Spring 2024

Results on k-Covers in the Game Quads

Daniel Melvin Rose-Levine
Bard College

Results on k-Covers in the Game *Quads*

A Senior Project submitted to
The Division of Science, Mathematics, and Computing
of
Bard College

by
Daniel Rose-Levine

Annandale-on-Hudson, New York
May, 2024

Abstract

This project explores the k-cover conjecture for the game *Quads*. The conjecture states that for all $k \in \mathbb{N}$, there exists a k-cover if and only if $k = 2$ or $k = \frac{2^n-2}{6}$ for some odd $n \in \mathbb{N}$ with $n > 1$. We show that if k is one of these values, then there exists a k-cover. We discuss partial results in the other direction, i.e., that these are the only values of k for which there exists a k-cover.

Contents

Dedication

This senior project is dedicated to everyone who has supported me this semester, in particular my good friends Josef Lazar and Husna Manalai.

Acknowledgments

First, I would like to thank my advisor Robert McGrail for guiding me through the process of senior project. Additionally, I would like to thank Arthur Collings and Lauren Rose for giving me lots of feedback on drafts in the final days, and hours, before the senior project was due.

Thank you to Darrion Thornburgh for sharing many relevant ideas and finding many relevant references, including the AB k-cover equivalence proof [vF03]. Without this, work on the k-cover conjecture would likely not have progressed nearly as much as it has.

Finally, thanks to Charles Doran for finding the textbook [Was08] which gives the needed result in order to prove the (i, j)-cover formulas for the values of i and j.

1
Introduction

In this project we explore questions relating to the game Quads, a card game that is similar to the popular game Set. The main difference is that a *quad* is a collection of *four* cards that has a special property, while a *set* is a collection of *three* cards that has a special property. The work in this project builds upon research conducted by a number of Bard students, including Julia Crager, Felicia Flores, Darrion Thornburgh, Max Redmond, Raphael Walker, and myself—as well as Bard professor Lauren Rose, and Bard alumnus and Lenoir-Rhyne University professor Timothy Goldberg. Prior work can be found in [Cra+23] and [RRW22].

As stated before, a *quad* is a collection of four cards that has a certain property. We are particularly interested in **caps**, which are sets of cards that *don't* contain a quad. A cap is said to be a k**-cover** if adding *any* card to it will create exactly k quads. The main goal of this project is to understand and classify all possible k-covers, in particular, to characterize the values of k for which there exists a k-cover. The K-Cover Conjecture, originally formulated by Lauren Rose and Raphael Walker, is as follows.

Conjecture 1.0.1 (K-Cover Conjecture)**.** *Let $k \in \mathbb{N}$. Then there exists a k-cover if and only if $k = 2$ or $k = \frac{2^n-2}{6}$ for some odd $n \in \mathbb{N}$ with $n > 1$.*

The main result of this project is a partial resolution to the k-Cover Conjecture. We provide a proof of one direction, that if k is of the form listed in the conjecture, then there exists a

k-cover. For the other direction, that for every k-cover, k is of this form, we establish partial results, and using computational methods, we establish this direction for values of k up to 10^7.

In the literature, a cap in the game Quads is referred to as a Sidon set or a 2-cap in $\mathbb{F}_{2^n}$. This work is directly connected to cryptography, specifically the study of certain classes of cryptographic functions $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ such as **almost perfect nonlinear** (APN) functions and **almost bent** (AB) functions.

We now provide a brief summary of the project.

In Chapter 2, we give an overview of the game Quads, explain how we can represent this game mathematically, prove some basic results, and introduce the Qap Finder to help visualize caps.

In Chapter 3, we derive the k-cover equation, classify k-covers for small values of $k = 1$ and $k = 2$, and also look at a specific example of a k-cover in depth—the 1-cover in dimension 11.

In Chapter 4, we introduce the machinery to prove one direction of the k-cover conjecture. We provide a modern proof that Gold functions in $\mathbb{F}_{2^n}$ for odd n are AB, and explicitly construct a k-cover for each value of k predicted by the k-cover conjecture, proving one direction of the conjecture.

In Chapter 5, we discuss the progress made thus far on proving the second direction of the k-cover conjecture. We provide Python code that we have used to verify the conjecture up to $k = 10^7$.

In Chapter 6, we look at i, j-covers, which arise when using the k-cover construction provided in Chapter 4 for n even rather than n odd. We discuss the analogous results to k-covers for i, j-covers, some of which have been proven, and some of which are conjectured.

2
The Quads Game

2.1 Quads Cards

The game Quads is a card game played with a deck of 64 cards. Each card can be described by a shape, a color, and a number. The shape, color, and number of a card are called its **attributes**. For example, Figure 2.1.1 displays the card with three blue circles.

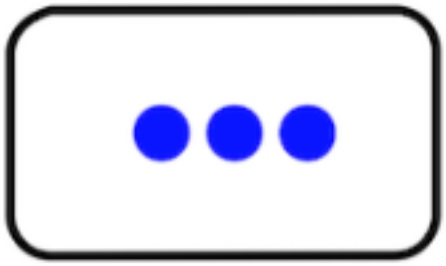

Figure 2.1.1: The card with three blue circles.

In the standard Quads deck, there are four **instances** of each attribute. There are 4 possible shapes, 4 possible colors, 4 possible numbers, and exactly one card for each choice of shape, color, and number. Hence, there are $4^3 = 64$ total cards. Figure 2.1.2 shows a complete Quads deck.

In this illustration of the game, the colors are red, yellow, green, and blue; the shapes are circle, triangle, square, and heart; and the numbers are one, two, three, and four.

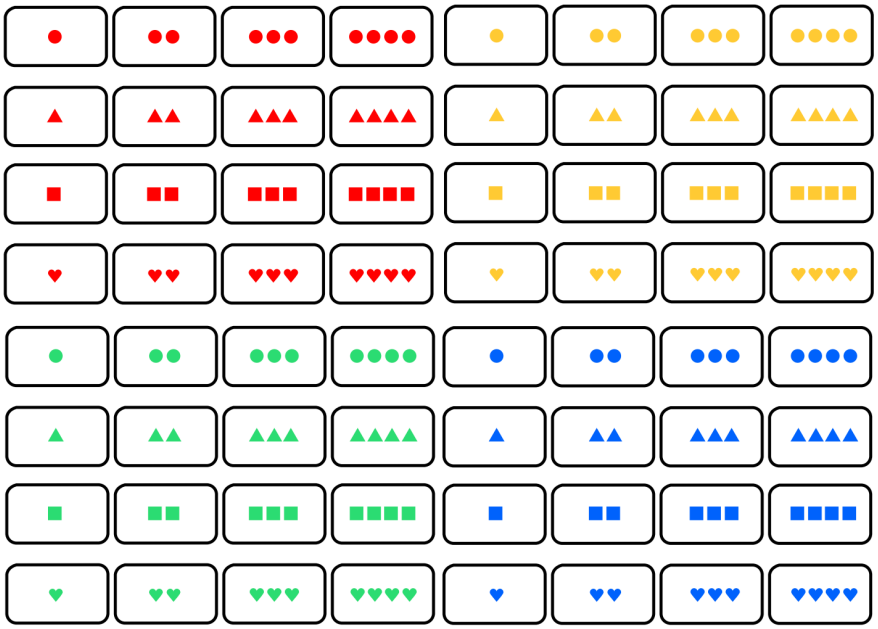

Figure 2.1.2: A complete Quads deck.

The game may also be played with fewer attributes, for example with color removed. Figure 2.1.3 shows an example of a Quads deck with only 16 cards, where the only attributes are shape and number.

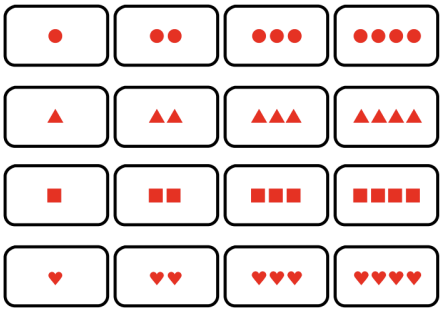

Figure 2.1.3: A Quads deck with 2 attributes.

We may also have a deck with only two instances of a given attribute. This attribute is then said to be a **half attributes**. In Figure 2.1.4, color is a half attribute as there are only two colors, blue and red. There are 32 cards in total, and we say that this deck has two and a half attributes.

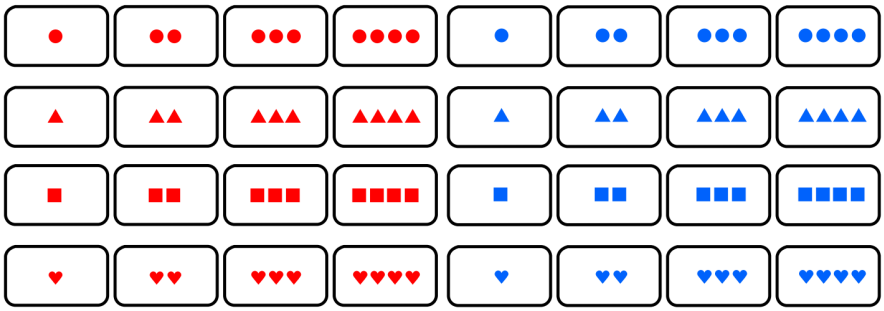

Figure 2.1.4: A Quads deck with 2.5 attributes.

2.2 Rules of the Game

We now go over the rules of the game. First, we must define what a quad is.

Definition 2.2.1. A set of four cards has the **quad property** in an attribute if among the four cards, the instances of this attribute are either all the same, all different, or two and two—meaning that two of the cards contain one instance, and the other two cards contain another. A **quad** is a set of cards that has the quad property in each attribute.

The best way to understand this definition is through examples. Here are a few examples of quads, and an example of four cards that do *not* form a quad.

Example 2.2.2. We first look at Figure 2.2.1. We may examine each attribute individually. First, note that among the four cards, the colors are all the same. Next, the shapes are all different. And finally, the numbers are all the same. So, this is a quad.

Figure 2.2.1: Example of a quad.

Example 2.2.3. We now look at Figure 2.2.2. For color, two of the cards are red and two of them are blue, so this fits under our "two and two" rule. For shape, they are all circles, and for number they are all different. So, this is also a quad.

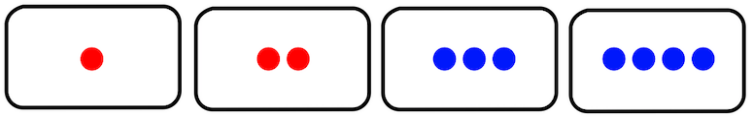

Figure 2.2.2: Example of a quad.

Example 2.2.4. In Figure 2.2.3, the colors are all different, the shapes are two and two (triangles and squares), and the numbers are all different. Thus, these cards form a quad.

Figure 2.2.3: Example of a quad.

Here is an example of four cards that do not form a quad.

Example 2.2.5. In Figure 2.2.4, the colors are all different and the numbers are all different. However, there are two cards with circles, one card with hearts, and one card with triangles. Therefore, these cards do not form a quad, because in the shape attribute, the four cards are neither all the same, all different, nor two and two.

Figure 2.2.4: Example of NOT a quad.

Now that we have established what a quad is, we may proceed with the rules of the game. First, some number of cards are laid out. It is up to the players to decide how many cards to lay out, but usually this number will be somewhere between 8 and 10.

Players will then simultaneously attempt to find four cards that form a quad. The first player to find a quad gets to keep the four cards that form the quad, and four new cards are dealt out in their place. Play continues until there are either no cards left, or all the cards have been laid out and there are no remaining quads. The winner is the player with the most cards at the end of the game.

2.3 Maximal Cap Size

Depending on how many cards have been dealt out, there may not necessarily be a quad among these cards. This is an important practical concern when considering the rules of the game. One would want to know how many cards must be laid out to guarantee a quad in order to determine how many cards they should lay out. This is equivalent to finding the maximum number of cards that can be laid out *without* containing a quad, called the **maximal cap size**. The analogous problem for the game Set, called the Cap Set Problem, is quite famous. It turns out that in Quads, the maximal cap size is 9, and so at least 10 cards must be laid out in order to guarantee a quad. However, there is usually a quad among 8 cards, and with only 8 cards the game is also a bit more challenging. So, depending on the skill level of the players, it is often preferable to start with 8 cards and only add another card if there aren't any quads.

Just as the Quads game can be played with fewer than 3 attributes, it can also be played with more attributes. For example, each card could also have one of four possible background colors. With this additional attribute of background color, there would be $4^4 = 256$ total cards. This can be generalized to an arbitrary number of attributes. If m is the total number of attributes, then there would be 4^m cards in total.

The problem of finding the maximal cap size for a Quads game with m attributes is unsolved in general, although it is known asymptotically up to a constant to be 2^m [TW21].

2.4 Mathematical Representation

In order to study this game mathematically, it is useful to find a more convenient representation for the cards.

The simplest way to do this would be to represent the cards as an ordered triple, with one attribute in each position. For example, the card in Figure 2.1.1 can be represented by (blue, circle, three). A better way would be to map the colors, shapes, and numbers to an element of a fixed set of four elements, say $\{0, 1, 2, 3\}$. Thus we do not need to worry about what the

attributes actually are, but just that there are four instances of each one. Choosing our scheme based on Figure 2.4.1, we may now represent the card (blue, circle, three) as $(3, 0, 2)$.

	Color	Shape	Number
0	red	circle	one
1	yellow	triangle	two
2	green	square	three
3	blue	heart	four

Figure 2.4.1: Associating instances of attributes with the numbers $\{0, 1, 2, 3\}$.

With this scheme, the standard Quads deck may now be represented by $\{0, 1, 2, 3\}^3 = \mathbb{Z}_4^3$. In general, a Quads deck with m attributes would be represented by $\mathbb{Z}_4^m$.

This is a useful way to represent Quads cards, but it turns out that choosing our four element set to be $\mathbb{Z}_2 \times \mathbb{Z}_2$ will end up being more useful mathematically. We denote the elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$ as the bit strings $\{00, 01, 10, 11\}$. Using Figure 2.4.2 as our scheme, the card (blue, circle, three) may now be represented as $(11, 00, 10)$. A Quads deck with m attributes is now represented by $(\mathbb{Z}_2 \times \mathbb{Z}_2)^m$, with which we choose to make the trivial identification with the vector space $\mathbb{Z}_2^{2m}$. Therefore, the standard Quads deck with 3 attributes may be represented by $\mathbb{Z}_2^6$.

	Color	Shape	Number
00	red	circle	one
01	yellow	triangle	two
10	green	square	three
11	blue	heart	four

Figure 2.4.2: Associating instances of attributes with the elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$.

We have shown that a Quads deck with m attributes may be represented by $\mathbb{Z}_2^{2m}$. From here on, we will not refer to the number of attributes in a Quads deck, but rather to the dimension of the vector space it is represented by, which we will call n. Thus, we say that the dimension of

a Quads deck is given by $n = 2m$. This allows for half attributes, and in fact, n can be thought of as the number of half attributes in the deck.

From here on, we will refer to the cards as elements of $\mathbb{Z}_2^n$.

2.5 Some Basic Results and Definitions

Proposition 2.5.1. *Let $a, b, c, d \in \mathbb{Z}_2^n$ be distinct. The cards associated with a, b, c, d form a quad if and only if $a + b + c + d = 0$.*

Proof. Suppose the cards associated with a, b, c, d form a quad. Consider the attribute represented by the first two bits of these elements. Among the four cards, this attribute must either be all the same, all different, or two and two. Thus, the first two bits of a, b, c, and d must either be all the same, all different, or two and two. If they are all the same $x \in \mathbb{Z}_2 \times \mathbb{Z}_2$, then we have

$$x + x + x + x = 00.$$

If they are all different, then they are all the elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$, and we have

$$00 + 01 + 10 + 11 = 00.$$

Finally, if they are two and two,$x \neq y \in \mathbb{Z}_2 \times \mathbb{Z}_2$, then we have

$$x + x + y + y = 00.$$

Regardless, summing the first two bits of each of these elements gives 00. This works with any attribute, not just the attribute represented by the first two bits. Thus the second two bits of each of these elements, third two bits, etc. must also sum to 00. Therefore the sum of the entire bit strings must be 0, i.e., $a + b + c + d = 0$.

Now, suppose the cards associated with a, b, c, d form a quad, i.e., that some attribute of the cards does not have the quad property. The elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$ associated with this attribute among a, b, c, and d must either be of the form x, x, y, z or x, x, x, y. But we have

$$x + x + y + z = y + z \neq 00$$

and

$$x + x + x + y = x + y \neq 00.$$

Thus, $a + b + c + d \neq 0$. □

Using this fact, we may define what it means for four elements of $\mathbb{Z}_2^n$ to be a quad as follows.

Definition 2.5.2. Let a, b, c, d be distinct elements of $\mathbb{Z}_2^n$. Then $\{a, b, c, d\}$ is a **quad** if $a + b + c + d = 0$.

Remark 2.5.3. Note that, as every element of $\mathbb{Z}_2^n$ is its own additive inverse, $a + b + c + d = 0$ is equivalent to $a + b + c = d$ and $a + b = c + d$. We will make use of this fact throughout the project.

