University of Tartu
Faculty of Science and Technology
Institute of Mathematics and Statistics

Martin Puškin

On Unequal Data Demand Private Information Retrieval Codes

Mathematics
Bachelor Thesis (9 ECTS)

Supervisors:
prof. Henk D.L. Hollmann
Dr. Ago-Erik Riet

Tartu 2022

ON UNEQUAL DATA DEMAND PRIVATE INFORMATION RETRIEVAL CODES

Bachelor thesis

Martin Puškin

Abstract

A t-PIR code allows the recovery of an encoded data symbol from each of t disjoint collections of code word symbols. We consider a generalization where some data symbols are in higher demand than other data symbols. We refer to such codes as $(t_1, \ldots, t_k)$-UDD PIR codes, where now the i-th data symbol can be recovered from each of t_i disjoint collections of code word symbols, for $i = 1, \ldots, k$.

We generalize the Griesmer bound for the length of the shortest possible t-PIR code to the $(t_1, \ldots, t_k)$-UDD PIR code case and provide two separate proofs for the bound. We show that the Griesmer bound is tight for $k \leq 3$ but not in general for $k = 4$. We also generalize other known upper and lower bounds for the shortest possible t-PIR codes to the $(t_1, \ldots, t_k)$-UDD PIR code case.

CERCS research specialisation: P110 Mathematical logic, set theory, combinatorics.

Key Words: PIR codes, UDD PIR Codes, Griesmer Bound.

EBAVÕRDSE ANDMENÕUDLUSEGA PRIVAATSE INFOOTSINGU KOODID

Bakalaureusetöö

Martin Puškin

Lühikokkuvõte

t-PIR kood võimaldab igat kodeeritud andmesümbolit t lõikumatust koodsõna sümbolite hulgast dekodeerida. Töös uuritakse ülesande üldistust, kus mõned andmesümbolid võivad olla kõrgema nõudluse all kui teised. Me kutsume selliseid koode $(t_1, \ldots, t_k)$-UDD PIR koodideks, kus nüüd on i-ndat andmesümbolit võimalik t_i lõikumatust koodsõna sümbolite hulgast dekodeerida, $i = 1, \ldots, k$.

Me üldistame Griesmeri tõket lühima võimaliku t-PIR koodi jaoks UDD PIR koodi juhule ja toome tõkke jaoks kaks erinevat tõestust. Me näitame, et Griesmeri tõke on $k \leq 3$ korral võrdus, kuid mitte üldiselt $k = 4$ korral. Samuti üldistame teisi tuntud alumisi ja ülemisi tõkkeid lühima t-PIR koodi jaoks $(t_1, \ldots, t_k)$-UDD PIR koodi juhule.

CERCS teaduseriala: P110 Matemaatiline loogika, hulgateooria, kombinatoorika

Märksõnad: PIR koodid, UDD PIR koodid, Griesmeri tõke.

Contents

1 Introduction **3**

2 Preliminaries **5**
2.1 Unequal Error Protection (UEP) codes 5
2.2 Examples of UDD PIR Codes 6
2.3 Some General Results About UDD PIR Codes 7

3 UDD PIR Codes With $k \leq 3$ **9**

4 The Griesmer Bound **11**
4.1 Proof Using Hyperplanes 11
4.2 Proof Using UEP Codes 14

5 Some other bounds **16**

6 UDD PIR codes for $k = 4$ **18**
6.1 Upper bound 18
6.2 Lower bound 20
6.2.1 Theoretical Approach 20
6.2.2 Computer-Based Approach 24

Conclusions **26**

References **27**

1 Introduction

This thesis concerns new type of codes which we coin unequal data demand (UDD) codes and which are a generalization of private information retrieval (PIR) codes. The latter were first discussed by Fazeli, Vardy and Yaakobi in [2] (this is the shorter refereed version of [3]) and are used in schemes where users can request any item from a database distributed among $n \in \mathbb{N}$ servers and must be able to receive it in such a way that no server gets any information about which piece of data was retrieved by the user.

An important characteristic of a PIR code is how many people can simultaneously ask for and obtain the same piece of data from the system of servers without having to wait for the previous query to be carried out. For positive integers k and t we define a (k,t)-PIR code to be a setup which allows up to t people to simultaneously request the same piece of data from a database of k items. Another way to advantage of a (k,t)-PIR code is that all data is still redeemable if any $t-1$ servers are simultaneously down.

In this work, $\mathbb{N}$ denotes the non-negative integers, i.e. $\mathbb{N} := \{0,1,2,\ldots\}$. We denote by $\mathbb{F}_2$ the field with two elements, by $\mathrm{Mat}_{k,n}(\mathbb{F}_2)$ the set of k-by-n matrices over $\mathbb{F}_2$ and $[n] := \{1,\ldots,n\}$ for $n \in \mathbb{N}$. We now formally define a PIR code.

For a k-dimensional subspace C of $\mathbb{F}_2^n$, a matrix for which $mG \in C$ holds for all $m \in \mathbb{F}_2^k$ is called the *generator matrix* $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$ of C.

Definition 1.1. Let C be a linear subspace of $\mathbb{F}_2^k$. We call a generator matrix $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$ of C a (binary) (k,t)-*PIR code* if it has the following property: for every $1 \leq i \leq k$, there exist t disjoint sets of column indices $I_1, ..., I_t \subseteq [n]$ such that for every $1 \leq j \leq t$, the columns of G with indices from I_j sum up to the i-th unit vector.

If we have such a PIR code G, then it is possible to distribute k pieces of data among n servers so that t users can simultaneously request any piece of data and the privacy condition is fulfilled for all of them. If the reader is interested in how exactly this is done, they are referred to a good introduction to the topic in [3].

The generalization that we make in this thesis is that of considering a vector $(t_1,\ldots,t_k) \in \mathbb{N}^k$ instead of the integer t. In this case, the value t_i, $i \in [k]$, of how many people can simultaneously ask for the i-th item can be different between all the k symbols. This corresponds to prescribing different values of importance or popularity to different pieces of data. We call such a generalization a $(k,(t_1,\ldots,t_k))$-UDD PIR code.

We can do this by just concatenating regular PIR codes, one for each different value of t_i, however, we can sometimes do better.

Definition 1.2. Let C be a linear subspace of $\mathbb{F}_2^k$. We call a generator matrix $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$ of C a (binary) $(k,(t_1,\ldots,t_k))$-*UDD PIR code* if it has the following property: for every $1 \leq i \leq k$, there exist t_i *disjoint* sets of column indices $I_1, ..., I_{t_i} \subseteq [n]\}$ such that for every $1 \leq j \leq t_i$, the columns of G with indices from I_j sum up to the i-th unit vector.

For simplicity, we usually call a $(k,(t_1,\ldots,t_k))$-UDD PIR code a $(t_1,\ldots,t_k)$-UDD code.

As a $(k,(t_1,\ldots,t_k))$-UDD code clearly exists for all choices of k and $(t_1,\ldots,t_k)$ (just consider the matrix with t_i columns of the i-th unit vector), we are interested in the shortest possible code which satisfies this condition for fixed k and $(t_1,\ldots,t_k)$. We call the length n of such a code $P(k,(t_1,\ldots,t_k))$. We may clearly always assume w.l.o.g. that $t_1 \geq \ldots \geq t_k$.

We can now formally phrase the main problem of this thesis.

Problem 1.3. Let $k \in \mathbb{N}$ and $T = (t_1,\ldots,t_k)$ with $t_1,\ldots,t_k \in \mathbb{N}$ and $t_1 \geq \ldots \geq t_k$. How long is the shortest T-UDD code i.e. what is the value of $P(k,T)$?

In the first chapter we introduce other relevant concepts for this thesis, give some examples of UDD codes and prove a few general statements about UDD codes.

In the second chapter we completely solve problem 1.3 for $k \leq 3$ by showing that in this case a general lower bound for $P(k, (t_1, \ldots, t_k))$ called the *Griesmer bound* can always be achieved. However, we do not prove the Griesmer bound in the second chapter.

In the third chapter we provide two different proofs for the Griesmer bound. The first proof uses hyperplanes in the vector space $\mathbb{F}_2^k$ and the second proof draws on the theory of unequal error protection (UEP) codes.

In the fourth chapter we provide several other useful upper and lower bounds for UDD PIR codes which are all generalizations of known similar bounds for regular PIR codes.

Finally, in the fifth chapter we use the above as well as a computer-based approach to solve the problem for $k = 4$.

2 Preliminaries

2.1 Unequal Error Protection (UEP) codes

We call a k-dimensional subspace C of $\mathbb{F}_2^n$ a binary $[n, k]$ linear code. The parameters n and k are called the *length* and *rank* of the code C, respectively. An *encoder* of the code C is a linear bijection $\epsilon\colon \mathbb{F}_2^k \to C$ which maps the *message word* $m \in \mathbb{F}_2^k$ to a corresponding *code word* $mG \in C$, where $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$. The matrix G is called a *generator matrix* of the code C. Clearly, the rows of G form a basis for C. But also conversely, every k-by-n matrix whose rows are linearly independent defines an $[n, k]$ linear code.

One aspect that one can study in a linear code is its resistance to corruption. The most commonly investigated form of corruption is that of additive noise: when a code word $c \in C$ is sent through a channel, sometimes some bits in the word are changed. This can be viewed as the addition of the error vector $e \in \mathbb{F}_2^n$ i.e. the word received on the other side of the channel is $c + e$ for an unknown vector e. The goal of error protection codes is to either detect an error or even make it possible recover c from $c + e$ provided the corruption is "small enough". This notion of a small error brings us to the concept of a weight of a vector.

Definition 2.1 (Hamming weight). The *(Hamming) weight* $wt(v)$ of the vector $v \in \mathbb{F}_2^n$ is the number of nonzero components of v.

Usually error protection codes are set up so as to be able to detect or correct and error given $wt(e) \leq t$ for some positive integer t. A closely related concept to Hamming weight is that of Hamming distance.

Definition 2.2 (Hamming distance). The *(Hamming) distance* $d(v, w)$ between two vectors $v, w \in \mathbb{F}_2^n$ is defined as $wt(v - w)$. The distance $d(v, S)$ between a vector $v \in \mathbb{F}_2^n$ and a subspace $S \leq \mathbb{F}_2^n$ is defined as $\min_{w \in S} d(v, w)$.

One can study a generalization of this problem where different bits of the message word m may not be of the same importance. In this case, some bits can be more protected than others and independently recovered when others cannot. Such codes are called *Unequal Error Protection codes* or UEP codes. In order to formally define such a code's error protection capability, we must first introduce the concept of the weight of a vector.

Dunning and Robins [1] introduced the concept of a separation vector to characterize the error protection capability of UEP codes.

Definition 2.3 (Separation vector). For a linear $[n, k]$ code C with a generator matrix G we define the *separation vector* $S(G) = (S(G)_1, \ldots, S(G)_k)$ by

$$S(G)_i := \min\{wt(mG) \mid m \in \mathbb{F}_2^k, m_i \neq 0\}. \tag{1}$$

The following theorem shows that the higher the value $S(G)_i$ the stronger the protection for the i-th data symbol.

