Assignment 6

In this assignment, we will study binary codes, which will be useful for the design of countermeasures against fault attacks.

Let n be a positive integer for the rest of the discussion. To study binary codes, we look at the vector space $\mathbb{F}_2^n$.

Definition 1.
- $\boldsymbol{w} = w_0 w_1 \ldots w_{n-1} \in \mathbb{F}_2^n$ is called a *binary word of length* n.
- A nonempty set $C \subset \mathbb{F}_2^n$ is called a *binary code* of *length* n.
- An element of a binary code C is called a *codeword* of C.
- Cardinality of C is called the *size* of C.
- A code of length n and size M is called a *binary* $(n, M)-$*code.*

Example 1.
- $C = \{ 00, 11 \}$ is a binary $(2,2)-$code.
- $C = \{ 010, 001, 110, 111 \}$ is a binary $(3,4)-$code.

Question 1. (1 mark) Let $C = \{ 0001, 0100, 1100 \}$. Then what is the length of C and what is the size of C?

Definition 2. For any $\boldsymbol{v},\ \boldsymbol{u} \in \mathbb{F}_2^n$, the *Hamming distance* between $\boldsymbol{v}$ and $\boldsymbol{u}$, denoted $\operatorname{dis}(\boldsymbol{v}, \boldsymbol{u})$, is defined as follows

$$\operatorname{dis}(\boldsymbol{v}, \boldsymbol{u}) = \sum_{i=0}^{n-1} \operatorname{dis}(v_i, u_i)\,, \text{ where } \operatorname{dis}(v_i, u_i) = \begin{cases} 1 & \text{if } v_i \neq u_i \\ 0 & \text{if } v_i = u_i \end{cases}. \tag{1}$$

Example 2. $\operatorname{dis}(001, 111) = 2$. $\operatorname{dis}(0000, 10101) = 3$

Definition 3. Let $C \subset \mathbb{F}_2^n$ be a binary code containing at least two codewords, the *(minimum) distance*, denoted $\operatorname{dis}(C)$, is given by

$$\operatorname{dis}(C) = \min\{ \operatorname{dis}(\boldsymbol{c}_1, \boldsymbol{c}_2) \mid \boldsymbol{c}_1, \boldsymbol{c}_2 \in C, \boldsymbol{c}_1 \neq \boldsymbol{c}_2 \}.$$

Definition 4. A binary code of length n, size M and distance d is called a *binary* $(n, M, d)-$*code.*

Example 3. Let $C = \{ 0011, 1101, 1000 \}$, we can calculate that

$$\operatorname{dis}(0011, 1101) = 3, \quad \operatorname{dis}(0011, 1000) = 3, \quad \operatorname{dis}(1101, 1000) = 2.$$

Thus C is a binary $(4, 3, 2)-$code

Question 2. (1 mark) Let $C = \{ 00000, 11100, 10101, 01010 \}$, what is the distance of C?

When the value of a bit is changed we say that the bit is flipped.

Definition 5. A binary code C is said to be $k-$*error-detecting* for a positive integer k if for any $\boldsymbol{c} \in C$, whenever at least 1 but at most k bits of $\boldsymbol{c}$ are flipped, the resulting word is not a codeword in C. If C is $k-$ error detecting but not $(k+1)-$error detecting, then we say C is *exactly* $k-$*error detecting.*

Example 4. Let $C = \{ 0011, 1101, 1000 \}$. Since

$$\text{dis}\,(0011, 1101) = \text{dis}\,(0011, 1000) = 3, \quad \text{dis}\,(1101, 1000) = 2,$$

with 1−bit flip from any codeword, we cannot get another codeword. But with 2−bit flips, we can change 1101 to 1000. Thus C is exactly 1−error detecting.

Theorem 1. A binary (n, M, d)−code C is k−error detecting if and only if $d \geq k + 1$, i.e. C is an exactly $(d - 1)$−error detecting code

Proof. $\Longleftarrow$ If $d \geq k+1$, take $\boldsymbol{c} \in C$ and $\boldsymbol{x} \in \mathbb{F}_2^n$ such that $1 \leq \text{dis}\,(\boldsymbol{c}, \boldsymbol{x}) \leq k$. Then $\boldsymbol{x} \notin C$, and C is k−error detecting

$\Longrightarrow$ If $d < k+1$, take $\boldsymbol{c}_1, \boldsymbol{c}_2 \in C$ such that $\text{dis}\,(\boldsymbol{c}_1, \boldsymbol{c}_2) = d$. Flipping d bits of $\boldsymbol{c}_1$ we can get $\boldsymbol{c}_2 \in C$. Hence C is not k−error detecting □

Question 3. (0.5 marks) Let $C = \{ 00000, 11100, 10101, 01010 \}$, C is exactly ?−error detecting.

Let us consider binary (n, M, d)−codes with $M = 2^k$ for some positive integer k. When a binary code is used for transmitting information, every information word $\boldsymbol{u} \in \mathbb{F}_2^k$ is assigned a unique codeword $\boldsymbol{c}(\boldsymbol{u}) \in C$. We say that $\boldsymbol{u}$ is *encoded* as $\boldsymbol{c}(\boldsymbol{u})$. Suppose Alice would like to send information $\boldsymbol{u}$ to Bob using C. Alice sends codeword $\boldsymbol{c}(\boldsymbol{u})$ to Bob. Due to transmission noise, Bob might receive a word $\boldsymbol{x} \in \mathbb{F}_2^n$ not equal to $\boldsymbol{c}(\boldsymbol{u})$. Thus we need to define a *decoding rule* for Bob that allows him to find $\boldsymbol{u}$ given $\boldsymbol{x}$.