Proposition 2.5.4. *Let a, b, c be distinct elements of $\mathbb{Z}_2^n$. Then there is a unique element $d \in \mathbb{Z}_2^n$ such that $\{a, b, c, d\}$ is a quad.*

Proof. We see that $a+b+c+(a+b+c) = 0$, and thus $\{a, b, c, a+b+c\}$ is a quad. For uniqueness, suppose $\{a, b, c, d\}$ is a quad. Then $a + b + c + d = 0$, which implies $d = a + b + c$. □

Definition 2.5.5. A subset $C \subseteq \mathbb{Z}_2^n$ is a **cap** if C contains no quads.

Definition 2.5.6. A cap $C \subseteq \mathbb{Z}_2^n$ is a **complete cap** if $C \cup \{x\}$ contains a quad for all $x \in \mathbb{Z}_2^n - C$.

In other words, a cap C is a complete cap if adding any element to C would create a quad. Equivalently, C is a complete cap if adding any element to C would render it no longer a cap.

Example 2.5.7. The set $\{1011, 0011, 1100, 0100\}$ is a quad in $\mathbb{Z}_2^4$ because

$$1011 + 0011 + 1100 + 0100 = 0000.$$

Example 2.5.8. The set

$$\{0000, 1000, 0100, 0010, 0001\}$$

is a cap in $\mathbb{Z}_2^4$. Adding any four of these bit strings will always result in a bit string that contains either 3 or 4 ones, and thus will certainly not be 0000.

Example 2.5.9. The set

$$C = \{0000, 1000, 0100, 0010, 0001, 1111\}$$

is a complete cap in $\mathbb{Z}_2^4$. Any bit string x not in this set will contain either two or three 1's. We must show that adding any such x to C will create a quad.

In the case that x has two 1's, for example $x = 1100$, we may choose 0000 and the two elements of C that have a 1 in the same positions as the 1's in x. In this case, those elements would be 1000 and 0100. Sure enough, we have

$$1100 + 0000 + 1000 + 0100 = 0000.$$

In the case that x has three 1's, for example $x = 1110$, we may choose 0000, 1111, and the element of C that has a 1 in the same position as the 0 in x. In this case, the element would be 0001. We have

$$1110 + 0000 + 1111 + 0001 = 0000.$$

We have thus shown that C is a complete cap.

Definition 2.5.10. Let $C \subseteq \mathbb{Z}_2^n$ be a cap. A point $x \in \mathbb{Z}_2^n - C$ is an **excluded point** if $C \cup \{x\}$ contains a quad.

Complete caps may also be characterized in terms of excluded points.

Remark 2.5.11. A cap $C \subseteq \mathbb{Z}_2^n$ is a complete cap if and only if every element of $\mathbb{Z}_2^n - C$ is an excluded point.

Definition 2.5.12. Let $C \subseteq \mathbb{Z}_2^n$ be a cap and let $x \in \mathbb{Z}_2^n - C$. The **multiplicity** of x is the number of quads in $C \cup \{x\}$. If the multiplicity of x is k, then x is said to be a **k-point**.

By Remark 2.5.3, the multiplicity of a point x is the number of unordered triples $\{a, b, c\} \subseteq C$ such that $a + b + c = x$. A k-point x would have k solutions $\{a, b, c\}$ to the equation

$$a + b + c = x.$$

2.6 Qap Finder

We now introduce the Qap Finder [Wal], a useful tool for visualizing caps in any Quads game from dimension 2 to 14, created by Raphael Walker. The URL is https://slickytail.github.io/QuadsVis/.

In the Qap Finder there is a grid, with each square in the grid representing a card in the deck, or alternatively an element of $\mathbb{Z}_2^n$. The dimension n may be toggled, which changes the size of the grid to have 2^n squares. Here is an example of a blank grid for dimension 6, the standard Quads deck.

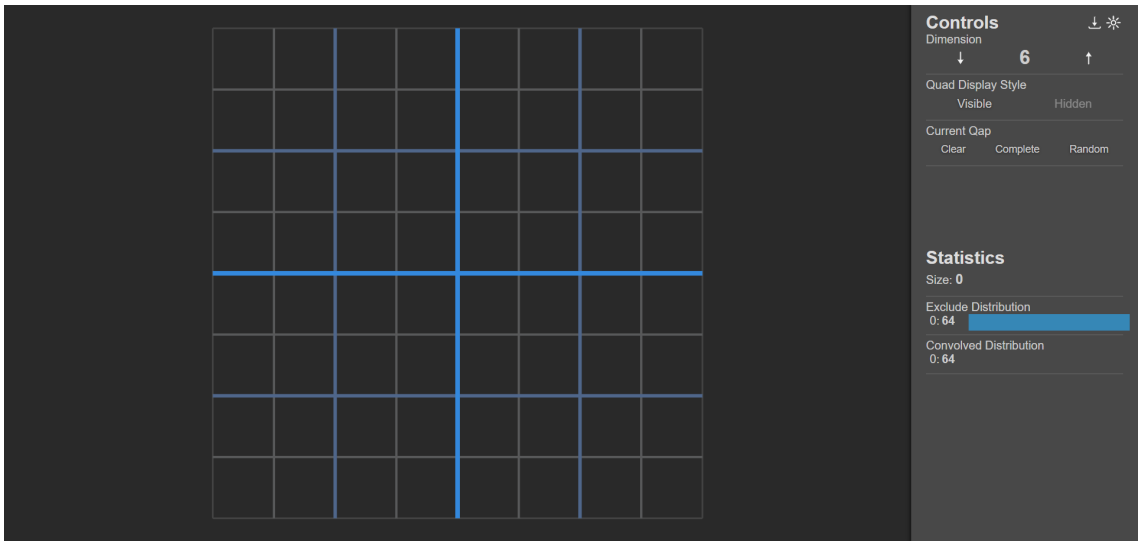

Figure 2.6.1: A blank grid in the Qap Finder.

We explain the coordinate system that associates each bit string in $\mathbb{Z}_2^n$ with a square on the grid, referred to as **recursive coordinates** [Cra+23]. Given a square in the grid, we will describe how to figure out which square in the grid is associated with this bit string, and vice-versa.

The following basic scheme is used. The grid is divided into four equal sub-grids. If the first two bits are 00, then the square in question is in the upper left sub-grid. If the first two bits are 01, then the square is in the upper right sub-grid—and so on according to Figure 2.6.2. Then, in this new sub-grid, the process is repeated with the next two bits in the bit string. Once all the bits have been exhausted, the final "sub-grid" will be a single square, and we are done. In

the case that n is odd, we must first look at the single first bit. If it is a 0, we go to the left sub-grid; if it is a one, the right sub-grid.

00	01
10	11

Figure 2.6.2: Grid layout of the elements of $\mathbb{Z}_2 \times \mathbb{Z}_2$.

Example 2.6.1. Consider the selected square in Figure 2.6.3. This square is in the top right 4x4 sub-grid, so by Figure 2.6.2, the bit string it is associated with starts with 01.

We may then zoom in on this sub-grid, as shown in Figure 2.6.4. Again, by Figure 2.6.2, as the selected square is in the bottom right 2x2 sub-grid, the next two bits are 11.

Zooming in again, shown in Figure 2.6.5, we see that the selected square is in the top left square. Thus, the final two bits are 00.

Putting these together, we see that the element of $\mathbb{Z}_2^4$ associated with the square in Figure 2.6.3 is 011100.

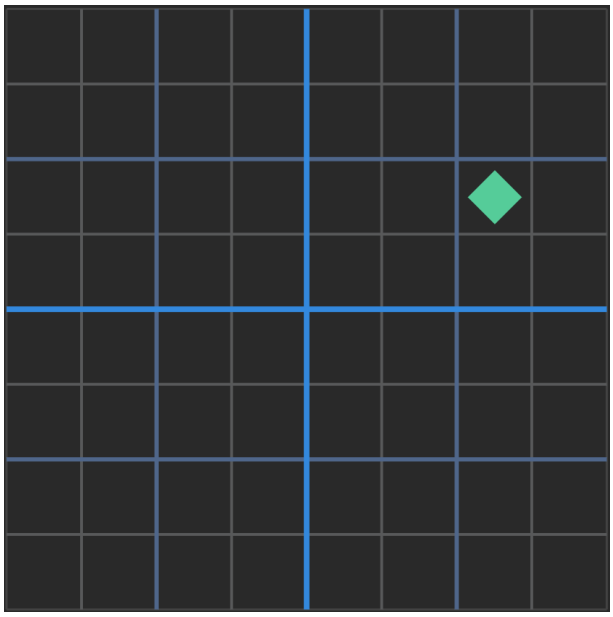

Figure 2.6.3: One square selected, starting with the bits 01.

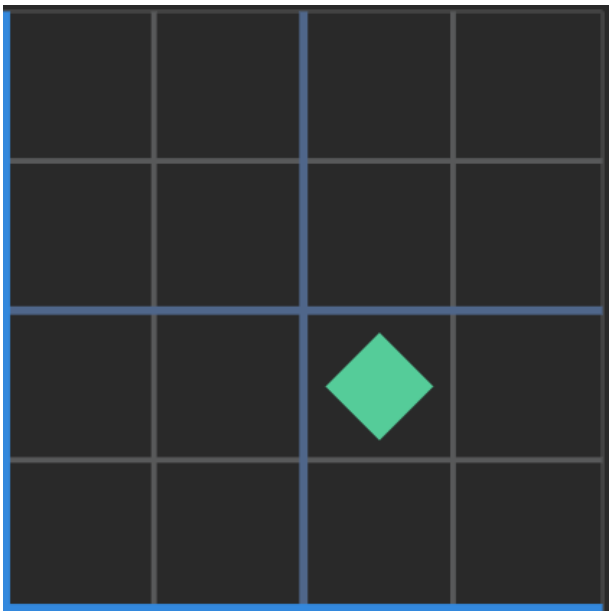

Figure 2.6.4: Contributing the bits 11.

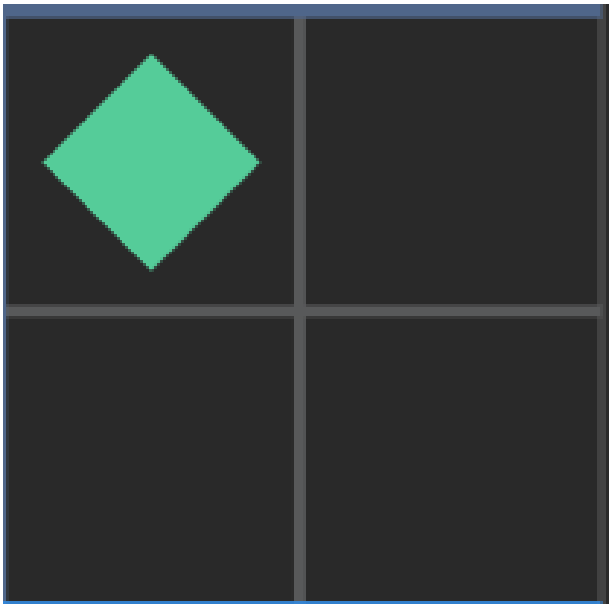

Figure 2.6.5: Contributing the bits 00.

Only caps may be created in the Qap Finder. Excluded points are not allowed to be selected. Displayed in each square representing an excluded point is the multiplicity of that point (the number of quads it completes in the cap). Hovering the cursor over the number will show the triples that sum to this point, as in Figure 2.6.7.

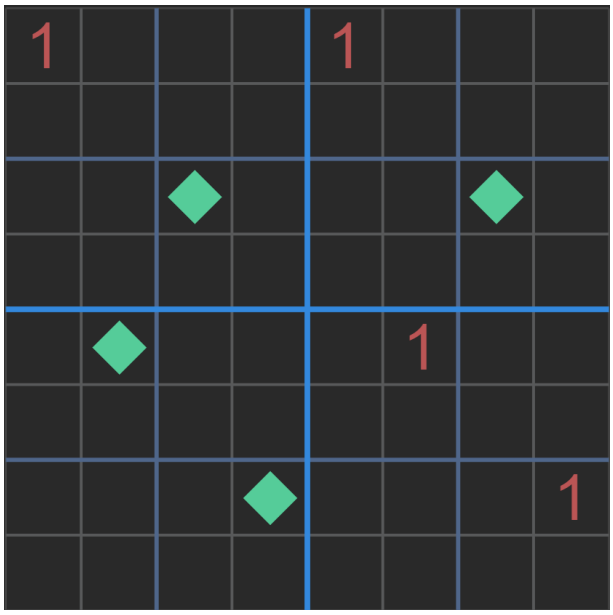

Figure 2.6.6: Some 1-points.

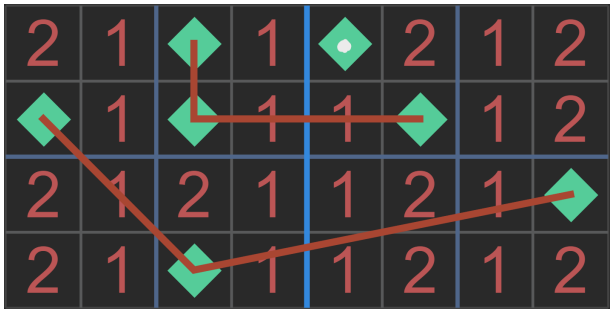

Figure 2.6.7: The two triples that complete a quad with the selected 2-point.

In Figure 2.6.8 the cap from Example 2.5.8 is displayed in the Qap Finder, and in Figure 2.6.9 the complete cap from Example 2.5.9 is displayed. Using the Qap Finder makes it quite easy to check whether a set of elements is a cap, or whether a cap is complete.

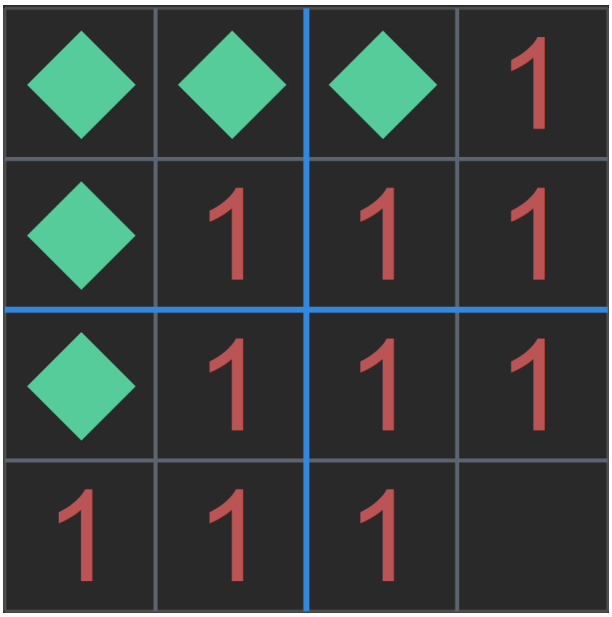

Figure 2.6.8: The cap from Example 2.5.8.

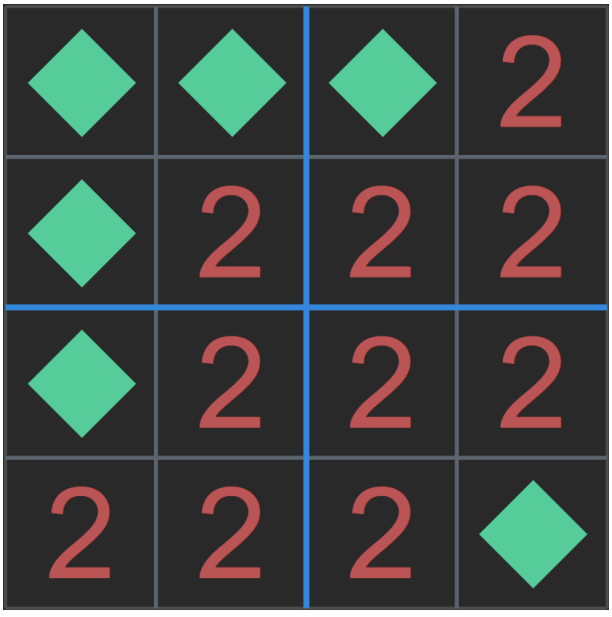

Figure 2.6.9: The complete cap from Example 2.5.9.

3
k-Covers

We now introduce the main topic of this project, k-covers.

Definition 3.0.1. Let $k \in \mathbb{N}$, and $C \subseteq \mathbb{Z}_2^n$ be a cap. C is a k-cover if each element of $\mathbb{Z}_2^n - C$ is a k-point.

Recall that this definition means that for each $x \in C$, there are exactly k unordered triples $\{a, b, c\} \subseteq C$ such that $a + b + c = x$.

The primary purpose of this project is to understand and classify k-covers. In particular, we examine the question: for which values of k does there exist a k-cover?

3.1 Examples

Here are some examples of k-covers, using the Qap Finder.

Example 3.1.1. Recall the complete cap from Example 2.5.9, the cap

$$C = \{0000, 1000, 0100, 0010, 0001, 1111\}.$$

This is a 2-cover. To see this, let $x \in \mathbb{Z}_2^4 - C$, and we must show that x is a 2-point. There are two cases, x has two 1's, and x has three 1's.

For the first case, we suppose $x = 1100$, but the same exact logic may be applied to any x containing two 1's. We see that

$$1000 + 0100 + 0000 = 0010 + 0001 + 1111 = 1100.$$

For the second case, we suppose $x = 1110$, and we see that

$$1000 + 0100 + 0010 = 0001 + 1111 + 0000 = 0001.$$

Therefore each $x \in \mathbb{Z}_2^4 - C$ has multiplicity at least 2.