Theorem 2.4. *Let C be a linear $[n, k]$ code with a generator matrix G and the separation vector $S(G)$ as in (1). If the weight of the error vector e satisfies $wt(e) < \frac{S(G)_i}{2}$, then the i-th data symbol can be recovered.*

Proof. Let us look at two message words m^1 and m^2 where the i-th coordinates of m^1 and m^2 differ, i.e. $m_i^1 \neq m_i^2$. This means that $(m^1 - m^2)_i \neq 0$ and by (1), $wt(m^1G - m^2G) = wt((m^1 - m^2)G) \geq S(G)_i$. This means that the code words m^1G and m^2G differ in at least $S(G)_i$ coordinates.

For $i \in [k]$ and $q \in \mathbb{F}_2$, we define $S_q^i = \{mG \mid m \in \mathbb{F}_2^k, m_i = q\}$.

Let m be a message word and $wt(e) < \frac{S(G)_i}{2}$. Then the distance between m and the subspace $S_{m_i}^i$ is

$wt(e) < \frac{S(G)_i}{2}$ and the distance between m and any other subspace S_q^i is at least $S(G)_i - wt(e) > \frac{S(G)_i}{2}$. Thus, we can decode the i-th symbol of m by finding the closest subspace S_q^i to the received vector $mG+e$. □

2.2 Examples of UDD PIR Codes

We recall the definition of a binary UDD PIR code.

Definition 1.2. Let C be a linear subspace of $\mathbb{F}_2^k$. We call a generator matrix $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$ of C a (binary) $(k, (t_1, \ldots, t_k))$-*UDD PIR code* if it has the following property: for every $1 \leq i \leq k$, there exist t_i *disjoint* sets of column indices $I_1, ..., I_{t_i} \subseteq [n]\}$ such that for every $1 \leq j \leq t_i$, the columns of G with indices from I_j sum up to the i-th unit vector.

Example 2.5. As mentioned in the introduction, there exists a $(k, (t_1, \ldots, t_k))$-UDD PIR code for all $k \in \mathbb{N}$ and $(t_1, \ldots, t_k) \in \mathbb{N}^k$ because we have the following trivial construction:

$$G = \begin{pmatrix} \overbrace{1 \ \ldots \ 1}^{t_1} & \overbrace{0 \ \ldots \ 0}^{t_2} & \ldots & \overbrace{0 \ \ldots \ 0}^{t_k} \\ 0 \ \ldots \ 0 & 1 \ \ldots \ 1 & \ldots & 0 \ \ldots \ 0 \\ \vdots \ \ddots \ \vdots & \vdots \ \ddots \ \vdots & \ddots & \vdots \ \ddots \ \vdots \\ 0 \ \ldots \ 0 & 0 \ \ldots \ 0 & \ldots & 0 \ \ldots \ 0 \\ 0 \ \ldots \ 0 & 0 \ \ldots \ 0 & \ldots & 1 \ \ldots \ 1 \end{pmatrix}.$$

This example also gives the upper bound $P(k, (t_1, \ldots, t_k)) \leq \sum_{i=1}^{k} t_i$.

Let us fix a $k \in \mathbb{N}$. We will henceforth denote by e_i the vector which if read from bottom to top gives the binary representation of the integer i (adding zeroes to the beginning if necessary). We denote the zero vector by $\mathbf{0}$. For example, for $k = 3$ we have the following vectors:

$$\left(\begin{array}{cccccccc} \mathbf{0} & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 \\ \hline 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right).$$

Note that the unit vectors are $e_{2^{i-1}}$ where $i \in [k]$.

If we have a matrix G, we denote by a_i the number of columns e_i in the matrix G.

The next example shows that the upper bound in Example 2.5 is far from being tight.

Example 2.6 (Simplex code). Let $k = 3$ and $t_1 = t_2 = t_3 = 4$. We consider the following matrix whose columns are all of the non-zero vectors of $\mathbb{F}_2^4$

$$G = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Let $i \in [3]$. All the columns of G except $e_{2^{i-1}}$ can be formed into pairs $\{v, v + e_{2^{i-1}}\}$. As these pairs are disjoint recovery sets for $e_{2^{i-1}}$ and $\{e_{2^{i-1}}\}$ is also a recovery set, we have found 4 disjoint recovery sets for $e_{2^{i-1}}$.

Obviously, nothing is special about $k = 3$ and we have $P(k, (2^{k-1}, \ldots, 2^{k-1})) \leq 2^k - 1$ for all $k \in \mathbb{N}$. Such

codes are very important and have the special name *simplex code*.

2.3 Some General Results About UDD PIR Codes

We will always assume that the k rows of the matrix G are linearly independent. Now, this is obviously true if G is the generator matrix for a linear subspace C as in Definition 1.2 but the following theorem shows that we can assume this in an even more general case.

Theorem 2.7. *Let $G = (g_{ij}) \in Mat_{k,n}(\mathbb{F}_2)$ be a matrix and $i_1, \ldots, i_l \in [k]$. Assume that the rows $g^{i_1}, \ldots, g^{i_l}$ of G sum to the 0-vector $\mathbf{0}$. Then G has no recovery sets for $e_{i_1}, \ldots, e_{i_l}$.*

Proof. We show that there is no recovery set for e_{i_1}. Assume for contradiction that the columns $g_{j_1}, \ldots, g_{j_m}$ are a recovery set for e_{i_1} i.e. $\sum_{r=1}^{m} g_{j_r} = e_{i_1}$. This is equivalent to $\sum_{r=1}^{m} g_{i,j_r} = \begin{cases} 1, & \text{if } i = i_1, \\ 0 & \text{otherwise.} \end{cases}$ But now summing both sides over $i_1, \ldots, i_l$ gives a contradiction, as the left side gives 0 by assumption and the right side gives 1. □

If the rows of G are not linearly independent, there must be a set of rows which sum to 0 (not over a general finite field but certainly over $\mathbb{F}_2$). As per the previous theorem, these rows are not useful and can be left out.

We call a system of recovery sets for a unit vector maximal if there exists no system of recovery sets for the same vector which is larger, i.e. has more recovery sets. We have the following useful result which greatly simplifies counting the recovery sets for a given unit vector.

Theorem 2.8. *Let G be a binary $k \times n$ matrix. For all $x \in \mathbb{F}_2^k \setminus \{\mathbf{0}\}$, let a_x be the number of the columns x in G. Then for any unit vector e_i, there exists a maximal system of recovery sets for e_i which for all $x \in \mathbb{F}_2^k \setminus \{\mathbf{0}, e_i\}$ has $\min\{a_x, a_{x+e_i}\}$ recovery sets of the form $\{x, x + e_i\}$.*

Proof. Let R_i be any maximal system of recovery sets for e_i. Let $x \in \mathbb{F}_2^k \setminus \{\mathbf{0}, e_i\}$ and assume that there are fewer than $\min\{a_x, a_{x+e_i}\}$ recovery sets of the form $\{x, x + e_i\}$. Then we have two cases:

- There exist recovery sets A and B of size 3 or larger in R_i with $x \in A$ and $x + e_i \in B$. We can replace A and B with two new recovery sets $\{x, x + e_i\}$ and $(A \cup B) \setminus \{x, x + e_i\}$.
- x is not in any recovery set in R_i and $x + e_i$ is in a recovery set B of size 3 or larger. Then we can use the recovery set $\{x, x + e_i\}$ instead of B.

Both of these moves retain the number of recovery sets for e_i but add a recovery set of the type $\{x, x + e_i\}$. These moves can be made until there are exactly $\min\{a_x, a_{x+e_i}\}$ recovery sets $\{x, x + e_i\}$, as desired. □

This theorem has a useful consequence.

Corollary 2.9. *Let G be a $(k, (t_1, \ldots, t_k))$-UDD PIR code of length n which has among its columns at least one of each non-zero vector of $\mathbb{F}_2^k$. Then*

$$n \geq P(k, (t_1 - 2^{k-1}, \ldots, t_k - 2^{k-1})) + 2^k - 1.$$

Proof. As G has one of each non-zero column, we can form the simplex code (cf. Example 2.6) from its columns. As the simplex code has only recovery sets of size 1 and 2, we can always assume that the columns are paired up in this way by virtue of Theorem 2.8.

Finally, it remains to note that a simplex code of dimension k has 2^{k-1} recovery sets for each unit vector and is of length $2^k - 1$. □

We define a recovery set to be minimal if it has no subset whose sum is the zero vector. If the latter is the case, then we can simply leave out the zero-sum set and still have a recovery set. By this consideration, we may clearly assume that all recovery sets are minimal. Similarly, we define a set to be a minimal zero-sum set or a *circuit* if none of its subsets have sum $\mathbf{0}$.

To end this chapter, we provide an algorithm to generate all minimal recovery sets for the unit vector e_1.

Algorithm 2.10. As all sets with more than k vectors are linearly dependent and thus must have a zero-sum subset (this is true in $\mathbb{F}_2^k$ but not in a general field), we can reduce ourselves to finding recovery sets of size up to k.

It is not difficult, even if cumbersome, to determine all the linearly independent sets of $\mathbb{F}_2^{k-1}$ (one can, for example, begin with one-dimensional subspaces of $\mathbb{F}_2^{k-1}$ and one by one add vectors which preserve linear independence). We can use this to find all the circuits of $\mathbb{F}_2^{k-1}$, as it is easy to see that all circuits are of the form $\{v_1, v_2, \ldots, v_m, v_1 + \ldots + v_m\}$ where $\{v_1, \ldots, v_m\}$ is a linearly independent set.

Now, let us consider the columns in a minimal recovery set for e_1 in $\mathbb{F}_2^k$. If we ignore the first row, we must get a circuit in $\mathbb{F}_2^{k-1}$. Indeed, we must get a zero-sum set and it were not minimal, then the original recovery set would also not be minimal.

It remains to note that any minimal recovery set for e_1 must have an odd number of columns with 1 as the first coordinate. Now every minimal recovery set in $\mathbb{F}_2^k$ can be built from a circuit in $\mathbb{F}_2^{k-1}$ by inserting them into $\mathbb{F}_2^k$ with the first coordinate 1 for an odd number of vectors.

3 UDD PIR Codes With $k \leq 3$

In this section we will look at Problem 1.3 with few data symbols ($k \leq 3$). We prove that given $t_1 \geq t_2 \geq t_3$, we have $P(3,(t_1,t_2,t_3)) = t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil$. Note that this immediately implies $P(2,(t_1,t_2)) = t_1 + \left\lceil \frac{t_2}{2} \right\rceil$ and $P(1,(t_1)) = t_1$ by setting $t_3 = 0$ and $t_2 = t_3 = 0$ respectively.

We first show that $P(3,\{t_1,t_2,t_3\}) \leq t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil$ by constructing a code of length $t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil$.

