenstående gjorde at man kan reparer en indgang i et kodeord ved kun at downloade 1 element i $\mathbb{B}$, som betyder at med repair bandwidth $n - 1$ kan vi rekonstruerer en indgang i et kodeord.
Nu er spørgsmålet: kan vi gøre det bedre? Ved at vise at den mindste repair width en MDS kode kan have er

$$b \geq (n-1) \log_{|\mathbb{B}|} \left(\frac{n-1}{n-k} \right) = (n-1) \left(1 + \log_p \left(\frac{n-1}{n} \right) \right).$$

Ved omskrivning kunne vi komme frem til et udtryk der for n der bliver større, går mod $n - 1$, men siden at en decimal repair bandwidth ikke giver mening, kunne vi runde op til nærmeste heltal. Dette gjorde at vi ramte et optimalt repair bandwidth tidligt i forhold til $n \to \infty$.

Bibliography

[1] Venkatesan Guruswami and Mary Wootters. Repairing reed-solomon codes. *CoRR*, abs/1509.04764, 2015.

[2] Niels Lauritzen. *Concrete Abstract Algebra*. Cambridge University Press, 2003.

[3] Steven H Weintraub. *Galois Theory / by Steven H. Weintraub*. Universitext. Springer Science+Business Media, Inc, New York, NY, elektronisk udgave edition, 2006.

[4] Alfred J. Menezes, Ian Blake, XuHong Gao, Ronald C. Mullin, Scott A. Vanstone, and Tomik Yaghoobian. *Applications of Finite Fields*. 01 1993.

[5] Rudolf Lidl and Harald Niederreiter. *Finite Fields*. 1997.

[6] Ruud Pellikaan, Xin-Wen Wu, Stanislav Bulygin, and Relinde Jurrius. *Codes, Cryptology and Curves with Computer Algebra*. Cambridge University Press, 2017.

[7] Jørn Justesen and Tom Høholdt. *A Course in Error-Correcting Codes*. European Mathematical Society Publishing House, first edition, 2004.

[8] A. G. Dimakis, K. Ramchandran, Y. Wu, and C. Suh. A survey on network codes for distributed storage. *Proceedings of the IEEE*, 99(3):476–489, March 2011.