As $|C| = 6$, there are $\binom{6}{3} = 20$ total triples of elements in C. There are $2^4 - 6 = 10$ total elements in $\mathbb{Z}_2^4 - C$, and we have already shown that each one can be summed to by two triples of elements in C. Thus we have exhausted all 20 triples, and the multiplicity of each point in $\mathbb{Z}_2^4 - C$ must be exactly 2. Therefore, C is a 2-cover.

This 2-cover is displayed in Figure 3.1.1.

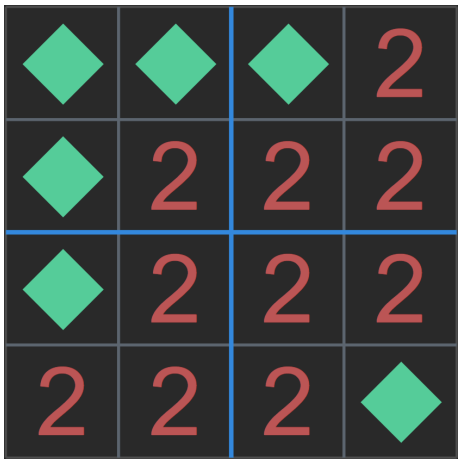

Figure 3.1.1: A 2-cover in dimension 4.

Example 3.1.2. We now show a few examples of 1-covers. Figure 3.1.2 displays a 1-cover in dimension 2 given by $\{00, 01, 10\}$, Figure 3.1.3 displays a 1-cover in dimension 3 given by $\{000, 001, 010, 100\}$, and Figure 3.1.4 displays a 1-cover in dimension 6.

The 1-covers and 2-cover shown thus far are quite easy to find just by playing around with the Qap Finder. In Section 3.5, we will show that the only 2-covers are in dimension 4, and in

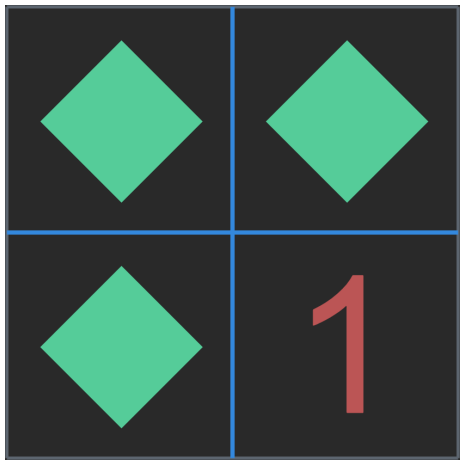

Figure 3.1.2: A 1-cover in dimension 2.

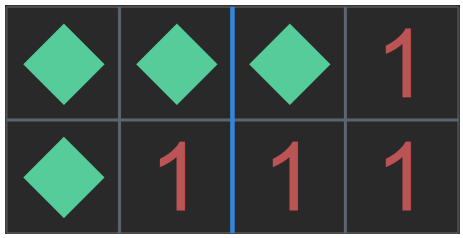

Figure 3.1.3: A 1-cover in dimension 3.

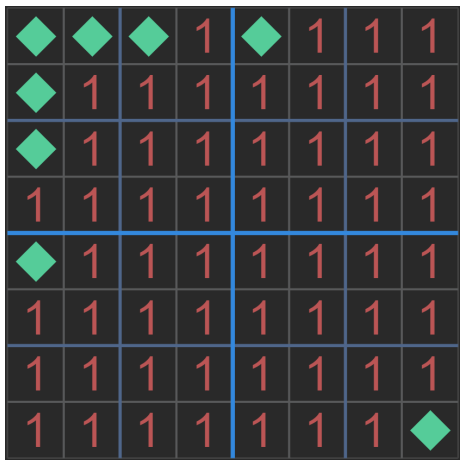

Figure 3.1.4: A 1-cover in dimension 6.

Section 3.4, that there is a 1-cover in dimension 11, but that this is the only other dimension with a 1-cover. In Chapter 4, we will show that there are many more k-covers for certain larger values of k.

3.2 k-Cover Equation

When classifying a k-cover $C \subseteq \mathbb{Z}_2^n$, we are primarily concerned with three quantities: k, n, and s. The first, k, is of course the multiplicity of the points in $\mathbb{Z}_2^n - C$. The second, n, is the dimension of the space that C is living in, $\mathbb{Z}_2^n$. The third, s, is the size of C. For the remainder of the project, we will say a k-cover is of the form (k, n, s). For example, the 2-cover of Example 3.1.1 is of the form $(2, 4, 6)$.

We will now derive an equation relating k, n, and s that will be satisfied by every k-cover of in dimension n of size s.

Proposition 3.2.1 (k-Cover Equation)**.** *Let $C \subset \mathbb{Z}_2^n$ be a k-cover and let $s = |C|$. Then*

$$k(2^n - s) = \binom{s}{3}. \tag{3.2.1}$$

Proof. For each point x outside of C, there are exactly k triples of points in C that sum to x. Additionally, as C is a cap, no triple of points in C can some to an element in C. Thus, this counts all of the triples. As there are $2^n - s$ points outside of C, there must be $k(2^n - s)$ triples of points in C.

Alternatively, we also know that there are $\binom{s}{3}$ triples of points in C. Thus,

$$k(2^n - s) = \binom{s}{3}.$$

□

This equation forms the basis for the rest of the section. We will explore which triples (k, n, s) satisfy this equation. This will ultimately allow us to rule out certain values of k for which there cannot exist a k-cover.

We will first turn the equation into a better form for our purposes. Expanding $\binom{s}{3}$ will make the equation a cubic in s. We have

$$\frac{s(s-1)(s-2)}{6} = k(2^n - s),$$

which can be written as

$$s(s^2 - 3s + 2) = 6k \cdot 2^n - 6ks.$$

This becomes

$$s(s^2 - 3s + 6k + 2) = 6k \cdot 2^n.$$

Thus, we have the following corollary.

Corollary 3.2.2. *$s \mid 6k \cdot 2^n$, and*

$$\frac{6k \cdot 2^n}{s} + 3s = s^2 + 6k + 2. \tag{3.2.2}$$

Note that this form of the equation only includes positive terms, and for this reason it will be particularly useful in sections 3.4 and 3.5.

3.3 A Result on k-Covers

We first note that it is possible to have a one-dimensional Quads deck, $\mathbb{Z}_2^1 = \{0, 1\}$, or even a zero-dimensional Quads deck, $\mathbb{Z}_2^0 = \{\emptyset\}$, where $\emptyset$ is the empty bit string. In either of these cases, the entire space is vacuously a k-cover for all $k \in \mathbb{N}$. There are not even enough elements for a quad to exist. Thus the entire space is a cap, and, as there are no excluded points, a k-cover for any k.

Additionally, for every dimension n, any zero, one, or two-element set will always be a 0-cover, as there will not be any excluded points. Any larger set will always have excluded points, and thus these are the only 0-covers.

Neither of these cases are very interesting. From here on, we require that a k-cover must not be equal to the entire space, in other words that there actually is a k-point in its complement (or simply that its complement is nonempty). Thus we require that $n \geq 2$. Additionally, we require that $k \geq 1$. Note also that given our restrictions on k and n, it must be that $s \geq 3$.

Theorem 3.3.1. *Fix two of (k, n, s). Then the third can take on exactly one value in $\mathbb{R}$ to satisfy the k-cover equation.*

Proof. If n and s or k and s are fixed, then 3.2.1 has a single real number solution for k and n respectively:

$$k(n, s) = \binom{s}{3}/(2^n - s) \tag{3.3.1}$$

and

$$n(k, s) = \log_2\left(\binom{s}{3}/k + s\right). \tag{3.3.2}$$

As we have stipulated that $n \geq 2$, it must not be that a k-cover $C = \mathbb{Z}_2^n$. In other words $s \neq 2^n$, and so 3.3.1 is well-defined. Additionally, $\binom{s}{3} > 0$ and $k \neq 0$, so the argument of the logarithm in 3.3.2 is greater than 0, and it is also well-defined.

Now suppose n and k are fixed. We write Equation 3.2.2 as

$$s^3 - 3s^2 + (6k + 2)s - 6k \cdot 2^n = 0, \tag{3.3.3}$$

and must show that this equation has exactly one solution s. We first show that the function $f : \mathbb{R} \to \mathbb{R}$ defined by $f(s) = s^3 - 3s^2 + (6k + 2)s - 6k \cdot 2^n$ is strictly increasing and thus that it must have at most one zero.

We first differentiate f with respect to s:

$$\frac{df}{ds} = 3s^2 - 6s + 6k + 2.$$

Differentiating again and setting equal to zero will allow us to find the minimum of $\frac{df}{ds}$. This gives

$$6s - 6 = 0,$$

which has solution $s = 1$. Thus, the minimum of $\frac{df}{ds}$ is at $s = 1$, and this minimum is

$$-3 + 6k + 2 \geq -3 + 8 = 5 > 0,$$

as $k \geq 1$. Therefore, $\frac{df}{ds}$ is always greater than 0. Thus f is strictly increasing, and can have at most one zero.

Using the cubic formula, we may explicitly find a zero of f. This solution was computed by Wolfram Alpha. First let

$$D = \sqrt[3]{81k \cdot 2^{n+1} + \sqrt{(81k \cdot 2^{n+1} - 162k)^2 + 4(18k-3)^3} - 162k}$$

The solution is

$$s(k,n) = -\frac{\sqrt[3]{2} \cdot (18k-3)}{3D} + \frac{D}{3 \cdot \sqrt[3]{2}} + 1. \tag{3.3.4}$$

Given that $k \geq 1$ and $n \geq 2$, it can be seen that the term inside the square root in the expression for D is positive, and that D is positive, and therefore nonzero. Thus, the expression (3.3.4) for $s(k,n)$ is always well-defined. □

3.4 1-Covers

We will now use the k-cover equation to classify the 1-covers.

The method used in the following results was originally used by Raphael Walker in [FRW21], and currently remains unpublished. The idea is that we derive an equation where each side is a sum of powers of two. Because every natural number has a unique binary representation, we know that both sides must actually reduce to the same sum of powers of two. For example,

$$2^2 + 2^5 = 36,$$

and

$$2^2 + 2^4 + 2^4 = 36.$$

But we may reduce $2^2 + 2^4 + 2^4$ to $2^2 + 2^5$ by adding the two 2^4 terms together.

Proposition 3.4.1. *Let $C \subseteq \mathbb{Z}_2^n$ be a 1-cover of size s. Then $(n,s) \in \{(2,3),(3,4),(6,8),(11,24)\}$*

Proof. By Corollary 3.2.2, $s \mid 6 \cdot 2^n$. Therefore, it must be that $s = 2^m$ or $s = 3 \cdot 2^m$ for some $m \geq 0$.

Case 1: $s = 2^m$

Recall equation 3.2.2:

$$\frac{6k \cdot 2^n}{s} + 3s = s^2 + 6k + 2.$$

This reduces to

$$6 \cdot 2^{n-m} + 3 \cdot 2^m = 2^{2m} + 8,$$

and we may rewrite this as

$$(2^2 + 2) \cdot 2^{n-m} + (2 + 1) \cdot 2^m = 2^{2m} + 2^3.$$

Simplified, this becomes

$$2^{n-m+2} + 2^{n-m+1} + 2^{m+1} + 2^m = 2^{2m} + 2^3.$$

Now each side is a sum of powers of 2.

As $2m \neq 3$, the right-hand side must have two terms in its binary expansion. Therefore, so does the left-hand side, which implies two of the left-hand side terms must be equal. The terms are of the form $2^a, 2^{a+1}, 2^b, 2^{b+1}$. Without loss of generality, suppose $a \leq b$, and then we have two cases: $b = a$ and $b = a + 1$. In the first case, we have

$$2^a + 2^{a+1} + 2^a + 2^{a+1} = 2^{a+1} + 2^{a+2}.$$

In the second case,

$$2^a + 2^{a+1} + 2^{a+1} + 2^{a+2} = 2^a + 2^{a+2} + 2^{a+2} = 2^a + 2^{a+3}.$$

Thus the powers of these terms must either have a difference of 1 or 3, and so $2m$ and 3 must have a difference of 1 or 3. This implies that $2m \in \{2, 4, 0, 6\}$, so $m \in \{0, 1, 2, 3\}$. As $s = 2^m$, this means $s \in \{1, 2, 4, 8\}$.

As we have stipulated that $n \geq 2$, we know that $s \notin \{1, 2\}$.

Recall Equation 3.3.2 from Theorem 3.3.1:

$$n(k, s) = \log_2 \left(\binom{s}{3} / k + s \right).$$

When $s = 4$, we see that $n = \log_2 \left(\binom{4}{3} + 4 \right) = \log_2(8) = 3$, and $s = 8$ yields $n = \log_2(64) = 6$.

Note that n could very well not be an integer. In this case, we know that there is no dimension with a k-cover of size s. If n is an integer, it is not necessarily the case that there exists a k-cover of size s in that dimension, but there may exist one, and furthermore, there necessarily does not exist a k-cover of size s in any other dimension.

Case 2: $s = 3 \cdot 2^m$

As before, Equation 3.2.1 yields

$$2 \cdot 2^{n-m} + 9 \cdot 2^m = 9 \cdot 2^{2m} + 8,$$

which can be written as

$$2^{n-m+1} + 2^{m+3} + 2^m = 2^{2m+3} + 2^{2m} + 2^3.$$

The three terms on the right-hand side may not be distinct if $2m + 3 = 3$. Otherwise, they must be distinct. Thus, first suppose that $2m + 3 = 3$. This implies $m = 0$, which gives $s = 3 \cdot 2^m = 3$, yielding, by Equation 3.3.2, $n = \log_2(4) = 2$. If they are distinct, then $m \neq 0$, and so it must be that the 2^m term on the left-hand side is equal to the 2^3 term on the right-hand side, and thus $m = 3$. This implies that $s = 24$, and so $n = \log_2(2048) = 11$.

Combining these results gives that

$$(n, s) \in \{(2, 3), (3, 4), (6, 8), (11, 24)\}.$$

□

As we have seen in Figures 3.1.2, 3.1.3, and 3.1.4, there do exist 1-covers for $(n, s) = (2, 3)$, $(3, 4)$, and $(6, 8)$. Additionally, as seen in Figure 3.4.1, there does exist a 1-cover in dimension 11 of size 24. This will be discussed in Section 3.6 in greater detail. Thus, we have:

Theorem 3.4.2. *The complete list of* (n, s) *values for which there exists a* 1*-cover is* $\{(2, 3), (3, 4), (6, 8), (11, 24)\}$.

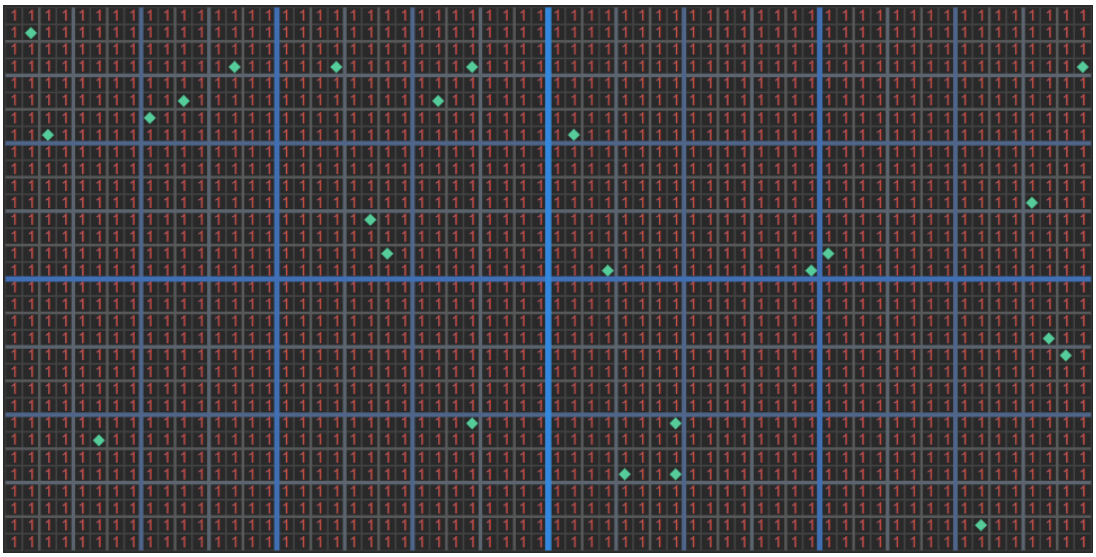

Figure 3.4.1: 1-cover in dimension 11.

3.5 2-Covers and 2^r-Covers

We may use a similar type of proof to classify the 2-covers as we did with the 1-covers.

Proposition 3.5.1. *Let $C \subseteq \mathbb{Z}_2^n$ be a 2-cover of size s. Then $(n, s) = (4, 6)$.*

Proof. Using the same logic as in the proof of Proposition 3.4.1, we obtain two cases: $s = 2^m$ and $s = 3 \cdot 2^m$.