Lemma 3.1. *Let $t_1 \geq t_2 \geq t_3$. Then $P(3,\{t_1,t_2,t_3\}) \leq G(t_1,t_2,t_3) := t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil$.*

Proof. To show that $P(3,\{t_1,t_2,t_3\}) \leq t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil$, we must give an algorithm for constructing a code for all triples $t_1 \geq t_2 \geq t_3$. We will do this in 3 parts.

1. a) Assume we have a (t_1,t_2,t_3)-UDD code with length $G(t_1,t_2,t_3)$. Then, as $G(t_1+1,t_2,t_3) = G(t_1,t_2,t_3)+1$, we can get a (t_1+1,t_2,t_3)-UDD code of length $G(t_1+1,t_2,t_3)$ just by increasing a_1 by 1.
   b) Note that $G(t_1+2,t_2+2,t_3) = G(t_1,t_2,t_3)+3$. Assume we already have a (t_1,t_2,t_3)-UDD code with length $G(t_1,t_2,t_3)$. We can get a (t_1+2,t_2+2,t_3)-UDD code of length $G(t_1+2,t_2+2,t_3)$ by adding the vectors e_1, e_2 and e_3.
   c) Note that $G(t_1+4,t_2+4,t_3+4) = G(t_1,t_2,t_3)+7$. Assume we already have a code for $T=(t_1,t_2,t_3)$ with length $G(t_1,t_2,t_3)$. We can get a code of length $G(t_1+4,t_2+4,t_3+4)$ for $T=(t_1+4,t_2+4,t_3+4)$ by increasing the value of each of $a_1,a_2,a_3,a_4,a_5,a_6,a_7$ in the previous code by 1 (i.e. adding the simplex code).
   d) Note that $G(t_1,t_2,4m) = G(t_1,t_2,4m-1) = G(t_1,t_2,4m-2) = G(t_1,t_2,4m-3)$. Thus, if we assume that we have a (t_1,t_2,t_3)-UDD code of length $G(t_1,t_2,t_3)$, then we can instead construct a code for (t_1,t_2,m) where $t_3 := \min\{t_2,m\}$ where m is the smallest multiple of 4 larger than t_3.
2. We now look at some special codes.
   a) $t_1 = t_2 = t_3 = 0$. A corresponding code of length $G(0,0,0) = 0$ is the empty code i.e. $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (0,0,0,0,0,0,0)$.
   b) $t_1 = t_2 = t_3 = 1$. A corresponding code of length $G(1,1,1) = 1 + \left\lceil \frac{1}{2} \right\rceil + \left\lceil \frac{1}{4} \right\rceil = 3$ is with $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (1,1,0,1,0,0,0)$.
   c) $t_1 = t_2 = t_3 = 2$. A corresponding code of length $G(2,2,2) = 2 + \left\lceil \frac{2}{2} \right\rceil + \left\lceil \frac{2}{4} \right\rceil = 4$ is $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (1,1,0,1,0,0,1)$.
   d) $t_1 = t_2 = t_3 = 3$. A corresponding code of length $G(3,3,3) = 3 + \left\lceil \frac{3}{2} \right\rceil + \left\lceil \frac{3}{4} \right\rceil = 6$ is $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (1,1,1,1,1,1,0)$.
   e) $t_1 = t_2 = t_3 = 4$. A corresponding code of length $G(4,4,4) = 4 + \left\lceil \frac{4}{2} \right\rceil + \left\lceil \frac{4}{4} \right\rceil = 7$ is $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (1,1,1,1,1,1,1)$.
   f) $t_1 = t_2 = 1$, $t_3 = 0$. A corresponding code of length $G(1,1,0) = 1 + \left\lceil \frac{1}{2} \right\rceil + \left\lceil \frac{0}{4} \right\rceil = 2$ is $(a_1,a_2,a_3,a_4,a_5,a_6,a_7) = (1,1,0,0,0,0,0)$.
3. By repeatedly using 1. on the codes presented in 2., we can construct a code of desired length for all vectors (t_1,t_2,t_3). This is explicitly demonstrated by the algorithm below.

□

Algorithm 3.2. Let $t_1 \geq t_2 \geq t_3$. We construct a matrix G of a (t_1, t_2, t_3)-UDD code of length $G(t_1, t_2, t_3)$ with the following steps.

- Add $\left\lfloor \frac{t_2 - t_3}{2} \right\rfloor$ vectors e_1, e_2 and e_3 to G. Set $t_1' := t_1 - 2\left\lfloor \frac{t_2 - t_3}{2} \right\rfloor$ and $t_2' = t_2 - 2\left\lfloor \frac{t_2 - t_3}{2} \right\rfloor$.
- Add $t_1' - t_2'$ columns of e_1 to the matrix G and set $t_1'' := t_2'$.
- Let m be the smallest multiple of 4 which is greater than or equal to t_3. We will instead construct a code for $T = (t_1'', t_2', t_3')$ where $t_3' := \min\{m, t_2'\}$.
- Add $\left\lfloor \frac{t_3'}{4} \right\rfloor$ columns of each type (i.e. $e_1, e_2, e_3, e_4, e_5, e_6, e_7$) and set $t_1''' := t_1'' - 4\left\lfloor \frac{t_3'}{4} \right\rfloor$, $t_2'' := t_2' - 4\left\lfloor \frac{t_3'}{4} \right\rfloor$, $t_3'' := t_3' - 4\left\lfloor \frac{t_3'}{4} \right\rfloor$.
- We have constructed the code of the desired length for $T' := (t_1''', t_2'', t_3'')$ in point 2 of the previous lemma. Add the corresponding columns to the matrix G.

We will also illustrate this construction with an example.

Example 3.3. Let $T = (t_1, t_2, t_3) = (13, 9, 6)$. We want a code of length $G(13, 9, 6) = 20$.

- As $\left\lfloor \frac{t_2 - t_3}{2} \right\rfloor = 1$, we add 1 vector of each of the types e_1, e_2 and e_3 to G. Set $t_1' = 13 - 2 = 11$ and $t_2' = 9 - 2 = 7$. We currently have $(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (1, 1, 1, 0, 0, 0, 0)$.
- $t_1' - t_2' = 11 - 7 = 4$ so we add 4 vectors e_1 and set $t_1'' = 7$. Now we have $(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (5, 1, 1, 0, 0, 0, 0)$.
- The smallest multiple of 4 which is larger than or equal to t_3 is $m = 8$. We set $t_3' = \min\{m, t_2\} = \min\{7, 8\} = 7$.
- We now add $\left\lfloor \frac{t_3'}{4} \right\rfloor = 1$ of each vector and set $t_1''' = t_1'' - 4 = 3$, $t_2'' = t_2' - 4 = 3$ and $t_3'' = t_3' - 4 = 3$. We now have $(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (6, 2, 2, 1, 1, 1, 1)$.
- In point 1.d) of Lemma 3.1 we constructed a code for $(t_1''', t_2'', t_3'') = (3, 3, 3)$. Thus, we add 1 of each vector except e_7. Our final matrix will have the following number of each vector: $(a_1, a_2, a_3, a_4, a_5, a_6, a_7) = (7, 3, 3, 2, 2, 2, 1)$.

The reverse inequality $P(3, (t_1, t_2, t_3)) \geq G(t_1, t_2, t_3)$ follows from a more general inequality, Theorem 4.1, which is a generalization of Theorem 4 in [5]. This will be proven in the next chapter.

4 The Griesmer Bound

A very important lower bound for UDD codes is the Griesmer bound:

Theorem 4.1 (Griesmer Bound). *Let $t_1, \ldots, t_k \in \mathbb{N} \cup \{0\}$ with $t_1 \geq \ldots \geq t_k$. Then*

$$P(k, (t_1, \ldots, t_k)) \geq G(t_1, \ldots, t_k) := \sum_{i=1}^{k} \left\lceil \frac{t_i}{2^{i-1}} \right\rceil.$$

4.1 Proof Using Hyperplanes

Before proving Theorem 4.1, we must do some preparation. We first provide some important definitions.

Definition 4.2 (Hyperplane). Let V be a k-dimensional vector space. Then a $k-1$-dimensional subspace of V is called a *hyperplane* in V.

Definition 4.3 (Scalar product). Let $v = (v_1, \ldots, v_k)$ and $w = (w_1, \ldots, w_k)$ be two vectors from $\mathbb{F}_2^k$. We define the *scalar product* $v \cdot w \colon \mathbb{F}_2^k \times \mathbb{F}_2^k \to \mathbb{F}_2$ of v and w as follows:

$$v \cdot w := \sum_{i=1}^{k} v_i w_i.$$

The scalar product is commutative and distributive over addition.

Definition 4.4 (Orthogonal complement). Let S be a subspace of $\mathbb{F}_2^k$. Then the *orthogonal complement* $S^\perp$ of S is defined as

$$S^\perp := \{v \in \mathbb{F}_2^k \mid v \cdot w = \mathbf{0}\ \forall w \in S\}.$$

It easy to check that the orthogonal complement of any subspace of $\mathbb{F}_2^k$ is itself a subspace of $\mathbb{F}_2^k$.

A well-known fact from linear algebra states that for a subspace S of a linear space V, we have

$$\dim S + \dim S^\perp = \dim V. \tag{2}$$

The identity (2) defines a one-to-one correspondence between hyperplanes and one-dimensional subspaces of $\mathbb{F}_2^k$.

We note that as in general the implication $v \cdot v = \mathbf{0} \Rightarrow v = \mathbf{0}$ does not hold in fields of finite characteristic, S and $S^\perp$ can have a non-trivial intersection.

We define $\psi(j) := \max\{k \in \mathbb{N} \mid 2^k \mid j\}$, that is, $\psi(j)$ returns the highest power of 2 that divides j. The following lemma is a generalization of Lemma 4 from [5].

Lemma 4.5. *Let $k \in \mathbb{N}$ and $T = (t_1, \ldots, t_k)$, where $t_1 \geq \ldots \geq t_k \geq 0$. Let $j \in [2^k - 1]$, then $e_j^\perp := H \leq \mathbb{F}_2^k$ is a hyperplane. Then*

$$\sum_{\substack{r \in [2^k - 1] \\ e_r \notin H}} a_r \geq t_{\psi(j)+1}. \tag{3}$$

Proof. We define H and $\psi(j)$ as above. Since the $2^{\psi(j)}$-th coordinate of e_j is 1, we have $e_{2^{\psi(j)}} \cdot e_j = 1$ i.e. the unit vector $e_{2^{\psi(j)}} \notin H$. As H is a subspace of $\mathbb{F}_2^k$, it is closed under addition. This means that any recovery set of $e_{2^{\psi(j)}}$ must include a vector that is not an element of H. Since there must be $t_{\psi(j)+1}$ disjoint recovery sets for $e_{2^{\psi(j)}}$, the inequality (3) follows. □

Let us now consider the $(2^k - 1) \times (2^k - 1)$ matrix of scalar products of vectors $e_i, i \in [2^k - 1]$. For $k = 2$,

we get the following matrix:

$$\begin{array}{c} \\ e_1 \\ e_2 \\ e_3 \end{array} \begin{array}{c} \begin{array}{ccc} e_1 & e_2 & e_3 \end{array} \\ \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{pmatrix} \end{array} \tag{4}$$

The next lemma proves that it is easy to inductively generate these matrices for all $k \geq 2$ and highlights many relevant symmetries in this sequence of matrices.