Case 1: $s = 2^m$

Using the same method as before, using Equation 3.2.2, we obtain the equation

$$2^{n-m+3} + 2^{n-m+2} + 2^{m+1} + 2^m = 2^{2m} + 2^3 + 2^2 + 2^1.$$

The four terms on the right-hand side are distinct unless $m = 1$, but this would imply $s = 2$. Therefore, the four terms must be distinct. The exponents on the left-hand side include two pairs of consecutive integers, so on the right-hand side, it must be that $2m \in \{0, 4\}$, which means $m \in \{0, 2\}$. If $m = 0$, then $s = 1$. Thus, $m = 2$, and so $s = 4$. Equation 3.3.2 gives $n = \log_2(6)$, which is not an integer. Thus there is no solution to the k-cover equation in this case.

Case 2: $s = 3 \cdot 2^m$

Here we have, by Equation 3.2.2,

$$2^{n-m+2} + 2^{m+3} + 2^m = 2^{2m+3} + 2^{2m} + 2^3 + 2^2 + 2^1.$$

In order for the right-hand side to have three or fewer distinct terms to match the left-hand side, either $2m = 2$ or $2m + 3 = 3$. But in the latter case, the right-hand side still ends up with four terms. Thus it must be that $2m = 2$, and so $m = 1$. Then we have $s = 6$, which implies $n = \log_2(16) = 4$.

Therefore, we have that $(n, s) = (4, 6)$. □

As seen in Figure 3.1.1, there does exist a 2-cover in dimension 4, and thus we have the following theorem.

Theorem 3.5.2. *The complete list of (n, s) values for which there exists a 2-cover is $\{(4, 6)\}$.*

We now show that this is the *only* case where a k-cover, with k a power of two, can exist.

Theorem 3.5.3. *There does not exist a 2^r-cover for $r > 1$.*

Proof. Suppose $k = 2^r$ for some $r > 1$. As before, either $s = 2^m$ or $s = 3 \cdot 2^m$.

Case 1: $s = 2^m$

We obtain

$$2^{n-m+r+2} + 2^{n-m+r+1} + 2^{m+1} + 2^m = 2^{2m} + 2^{r+2} + 2^{r+1} + 2^1.$$

The exponents on the left-hand side are two pairs of consecutive integers, thus by the argument in case 1 of the proof of Proposition 3.4.1, if they are not all distinct, there must be precisely two distinct terms. As $r > 1$, the three rightmost terms of the RHS are all distinct. There is one way to make there be two distinct terms, by setting $2m = r + 1$. In this case, the right-hand side becomes $2^{r+3} + 2^1$. Again, by case 1 of the proof of Proposition 3.4.1, these exponents must either have a difference of 1 or 3. The difference can't be 1 as $r > 1$, so it must be 3, whence $r = 1$. Thus, there is no solution in this case.

Next suppose that the four terms on each side are distinct. Then the exponents of the right-hand side must be two pairs of consecutive integers. If $2m$ is paired with 1, it must either be that $2m = 0$ or $2m = 2$. Otherwise, the four exponents must be consecutive and $2m = 4$. Thus $m \in \{0, 1, 2\}$. The first two cases imply $s = 1$ or $s = 2$, thus $m = 2$. So $s = 4$, and $n = \log_2(4 \cdot 2^{-r} + 4)$. In order for $4 \cdot 2^{-r} + 4$ to be a power of 2, it must be that $r = 0$. Thus there is no solution for $r > 1$.

Case 2: $s = 3 \cdot 2^m$

We obtain

$$2^{n-m+r+1} + 2^{m+3} + 2^m = 2^{2m+3} + 2^{2m} + 2^{r+2} + 2^{r+1} + 2^1.$$

The only way for the right-hand side to have at most three distinct terms is for $\{2m+3, 2m\} \cap \{r+2, r+1\} \neq \emptyset$. If $2m = r+2$, then there are four distinct terms, so there can't be a solution. Suppose $2m = r + 1$. Then the right-hand side becomes

$$2^{r+4} + 2^{r+3} + 2^1.$$

Note that these terms must be distinct, as $r > 1$. Thus it must be that $m = 1$, but then $r = 1$, and so there is no solution.

Suppose that $2m + 3 = r + 2$. The right-hand side is

$$2^{r+3} + 2^{r+1} + 2^{r-1} + 2^1.$$

The only way for this to have three terms is if $r = 2$. But by assumption, r must be odd, as $r = 2m + 1$. Therefore, there is no solution.

Finally, suppose that $2m + 3 = r + 1$. Then the right-hand side is

$$2^{r+3} + 2^{r-2} + 2^1.$$

These three terms must be distinct as otherwise it would be that $r = 3$. But by assumption, r must be even, as $r = 2m + 2$. Thus, the right-hand side contains 2^{m+3} and 2^m. This can only

happen if $m = 1$, as neither of the other exponents on the right-hand side besides 1 can be 3 smaller than another. Therefore, $r = 4$, and the right-hand side becomes

$$2^7 + 2^2 + 2^1.$$

But no pair of exponents have a difference of 3. Thus the right-hand side does not match the left-hand side, and there is no solution. □

3.6 1-Cover in Dimension 11

We now informally describe the construction of a 1-cover in dimension 11. This section does not contain any formal proofs. Its main purpose is to attempt to understand the structure of a 1-cover in dimension 11.

This 1-cover can be represented many ways. First, without loss of generality, we may choose the standard affine basis in $\mathbb{Z}_2^{11}$, which is just the standard linear basis plus the zero vector (see [Cra+23]):

$$00000000000, 10000000000, 01000000000, 00100000000, \ldots, 00000000001.$$

Next we add the all-ones vector:

$$11111111111.$$

There are many ways to complete the cap and make a 1-cover. Using Python, the rows of the following array have been found to complete a 1-cover with the basis and all-ones vector:

1	1	1	1	1	1	0	0	0	0	0
1	1	1	0	0	0	1	1	1	0	0
1	0	0	0	1	1	0	1	1	1	0
0	1	0	0	1	1	1	0	1	0	1
1	0	1	0	1	0	1	0	0	1	1
0	0	1	1	0	1	1	0	1	1	0
1	0	0	1	0	1	1	1	0	0	1
1	1	0	1	0	0	0	0	1	1	1
0	0	1	1	1	0	0	1	1	0	1
0	1	1	0	0	1	0	1	0	1	1
0	1	0	1	1	0	1	1	0	1	0

This 1-cover is displayed in Figure 3.6.1.

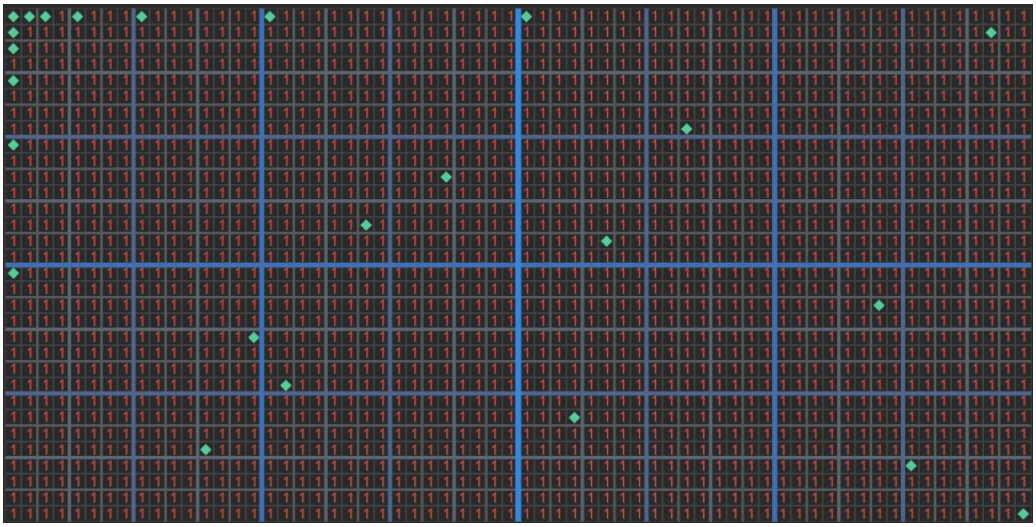

Figure 3.6.1: 1-cover in dimension 11 in the Qap Finder.

Observe that each of these 11 vectors has 6 ones, any two have 3 ones in common, and any pair of indices has exactly three vectors that have ones in those indices. Interestingly, both the rows and the columns of the 11x11 array have these properties.

It has not been proven, but has been observed, that any 11 vectors that complete a 1-cover starting from the standard basis and the all-ones vector seem to have this property. We conjecture that this property is necessary and sufficient to complete a 1-cover.

We now give an explanation as to how each non-cap point can be attained by a sum of three of the 24 cap points. Note that we are trying to reach every point except for the single point with no ones, the points with 1 one, the 11 points in the cap with 6 ones, and the single point with 11 ones. We treat each case separately.

The easy cases are the points with 2, 3, 9, or 10 ones. These points can not be attained by using any of points with 6 ones. This is not obvious, but what is clear is that they *can* be attained by using just the standard basis and the all-ones vector. As long as every point can be shown to be reached at least once, the combinatorics will show that every point can also only be reached at most once.

The following are the ways to sum two or three vectors from the elements of the cap with 6 ones.

$$\begin{array}{r} 11111100000 \\ +11100011100 \\ \hline 00011111100 \end{array}$$

Note that any way to sum two of these elements will yield an element with 6 ones. The 3 ones in common will cancel out, and the other 6 ones will become the 6 ones in the sum.

$$\begin{array}{r} 11111100000 \\ 11100011100 \\ +11010000111 \\ \hline 11001111011 \end{array}$$

In this case the three vectors have 2 ones in common. So those 2 ones are in the sum. Each pair of vectors also has a one in common that isn't in the third, thus these become zeros in the sum. There are 2 remaining ones in each vector that aren't in the other two vectors. So there are 6 more ones in the sum, and thus there are 8 ones in total.

$$\begin{array}{r} 11111100000 \\ 11100011100 \\ +10001101110 \\ \hline 10010010010 \end{array}$$

In this case the three vectors have just 1 one in common. So this one is in the sum. Each pair of vectors also have 2 ones in common that aren't in the third, thus these are zeros in the sum. There is one remaining one in each vector that isn't in the other two vectors. So there are 3 more ones in the sum, and thus there are 4 ones in total.

So, any combination of one, two, or three of the vectors with 6 ones will sum to a vector with 4, 6, or 8 ones.

We may easily show that the following cases can be attained with a sum of three of the cap vectors:

2 ones: the two basis vectors with each of the ones, along with the zero vector

3 ones: the three basis vectors with each of the ones

9 ones: the two basis vectors that coincide with the 2 zeros, along with the all-ones vector

10 ones: the one basis vector that coincides with the 1 zero, along with the all-ones and zero vector

Observe that none of these cases could be attained if one of the 6 ones vectors were used, based on the previous discussion. Using one of these vectors, one can add two vectors from the basis and the all-ones vector to attain 4, 5, 6, 7, and 8 ones. Using two of these vectors, we have seen that we get a new vector with 6 ones, and thus adding a third vector from the basis or the all-ones vector will yield a vector with 5, 6, or 7 ones. Finally, using three of these vectors, we have seen that we either get 4 or 8 ones.

It is not necessary to prove directly that there is only one way to attain each non-cap point, but the fact that the vectors with 2, 3, 9, or 10 ones cannot be reached using a 6 ones vector, and thus only in the ways described above, certainly supports the hypothesis that this is a 1-cover.

We now cover the difficult cases. Here we show that there are $\binom{11}{i}$ ways to sum to vectors with i ones, except for the case $i = 6$, for which there are $\binom{11}{6} - 11$ ways. In order for the 1-cover proof to be complete, it must be shown that each of these are distinct. This has yet to be done in a theoretical way, although it is certainly true as this has been computationally verified to be a 1-cover.

4 ones: For each of the 11 6-ones vectors, 2 ones can be removed using basis vectors, or a one can be added and the remaining 4 zeros turned into ones with the all-ones vector. Finally, three 6-ones vectors that only have 1 one in common can be summed together to get 4 ones. The number of ways to do these is

$$11 \cdot \binom{6}{2} + 11 \cdot \binom{5}{1} + \frac{11 \cdot 10 \cdot 6}{3!} = \binom{11}{4}.$$

5 ones: For each of the 11 6-ones vectors, a one can be removed and the zero vector added, or the vector can be inverted with the all-ones vector and the zero vector added. Additionally, two 6-ones vectors can be added and a one can be removed from the result, or the result can be inverted. This gives

$$11 \cdot \binom{6}{1} + 11 + \binom{11}{2} \cdot \binom{6}{1} + \binom{11}{2} = \binom{11}{5}.$$

6 ones: For each of the 11 6-ones vectors, a one can be removed and another one can be added in place of a zero, using two basis vectors. A one can also be removed and the vector inverted

using the all-ones vector. Finally, two 6-ones vectors can be added along with the zero vector. This gives

$$11 \cdot 6 \cdot 5 + 11 \cdot 6 + \binom{11}{2} = \binom{11}{6} - 11.$$

7 ones: For each of the 11 6-ones vectors, a one can be added along with the zero vector. For each pair of 6-ones vectors, a one can be added. This gives

$$11 \cdot 5 + \binom{11}{2} \cdot 5 = \binom{11}{7}.$$

8 ones: For each of the 11 6-ones vectors, two ones can be added. Additionally, three 6-ones vectors with 2 ones in common can be added together. This gives

$$11 \cdot \binom{5}{2} + \frac{11 \cdot 10 \cdot 3}{3!} = \binom{11}{8}.$$

This completes the construction of the 1-cover of size 24 in dimension 11.

4
APN and AB Functions

4.1 Finite Fields

In order to construct more k-covers, it will be useful to talk about certain kinds of functions on $\mathbb{Z}_2^n$. For this, we consider the finite field of order 2^n, $\mathbb{F}_{2^n}$, for which the additive structure can be identified with $\mathbb{Z}_2^n$. From here on, we use $\mathbb{F}_{2^n}$ to represent a Quads deck of dimension n, rather than $\mathbb{Z}_2^n$.

We will not go into the explicit construction of $\mathbb{F}_{2^n}$, but this can be found in detail in Section 33 of [Fra03].

The field $\mathbb{F}_{2^n}$ has characteristic 2, i.e., $x + x = 0$ for all $x \in \mathbb{F}_{2^n}$.

We have the following identity:

Proposition 4.1.1. *Let $x, y \in \mathbb{F}_{2^n}$ and $k \in \mathbb{N}$. Then $(x+y)^{2^k} = x^{2^k} + y^{2^k}$.*

Proof. We prove this by induction. For $k = 1$, we have

$$(x+y)^2 = x^2 + 2xy + y^2 = x^2 + y^2,$$

with the last equality holding because $2xy = 0$, as $\mathbb{F}_{2^n}$ has characteristic 2.

Now, assume that $(x+y)^{2^k} = x^{2^k} + y^{2^k}$ for all $x, y \in \mathbb{F}_{2^n}$. Let $x, y \in \mathbb{F}_{2^n}$, and we have

$$(x+y)^{2^{k+1}} = ((x+y)^{2^k})^2.$$

By the induction hypothesis, this is equal to

$$(x^{2^k} + y^{2^k})^2.$$

By applying the base case, this is equal to

$$x^{2^{k+1}} + y^{2^{k+1}}.$$

□

As already stated, the additive structure of $\mathbb{F}_{2^n}$ is isomorphic to that of $\mathbb{Z}_2^n$. Additionally, the multiplicative group, i.e., the group of all nonzero elements of $\mathbb{F}_{2^n}$, is cyclic. Thus for some $\alpha \in \mathbb{F}_{2^n}$, we have that

$$\mathbb{F}_{2^n} - \{0\} = \{\alpha^i \mid i \in \mathbb{Z}\},$$

as outlined in the following proposition.

Proposition 4.1.2. *There exists $\alpha \in \mathbb{F}_{2^n}$ such that each nonzero element of $\mathbb{F}_{2^n}$ can be written as α^i for some $i \in \mathbb{Z}$. Additionally, $\alpha^i = \alpha^j$ if and only if $i \equiv j \pmod{2^n - 1}$*

Proof. A proof of the first statement can be found in Corollary 23.6 on page 213 of [Fra03]. Now, suppose $i \equiv j \pmod{2^n - 1}$. So $i - j = (2^n - 1)k$ for some $k \in \mathbb{Z}$. Then

$$\alpha^i \alpha^{-j} = \alpha^{i-j} = \alpha^{(2^n-1)k} = (\alpha^{2^n-1})^k = 1^k = 1$$

. Therefore, $\alpha^i = \alpha^j$. □

Thus, we may write

$$\mathbb{F}_{2^n} = \{0\} \cup \{\alpha^i \mid 0 \leq i \leq 2^n - 2\}.$$

The next proposition shows that a field element raised to two different powers of 2 gives the same result if the exponent of those powers of 2 are equivalent mod n. The **trace** of a field element, introduced later, will be the sum of all of these powers of 2 powers, with exponents ranging from 0 to $n - 1$. This proposition will help to understand some of its properties.

Proposition 4.1.3. *Let $x \in \mathbb{F}_{2^n}$ and $i, j \in \mathbb{Z}$ such that $i \equiv j \pmod{n}$. Then $x^{2^i} = x^{2^j}$.*

Proof. We know that $i = j + kn$ for some $k \in \mathbb{Z}$. We first have that

$$x^{2^{kn}} = x^{2^{(k-1)n}2^n} = (x^{2^n})^{2^{(k-1)n}} = x^{2^{(k-1)n}},$$

using the fact that $x^{2^n} = x$. By induction, $x^{2^{kn}} = x$. Now, we have

$$x^{2^i} = x^{2^{j+kn}} = x^{2^j 2^{kn}} = (x^{2^{kn}})^{2^j} = x^{2^j}.$$

□

Definition 4.1.4. The **graph** of a function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined as $\{(x, f(x)) \mid x \in \mathbb{F}_{2^n}\} \subseteq \mathbb{F}_{2^n} \times \mathbb{F}_{2^n}$. This can be thought of as a subset of $\mathbb{F}_{2^{2n}}$, and we will treat it as such for the remainder of this project.

In the next two sections, we will introduce two kinds of functions from $\mathbb{F}_{2^n}$ to $\mathbb{F}_{2^n}$, **almost perfect nonlinear** (APN) functions and **almost bent** (AB) functions. Both of these classes of functions attempt to define functions that are as nonlinear as possible, in two different ways.

4.2 APN Functions

Definition 4.2.1. A function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is **APN** if for all $a, b \in \mathbb{F}_{2^n}$ with $a \neq 0$, the equation

$$f(x + a) - f(x) = b$$

has at most two solutions.

Note that, as we are only working in characteristic 2, the equation can be modified to be

$$f(x + a) + f(x) = b, \tag{4.2.1}$$

and this is what we will use from now on.

Example 4.2.2 (x^3 is APN)**.** Define $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ by $f(x) = x^3$, and let $a, b \in \mathbb{F}_{2^n}$ with $a \neq 0$. In order to determine whether or not f is APN, we must look at the solutions to Equation 4.2.1. This becomes

$$(x + a)^3 + x^3 = b,$$

which becomes

$$x^3 + 3ax^2 + 3a^2x + a^3 + x^3 = b.$$

As we are in characteristic 2, the x^3 terms cancel and the 3's turn into 1's:

$$ax^2 + a^2x + a^3 = b.$$

This is a quadratic over a field, and thus has at most 2 solutions $x \in \mathbb{F}_{2^n}$. Therefore, f is APN.

The rough idea of this definition is that if the function f were linear, then given an a and a b, we would have

$$f(x + a) + f(x) = f(a). \tag{4.2.2}$$

If $b \neq f(a)$, then Equation 4.2.1 would have no solutions, but if $b = f(a)$, then every $x \in \mathbb{F}_{2^n}$ is a solution. In some sense then, what this definition is saying is that given any $a \in \mathbb{F}_{2^n}$, there is no good candidate for $f(a)$ that would have f satisfy the linearity equation (4.2.2). If $f(a)$ were to equal any value of b, then the linearity equation would be satisfied by at most two values of x.

One obvious question about this definition is why it is at most *two* solutions to Equation 4.2.1? Why not at most *one* solution? The answer is that the latter would actually make it a **perfect nonlinear** function. But because we are in a field of characteristic 2, there are no perfect nonlinear functions. This can be seen in the following proposition.

Proposition 4.2.3. *Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function and $a, b \in \mathbb{F}_{2^n}$ with $a \neq 0$. If Equation 4.2.1 has a solution $x \in \mathbb{F}_{2^n}$, then it has at least two solutions.*

Proof. Suppose that x satisfies Equation 4.2.1, i.e., $f(x + a) + f(x) = b$. Then

$$f((x + a) + a) + f(x + a) = f(x) + f(x + a) = b.$$

Therefore $x + a$ also satisfies Equation 4.2.1, and as $a \neq 0$, we have $x + a \neq x$. □

Corollary 4.2.4. *A function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is APN if and only if for all $a, b \in \mathbb{F}_{2^n}$ with $a \neq 0$, Equation 4.2.1 has either 0 or 2 solutions $x \in \mathbb{F}_{2^n}$.*

Therefore, if the equation has one solution, it will necessarily have another, and so the definition of APN is what makes the most sense in characteristic 2 as a general definition.

We will now show a relationship between APN functions and caps. The following theorem shows that the graph of a function is a cap if and only if that function is APN. One direction of this theorem, that the graph of an APN function is a cap, was proven by Michael Tait and Robert Won in [TW21]. We provide a proof of the other direction.

Theorem 4.2.5. *A function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is APN if and only if the graph of f is a cap in $\mathbb{F}_{2^{2n}}$.*

Proof. Suppose f is not APN. Then for some $a, b\mathbb{Z}_2^n$ with $a \neq 0$, Equation 4.2.1 has three solutions. Let one of the solutions be x. By Proposition 4.2.3, $x + a$ is also a solution. Let y be the third solution. Note that as y is distinct from x, and $x + a$, so is $y + a$. We show that

$$\{(x, f(x)), (x + a, f(x + a)), (y, f(y)), (y + a, f(y + a))\}$$

is a quad, and thus that the graph of f is not a cap. We have

$$x + (x + a) + y + (y + a) = 0$$

and

$$f(x) + f(x + a) + f(y) + f(y + a) = b + b = 0.$$

See Theorem 5.1 in [TW21] for a proof of the other direction, that if f is APN then its graph is a cap. □

This result provides a recipe for constructing a cap of size 2^n in any even dimension $2n$, given an APN function in dimension n. In order to use this result to actually construct a cap in each dimension, we must have a specific example of an APN function in each dimension n. As we have shown, in 4.2.2, x^3 is APN in each dimension. Thus we have the following corollary.

Corollary 4.2.6. *There exists a cap of size 2^n in each even dimension $2n$.*

We now define the *Gold functions*, of which x^3 is an example.

Definition 4.2.7. A function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is **Gold** if $f(x) = x^{2^\ell+1}$ where $(n, \ell) = 1$.

By Table 1.1 in [Dav21], we have the following theorem.

Theorem 4.2.8. *Gold functions are APN.*

Note that $(n, 1) = 1$ for all n, and thus $x^{2^1+1} = x^3$ is Gold for all n.

In Figures 4.2.1-4.2.7 we show the graph of the APN function x^3 in dimensions $n = 1$ to $n = 7$ in the Qap Finder. Note that the graph will then be in $\mathbb{F}_{2^{2n}}$, and will thus be in the even dimensions from 2 to 14. For $n = 6$ and $n = 7$, we only show a portion of the graph, as they are quite large.

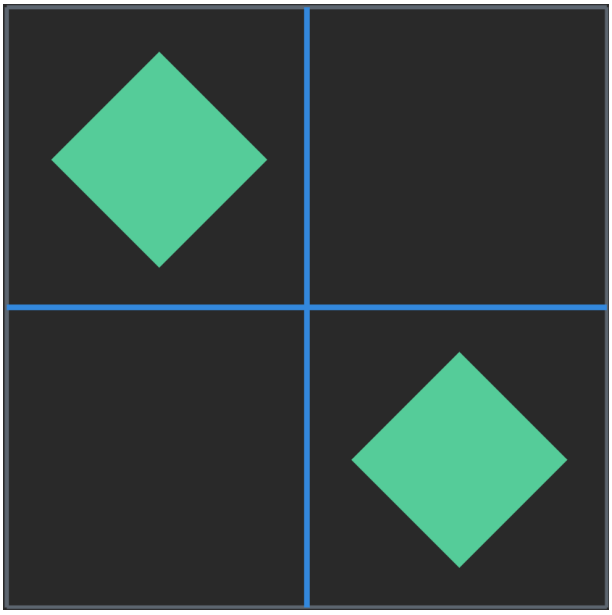

Figure 4.2.1: The graph of x^3 in $\mathbb{F}_{2^2}$ $(n = 1)$.

When looking at these caps in the Qap Finder, some immediate observations can be made. When n is odd, the construction seems to always yield a k-cover. When $n = 1$, a 0-cover; when $n = 3$, a 1-cover; when $n = 5$, a 5-cover; and when $n = 7$, a 21-cover. In fact, these values appear to satisfy

$$k(n) = \frac{2^n - 2}{6}. \tag{4.2.3}$$

When n is even, the non-cap points seem to always have one of two different multiplicities. In addition, it can be observed that one of these two multiplicities appears exactly twice as many

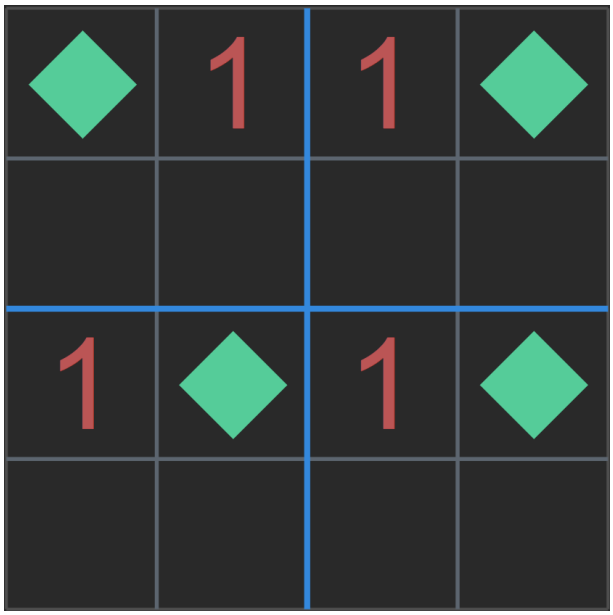

Figure 4.2.2: The graph of x^3 in $\mathbb{F}_{2^4}$ $(n = 2)$.

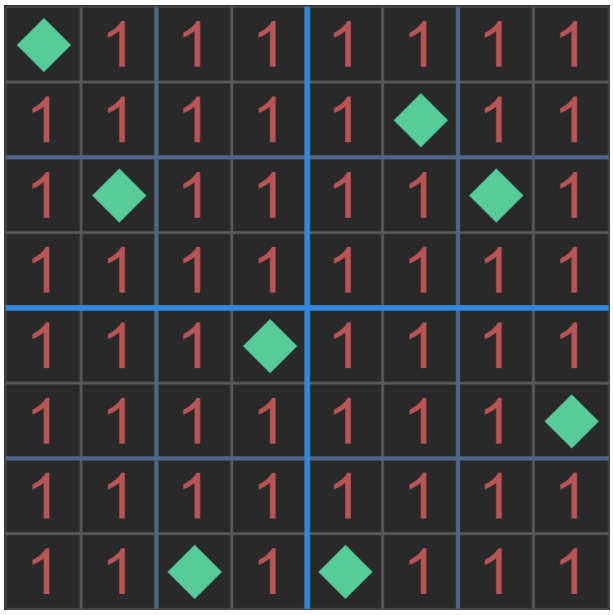

Figure 4.2.3: The graph of x^3 in $\mathbb{F}_{2^6}$ $(n = 3)$.

Figure 4.2.4: The graph of x^3 in $\mathbb{F}_{2^8}$ $(n = 4)$.

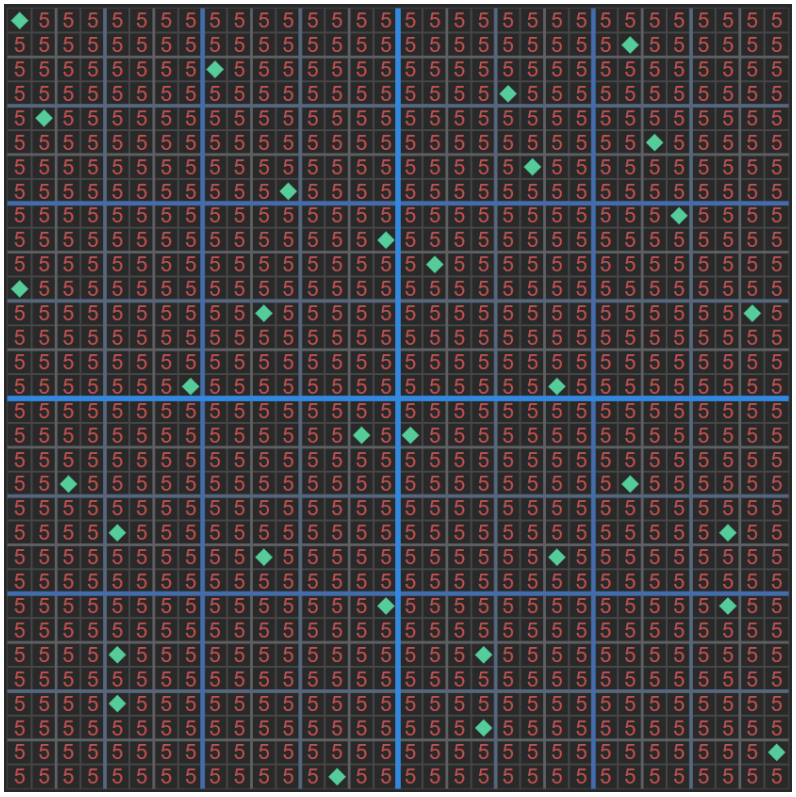

Figure 4.2.5: The graph of x^3 in $\mathbb{F}_{2^{10}}$ $(n = 5)$.

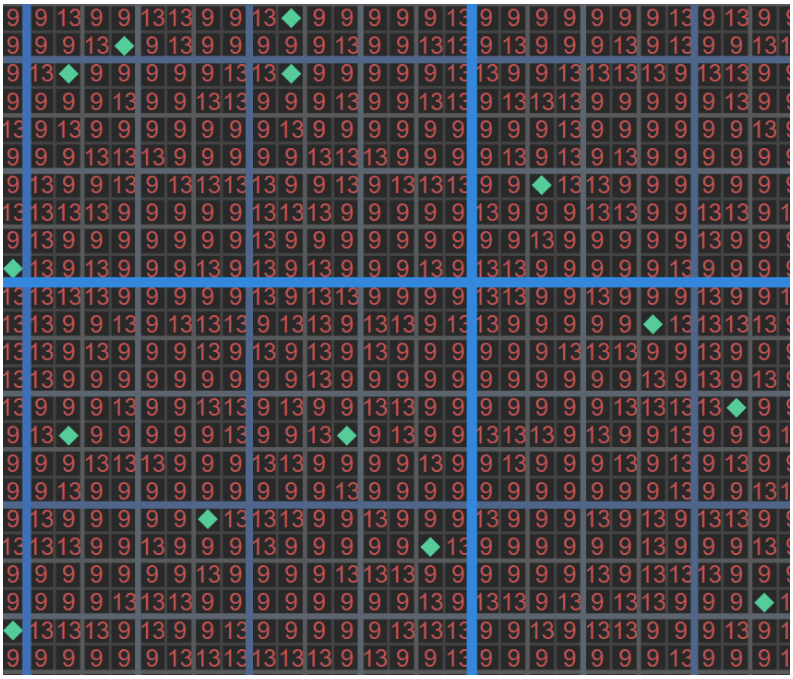

Figure 4.2.6: A portion of the graph of x^3 in $\mathbb{F}_{2^{12}}$ $(n = 6)$.

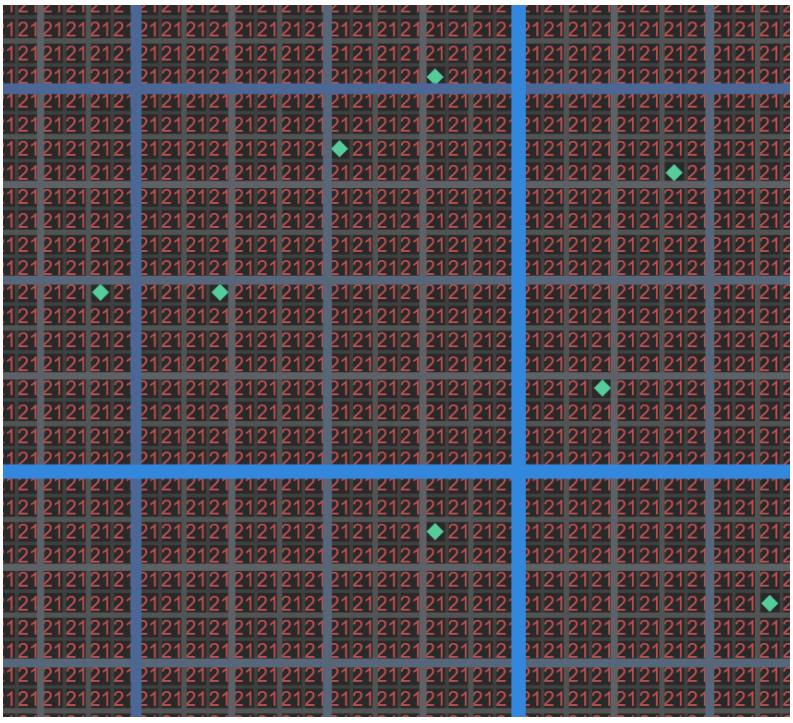

Figure 4.2.7: A portion of the graph of x^3 in $\mathbb{F}_{2^{14}}$ $(n = 7)$.

times as the other. For example, when $n = 2$, there are eight 0-points and four 1-points. When $n = 3$, there are 160 3-points and 80 1-points. We call these (i, j)-**covers**. When $n = 2$, we have a $(0, 1)$-cover; when $n = 4$, a $(3, 1)$-cover; and when $n = 6$, a $(9, 13)$-cover. Note that i and j appear to satisfy

$$i(n) = \frac{2^n - 2}{6} - \frac{(-2)^{n/2}}{6} \tag{4.2.4}$$

and

$$j(n) = \frac{2^n - 2}{6} + \frac{(-2)^{n/2}}{3}, \tag{4.2.5}$$

where i-points make up two-thirds of the non-cap points, and j-points make up one-third of them. We discuss i, j-covers in more detail in Chapter 6.