Lemma 4.6. *Let $k \in \mathbb{N}$ and H_k be the $(2^k-1)\times(2^k-1)$ matrix of scalar products of the vectors $e_1, \ldots, e_{2^k-1}$. We denote $\mathbf{1}_{m\times n}$ the m-by-n dimensional matrix consisting of ones and $\mathbf{0}_{m\times n}$ the m-by-n dimensional matrix consisting of zeroes. The following statements hold.*

1. *The $(2^{k+1}-1)\times(2^{k+1}-1)$ matrix of scalar products of the vectors $e_1, \ldots, e_{2^{k+1}-1}$ has the following form:*

$$H_{k+1} = \left(\begin{array}{c|c|c} H_k & \mathbf{0}_{(2^k-1)\times 1} & H_k \\ \hline \mathbf{0}_{1\times(2^k-1)} & 1 & \mathbf{1}_{1\times(2^k-1)} \\ \hline H_k & \mathbf{1}_{(2^k-1)\times 1} & \mathbf{1}_{(2^k-1)\times(2^k-1)} - H_k \end{array}\right) \tag{5}$$

2. *Every row of H_k has exactly $2^{k-1}-1$ zeroes and 2^{k-1} ones.*
3. *Adding (in $\mathbb{F}_2^{2^k-1}$) any two odd-numbered rows of H_k gives an even-numbered row of H_k.*
4. *Let $k \geq 2$ and $i \in [2^k-1]$ be odd. The submatrix which consists of the columns where there is 0 in the i-th row and of all the even-numbered rows of H_k is H_{k-1}.*

Proof.

1. The product of two vectors of length $k+1$ will only be different from the results in H_k when the $k+1$-st coordinate of both of these vectors is 1. The identity (5) is now easy to check.
2. This is a direct consequence of (5) and $H_1 = (1)$. Alternatively, the elements whose scalar product with e_i is 0 form a hyperplane and a hyperplane has exactly 2^k elements, including the zero vector.
3. Let $i, j \in [2^k-1]$ be odd numbers. As the scalar product is distributive over addition, the l-th coordinate in the sum of the i-th and j-th row is

$$e_i \cdot e_l + e_j \cdot e_l = (e_i + e_j) \cdot e_l.$$

   As i and j are both odd, the first coordinate of both of them is 1. This means that the first coordinate of their sum is 0 and $i+j$ is even (and $e_i + e_j \neq \mathbf{0}$ as $i \neq j$).

4. We prove a stronger claim:
   a) *The submatrix which consists of the columns where there is 0 in the i-th row and of all the even-numbered rows of H_k is H_{k-1}.*
   b) *The submatrix which consists of the columns where there is 1 in the i-th row and of all the even-numbered rows of H_k is $(\mathbf{0}_{(2^{k-1}-1)\times 1}|H_{k-1})$.*

   It is routine to check that the claim is true for $k=2$ (the matrix can be seen in (4)). Let us assume that the claim holds for H_k and show that it then also holds for H_{k+1}.

   In all the following cases, the investigated submatrix includes the middle row of H_{k+1} and it will always have the right form. Thus, to use (5), we must prove that the top left, bottom left and top right quadrants of the submatrix are H_{k-1} and that the bottom right quadrant is $\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1}-1)} - H_{k-1}$. We must additionally either have the middle column of H_{k+1} in our submatrix or have an extra column to replace it.

a)
- Let $i \in [2^k - 1]$ i.e. we fix a row in the top half of the matrix H_{k+1}. Then the submatrix restricted to the top left quadrant of H_{k+1} will be H_{k-1} by a) in the induction hypothesis. Symmetrically, this will also be true for the bottom left quadrant and top right quadrant. As the bottom right quadrant is just the top right quadrant with the zeroes and ones swapped, we get $\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1}-1)} - H_{k-1}$ there. Finally, as the i-th row has a zero in the middle column of H_{k+1}, we also have this column in the submatrix and will get exactly H_k by (5).
- Let $i \in [2^{k+1} - 1] \setminus [2^k - 1]$ i.e. we fix a row in the bottom half of the matrix H_{k+1}. As the bottom left quadrant is H_k, we get from the induction hypothesis a) that the bottom left quadrant of our submatrix is H_{k-1}. This extends to the whole left half of the submatrix as we choose the same columns in the top left quadrant of H_{k+1}.

  The columns which have 0 in the i-th row on the right side of H_{k-1} are exactly the columns which have 1 in the $i - 2^k$-th row in the top right quadrant. By the induction hypothesis b), we get the matrix $(\mathbf{0}_{(2^{k-1}-1)\times 1}|H_{k-1})$ in the top right quadrant and thus

$$\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1}-1)} - (\mathbf{0}_{(2^{k-1}-1)\times 1}|H_{k-1}) = (\mathbf{1}_{(2^{k-1}-1)\times 1}|\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1}-1)} - H_{k-1})$$

  in the bottom right quadrant.

  Again by (5), the four quadrants together give us H_k, as desired.

b)
- Let $i \in [2^k - 1]$ i.e. we fix a row in the top half of the matrix H_{k+1}. The induction hypothesis b) on H_k in the top left quadrant gives us $(\mathbf{0}_{(2^{k-1}-1)\times 1}|H_{k-1})$ in the top left quadrant of the submatrix. Symmetry gives us the same matrix in the bottom left and top right quadrants. In the bottom right quadrant we get

$$\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1})} - (\mathbf{0}_{(2^{k-1}-1)\times 1}|H_{k-1}) = (\mathbf{1}_{(2^{k-1}-1)\times 1}|H_{k-1}).$$

  By (5), the submatrix will be H_k, as desired.
- Let $i \in [2^{k+1} - 1] \setminus [2^k - 1]$ i.e. we fix a row in the bottom half of the matrix H_{k+1}. In the left half everything will go as before. We also choose the middle column of H_{k+1} as the i-th row has 1 there. The bottom right quadrant has ones exactly where the top right quadrant has zeroes. Thus, by the induction hypothesis a) on the $i - 2^k$-th row in the top right quadrant, we get H_{k-1} in the top right quadrant and consequently $\mathbf{1}_{(2^{k-1}-1)\times(2^{k-1})} - H_{k-1}$ in the bottom right quadrant.

  By (5), the submatrix will be H_k, as desired.

□

We are now ready to prove the Griesmer bound.

Proof of theorem 4.1. The proof will be by induction. The lower bound $P(1,(t_1)) \geq t_1$ obviously holds for all $t_1 \geq 0$. We now assume that for all tuples $(t_1, \ldots, t_k)$ with $t_1 \geq \ldots \geq t_k \geq 0$ we have $P(k,(t_1,\ldots,t_k)) \geq G(t_1,\ldots,t_k)$. We show that then the bound $P(k+1,(t_1,\ldots,t_{k+1})) \geq G(t_1,\ldots,t_{k+1})$ holds for any tuple $(t_1,\ldots,t_{k+1})$ with $t_1 \geq \ldots \geq t_{k+1} \geq 0$.

We assume that we have an optimal $(t_1,\ldots,t_{k+1})$-UDD code of length $n = P(k+1,(t_1,\ldots,t_{k+1}))$. Lemma 4.5 gives us $2^{k+1} - 1$ inequalities for $a_1,\ldots,a_{2^{k+1}-1}$, one for each hyperplane. If we order the inequalities according to the hyperplanes that generate them, i.e. in the order $e_1^\perp,\ldots,e_{2^{k+1}-1}^\perp$, then the system of inequalities will have exactly the matrix H_{k+1} as in Lemma 4.6 (a vector is not in the hyperplane $e_j^\perp$ exactly when its scalar product with e_j is 1).

Every odd row of the matrix will correspond to an inequality $\sum_{\substack{r\in[2^{k+1}-1]\\ e_r\notin H}} a_r \geq t_1$. We notice these are the only inequalities that have the variable a_1 in them. Now, if none of these inequalities were sharp then we could reduce a_1 by 1 and all the inequalities would still hold and we would not have a minimal solution. Hence, we can assume that one of these inequalities, let it be the i-th one (i is odd), is actually an equality.

We can now subtract the i-th inequality (equality) from any other inequality and the inequality will still hold. Thus, we subtract (in $\mathbb{Z}^{2^{k+1}-1}$) the i-th row of H_{k-1} from all the other odd rows. This will give inequalities of the form

$$\sum_{\substack{r\in[2^{k+1}-1]\\ e_r\notin H}} a_r - \sum_{\substack{r\in[2^{k+1}-1]\\ e_r\notin e_i^\perp}} a_r \geq 0. \tag{6}$$

Point 3. of Lemma 4.6 says that each odd row except the i-th one now have 1s and -1s in exactly the same columns as a certain even row has 1s. Of these columns, the 1s will be in the ones where the i-th row has a 0 and the -1s in the other ones.

This allows us to pair up the even rows with the odd rows other than the i-th one. Let us add the odd row in each pair to the corresponding even row, this won't change the right hand side of the inequality. Now, the 1s and -1s cancel out and the even row will have 2s in exactly the columns where it originally had 1s and the i-th row had zeroes. Point 4. in Lemma 4.6 says that reading from top to bottom, this gets us the matrix H_k with 2s instead of 1s.

Dividing each inequality through by 2 and restricting ourselves only to the even rows gives a new system of inequalities. Its matrix is H_k and the right hand side has $\frac{t_{i_j+2}}{2}$ in the j-th row. As the left hand side of the inequalities is integral, we can write $\left\lceil \frac{t_{i_j+2}}{2} \right\rceil$ instead. As $\left\lceil \frac{t_2}{2} \right\rceil \geq \ldots \geq \left\lceil \frac{t_{k+1}}{2} \right\rceil$, we can apply the induction hypothesis. Accordingly, we have

$$\begin{aligned}\sum_{\substack{j\in[2^{k+1}-1]\\ e_j\in e_i^\perp}} a_j &\geq Gr\left(\left\lceil \frac{t_2}{2} \right\rceil, \ldots, \left\lceil \frac{t_{k+1}}{2} \right\rceil\right)\\ &= \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{\left\lceil \frac{t_3}{2} \right\rceil}{2} \right\rceil + \ldots + \left\lceil \frac{\left\lceil \frac{t_{k+1}}{2} \right\rceil}{2^{k-1}} \right\rceil\\ &= \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil + \cdots + \left\lceil \frac{t_{k+1}}{2^k} \right\rceil.\end{aligned}$$

Now, as $\sum_{\substack{j\in[2^{k+1}-1]\\ e_j\notin e_i^\perp}} a_j = t_1$, we have

$$n = \sum_{j\in[2^{k+1}-1]} a_j = \sum_{\substack{j\in[2^{k+1}-1]\\ e_j\in e_i^\perp}} a_j + \sum_{\substack{j\in[2^{k+1}-1]\\ e_j\notin e_i^\perp}} a_j \geq t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \cdots + \left\lceil \frac{t_{k+1}}{2^k} \right\rceil = G(t_1, \ldots, t_k).$$

□

4.2 Proof Using UEP Codes

In this chapter we establish a connection between unequal error protection (UEP) codes and UDD codes. UEP codes were discussed in chapter 2.1.