In fact, the graph of every Gold function, not just x^3, seems to be a k-cover for odd n, and an (i, j)-cover for even n (with the same values of k and i, j). We prove this in Section 4.6 for k-covers and Chapter 6 for (i, j)-covers. However, there are APN functions which are not Gold. For example, the **inverse** function x^{15} in $\mathbb{F}_{2^5}$, found in Table 1.1 of [Dav21]. See Figure 4.2.8 for the graph of this APN function. Notice that it is not a k-cover. It turns out that the functions whose graphs are k-covers belong to a more restricted class of functions called AB functions. The Gold functions for n odd happen to be AB, but x^{15} in $\mathbb{F}_{2^5}$ is not.

It does, however, appear that the graph of an APN function is always a complete cap, for $n \geq 3$. This is still an open problem (see Conjecture 3 in [Car22]).

4.3 AB functions

We know that the graph of an APN function is always a cap (and maybe a complete cap), but in order to understand for which functions the graph is a k-cover, we have to introduce another kind of function, **almost bent** (AB) functions.

The definition of an AB function is considerably more involved than that of APN functions. We start with some preliminary definitions. We only show what is necessary here for our

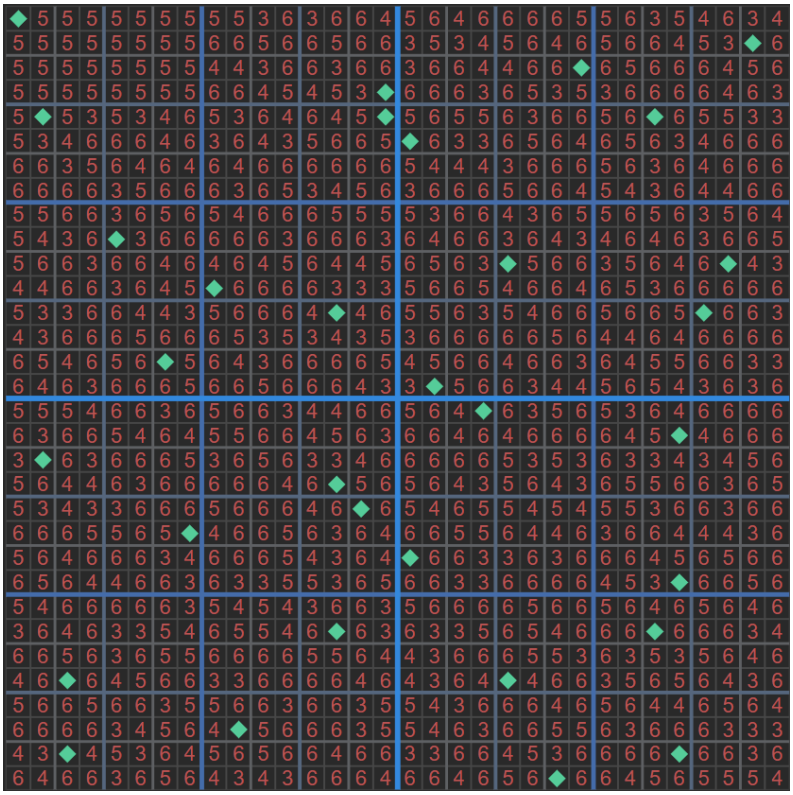

Figure 4.2.8: The graph of the APN function x^{15} in $\mathbb{F}_{2^5}$.

purposes, but for more details, including a characterization of AB functions in terms of their **nonlinearity** and discussion of **bent** functions, see [Dav21].

Definition 4.3.1. The **trace**, $\mathrm{Tr} : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is defined by

$$\mathrm{Tr}(x) = \sum_{i=0}^{n-1} x^{2^i}$$

An interesting result that may help to understand the trace is the following, that the trace can be represented as the sum of x raised to *any* n consecutive powers of two.

Proposition 4.3.2. *Let $k \in \mathbb{Z}$ and $x \in \mathbb{F}_{2^n}$. Then*

$$\mathrm{Tr}(x) = \sum_{i=k}^{k+n-1} x^{2^i}.$$

Proof. Use the division algorithm to write $k = qn + r$ with $0 \leq r < n$. Then, we have

$$\sum_{i=k}^{k+n-1} x^{2^i} = \sum_{i=qn+r}^{qn+(n-1)} x^{2^i} + \sum_{i=q(n+1)}^{q(n+1)+(r-1)} x^{2^i}.$$

By Proposition 4.1.3, this is equal to

$$\sum_{i=r}^{n-1} x^{2^i} + \sum_{i=0}^{r-1} x^{2^i} = \sum_{i=0}^{n-1} x^{2^i} = \mathrm{Tr}(x).$$

□

Corollary 4.3.3. *Let $j, k \in \mathbb{Z}$. Then $\mathrm{Tr}(x^{2^j}) = \mathrm{Tr}(x^{2^k})$.*

It turns out that the trace of any field element is 0 or 1, and the trace is often defined to have codomain $\{0, 1\} = \mathbb{F}_2$. This is justified in the following proposition.

Proposition 4.3.4. $\mathrm{Tr}(\mathbb{F}_{2^n}) \subseteq \{0, 1\}$.

Proof. Let $x \in \mathbb{F}_{2^n}$. Then

$$\mathrm{Tr}(x)^2 = \left(\sum_{i=0}^{n-1} x^{2^i}\right)^2,$$

which by Proposition 4.1.1, is equal to

$$\sum_{i=0}^{n-1} (x^{2^i})^2 = \sum_{i=0}^{n-1} x^{2^{i+1}} = \sum_{i=1}^{n} x^{2^i}.$$

By 4.3.2, this is equal to $\mathrm{Tr}(x)$. Thus, $\mathrm{Tr}(x)^2 = \mathrm{Tr}(x)$, which implies $\mathrm{Tr}(x)(\mathrm{Tr}(x) - 1) = 0$, and so $\mathrm{Tr}(x) = 0$ or $\mathrm{Tr}(x) = 1$. □

Note that Tr is linear.

Proposition 4.3.5. *Let $x, y \in \mathbb{F}_{2^n}$. Then $\mathrm{Tr}(x + y) = \mathrm{Tr}(x) + \mathrm{Tr}(y)$.*

Proof. Using Proposition 4.1.1, we have

$$\mathrm{Tr}(x + y) = \sum_{i=0}^{n-1} (x + y)^{2^i} = \sum_{i=0}^{n-1} x^{2^i} + y^{2^i} = \mathrm{Tr}(x) + \mathrm{Tr}(y).$$

□

Definition 4.3.6. Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function. The **Walsh transform**, $W_f : \mathbb{F}_{2^n} \times \mathbb{F}_{2^n} \to \mathbb{Z}$, of f, is defined by

$$W_f(a, b) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\mathrm{Tr}(ax + bf(x))}.$$

We may also write $W_f(a,b)$ as $W(a,b)$ when f is clear by the context.

We define $S : \mathbb{F}_{2^n} \to \{-1,1\}$ by

$$S(x) = (-1)^{\mathrm{Tr}(x)}.$$

Thus, we may write

$$W_f(a,b) = \sum_{x \in \mathbb{F}_{2^n}} S(ax + bf(x)).$$

Note the following fact that follows from the linearity of the trace:

Proposition 4.3.7. *Let $x, y \in \mathbb{F}_{2^n}$. Then $S(x+y) = S(x)S(y)$.*

Proof. We have

$$S(x+y) = (-1)^{\mathrm{Tr}(x+y)} = (-1)^{\mathrm{Tr}(x)+\mathrm{Tr}(y)} = (-1)^{\mathrm{Tr}(x)}(-1)^{\mathrm{Tr}(y)}.$$

□

Additionally, $\langle x, y \rangle = \mathrm{Tr}(xy)$ defines an inner product, and the Walsh transform may be written as

$$W_f(a,b) = \sum_{x \in \mathbb{F}_{2^n}} (-1)^{\langle a,x \rangle + \langle b, f(x) \rangle}.$$

The Walsh transform may be defined using any inner product, including the dot product over $\mathbb{Z}_2^n$. Thus it is technically not necessary to have a field structure to discuss the Walsh transform. However, we do require this, as we will be taking the Walsh transform of functions which require the field structure to describe.

What we are most interested in is the **Walsh spectrum** of f.

Definition 4.3.8. Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function. The **Walsh spectrum** of f is the image of W_f.

We are now ready to define AB functions.

Definition 4.3.9. A function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is **AB** if its Walsh spectrum is contained in $\{0, \pm 2^{\frac{n+1}{2}}\}$.

We now prove a basic result about the Walsh transform, Parseval's equation, which is stated but not proved in [Dav21].

The following Lemma is true, but more difficult to prove, for n even.

Lemma 4.3.10. *Let $n \in \mathbb{N}$ be odd. Then* Tr *is* ***balanced****, i.e.,* $\mathrm{Tr}(x) = 0$ *for exactly half of the elements $x \in \mathbb{F}_{2^n}$, and* $\mathrm{Tr}(x) = 1$ *for the other half.*

Proof. As n is odd,

$$\mathrm{Tr}(1) = \sum_{i=0}^{n-1} 1 = n \cdot 1 = 1.$$

Thus, the mapping $x \mapsto x + 1$ maps elements whose trace is 1 to elements whose trace is 0, and vice-versa. Additionally, it is its own inverse. Therefore, it is a bijection between the elements whose trace is 1 and the elements whose trace is 0. □

Proposition 4.3.11. *Let n be odd. Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function and $b \in \mathbb{F}_{2^n}$. Then*

$$\sum_{a \in \mathbb{F}_{2^n}} W_f(a, b)^2 = 2^{2n}.$$

Proof.

$$\sum_{a \in \mathbb{F}_{2^n}} W_f(a, b)^2 = \sum_{a \in \mathbb{F}_{2^n}} \sum_{x,y \in \mathbb{F}_{2^n}} S(ax + bf(x))S(ay + bf(y)).$$

Using Proposition 4.3.7, this is equal to

$$\sum_{a \in \mathbb{F}_{2^n}} \sum_{x,y \in \mathbb{F}_{2^n}} S(a(x + y) + b(f(x) + f(y))) = \sum_{x,y \in \mathbb{F}_{2^n}} \left[S(b(f(x) + f(y))) \sum_{a \in \mathbb{F}_{2^n}} S(a(x + y)) \right]$$

The inner sum is equal to 0 unless $x = y$, as Tr, and therefore S, is balanced, by Lemma 4.3.10. Thus the sum is equal to

$$\sum_{x \in \mathbb{F}_{2^n}} S(0) \sum_{a \in \mathbb{F}_{2^n}} S(0) = 2^{2n}.$$

□

This shows, in some sense, that there is a bound on the value of the Walsh transform ranging over all inputs. Roughly speaking, if the absolute value of the Walsh transform is higher for some inputs, then it will be lower for other inputs, and vice-versa.

4.4 Gold Functions in Odd Dimension are AB

We now prove the existence of AB functions. We show that the Gold functions are AB in odd dimension n.

There are many papers that state that Gold functions are AB, but it is not easy to find a proof of this fact. Every paper that states this result seems to reference the same paper from 1968 by Robert Gold [Gol68]. This paper, with a bit of work, can be understood to be proving that Gold functions are AB. However, it uses different notation and terminology (does not make mention of AB functions at all, and instead of the Walsh transform, talks about the cross-correlation function). It also proves a slightly weaker result, and additionally has a minor mistake.

The following is a reworking of Gold's proof to fit the context we are working in.

Recall that a function $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ is **Gold** if $f(x) = x^{2^l+1}$ where $(n, \ell) = 1$.

From [N18], we have the following lemma.

Lemma 4.4.1. *Suppose* $(n, \ell) = 1$. *Then* $(2^n - 1, 2^\ell - 1) = 1$.

In the following, you will see the notation $x^{2^{-l}}$. Here, we justify the use of this notation.

Proposition 4.4.2. *Let* $n, k \in \mathbb{N}$ *such that* $(2^n - 1, k) = 1$. *Then for each* $x \in \mathbb{F}_{2^n}$, *there exists a unique* $y \in \mathbb{F}_{2^n}$ *such that* $y^k = x$, *i.e., every* $x \in \mathbb{F}_{2^n}$ *has a unique* k*th root.*

Proof. From basic number theory, we know that k has a unique multiplicative inverse in $\mathbb{Z}_{2^n-1}$, i.e., there is a unique $m \in \mathbb{Z}$ with $0 \leq m \leq 2^n - 2$ such that $km \equiv 1 \pmod{2^n - 1}$. Let $x \in \mathbb{F}_{2^n}$, and let $y = x^m$. Then $y^k = (x^m)^k = x^{mk} = x$.

Now, suppose $y^k = z^k$. We know that $y = \alpha^i$ and $z = \alpha^j$ for some $i, j \in \mathbb{Z}$, where α is a multiplicative generator of $\mathbb{F}_{2^n} - \{0\}$. We have $(\alpha^i)^k = (\alpha^j)^k$, which implies $\alpha^{ik} = \alpha^{jk}$. In order for this to be true, it must be that $ik \equiv jk \pmod{2^n - 1}$, by Proposition 4.1.2. This means that $2^n - 1 \mid (i - j)k$, but as $(2^n - 1, k) = 1$, it must be that $2^n - 1 \mid i - j$, and thus $i \equiv j \pmod{2^n - 1}$. Therefore, $\alpha^i = \alpha^j$, and so $y = z$. □

Definition 4.4.3. Let $n, k \in \mathbb{N}$ such that $(2^n - 1, k) = 1$. For $x \in \mathbb{F}_{2^n}$, define $x^{1/k}$ to be the unique kth root of x, from Proposition 4.4.2.

As $2^n - 1$ is odd, for any $\ell \in \mathbb{Z}$, we have $(2^n - 1, 2^\ell) = 1$, therefore $x^{1/2^\ell}$, or equivalently $x^{2^{-\ell}}$, is well-defined.

Lemma 4.4.4. *Let $c \in \mathbb{F}_{2^n}$ such that $\mathrm{Tr}(c) = 0$ and $(n, \ell) = 1$. Then there exists $x \in \mathbb{F}_{2^n}$ such that $x^{2^\ell} + x^{2^{-\ell}} = c$.*

Proof. First, define $\psi : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ by $\psi(x) = x^{2^\ell} + x^{2^{-\ell}}$. By Propositions 4.3.5 and 4.1.1, Tr and ψ are linear maps. We have that $\mathrm{im}(\mathrm{Tr}) = \{0, 1\}$ by Lemma 4.3.10 so $\dim(\mathrm{im}(\mathrm{Tr})) = 1$. By rank-nullity, $\dim(\ker(\mathrm{Tr})) = n - 1$.

Now, suppose $x^{2^\ell} + x^{2^{-\ell}} = 0$. Raising each side to the power of 2^ℓ gives $x^{2^{2\ell}} + x = 0$, which factors into $x(x^{2^{2\ell}-1} + 1) = 0$. Thus, $x = 0$ or $x^{2^{2\ell}-1} = 1$. In the second case, this implies that $\mathrm{ord}(x) \mid 2^{2\ell} - 1$. Additionally, we know that $\mathrm{ord}(x) \mid 2^n - 1$. But we have that $(n, 2\ell) = 1$, as n is odd. Therefore, by Lemma 4.4.1, $(2^n - 1, 2^{2l} - 1) = 1$, and so $\mathrm{ord}(x) = 1$, and thus $x = 1$. This shows that $\ker(\psi) = \{0, 1\}$, so $\dim(\ker(\psi)) = 1$. By rank-nullity, $\dim(\mathrm{im}(\psi)) = n - 1$.

Let $c \in \mathrm{im}(\psi)$. Then $c = x^{2^\ell} + x^{2^{-\ell}}$ for some $x \in \mathbb{F}_{2^n}$. Then we have

$$\mathrm{Tr}(c) = \mathrm{Tr}(x^{2^\ell} + x^{2^{-\ell}}) = \mathrm{Tr}(x^{2^\ell}) + \mathrm{Tr}(x^{2^{-\ell}}).$$

By Corollary 4.3.3, this is equal to

$$\mathrm{Tr}(x^{2^\ell}) + \mathrm{Tr}(x^{2^\ell}) = 0.$$

Thus, $\mathrm{im}(\psi) \subseteq \ker(\mathrm{Tr})$.

As $\dim(\mathrm{im}(\psi)) = \dim(\ker(\mathrm{Tr}))$, and $\mathrm{im}(\psi) \subseteq \ker(\mathrm{Tr})$, it must be that $\mathrm{im}(\psi) = \ker(\mathrm{Tr})$.

Therefore, if $\mathrm{Tr}(c) = 0$, then $c \in \mathrm{im}(\psi)$, and so $x^{2^\ell} + x^{2^{-\ell}} = c$ for some $x \in \mathbb{F}_{2^n}$. □

Theorem 4.4.5 (Gold functions are AB)**.** *Let n be odd, and let f be the Gold function $f(x) = x^{2^\ell+1}$. Then f is AB.*

Proof. We must show for all inputs to the Walsh transform of f, that the output is an element of $\{0, \pm 2^{\frac{n+1}{2}}\}$. We first consider the case where the input to the Walsh transform is of the form $(c, 1)$ with $c \in \mathbb{F}_{2^n}$. This is the only case considered in Gold's proof, and we will show how this extends to the general case of an input (a, b) at the end of this proof.