We will need the following result, proved by van Gils in [4]:

Theorem 4.7 (Griesmer Bound for UEP Codes). [[4], Part I Corollary 14] *Let C be an [n,k] linear code with the separation vector $S = (S_1, \ldots, S_k) \in (\mathbb{N} \cup \{0\})^k$ with $S_1 \geq S_2 \geq \ldots \geq S_k$. Then*

$$n \geq \sum_{i=1}^{k} \left\lceil \frac{S_i}{2^{i-1}} \right\rceil .$$

Theorem 4.1 will now be an immediate consequence of the following lemma.

Lemma 4.8. *Let $t_1 \geq \ldots \geq t_k \geq 0$ and $G = (g_{ij})$ be a matrix for a $(t_1, \ldots, t_k)$-UDD code. Then G has a separation vector of $(S(G)_1, \ldots, S(G)_k)$ where $S(G)_i \geq t_i$ for all $i \in [k]$.*

Proof. Let $i \in [k]$. Assume w.l.o.g. that $t_i \geq 1$. Then we have t_i disjoint sets of column indices $I_{i_1}, \ldots, I_{i_{t_i}}$, all of whose respective columns in G sum to e_i. We will show that given a message vector $m = (m_1, \ldots, m_k) \in \mathbb{F}_2^k$ with $m_i \neq 0$ and a recovery set $I_{i_j} = \{i_{j_1}, \ldots, i_{j_l}\}$, the restriction of mG to I_{i_j} will not be the zero vector (i.e. $mG|_{I_{i_j}} \neq \mathbf{0}$). Then each recovery set increases the weight of mG by at least one which proves the claim.

Assume for contradiction that $mG|_{I_{i_j}}$ is the zero-vector. Then $\sum_{r=1}^{k} m_r g_{r,i_{j_s}} = 0$ for all $s = 1, \ldots, l$. But this gives the contradiction

$$0 = \sum_{s=1}^{l} \sum_{r=1}^{k} m_r g_{r,i_{j_s}} = \sum_{r=1}^{k} \sum_{s=1}^{l} m_r g_{r,i_{j_s}} = \sum_{r=1}^{k} m_r \sum_{s=1}^{l} g_{r,i_{j_s}} = \sum_{r=1}^{k} m_r \delta_{ri} = 1,$$

where $\delta_{ri} = \begin{cases} 1 & \text{if } r = i, \\ 0 & \text{otherwise} \end{cases}$ is the Kronecker delta symbol. □

Proof of Theorem 4.1. Let G be the matrix form of a $(t_1, \ldots, t_k)$-UDD code. Then G is a generator matrix for an $[n, k]$ linear code with the separation vector $(S(G)_1, \ldots, S(G)_k)$ by Lemma 4.8. Now the bound follows from Theorem 4.7. □

5 Some other bounds

In this chapter we generalize some known bounds for regular PIR codes to the UDD PIR code case.

Theorem 5.1 (Puncturing constraint). [[3], Lemma 13] *Let $k \in \mathbb{N}$ and $t_1 \geq \ldots \geq t_k \geq 1$. Then*

$$P(k, (t_1, \ldots, t_k)) \geq P(k, (t_1 - 1, \ldots, t_k - 1)) + 1.$$

Proof. Let $c = (a_1, \ldots, a_{2^k-1})$ be a $(t_1, \ldots, t_k)$-UDD code of length $P(k, (t_1, \ldots, t_k))$. Puncturing (removing) any one vector from the code can decrease the number of recovery sets for each symbol by at most one. Thus

$$P(k, (t_1, \ldots, t_k)) - 1 \geq P(k, (t_1 - 1, \ldots, t_k - 1)).$$

□

Theorem 5.2 (Concatenating codes). [[3], Lemma 13] *Let $k \in \mathbb{N}$, $t'_1 \geq \ldots, t'_k$, $t''_1 \geq \ldots \geq t''_k$ and $t_i = t'_i + t''_i$ for all $i = 1, \ldots, k$. Then*

$$P(k, (t_1, \ldots, t_k)) \leq P(k, (t'_1, \ldots, t'_k)) + P(k, (t''_1, \ldots, t''_k)).$$

Proof. Let G' be a generator matrix for a $(t'_1, \ldots, t'_k)$-UDD code and G'' be a generator matrix for a $(t''_1, \ldots, t''_k)$-UDD code. Then $(G' \ G'')$ is a generator matrix for a $(t_1, \ldots, t_k)$-UDD code. □

Theorem 5.3. [[3], Lemma 14] *Let $k \in \mathbb{N}$ and $t_1 \geq \ldots \geq t_k \geq 0$. Then*

$$P\left(k, \left(2\left\lfloor \frac{t_1 + 1}{2} \right\rfloor, \ldots, 2\left\lfloor \frac{t_k + 1}{2} \right\rfloor\right)\right) \leq P(k, (t_1, \ldots, t_k)) + 1.$$

Proof. Assume G is a generator matrix for a $(t_1, \ldots, t_k)$-UDD code of length $P(k, (t_1, \ldots, t_k))$. Let g be the sum of the column vectors of G. If $g \neq \mathbf{0}$, define G' to be G with the additional column g. If $g = \mathbf{0}$, define $G' := G$.

Let t_i be odd. Then the sum of all the recovery sets for t_i in G is the i-th unit vector. Now, as the sum of all the columns of G' is 0, the remaining vectors of G must also sum up to the i-th unit vector, meaning that G' has $t_i + 1$ recovery sets for the i-th unit vector. □

Theorem 5.4 (Simplex codes). *Let $k \in \mathbb{N}$. Then*

$$P(k, (2^{k-1}, \ldots, 2^{k-1})) = G(2^{k-1}, \ldots, 2^{k-1}) = 2^k - 1$$

where $G(t_1, \ldots, t_k) = \sum_{j=1}^{k} \left\lceil \frac{t_j}{2^{j-1}} \right\rceil$ is the Griesmer bound.

Proof. We argued that $P(k, (2^{k-1}, \ldots, 2^{k-1})) \leq 2^k - 1$ after Example 2.6. The reverse inequality follows from Theorem 4.1. □

Corollary 5.5. *Let $k \in \mathbb{N}$ and $t_1 \geq \ldots \geq t_k \geq 0$. Then*

$$P(k, (t_1, \ldots, t_k)) \leq \left\lceil \frac{t_1}{2^{k-1}} \right\rceil (2^k - 1).$$

Proof. We get this bound by concatenating $\left\lceil \frac{t_1}{2^{k-1}} \right\rceil$ simplex codes. □

Theorem 5.6. [[6], Theorem 3]

$$P(k,(3,\dots,3)) \geq k + \left\lceil \sqrt{2k+\frac{1}{4}} + \frac{1}{2} \right\rceil.$$

Furthermore, for $t_1 \geq \ldots \geq t_k \geq 3$, we have

$$P(k,(t_1,\dots,t_k)) \geq k + \left\lceil \sqrt{2k+\frac{1}{4}} + \frac{1}{2} \right\rceil + t_k - 3.$$

Proof. See Theorem 3 in [6]. The second claim follows from repeated application of Theorem 5.1. □

Theorem 5.7. [[5], Proposition 18] *Let ℓ be a positive integer with $\ell \geq P(k,(t_1,\dots,t_k)) - 2t_k$. Let $t_i' = t_i + 2^{k-1}\ell$ and $t_i'' = t_i + 2^{k-1}(\ell-1)$, $i = 1,\dots,k$. Then*

$$P(k,(t_1',\dots,t_k')) = P(k,(t_1'',\dots,t_k'')) + 2^k - 1.$$

Proof. Since the k-dimensional simplex code has length 2^k-1 and 2^{k-1} recovery sets for each unit vector, the inequality $P(k,(t_1',\dots,t_k')) \leq P(k,(t_1'',\dots,t_k'')) + 2^k - 1$ follows from Theorem 5.2.

For the opposite inequality, we first assume that every non-zero vector appears as a column of a generator matrix G of a $(t_i',\dots,t_k')$-UDD code. Then Lemma 2.8 allows us to further assume that each system of recovery recovery sets includes the recovery sets of the k-dimensional simplex code (i.e. recovery sets of the form $\{e_i\}$ and $\{x, x+e_i\}$ for the unit vector e_i). The inequality $P(k,(t_1',\dots,t_k')) \geq P(k,(t_1'',\dots,t_k'')) + 2^k - 1$ follows, as the simplex code has length 2^k-1 and has 2^{k-1} recovery sets for all unit vectors.

Thus, we finally assume that there exists a non-zero vector $v \in \mathbb{F}_2^k$ which does not appear as a column vector in $G \in \mathrm{Mat}_{k,n}(\mathbb{F}_2)$. Let a_j denote the number of occurrences of the vector j (written in binary) as a column vector of G. Lemma 4.5 gives $\sum_{j \notin H} a_j \geq t_k'$ for every hyperplane H of $\mathbb{F}_2^k$.

Now, summing this over the 2^{k-1} hyperplanes which do not contain v, gives

$$\sum_{H \leq \mathbb{F}_2^k :\ \dim(H) = k-1, v \notin H} \ \sum_{j \notin H} a_j \geq 2^{k-1} \cdot t_k'.$$

The coefficient of a_v on the left hand side is 2^{k-1} and the coefficient of a_w for every other non-zero vector $w \neq v$ is

$$|(\mathbb{F}_2^k \setminus v^\perp) \cap (\mathbb{F}_2^k \setminus w^\perp)| = |\mathbb{F}_2^k| - |v^\perp| - |w^\perp| + |v^\perp \cap w^\perp| = 2^k - 2^{k-1} - 2^{k-1} + 2^{k-2} = 2^{k-2}.$$

Using $a_v = 0$, we get

$$2^{k-2} n = 2^{k-2} \sum_{j \in \mathbb{F}_2^k \setminus \{\mathbf{0}\}} a_j \geq 2^{k-1} \cdot t_k',$$

so $n \geq 2t_k' = (2t_k + \ell) + (2^k - 1) \cdot \ell$. Now, since $P(k,(t_1'',\dots,t_k'')) \leq P(k,(t_1,\dots,t_k)) + (2^k-1)(\ell-1)$ (this follows from theorem 5.4), we get

$$\begin{aligned} n &\geq (2t_k + \ell) + (2^k - 1) \cdot \ell \\ &\geq P(k,(t_1,\dots,t_k)) + (2^k-1) \cdot \ell \\ &\geq P(k,(t_1'',\dots,t_k'')) + 2^k - 1. \end{aligned}$$

□

6 UDD PIR codes for $k = 4$

We saw in chapter 3 that the Griesmer bound is tight for $k \in \{1, 2, 3\}$. This will not be true in general and counterexamples exist for $k = 4$ already. For $m \in \mathbb{N}$ we define

$$\begin{aligned} E_m = \{&(3+8m, 3+8m, 3+8m, 3+8m), (4+8m, 4+8m, 3+8m, 3+8m),\\ &(4+8m, 4+8m, 4+8m, 3+8m), (4+8m, 4+8m, 4+8m, 4+8m),\\ &(5+8m, 4+8m, 4+8m, 4+8m)\} \end{aligned}$$

and set

$$E := \bigcup_{m \in \mathbb{N} \cup \{0\}} E_m.$$

The main result of this chapter will be the following theorem:

Theorem 6.1. *Let $t_1 \geq t_2 \geq t_3 \geq t_4 \geq 0$. Let $Gr(t_1, t_2, t_3, t_4)$ denote the Griesmer bound i.e. $Gr(t_1, t_2, t_3, t_4) = t_1 + \left\lceil \frac{t_2}{2} \right\rceil + \left\lceil \frac{t_3}{4} \right\rceil + \left\lceil \frac{t_4}{8} \right\rceil$. Then*

$$P(4, (t_1, t_2, t_3, t_4)) = H(t_1, t_2, t_3, t_4) := \begin{cases} Gr(t_1, t_2, t_3, t_4) + 1 & \text{if } (t_1, t_2, t_3, t_4) \in E, \\ Gr(t_1, t_2, t_3, t_4) & \text{otherwise.} \end{cases}$$

6.1 Upper bound

We shall first prove the upper bound.

Lemma 6.2. *Let H be as in Theorem 6.1. Then*

$$P(4, (t_1, t_2, t_3, t_4)) \leq H(t_1, t_2, t_3, t_4).$$

Proof. 1. We first show that we only need to consider codes with $t_1 \leq 8$.

a) The simplex code

$$(a_1, \ldots, a_{15}) = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$$

is an $(8, 8, 8, 8)$-UDD code of length 15.

Thus, as $H(t_1 + 8, t_2 + 8, t_3 + 8, t_4 + 8) = H(t_1, t_2, t_3, t_4) + 15$, if we have a (t_1, t_2, t_3, t_4)-UDD code of length $H(t_1, t_2, t_3, t_4)$, we can get a $(t_1 + 8, t_2 + 8, t_3 + 8, t_4 + 8)$-UDD code of length $H(t_1 + 8, t_2 + 8, t_3 + 8, t_4 + 8)$ by adding one of each non-zero vector to the code.

Considering this, we may assume $t_4 \leq 7$.

b) Now, assume $t_3 \geq 9$.

Let us first assume that $t_4 \geq 5$. Then $H(t_1, t_2, t_3, t_4) = Gr(t_1, t_2, t_3, t_4)$ and thus

$$H(t_1, t_2, t_3, t_4) = Gr(t_1, t_2, t_3, 8) = H(t_1 - 8, t_2 - 8, t_3 - 8, 0) + 15$$

and we reduced the problem to a code with $t_4 \leq 4$.

Let now $t_4 \leq 4$. Consider the code

$$(a_1, \ldots, a_{15}) = (1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0).$$

It is a $(4, 4, 4, 0)$-UDD code with length 7. As $Gr(t_1 + 4, t_2 + 4, t_3 + 4, t_4) = Gr(t_1, t_2, t_3, t_4) + 7$, if we have a (t_1, t_2, t_3, t_4)-UDD code with length $H(t_1, t_2, t_3, t_4) = Gr(t_1, t_2, t_3, t_4)$, we can get a

$(t_1+4, t_2+4, t_3+4, t_4)$-UDD code of length $H(t_1+4, t_2+4, t_3+4, t_4) = Gr(t_1+4, t_2+4, t_3+4, t_4)$ by increasing by 1 the number of each vector whose fourth coordinate is 0.

Since given $t_4 \leq 7$ we have $H(t_1, t_2, t_3, t_4) \neq Gr(t_1, t_2, t_3, t_4)$ only if $t_3 \leq 4$, we certainly have $H(t_1 - 4, t_2 - 4, t_3 - 4, t_4) = Gr(t_1 - 4, t_2 - 4, t_3 - 4, t_4)$ (as $t_3 - 4 \geq 5$). Repeatedly applying this allows to assume that $t_3 \leq 8$.

c) Assume $t_2 \geq 9$.

Let us first assume that $t_3 = 8$. Then $H(t_1, t_2, t_3, t_4) = H(t_1, t_2, 8, 8) = H(t_1 - 8, t_2 - 8, 0, 0)$ and we reduced the problem to a case where $t_3 \leq 7$.

Setting

$$(a_1, \ldots, a_{15}) = (1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0),$$

we get a $(2, 2, 0, 0)$-UDD code of length 3. As $Gr(t_1 + 2, t_2 + 2, t_3, t_4) = Gr(t_1, t_2, t_3, t_4) + 4$, if we have a (t_1, t_2, t_3, t_4)-UDD code with length $H(t_1, t_2, t_3, t_4) = Gr(t_1, t_2, t_3, t_4)$, we can get a $(t_1 + 2, t_2 + 2, t_3, t_4)$-UDD code of length $H(t_1 + 2, t_2 + 2, t_3, t_4) = Gr(t_1 + 2, t_2 + 2, t_3, t_4)$ by increasing by 1 the number of each vector whose third and fourth coordinate are 0.

Since given $t_3 \leq 8$ we have $H(t_1, t_2, t_3, t_4) \neq Gr(t_1, t_2, t_3, t_4)$ only if $t_2 \leq 5$, we certainly have $H(t_1 - 2, t_2 - 2, t_3, t_4) = Gr(t_1 - 2, t_2 - 2, t_3, t_4)$. Repeatedly applying this allows to assume that $t_2 \leq 8$.

d) Finally, assume that $t_1 \geq 9$. Since $t_2 \leq 8$, we have $H(t_1, t_2, t_3, t_4) \neq Gr(t_1, t_2, t_3, t_4)$ only if $t_1 \leq 5$ and thus $H(t_1 - 1, t_2, t_3, t_4) = Gr(t_1 - 1, t_2, t_3, t_4)$. Repeatedly applying this, we have $H(t_1, t_2, t_3, t_4) = H(8, t_2, t_3, t_4) + t_1 - 8$ and we reduced the problem to $t_1 \leq 8$.

2. We now show that $H(t_1, t_2, t_3, t_4)$ is indeed an upper bound to the minimum length of a (t_1, t_2, t_3, t_4)-UDD code. We define $(t_1, t_2, t_3, t_4) \leq (t'_1, t'_2, t'_3, t'_4) \Leftrightarrow t_i \leq t'_i \; \forall i \in [4]$.

Now, let $T = (t_1, t_2, t_3, t_4)$ and $T' = (t'_1, t'_2, t'_3, t'_4)$ where $t'_1 \geq t_1$ and $t'_i = \min\{t'_{i-1}, 2^i \lceil t_i/2^i \rceil\}$ for $i = 2, 3, 4$ (here we define the values t'_i in the order $i = 1, 2, 3, 4$).

Let now $T'' = (t''_1, t''_2, t''_3, t''_4)$ and $T \leq T'' \leq T'$. Then, if $T, T'' \in E$ or $T, T'' \notin E$ then proving the upper bound for T'' proves the upper bound for T too.

We finally note that if $t_4 = 0$, then the Griesmer bound is tight by Lemma 3.1 (taking $t_4 = 0$).

We provide sample codes which by virtue of the preceding considerations prove the lemma.

a) A $(2, 2, 2, 2)$-UDD code of length $H(2, 2, 2, 2) = Gr(2, 2, 2, 2) = 5$ is given by

$$(a_1, \ldots, a_{15}) = (1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1).$$

b) A $(4, 4, 3, 2)$-UDD code of length $H(4, 4, 3, 2) = Gr(4, 4, 3, 2) = 8$ is given by

$$(a_1, \ldots, a_{15}) = (0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0).$$

c) A $(4, 3, 3, 3)$-UDD code of length $H(4, 3, 3, 3) = Gr(4, 3, 3, 3) = 8$ is given by

$$(a_1, \ldots, a_{15}) = (1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0).$$

This is also a $(3, 3, 3, 3)$-UDD code of length $H(3, 3, 3, 3) = Gr(3, 3, 3, 3) + 1 = 8$.

d) A $(4, 4, 4, 2)$-UDD code of length $H(4, 4, 4, 2) = Gr(4, 4, 4, 2) = 8$ is given by

$$(a_1, \ldots, a_{15}) = (0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0)$$

e) A $(4, 4, 4, 4)$-UDD code of length $H(4, 4, 4, 4) = Gr(4, 4, 4, 4) + 1 = 9$ is given by

$$(a_1, \ldots, a_{15}) = (0, 0, 1, 1, 1, 1, 1, 2, 1, 1, 0, 0, 0, 0, 0).$$

Taking here $a_1 = 1$ we get a $(5,4,4,4)$-UDD code of length $H(5,4,4,4) = Gr(5,4,4,4)+1 = 10$.

f) A $(5,5,4,4)$-UDD code of length $H(5,5,4,4) = Gr(5,5,4,4) = 10$ is given by

$$(a_1,\ldots,a_{15}) = (1,1,0,1,1,1,1,1,1,1,1,0,0,0,0).$$

g) A $(5,5,5,5)$-UDD code of length $H(5,5,5,5) = Gr(5,5,5,5) = 11$ is given by

$$(a_1,\ldots,a_{15}) = (1,1,0,1,1,1,1,1,1,1,1,1,0,0,0).$$

h) A $(6,6,4,4)$-UDD code of length $H(6,6,4,4) = Gr(6,6,4,4) = 11$ is given by

$$(a_1,\ldots,a_{15}) = (1,1,1,1,1,1,1,1,1,1,1,0,0,0,0).$$

i) A $(6,6,6,6)$-UDD code of length $H(6,6,6,6) = Gr(6,6,6,6) = 12$ is given by

$$(a_1,\ldots,a_{15}) = (0,1,1,1,1,1,1,1,1,1,1,1,1,0,0).$$

j) A $(7,7,4,4)$-UDD code of length $H(7,7,4,4) = Gr(7,7,4,4) = 13$ is given by

$$(a_1,\ldots,a_{15}) = (2,2,1,1,1,1,1,1,1,1,1,0,0,0,0).$$

k) A $(7,7,7,7)$-UDD code of length $H(7,7,7,7) = Gr(7,7,7,7) = 14$ is given by

$$(a_1,\ldots,a_{15}) = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,0).$$

l) A $(8,8,4,4)$-UDD code of length $H(8,8,4,4) = Gr(8,8,4,4) = 14$ is given by

$$(a_1,\ldots,a_{15}) = (2,2,2,1,1,1,1,1,1,1,1,0,0,0,0).$$

m) A $(8,8,8,8)$-UDD code of length $H(8,8,8,8) = Gr(8,8,8,8) = 15$ is given by

$$(a_1,\ldots,a_{15}) = (1,1,1,1,1,1,1,1,1,1,1,1,1,1,1).$$

□

6.2 Lower bound

For the cases where $H(t_1,t_2,t_3,t_4) = Gr(t_1,t_2,t_3,t_4)$, Theorem 4.1 suffices for the lower bound. However, when $H(t_1,t_2,t_3,t_4) = Gr(t_1,t_2,t_3,t_4)+1$, additional work needs to be done.