First, we have

$$W(c,1) = \sum_{x\in\mathbb{F}_{2^n}} S(cx + x^{2^l+1}).$$

For any $\beta \in \mathbb{F}_{2^n}$, the function $x \mapsto x + \beta$ is a bijection, so we may replace the occurrences of x in the summand with $x + \beta$, for any $\beta \in \mathbb{F}_{2^n}$:

$$\sum_{x\in\mathbb{F}_{2^n}} S(c(x+\beta) + (x+\beta)^{2^\ell+1}) = \sum_{x\in\mathbb{F}_{2^n}} cx + c\beta + x^{2^\ell+1} + x^{2^\ell}\beta + x\beta^{2^\ell} + \beta^{2^\ell+1}.$$

By Corollary 4.3.3, $\mathrm{Tr}(x^{2^\ell}\beta) = \mathrm{Tr}(x\beta^{2^{-\ell}})$. Thus the sum above is equal to

$$\sum_{x\in\mathbb{F}_{2^n}} cx + c\beta + x^{2^\ell+1} + x\beta^{2^{-\ell}} + x\beta^{2^\ell} + \beta^{2^\ell+1} = \sum_{x\in\mathbb{F}_{2^n}} S(x(c + \beta^{2^{-\ell}} + \beta^{2^\ell}) + x^{2^\ell+1} + c\beta + \beta^{2^\ell+1}).$$

Now, if $\mathrm{Tr}(c) = 0$, then by Lemma 4.4.4, we may choose β such that $\beta^{2^{-l}} + \beta^{2^l} = c$. If $\mathrm{Tr}(c) = 1$, then $\mathrm{Tr}(c+1) = \mathrm{Tr}(c) + \mathrm{Tr}(1) = 1 + 1 = 0$. In this case, again by Lemma 4.4.4, we may choose β such that $\beta^{2^{-\ell}} + \beta^{2^\ell} = c + 1$. Notice that in each case, we have chosen β such that $\beta^{2^{-\ell}} + \beta^{2^\ell} = c + \mathrm{Tr}(c)$. The previous sum becomes

$$\sum_{x\in\mathbb{F}_{2^n}} S(\mathrm{Tr}(c)x + x^{2^\ell+1} + c\beta + \beta^{2^\ell+1})$$

$$= S(c\beta + \beta^{2^l+1}) \sum_{x\in\mathbb{F}_{2^n}} S(\mathrm{Tr}(c)x + x^{2^\ell+1}) = \pm W(\mathrm{Tr}(c), 1).$$

Thus, we have that $W(c,1) \in \{\pm W(0,1), \pm W(1,1)\}$. By Lemma 4.3.10, we have that $W(0,1) = 0$, and by Proposition 4.3.11, $W(c,1) = \pm 2^{\frac{n+1}{2}}$ for all values of c.

Finally, we must show that this holds not just for $W(c,1)$, but for $W(a,b)$ where $a, b \in \mathbb{F}_{2^n}$. Because $(n,\ell) = 1$ and n is odd, we have $(2^n - 1, 2^\ell + 1) = 1$, and thus f is invertible. We now express $W(a,b)$ in the form $W(c,1)$:

$$W(a,b) = \sum_{x\in\mathbb{F}_{2^n}} S(ax + bf(x)) = \sum_{x\in\mathbb{F}_{2^n}} S(ax + f(f^{-1}(b)x))$$

$$= \sum_{x\in\mathbb{F}_{2^n}} S(af^{-1}(b)^{-1}x + f(x)) = W(af^{-1}(b)^{-1}, 1).$$

□

4.5 AB Function k-Cover Equivalence

The graph of an AB function is a k-cover, and moreover these are equivalent, i.e., that a function f is AB if and only if its graph is a k-cover. This, along with the fact that Gold functions are AB, proves the existence of the entire class of k-covers predicted in Section 4.2, with k predicted by Equation 4.2.3.

This theorem is directly implied by Theorem 1 of [vF03].

Theorem 4.5.1. *Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be a function. Then f is AB if and only if its graph is a k-cover, with*

$$k = \frac{2^n - 2}{6}.$$

The method used in the proof of this theorem is quite interesting. It involves constructing a matrix equation that is equivalent to the statement *f is AB*, and another matrix equation that is equivalent to the statement *The graph of f is a k-cover*. Then the first equation is shown to be satisfied if and only if the second is satisfied.

In the following section, we provide a more direct proof that the graph of a Gold function in odd dimension n is a k-cover.

4.6 Explicit Construction of k-Covers Using Gold Functions

The previous two sections together give a proof of the existence of a certain class of k-covers. We now provide a direct proof that the graph of a Gold function for odd n is a k-cover. In fact, combined with Theorem 4.5.1, this provides an alternative proof that Gold functions are AB.

We first prove this just for the function x^3. The proof is slightly simpler, and it is all that is necessary for showing the existence of these k-covers.

Lemma 4.6.1. *Let $n \in \mathbb{N}$ be odd, and let $a, b \in \mathbb{F}_{2^n}$ such that $a^3 \neq b$. Then there are exactly $2^n - 2$ ordered triples (x, y, z) of elements in $\mathbb{F}_{2^n}$ such that*

$$x + y + z = a$$

and

$$x^3 + y^3 + z^3 = b.$$

Proof. Let $\alpha \in \mathbb{F}_2^n - \{0, 1\}$, and let

$$r_\alpha = \left(\frac{a^3 + b}{\alpha + \alpha^2}\right)^{\frac{1}{3}}.$$

Note that because n is odd, $(2^n - 1, 3) = 1$, and so by Proposition 4.4.2, every element of $\mathbb{F}_{2^n}$ has a unique cube root. Let

$$x = r_\alpha + a,$$

$$y = r_\alpha \alpha + a,$$

and

$$z = r_\alpha(1 + \alpha) + a.$$

We see that

$$x + y + z = r_\alpha(1 + \alpha + (1 + \alpha)) + a = a.$$

Next, observe that

$$x^3 + y^3 + z^3$$

$$= r_\alpha^3(1 + \alpha^3 + (1 + \alpha)^3) + r_\alpha^2 a(1 + \alpha^2 + (1 + \alpha)^2) + r_\alpha a^2(1 + \alpha + (1 + \alpha)) + a^3$$

$$= r_\alpha^3(1 + \alpha^3 + (1 + \alpha + \alpha^2 + \alpha^3)) + a^3$$

$$= r_\alpha^3(\alpha + \alpha^2) + a^3$$

$$= (a^3 + b) + a^3$$

$$= b.$$

Suppose $\alpha \neq \beta \in \mathbb{F}_{2^n}$. If $r_\alpha = r_\beta$, then $r_\alpha \alpha + a \neq r_\beta \beta + a$. Otherwise, $r_\alpha + a \neq r_\beta + a$. Thus for each choice of $\alpha \in \mathbb{F}_2^n - \{0, 1\}$, the solution (x, y, z) described above is distinct, and so there are at least $2^n - 2$ solutions.

There are $2^{2n} - 2^n$ choices for (a, b) such that $a^3 \neq b$. We have shown that each choice has at least $2^n - 2$ solutions (x, y, z). In total this gives at least

$$(2^{2n} - 2^n)(2^n - 2) = 2^n(2^n - 1)(2^n - 2)$$

triples. But this is how many ordered triples there are in total. Thus, there must be exactly $2^n - 2$ solutions for each choice of (a, b). □

Theorem 4.6.2. *Let $n \in \mathbb{N}$ be odd. Then the graph of x^3 in $F_{2^{2n}}$ is a k-cover, with $k = \frac{2^n-2}{6}$.*

Proof. As x^3 is APN, we know that its graph is a cap. By Lemma 4.6.1, for each (a, b) not in the graph, there are $2^n - 2$ ordered triples of elements in the graph that sum to (a, b). As each unordered triple appears $3! = 6$ times, there are $\frac{2^n-2}{6}$ unordered triples of elements in the graph that sum to (a, b). Therefore, the graph is a $\left(\frac{2^n-2}{6}\right)$-cover. □

We now generalize this result to include all Gold functions.

Lemma 4.6.3. *Let $n, \ell \in \mathbb{N}$ such that n is odd and $(n, \ell) = 1$. Let $a, b \in \mathbb{F}_{2^n}$ such that $a^{2^\ell+1} \neq b$. Then there are exactly $2^n - 2$ ordered triples (x, y, z) of elements in $\mathbb{F}_{2^n}$ such that*

$$x + y + z = a$$

and

$$x^{2^\ell+1} + y^{2^\ell+1} + z^{2^\ell+1} = b.$$

Proof. Let $\alpha \in \mathbb{F}_2^n - \{0, 1\}$, and let

$$r_\alpha = \left(\frac{a^{2^\ell+1} + b}{\alpha + \alpha^{2^\ell}}\right)^{\frac{1}{2^\ell+1}}.$$

Note that because n is odd and $(n, \ell) = 1$, $(2^n - 1, 2^\ell + 1) = 1$, and so by Proposition 4.4.2, every element of $\mathbb{F}_{2^n}$ has a unique $(2^\ell + 1)$th root. Let

$$x = r_\alpha + a,$$

$$y = r_\alpha \alpha + a,$$

and

$$z = r_\alpha(1+\alpha) + a.$$

We see that

$$x + y + z = r_\alpha(1 + \alpha + (1+\alpha)) + a = a.$$

Next, observe that

$$x^{2^\ell+1} + y^{2^\ell+1} + z^{2^\ell+1}$$

$$= (r_\alpha + a)(r_\alpha^{2^\ell} + a^{2^\ell}) + (r_\alpha\alpha + a)(r_\alpha^{2^\ell}\alpha^{2^\ell} + \alpha^{2^\ell}) + (r_\alpha(1+\alpha) + a)(r_\alpha^{2^\ell}(1+\alpha)^{2^\ell} + a^{2^\ell})$$

$$= r_\alpha^{2^\ell+1}(1 + \alpha^{2^\ell+1} + (1+\alpha)^{2^\ell+1}) + r_\alpha^{2^\ell}a(1 + \alpha^{2^\ell} + (1+\alpha)^{2^\ell}) + r_\alpha a^{2^\ell}(1 + \alpha + (1+\alpha)) + a^{2^\ell+1}$$

$$= r_\alpha^{2^\ell+1}(1 + \alpha^{2^\ell+1} + (1 + \alpha + \alpha^{2^\ell} + \alpha^{2^\ell+1})) + a^{2^\ell+1}$$

$$= r_\alpha^{2^\ell+1}(\alpha + \alpha^{2^\ell}) + a^{2^\ell+1}$$

$$= (a^{2^\ell+1} + b) + a^{2^\ell+1}$$

$$= b.$$

By the exact same argument as in Lemma 4.6.1, there are $2^n - 2$ solutions for each choice of (a, b). □

Theorem 4.6.4. *Let $n, \ell \in \mathbb{N}$ such that n is odd and $(n, \ell) = 1$. Let $f : \mathbb{F}_{2^n} \to \mathbb{F}_{2^n}$ be the Gold function defined by $f(x) = x^{2^\ell+1}$. Then the graph of $x^{2^\ell+1}$ in $F_{2^{2n}}$ is a k-cover, with $k = \frac{2^n-2}{6}$.*

Proof. Use the same argument as the proof of Theorem 4.6.2. □

5
k-Cover Conjecture

We now have the tools to explore the main conjecture of this project, the complete characterization of values of k such that there exists a k-cover.

Conjecture 5.0.1 (The k-cover conjecture)**.** *Let $k \in \mathbb{N}$. Then there exists a k-cover if and only if $k = 2$ or $k = \frac{2^n-2}{6}$ for some odd $n \in \mathbb{N}$ with $n > 1$.*

This conjecture was originally formulated by Lauren Rose and Raphael Walker.

We have the tools to prove one direction of this conjecture. We are able to construct a k-cover for each value of k in the conjecture.

Theorem 5.0.2. *Suppose $k = 2$ or $k = \frac{2^n-2}{6}$ for some odd $n \in \mathbb{N}$ with $n > 1$. Then there exists a k-cover.*

Proof. The existence of a 2-cover has been established by Example 3.1.1, and the existence of a $\left(\frac{2^n-2}{6}\right)$-cover for each odd $n > 1$ has been established by Theorem 4.6.2. □

The second direction of the conjecture, that there only exists a k-cover if $k = 2$ or $k = \frac{2^n-2}{6}$, is quite a bit more difficult. We have established the result computationally for the first 10^7 values of k.

Here we give a Python program for checking these values. We refer back to Theorem 3.3.1 that proves that given two of (k, n, s), the third is completely determined.

The basic idea of the program is to iterate through each value of k, and then through every possible value of n and s given k, until there is a solution to the k-cover equation. In order to do this practically, we must figure out a way to limit the number of possible n and s values given k.

First, we know that given k, fixing either n *or* s determines the other. If the other is an integer, then (k, n, s) is a solution to the k-cover equation, and thus there *may* exist a k-cover for that particular k. Knowing this, we must now restrict the possible values of either n or s, given k.

Given k, there are upper and lower bounds on the dimension n for which it is possible for there to be a k-cover.

First, we must prove the following upper bound on the **maximal cap size** in each dimension n.

Lemma 5.0.3. *Let $C \subseteq \mathbb{F}_{2^n}$ be a cap of size s. Then $s \leq 2^{\frac{n+1}{2}} + 1$.*

Proof. We first show that any two pairs of elements in C must sum to distinct elements. Let (a, b) and (c, d) be pairs of elements in C. If the pairs do not have elements in common, then $a + b \neq c + d$, as otherwise $\{a, b, c, d\}$ would be a quad. If they do have an element in common, say $a = c$, then $a + b = c + d$ would imply that $b = d$, and these would be the same pair. Therefore, any two pairs of elements must have distinct sums. As there are 2^n elements in total that the pairs could sum to, there must be at most 2^n pairs. Thus,

$$\binom{s}{2} \leq 2^n.$$

We write this as

$$\frac{s(s-1)}{2} \leq 2^n,$$

which becomes

$$s(s-1) \leq 2^{n+1}. \tag{5.0.1}$$

As $(s-1)^2 \leq s(s-1)$, we have that

$$(s-1)^2 \leq 2^{n+1}.$$

Taking the square root gives

$$s - 1 \leq 2^{\frac{n+1}{2}},$$

which finally yields

$$s \leq 2^{\frac{n+1}{2}} + 1.$$

□

Definition 5.0.4. Let $n \in \mathbb{N}$. Express n in its unique binary representation:

$$n = \sum_{i=0}^{j} c_i 2^i.$$

Define the **least bit** of n to be the smallest i such that $c_i = 1$.

Example 5.0.5. The binary representation of 24 is 11000, and there is a 1 in the 2^3 place, with only 0's to the right. Therefore, the least bit of 24 is 3.

Proposition 5.0.6. *Let $k \in \mathbb{N}$ and suppose $C \subseteq \mathbb{F}_{2^n}$ is a k-cover. Let b be the least bit of $6k+2$. Then $2\log_2(3k-1) - 1 \leq n \leq \max(2b+2, \log_2(42k^3 2^{3b}))$.*

Proof. For the lower bound, we know there exists an element $x \in \mathbb{F}_{2^n} - C$. This element x is a k-point, meaning there are k unordered triples $\{a, b, c\}$ such that $a + b + c = x$. If two of these triples contain a point in common, Then $a+b+c = a+d+e$, and so $b+c = d+e$, which would imply that C is not a cap. This is a contradiction, so the triples must be disjoint. As there are k triples, C must then have at least $3k$ elements. Combining this with Lemma 5.0.3, we have

$$2^{\frac{n+1}{2}} + 1 \geq s \geq 3k.$$

This gives

$$2^{\frac{n+1}{2}} \geq 3k - 1,$$

which implies

$$\frac{n+1}{2} \geq \log_2(3k-1),$$

which finally gives

$$n \geq 2\log_2(3k-1) - 1.$$

The upper bound was proven by Raphael Walker in [FRW21]. □

This result allows us to perform a much more efficient search. Here is the Python code for the lower bound:

```
def lower_n(k):
    return math.ceil(2 * math.log2(3*k - 1) - 1)
```

Here is the code for the least bit:

```
def least_bit(a):
    i = 0
    while True:
        if (a >> i) % 2 == 1:
            return i
        i += 1
```

And here is the code for the upper bound:

```
def upper_n(k):
    b = least_bit(6*k + 2)
    f1 = 2*b + 2
    f2 = math.floor(math.log2(42 * k**3 * 2**(3*b)))
    return max(f1, f2)
```

Now, we use Equation 5.0.1 to place the max-cap restriction on n and s.

```
def cap_allowed(n, s):
    return s*(s - 1) <= 2**(n + 1)
```

Recall the form of the k-cover equation given in Equation 3.3.3:

$$s^3 - 3s^2 + (6k + 2)s - 6k \cdot 2^n = 0.$$

This is a cubic equation in s. We define `p(k, n, s)` to be the result of the cubic given k, n, s, and `cubic_sol(k, n)` to be the solution to the cubic given k and n, using Equation 3.3.4.

```
def p(k, n, s):
    return s**3 - 3*s**2 + (6*k + 2)*s - 6*k*2**n


def cubic_sol(k, n):
    d = (81 * k * 2**(n + 1)
         + ((81 * k * 2**(n + 1) - 162 * k)**2 + 4*(18*k - 3)**3)**(1/2)
         - 162*k)**(1/3)
    a = 2**(1/3) * (18*k - 3)
    return -a/(3*d) + d/(3*2**(1/3)) + 1
```

Finally, here is the code for iterating through each value of k and checking whether a k-cover may be possible by the k-cover equation.

```
for k in range(1, 100000):
    for n in range(max(lower_n(k), 2), upper_n(k) + 1):
        s = round(cubic_sol(k, n))
        if p(k, n, s) == 0 and cap_allowed(n, s):
            print((k, n, s))
```

Due to floating-point error, we first round the solution to the cubic to the nearest integer, and then check if this integer satisfies the cubic. If it does, and s does not exceed the max-cap size, then (k, n, s) is printed. In the code above, the first $100,000$ values of k are checked. The results are displayed in Figure 5.0.1.