We remind the reader that

$$E_0 = \{(3,3,3,3),(4,4,3,3),(4,4,4,3),(4,4,4,4),(5,4,4,4)\}.$$

Considering Theorem 5.7, it suffices to prove the lower bound for

$$(t_1,t_2,t_3,t_4) \in E_0 \cup \{(12,12,11,11),(12,12,12,11),(13,12,12,12),(20,20,19,19),(20,20,20,19)\}.$$

6.2.1 Theoretical Approach

In this chapter we use the notation d_i^j to denote the number of recovery sets of size j for the vector $e_{2^{i-1}}$ in a given code. Obviously, $d_i^1 = a_{2^{i-1}}$. Also, we only ever consider minimal recovery sets.

Lemma 6.3. *Let $k \in \mathbb{N}$ and $t_1, \geq \ldots \geq t_k \geq 0$. Let G be a generator matrix for a $(t_1, \ldots, t_k)$-UDD code of length n. Let $i \in [k]$, then*

$$2t_i - n \leq d_i^1 \leq n - Gr(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k).$$

Proof. For the inequality $d_i^1 \geq 2t_i - n$, it suffices to see that any code with $d_i^1 < 2t_i - n$ would have at least $2t_i - d_i^1$ recovery sets of size 2 or larger for d_i^1 and we would consequently have

$$2t_i - d_i^1 > n \geq d_i^1 + 2(2t_i - d_i^1) = 4t_i - d_i^1$$

which gives the contradiction $t_i > 2t_i$.

For the inequality $d_i^1 \leq n - Gr(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$, let us consider the matrix G' which we get by removing all the columns $e_{2^{i-1}}$ and the 2^{i-1}-st row from G. Now for any other unit vector $e_{2^{j-1}}$, $j \in [k] \setminus \{i\}$, every recovery set for it in G is also a recovery set for it in G'. However we can ignore the columns $e_{2^{i-1}}$, as they will be zero-vectors in G'. Thus, G' is a generator matrix of length $n - d_i^1$ for a $(t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_k)$-UDD code and must satisfy the Griesmer bound. □

Lemma 6.4. *Let $k \in \mathbb{N}$ and $t_1, \geq \ldots \geq t_k \geq 0$. Let G be a generator matrix for a $(t_1, \ldots, t_k)$-UDD code of length n. Let $i \in [k]$, then*

$$d_i^2 \geq 3t_i - 2d_i^1 - n.$$

Proof. We need $t_i - d_i^1$ recovery sets of size 2 or larger for $e_{2^{i-1}}$ and have $t_i - d_i^1 - d_i^2$ recovery sets of size 3 or larger. Thus,

$$n \geq d_i^1 + 2d_i^2 + 3(t_i - d_i^1 - d_i^2) = 3t_i - 2d_i^1 - d_i^2.$$

□

Lemma 6.5. *Let $k \in \mathbb{N}$ and $t_1, \geq \ldots \geq t_k \geq 0$. Let G be a generator matrix for a $(t_1, \ldots, t_k)$-UDD code of length n. Let $i \in [k]$ and assume $a_1 \geq a_2$. Then*

$$a_3 \geq a_2 - a_1 + 2t_1 - n.$$

Proof. Since $d_1^2 \geq 3t_1 - 2d_1^1 - n = 3t_1 - 2a_1 - n$ by (6.4), there can be at most $n - a_1 - 2(3t_1 - 2a_1 - n) = 3n - 6t_1 + 3a_1$ columns of G in recovery sets of size 3 or larger for e_1. This means that we have at most $\frac{3n - 6t_1 + 3a_i}{3} = n - 2t_1 + a_1$ recovery sets of size 3 or larger for e_1. Now, since none of these recovery sets can include the vector e_2 more than once, we have at least $a_2 - (n - 2t_1 - a_1) = a_2 - a_1 + 2t_1 - n$ recovery sets of size 2 which include the vector e_2. The other vector in each of these sets must be e_3 and (6.5) follows. □

For $a, b \in \{0, 1\}$, we define

$$B(a, b) := \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ a & a & a & a \\ b & b & b & b \end{pmatrix}$$

and N as the number of disjoint (i.e. sharing no columns) submatrices of G of the form $B(a, b)$.

Lemma 6.6. *Let $k \in \mathbb{N}$ and $t_1, \geq \ldots \geq t_k \geq 0$. Let G be a generator matrix for a $(t_1, \ldots, t_k)$-UDD code of length n. Let $i \in [k]$ and assume $a_1 \geq a_2$. Then*

$$N \geq \left\lceil \frac{8t_1 + 3t_2 - 5n - 5a_1 - 2a_2}{2} \right\rceil.$$

Proof. Because of Lemmas 6.3 and 6.5, we can write G in the following form:

$$G = \begin{pmatrix} \overbrace{\begin{matrix}1 & \dots & 1\\ 0 & \dots & 0\\ 0 & \dots & 0\\ 0 & \dots & 0\end{matrix}}^{a_1} & \overbrace{\begin{matrix}0 & \dots & 0\\ 1 & \dots & 1\\ 0 & \dots & 0\\ 0 & \dots & 0\end{matrix}}^{a_2} & \overbrace{\begin{matrix}1 & \dots & 1\\ 1 & \dots & 1\\ 0 & \dots & 0\\ 0 & \dots & 0\end{matrix}}^{a_2-a_1+2t_1-n} & \overbrace{\begin{matrix}\\ \\ \\ \\ \end{matrix}}^{2n-2t_1-2a_2} \\ & & & G' \end{pmatrix}.$$

We denote $G \setminus G'$ the submatrix formed of the columns of G not in G'. G' is of length $n - a_1 - a_2 - (a_2 - a_1 + 2t_1 - n) = 2n - 2t_1 - 2a_2$.

Assume we have two different recovery sets of size 2 for e_1, i.e. we have a submatrix

$$B = \begin{pmatrix} 1 & 0 & 1 & 0 \\ a & a & d & d \\ b & b & e & e \\ c & c & f & f \end{pmatrix}$$

If there is a recovery set of size 2 for e_2 among these columns, then we necessarily get $d = a + 1$, $b = e$ and $c = f$, $B = B(b, c)$. We also get a second recovery set for e_2. This means that in order to get a lower bound on N, we can count how many recovery sets of size 2 for e_2 have both columns as parts of recovery sets of size 2 for e_1. Also, we have no submatrices $B(0, 0)$, as then G would have a zero-column.

By Lemma 6.4, G has at least $3t_1 - 2a_1 - n$ recovery sets of size two for e_1. As $a_2 \geq a_2 - a_1 + 2t_1 - n$ by Lemma 6.3, we have that every such recovery set which includes a column from $G \setminus G'$ must include a column e_2. Hence, there must be at least $3t_1 - 2a_1 - n - a_2$ recovery sets for e_1 whose both columns are in G'. Similarly, we have at least $3t_2 - 2a_2 - n - a_1$ recovery sets of size 2 for e_2 whose both columns are in G'.

We want to find how many of these $3t_2 - 2a_2 - n - a_1$ recovery sets for e_2 in G' must have both columns among the $3t_1 - 2a_1 - n - a_2$ recovery sets for e_1 in G'. For this not to be the case for a recovery set, at least one of its columns must be among the $(2n - 2t_1 - 2a_2) - 2(3t_1 - 2a_1 - n - a_2) = 4n - 8t_1 + 4a_1$ columns which are not in some recovery set of size 2 for e_1 in G'. Hence, there are at least $3t_2 - 2a_2 - n - a_1 - (4n - 8t_1 + 4a_1) = 8t_1 + 3t_2 - 5n - 5a_1 - 2a_2$ suitable recovery sets for e_2. Such recovery sets come in pairs (we have the pairs $\{(1, 1, b, c)^T, (1, 0, b, c)\}$ and $\{(0, 1, b, c)^T, (0, 0, b, c)\}$) and each pair gives a submatrix $B(b, c)$. Thus

$$N \geq \left\lceil \frac{8t_1 + 3t_2 - 5n - 5a_1 - 2a_2}{2} \right\rceil.$$

□

Theorem 6.7. *We have* $P(4, (20, 20, 19, 19)) = P(4, (12, 12, 11, 11)) + 15$.

Proof. We will show that any $(20, 20, 19, 19)$-UDD code of length $n = Gr(20, 20, 19, 19) = 38$ must necessarily contain one of each non-zero vector in $\mathbb{F}_2^4$. This gives $P(4, (20, 20, 19, 19)) \geq P(4, (12, 12, 11, 11))$ by Corollary 2.9. The opposite inequality follows from Theorem 5.2 in conjunction with Theorem 5.4.

We assume that we have a matrix G corresponding to a $(20, 20, 19, 19)$-UDD code of length 38. Showing that G has one of each non-zero vector as a column is equivalent to showing that G contains the columns e_1, e_2, e_3 as well as the submatrices $B(1, 1)$, $B(1, 0)$ and $B(0, 1)$.

By Lemma 6.3, we have $2 \leq a_1, a_2 \leq 3$. We can w.l.o.g. assume $a_1 \geq a_2$.

1. We first assume $a_1 = a_2 = 3$.

Using the Lemmas 6.3, 6.4, 6.5 and 6.6, we get the following bounds:

- $d_1^1 = d_2^1 = 3$, $d_3^1, d_4^1 \leq 3$.
- $d_1^2, d_2^2 \geq 16$, $d_3^2, d_4^2 \geq 13$.
- $a_3 \geq 2$ and $N \geq 5$.

This gets us the following form for G:

$$\left(\begin{array}{cccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \begin{array}{ccccc} B(a,b) & B(c,d) & B(e,f) & B(g,h) & B(k,l) \end{array} \begin{array}{cccccccccc} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & \alpha & \epsilon \\ m & m & q & q & t & t & x & x & \beta & \zeta \\ n & n & r & r & u & u & y & y & \gamma & \eta \\ p & p & s & s & v & v & z & z & \delta & \theta \end{array}\right)$$

We also make the following observations:

- Every recovery set for $e_{2^{i-1}}$ has a column with 1 in the i-th row. Thus, there are at least t_i ones in the i-th row.
- Any recovery set for e_8 must have a column of the form $(-,-,1,0)^T$ or $(-,-,0,1)^T$. Thus, there must be at least $t_8 \geq 19$ columns of the form $(-,-,1,0)^T$ or $(-,-,0,1)^T$.