Note that just because a triple (k, n, s) appears in this list doesn't mean that there exists a k-cover in dimension n of size s. However, if there does, then (k, n, s) must appear in this list.

Looking at the list, we see that it includes the four 1-covers and the 2-cover from Chapter 3. As expected, it also includes the $\left(\frac{2^r-2}{6}\right)$-cover in dimension $2r$ of size 2^r for each odd value of

k	n	s
1	2	3
1	3	4
1	6	8
1	11	24
2	4	6
5	7	16
5	10	32
21	11	64
21	14	128
85	15	256
85	18	512
341	19	1024
341	22	2048
1365	23	4096
1365	26	8192
5461	27	16384
5461	30	32768
21845	31	65536
21845	34	131072
87381	35	262144
87381	38	524288

Figure 5.0.1: Output of Python k-cover search code.

$r > 1$. For example, for $r = 5$, we have $(k, n, s) = (5, 10, 32)$, which appears on the list. Oddly enough, it *also* includes $(5, 7, 16)$. There certainly cannot exist a 5-cover in dimension 7 of size 16, as there does not even exist a *cap* of size 16 in dimension 7. For every $k = \left(\frac{2^r-2}{6}\right)$ on the list, there is an entry with $n = 2r$ and $s = 2^r$, which was expected. But there is *also* an entry with $n = 2r - 3$ and $s = 2^{r-1}$. We don't know if any of these entries actually correspond to a k-cover, except for the case of $r = 3$, for which both of the 1-covers *do* exist.

Nonetheless, these results verify the k-cover conjecture up to $k = 100,000$, and we have so far additionally used this code to check the first 10^7 values of k, and the pattern described still holds.

6
(i, j)-Covers

We now discuss (i, j)-covers in more detail.

Theorem 6.0.1. *Let $n \in \mathbb{N}$ be even, and let $a, b \in \mathbb{F}_{2^n}$ such that $a^3 \neq b$. There are integers i and j such that if $a^3 + b$ has a cube root, then there are i unordered triples, and otherwise there are j unordered triples (x, y, z) of elements in $\mathbb{F}_{2^n}$ such that*

$$x + y + z = a$$

and

$$x^3 + y^3 + z^3 = b.$$

Proof. We may use the exact same construction as in Lemma 4.6.1. For $\alpha \in \mathbb{F}_2^n - \{0, 1\}$, let

$$r_\alpha = \left(\frac{a^3 + b}{\alpha + \alpha^2} \right)^{\frac{1}{3}},$$

let

$$x = r_\alpha + a,$$

$$y = r_\alpha \alpha + a,$$

and

$$z = r_\alpha (1 + \alpha) + a.$$

We see that

$$x + y + z = r_\alpha(1 + \alpha + (1 + \alpha)) + a = a.$$

By the same argument as in Lemma 4.6.1, (x, y, z) is a solution.

The difference is that because n is even, $(2^n - 1, 3) = 3$. It is no longer the case that every element has a unique cube root. In fact, one third of the elements in $\mathbb{F}_{2^n} - \{0\}$ have cube roots, and they each have 3 distinct cube roots.

Suppose $a^3 + b$ has a cube root. Then r_α exists if and only if $\alpha + \alpha^2$ has a cube root as well. Let I be the number of cube roots in the image of the map $\alpha \mapsto \alpha + \alpha^2$. Thus, regardless of what a and b are, whenever $a^3 + b$ has a cube root, there are $3I$ solutions, as when r_α exists, it can take on 3 different values.

Suppose $a^3 + b$ doesn't have a cube root. Then it is either of the form s^{3i+1} or s^{3i+2}, where s is a multiplicative generator of $\mathbb{F}_2^n$. In either case, the remainder of the exponent of $\alpha + \alpha^2$ mod 3 must match this remainder. Observe that $\alpha + \alpha^2$ is of the form s^{3i+1} if and only if $\alpha^2 + \alpha^4 = \alpha^2 + (\alpha^2)^2 = s^{6i+2}$, which is of the form s^{3i+2}. Thus there is a bijection between elements in the image of $\alpha \mapsto \alpha + \alpha^2$ of the form s^{3i+1} and those of the form s^{3i+2}. Therefore, regardless of whether $a^3 + b$ is of the form s^{3i+1} or s^{3i+2}, there are the same number of solutions. □

This theorem proves that the graph of the function x^3 in even dimension n is always an (i, j)-cover, with i-points making up exactly two-thirds of the non-cap points, and j-points making up one-third, as observed in 4.2. This theorem does NOT, however, tell us what i and j actually are.

Recall Equations 4.2.4 and 4.2.5, where we noted formulas that i and j appear to satisfy:

$$i(n) = \frac{2^n - 2}{6} - \frac{(-2)^{n/2}}{6}$$

and

$$j(n) = \frac{2^n - 2}{6} + \frac{(-2)^{n/2}}{3}.$$

In order to prove these formulas, we must be able to count the number of cubes in the image of the function $\alpha \mapsto \alpha + \alpha^2$, where $\alpha \notin \{0, 1\}$. This would give us $6i$, from which we can derive i and j.

This can also be framed as finding the number of solutions to the elliptic curve

$$x^3 + y^2 + y = 0.$$

This can be done. Exercise 4.2 on page 139 of [Was08] shows that for n even, this elliptic curve has $2^n + 1 - 2(-2)^{n/2}$ solutions, which justifies our formulas for i and j.

7
Future Work

In the future, we hope to prove the second direction of the k-cover conjecture. We would additionally like to understand exactly for which (k, n, s) there exists a k-cover, and even stronger, the complete classification of k-covers up to affine equivalence. Additionally, we hope to prove the properties observed in Chapter 3 about the 1-cover in dimension 11.

We also would also like to understand more about (i, j)-covers. We have shown that the graph of x^3 in even dimension is an (i, j)-cover with i and j given by Equations 4.2.4 and 4.2.5. We would like to show this for all Gold functions. Additionally, we would like to find the complete class of functions whose graph is an (i, j)-cover. We conjecture that these are precisely the functions in $\mathbb{F}_{2^n}$ whose Walsh spectrum is $\{0, \pm 2^{\frac{n}{2}}, \pm 2^{\frac{n}{2}+1}\}$.

Appendix A
Python Code

A.1 Finite Fields

The following is a finite field library using `pyfinite.ffield` that can compute many things such as the trace of a field element, the Walsh transform and Walsh spectrum of a function, as well as the nonlinearity of a function (related to the Walsh spectrum and discussed in detail in [Dav21]). At the end is a construction of the graph of x^3 in dimension n. The printed string may be directly input into the Qap Finder, in the browser's developer console, to view the k-cover in dimension $2n$.

```
from pyfinite import ffield
import math
import numpy as np

class Field:
    def __init__(self, n):
        self.n = n
        self.field = ffield.FField(n)
        self.gen = self._gen()
        self.list = [self.FieldElem(0, self)]\
```

```
                + [self.FieldElem(self.gen, self) ** i for i in range(2 ** self.n - 1)]

        self.l = 1

        def gold(x):
            assert math.gcd(self.l, n) == 1
            return x ** (2 ** self.l + 1)
        self.gold = gold
        self.A = {lambda x: (self.trace(a * x) + b) % 2 for a in self for b in [0, 1]}

    def _order(self, x):
        r = x
        for i in range(1, 2 ** self.n):
            if r == 1:
                return i
            r = self.field.Multiply(r, x)

    def _gen(self):
        for i in range(1, 2 ** self.n):
            if self._order(i) == 2 ** self.n - 1:
                return i

    def __getitem__(self, index):
        return self.list[index]

    def __iter__(self):
        return iter(self.list)
```

```
    def fsum(self, xs):
        return sum(xs, start=self[0])

    def trace(self, x):
        return self.fsum(x ** 2 ** i for i in range(self.n)).i

    def s(self, x):
        return (-1) ** self.trace(x)

    def W(self, f, a, b):
        return sum(self.s(a * x + b * f(x)) for x in self)

    def matrix(self, m):
        return np.array([[m(a, b) for b in self] for a in self])

    def spectrum(self, f):
        s = [self.W(f, a, b) for a in self for b in self if b != self[0]]
        return {i: s.count(i) for i in set(s)}

    def dH(self, f, g):
        return len({x for x in self if f(x) != g(x)})

    def _NL(self, f):
        return min(self.dH(f, a) for a in self.A)

    def NL(self, f):
```

```
        return min(self._NL(lambda x: self.trace(v * f(x))) for v in self if v != 0)


    class FieldElem:
        def __init__(self, i, F):
            self.i = i
            self.F = F
            self.inverse = F.field.Inverse(i) if i != 0 else 'NaN'
            r = 1
            if i == 0:
                self.exp = 'NaN'
                return
            for j in range(2 ** F.n - 1):
                if r == i:
                    self.exp = j
                    break
                r = F.field.Multiply(r, F.gen)


        def __eq__(self, other):
            return self.i == other.i


        def __add__(self, other):
            return Field.FieldElem(self.F.field.Add(self.i, other.i), self.F)


        def __mul__(self, other):
            return Field.FieldElem(self.F.field.Multiply(self.i, other.i), self.F)


        def __truediv__(self, other):
```

```
            return self * Field.FieldElem(other.inverse, self.F)

        def __pow__(self, power):
            if power < 0:
                return F[1] / self ** (-power)
            if 0 < power < 1:
                p = round(1 / power)
                for x in self.F:
                    if x ** p == self:
                        return x
            r = Field.FieldElem(1, self.F)
            for i in range(power):
                r *= self
            return r

        def __repr__(self):
            if self.i == 0:
                return '0'
            if self.exp == 0:
                return '1'
            if self.exp == 1:
                return '{alpha}'
            return f'{{alpha}}^{self.exp}'

        def __hash__(self):
            return self.i
```

```
n = 5

F = Field(n)

cap = [2**n * x.i + (x ** 3).i for x in F]

print(f'loadQap({cap})')
```

A.2 k-Cover Equation Search

The following is the complete code that prints out the values of (k, n, s) that satisfy the k-cover equation, as outlined in Chapter 5.

```
import math


def p(k, n, s):
    return s**3 - 3*s**2 + (6*k + 2)*s - 6*k*2**n


def cubic_sol(k, n):
    d = (81 * k * 2**(n + 1)
         + ((81 * k * 2**(n + 1) - 162 * k)**2 + 4*(18*k - 3)**3)**(1/2)
         - 162*k)**(1/3)
    a = 2**(1/3) * (18*k - 3)
    return -a/(3*d) + d/(3*2**(1/3)) + 1


def cap_allowed(n, s):
    return s*(s - 1) <= 2**(n + 1)


def lower_n(k):
    return math.ceil(2 * math.log2(3*k - 1) - 1)


def least_bit(a):
```

```
    i = 0
    while True:
        if (a >> i) % 2 == 1:
            return i
        i += 1


def upper_n(k):
    b = least_bit(6*k + 2)
    f1 = 2*b + 2
    f2 = math.floor(math.log2(42 * k**3 * 2**(3*b)))
    return max(f1, f2)


for k in range(1, 10**7):
    for n in range(max(lower_n(k), 2), upper_n(k) + 1):
        s = round(cubic_sol(k, n))
        if p(k, n, s) == 0 and cap_allowed(n, s):
            print((k, n, s))
```

A.3 1-Cover in Dimension 11 Search

The following is the code used in Section 3.6 to construct a 1-cover in dimension 11. As in Section A.1, the printed string may be directly input into the Qap Finder, in the browser's developer console, to view the 11-cover.

```
from itertools import combinations
import random


class Cap:
```

```
    def __init__(self, n):
        self.n = n
        self.points = range(2 ** n)
        self.cap = []
        self.safe = list(self.points)
        self.non_cap = {i: 0 for i in self.points}
        self.unsafe = set()

    def update(self, x):
        assert x in self.safe
        for a, b in combinations(self.cap, 2):
            self.non_cap[a ^ b ^ x] += 1
            if a ^ b ^ x in self.safe:
                self.safe.remove(a ^ b ^ x)
        self.cap.append(x)
        self.non_cap.pop(x)
        self.safe.remove(x)

    def remove(self, x):
        assert x in self.cap
        self.cap.remove(x)
        self.non_cap[x] = 0
        self.safe.append(x)
        for a, b in combinations(self.cap, 2):
            self.non_cap[a ^ b ^ x] -= 1
            if self.non_cap[a ^ b ^ x] == 0 and a ^ b ^ x not in self.unsafe:
                self.safe.append(a ^ b ^ x)
```

```
    def max_exclude(self):
        return max(self.non_cap[i] for i in self.non_cap)

    def complete(self):
        while self.safe:
            self.update(random.choice(self.safe))

    def k_cover(self, k, start=()):
        # make it do arbitrary backtracking
        while True:
            self.clear(start)
            while self.safe:
                i = random.choice(self.safe)
                # try to add the next point
                self.update(i)
                if self.max_exclude() > k:
                    self.remove(i)
                    self.safe.remove(i)
                    self.unsafe.add(i)
            if {self.non_cap[i] for i in self.non_cap} == {k}:
                return

    def clear(self, points=()):
        self.__init__(self.n)
        for i in points:
            self.update(i)
```

```
    def __str__(self):
        return f'loadQap({self.cap})'


n = 11
k = 1
start = [0] + [2**i for i in range(n)]
c = Cap(n)
c.k_cover(k, start=start)
print(c)
```

Bibliography

[Car22] *On APN Functions Whose Graphs are Maximal Sidon Sets*. Vol. 13568. Lecture Notes in Computer Science. Springer International Publishing, Oct. 2022, pp. 243–254. DOI: 10.1007/978-3-031-20624-5_15. URL: https://hal.science/hal-03965443.

[Cra+23] Julia Crager et al. "How many cards should you lay out in a game of EvenQuads: a detailed study of caps in AG$(n, 2)$". In: *Matematica* 2.2 (2023), pp. 382–419. ISSN: 2730-9657. DOI: 10.1007/s44007-023-00047-0. URL: https://doi.org/10.1007/s44007-023-00047-0.

[Dav21] Diana Davidova. "On properties of bent and almost perfect nonlinear functions". PhD thesis. University of Bergen, Norway, 2021.

[Fra03] John B. Fraleigh. *A First Course In Abstract Algebra*. 7th ed. Pearson, 2003. ISBN: 9780201763904.

[FRW21] Felicia Flores, Maximus Redman, and Raphael Walker. "Quads Proofs". 2021.

[Gol68] R. Gold. "Maximal recursive sequences with 3-valued recursive cross-correlation functions (Corresp.)" In: *IEEE Transactions on Information Theory* 14.1 (1968), pp. 154–156. DOI: 10.1109/TIT.1968.1054106.

[N18] Annamalai N. $gcd(a^n - 1, a^m - 1) = a^{gcd(n,m)} - 1$. Youtube. 2018. URL: https://www.youtube.com/watch?v=TnSsOpltz_M.

[RRW22] Maximus Redman, Lauren Rose, and Raphael Walker. "A Small Maximal Sidon Set in $\mathbb{Z}_2^n$". In: *SIAM Journal on Discrete Mathematics* 36.3 (2022), pp. 1861–1867. DOI: 10.1137/21M1454663. eprint: https://doi.org/10.1137/21M1454663. URL: https://doi.org/10.1137/21M1454663.

[TW21] Michael Tait and Robert Won. "Improved bounds on sizes of generalized caps in $AG(n, q)$". In: *SIAM J. Discrete Math.* 35.1 (2021), pp. 521–531. ISSN: 0895-4801,1095-7146. DOI: 10.1137/20M1369439. URL: https://doi.org/10.1137/20M1369439.

[vF03] E.R. van Dam and D. Fon-Der-Flaass. "Codes, graphs, and schemes from nonlinear functions". In: *European Journal of Combinatorics* 24.1 (2003), pp. 85–98. ISSN: 0195-6698. DOI: https://doi.org/10.1016/S0195-6698(02)00116-6. URL: https://www.sciencedirect.com/science/article/pii/S0195669802001166.

[Wal] Raphael Walker. *Qap Visualizer*. https://slickytail.github.io/QuadsVis/index.html. Accessed: 2022-07-18.

[Was08] Lawrence C. Washington. *Elliptic Curves: Number Theory and Cryptography, Second Edition*. 2nd ed. Chapman & Hall/CRC, 2008. ISBN: 9781420071467.