The third and fourth row must have 19 ones and thus 3 of the submatrices in the middle $B(-,-)$ must be of the form $B(1,-)$ and three of the form $B(-,1)$. By the pigeonhole principle, there must be at least one submatrix $B(1,1)$ and we assume w.l.o.g. that $a = b = 1$. If there also existed submatrices of both the forms $B(1,0)$ and $B(0,1)$, we would already have the full simplex subcode and be done. Thus, we assume w.l.o.g. that we do not have a submatrix $B(0,1)$.

We previously argued that we must have at least 3 submatrices $B(-,1)$, so we have 3 submatrices $B(1,1)$. But now we cannot have $t_4 = 19$ columns $(-,-,1,0)^T$ or $(-,-,0,1)^T$. This is a contradiction.

2. Assume now $a_1 = 3$ and $a_2 = 2$. Using the Lemmas 6.3, 6.4, 6.5 and 6.6, we get the following bounds (we omit the argumentation in a)):
   - $d_1^1 = 3$, $d_2^1 = 2$, $d_3^1, d_4^1 \leq 3$.
   - $d_1^2 \geq 16$, $d_2^2 = 18$, $d_3^2, d_4^2 \geq 13$.
   - $a_3 \geq 1$ and $N \geq 6$.

   We get the following form for G:

$$\left(\begin{array}{cccccc} 1 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \begin{array}{ccccc} B(a,b) & B(c,d) & B(e,f) & B(g,h) & B(k,l) \end{array} \begin{array}{cccccccc} 1 & 0 & 1 & 0 & 1 & 0 & \alpha & \epsilon \\ p & p & s & s & v & v & \beta & \zeta \\ q & q & t & t & w & w & \gamma & \eta \\ r & r & u & u & x & x & \delta & \theta \end{array}\right)$$

   We make the same additional observations as before.

   The third and fourth row must both have at least 19 ones, so the middle part of G must have at least 4 submatrices of the form $B(1,-)$ and 4 of the form $B(-,1)$. By the pigeonhole principle, we must have at least 2 submatrices $B(1,1)$, so w.l.o.g., we let $a = b = c = d = 1$.

   As we must have 19 columns of the form $(-,-,1,0)^T$ or $(-,-,0,1)^T$, we must have at least 3 of the middle $B(-,-)$ submatrices be $B(1,0)$ or $B(0,1)$. If we have both $B(1,0)$ and $B(0,1)$, we are done. We thus assume that we do not have $B(0,1)$ and $e = g = k = m = 1$ and $h = l = n = 0$. But now we cannot have $t_3 = 19$ zeroes in the third row.

3. Let $a_1 = a_2 = 2$. Using the Lemmas 6.4, 6.5 and 6.6, we get the following bounds:
   - $d_1^1 = 2$, $d_2^1 = 2$, $d_3^1, d_4^1 \leq 3$.
   - $d_1^2 = 18$, $d_2^2 = 18$, $d_3^2, d_4^2 \geq 13$.

- $a_3 \geq 2$ and $N = 8$.

Thus, we have the following form for G:

$$\left(\begin{array}{cccccc} 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{array} \quad B(a,b) \quad B(c,d) \quad B(e,f) \quad B(g,h) \quad B(k,l) \quad B(m,n) \quad B(p,q) \quad B(r,s)\right)$$

We make the following additional observations:

- The i-th row has at least d_i^2 zeroes, as every recovery set of size 2 has a column with a zero in the i-th row)
- The i-th row has at least t_i ones.

Since the third and fourth row must have 19 ones, there must be at least 5 submatrices $B(1,-)$ and 5 submatrices $B(-,1)$. By the pigeonhole principle, there are at least two submatrices $B(1,1)$. As the third and fourth row must both have at least 13 zeroes (every recovery set of size 2 has one), there must be submatrices $B(-,0)$ and $B(0,-)$. As we cannot have any submatrices $B(0,0)$, because G does not have zero-columns, G has the submatrices $B(1,0)$ and $B(0,1)$ and we are done.

□

A similar method can be used to reduce determining $P(4,(12,12,11,11))$ and $P(4,(13,12,12,12))$ to determining $P(4,(4,4,3,3))$ and $P(4,(5,4,4,4))$. However, the arguments will be longer and much more convoluted without having lots of theoretical value.

We will therefore not do this and instead propose another solution.

6.2.2 Computer-Based Approach

As we have reduced the infinite problem to a finite problem, we can use a computer to determine the lower bounds. Let $t_1 \geq t_2 \geq t_3 \geq t_4$ as always. We propose the following algorithm to determine if the Griesmer bound $Gr(t_1,t_2,t_3,t_4)$ can be achieved:

Algorithm 6.8.

- We first generate all the recovery sets for each unit vector (for example using algorithm 2.10).
- We now write a function which takes in a vector $(a_1,\ldots,a_{15})$ and returns the (maximal) number of recovery sets for each unit vector in a matrix with a_i columns e_i, $i \in [15]$. This can be implemented as follows:
  - As recovery sets of size 1 and 2 do not intersect, we first count how many of these we have. We can assume that we use up all these columns by Theorem 2.8.

    We count and remove these recovery sets from G and consequently reduce $(a_1,\ldots,a_{15})$ to a vector with at most 7 non-zero entries.
  - We check which recovery sets from our list of recovery sets of size 3 or 4 do not include columns of which we have 0 in our new $(a_1,\ldots,a_15)$.
  - We recursively go through all combinations of such recovery sets and determine the best one. We use dynamic programming for efficiency.
- We now set up a 15-fold for-loop to go through all the vectors $(a_1,\ldots,a_{15})$ with $\sum_{i=1}^{15} a_i = Gr(t_1,t_2,t_3,t_4)$.

  For each one, we find the number of recovery set for each unit vector and check whether they satisfy the UDD condition.

The efficiency of this step can be greatly improved by using the Lemmas 6.3 and 6.4 to set up additional restrictions to which vectors $(a_1, \ldots, a_{15})$ are considered.

- If a suitable solution $(a_1, \ldots, a_{15})$ is found, we return it and stop the program. Otherwise we return that the Griesmer bound can not be attained.

The bound $P(4, (3, 3, 3, 3)) \geq 8$ follows from Theorem 5.6.

Using algorithm 6.8, we determine that $P(4, (4, 4, 3, 3)) > Gr(4, 4, 3, 3)$, $P(4, (5, 4, 4, 4)) > Gr(5, 4, 4, 4)$, $P(4, (12, 12, 11, 11)) > Gr(12, 12, 11, 11)$ and $P(4, (13, 12, 12, 12)) > Gr(13, 12, 12, 12)$. By virtue of Theorem 6.7, we also get $P(4, (20, 20, 19, 19)) > Gr(20, 20, 19, 19)$.

It is easy to see that the above implies the lower bound $P(4, (t_1, t_2, t_3, t_4)) \geq H(t_1, t_2, t_3, t_4)$ for all the vectors

$$(t_1, t_2, t_3, t_4) \in E_0 \cup \{(12, 12, 11, 11), (12, 12, 12, 11), (13, 12, 12, 12), (20, 20, 19, 19), (20, 20, 20, 19)\}.$$

This in conjunction with Lemma 6.2 and Theorem 4.1 proves Theorem 6.1.

To finish this chapter, we state the conjecture 24 from [5] in terms of UDD PIR codes:

Conjecture 6.9.

$$\lim_{k \to \infty} P(k, T) - Gr(T) = \infty,$$

where $T = (2^{k-2}, \ldots, 2^{k-2})$ and $Gr(T) = Gr(2^{k-2}, \ldots, 2^{k-2}) = 2^{k-1}$.

In essence, this conjecture states that the Griesmer bound can arbitrarily loose and suggests that the difference between $P(k, (t_1, \ldots, t_k))$ and $Gr(t_1, \ldots, t_k)$ is especially large when $(t_1, \ldots, t_k)$ is near $(2^{k-2}, \ldots, 2^{k-2})$ i.e. when $\sum_{i=1}^{k} |t_i - 2^{k-2}|$ is small.

Conclusions

In this thesis, we defined the concept of a UDD PIR code as a generalization of a (regular) PIR code. We provided two proofs for an important lower bound called the Griesmer bound for such codes. Furthermore, we proved that this bound is tight for sufficiently small codes ($k \leq 3$) but showed in chapter 6 that it is not tight in general by solving the problem for $k = 4$. In chapter 5, we generalized other known bounds from the PIR case to the more general UDD PIR case.

In the future, the problem could be generalized to non-binary codes. Additionally, using a computer to prove lower bounds is only feasible for small parameters and it would be useful to establish other general lower bounds that are in some cases better than the Griesmer bound (like the bound in Theorem 5.6). As UEP codes have a strong link with UDD codes (cf. Lemma 4.8), this could be done by investigating the PIR capabilities of UEP codes.

References

[1] L. A. Dunning and W. E. Robbins. "Optimal encodings of linear block codes for unequal error protection". *Information and control* 37.2 (1978), pp. 150–177.

[2] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. "Codes for distributed PIR with low storage overhead". *2015 IEEE International Symposium on Information Theory (ISIT)*. 2015, pp. 2852–2856.

[3] Arman Fazeli, Alexander Vardy, and Eitan Yaakobi. "PIR with low storage overhead: coding instead of replication". *arXiv:1505.06241* (2015).

[4] W. J. van Gils. "Design of error-control coding schemes for three problems of noisy information transmission, storage and processing". `https://doi.org/10.6100/IR274904`. PhD thesis. Eindhoven University of Technology, 1988.

[5] Sascha Kurz and Eitan Yaakobi. "PIR Codes with Short Block Length". *Designs, Codes and Cryptography* 89.3 (2021), pp. 559–587.

[6] Sankeerth Rao and Alexander Vardy. "Lower bound on the redundancy of PIR codes". *arXiv:1605.01869* (2017).

Lihtlitsents lõputöö reprodutseerimiseks ja üldsusele kättesaadavaks tegemiseks

Mina, Martin Puškin,

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) minu loodud teose On Unequal Data Demand Private Information Retrieval Codes, mille juhendajad on Ago-Erik Riet ja Henk D.L. Hollmann, reprodutseerimiseks eesmärgiga seda säilitada, sealhulgas lisada digitaalarhiivi DSpace kuni autoriõiguse kehtivuse lõppemiseni.
2. Annan Tartu Ülikoolile loa teha punktis 1 nimetatud teos üldsusele kättesaadavaks Tartu Ülikooli veebikeskkonna, sealhulgas digitaalarhiivi DSpace kaudu Creative Commonsi litsentsiga CC BY NC ND 3.0, mis lubab autorile viidates teost reprodutseerida, levitada ja üldsusele suunata ning keelab luua tuletatud teost ja kasutada teost ärieesmärgil, kuni autoriõiguse kehtivuse lõppemiseni.
3. Olen teadlik, et punktides 1 ja 2 nimetatud õigused jäävad alles ka autorile.
4. Kinnitan, et lihtlitsentsi andmisega ei riku ma teiste isikute intellektuaalomandi ega isikuandmete kaitse õigusaktidest tulenevaid õigusi.

Martin Puškin
10.05.2022