We are interested in a *minimum distance decoding rule*, which specifies that after receiving $\boldsymbol{x}$, Bob computes[1]

$$\boldsymbol{c}_{\boldsymbol{x}} = \arg\min_{\boldsymbol{c}} \{ \text{dis}\,(\boldsymbol{x}, \boldsymbol{c}) \mid \boldsymbol{c} \in C - \{ \boldsymbol{x} \} \}, \quad \text{i.e.} \quad \text{dis}\,(\boldsymbol{c}_{\boldsymbol{x}}, \boldsymbol{x}) = \min_{\boldsymbol{c}} \{ \text{dis}\,(\boldsymbol{x}, \boldsymbol{c}) \mid \boldsymbol{c} \in C - \{ \boldsymbol{x} \} \}.$$

If more than one codeword is identified as $\boldsymbol{c}_{\boldsymbol{x}}$, there are two options. An *incomplete decoding rule* says that Bob should request Alice for another transmission. Following a *complete decoding rule*, Bob would then randomly select one codeword.

Example 5. Let $C = \{ 0000, 0111, 1110, 1111 \}$. We use C to encode information words $\boldsymbol{u} \in \mathbb{F}_2^2$ with encoding designed as follows:

$$c(00) = 0000, \quad c(01) = 0111, \quad c(10) = 1110, \quad c(11) = 1111.$$

Suppose Alice was sending information 00 with codeword 0000 to Bob. Due to an error during the transmission, Bob received 0001. By the minimum distance decoding rule, Bob computes the distances between 0001 and codewords in C.

$$\text{dis}\,(0001, 0000) = 1, \quad \text{dis}\,(0001, 0111) = 2, \quad \text{dis}\,(0001, 1110) = 4, \quad \text{dis}\,(0001, 1111) = 3$$

Thus $\boldsymbol{c}_{0001} = 0000$ and Bob gets the correct information 00.

Question 4. (1 mark) Continuing from Example 5, if Bob receives 0011, what is the decoding result following the minimum distance decoding rule?

Definition 6. A binary code C is said to be k−*error correcting* if with the incomplete decoding rule, minimum distance decoding outputs the correct codeword when k or fewer bits are flipped. If C is k−error correcting but not $k+1$−error correcting, then we say that C is *exactly k-error correcting*

[1]Recall difference between sets defined in week 1.

Example 6. Let $C = \{ 000, 111 \}$.

- If 000 was sent and 1 bit flip occurred, the received word $\{ 001, 010, 100 \}$ will be decoded to 000.
- If 111 was sent and 1 bit flip occurred, the received word $\{ 110, 011, 101 \}$ will be decoded to 111.
- If 000 was sent and 011 was received, the decoding result will be 111.

Thus C is exactly $1-$error correcting.

Theorem 2. A binary $(n, M, d)-$code C is $k-$error correcting if and only if $d \geq 2k + 1$, i.e. C is an exactly $\lfloor (d-1)/2 \rfloor -$error correcting code

Example 7 (Repetition code)**.** Let

$$C = \{ 00\ldots00, 11\ldots11 \} \subseteq \mathbb{F}_2^n.$$

C is a binary $(n, 2, n)-$code. C is called the *binary $n-$repetition code.* By Theorems 1 and 2, C is exactly $(n-1)-$error detecting and exactly $\lfloor (n-1)/2 \rfloor -$error correcting.

Question 5. Let C be the binary $5-$repetition code.

a) (0.5 marks) What are the codewords in C?

b) (0.5 marks) C is exactly ?$-$error detecting.

c) (0.5 marks) C is exactly ?$-$error correcting.

Example 8 (Parity-check code)**.** Suppose we would like to encode information words

$$\boldsymbol{u} = (u_0, u_1, \ldots, u_{n-2}) \in \mathbb{F}_2^{n-1}.$$

We add one parity-check bit and encode $\boldsymbol{u}$ using

$$\boldsymbol{c} = (u_0, u_1, \ldots, u_{n-2}, c_{n-1}), \text{ where } c_{n-1} = \sum_{i=0}^{n-2} u_i.$$

The corresponding code C consists of codewords that have an even number of 1s.

$$C = \left\{ (c_0, c_1, \ldots, c_{n-2}, c_{n-1}) \;\middle|\; c_{n-1} = \sum_{i=0}^{n-2} c_i \right\} \subseteq \mathbb{F}_2^n. \tag{2}$$

C is called the *binary parity-check code with length n.* W note that the minimum distance between the first $n-1$ bits of codewords in C is 1. The parity-check bit for two codewords will be different if they differ only at one position in the first $n-1$ bits. Thus, the minimum distance of C is 2. By Theorems 1 and 2, C is exactly $1-$error detecting and cannot correct errors.

Definition 7. Let C be a binary $(n, M, d)-$code. We define the *maximum distance* of C to be

$$\text{maxdis}(C) := \max \{ \text{dis}(\boldsymbol{c}_1, \boldsymbol{c}_2) \;\mid\; \boldsymbol{c}_1, \boldsymbol{c}_2 \in C \}.$$

If $\text{maxdis}(C) = \delta$, C is called a *binary $(n, M, d, \delta)-$anticode.*

The notion of anticode was first defined in [Far70], where an anticode refers to a 2−dimensional array of bits such that the maximum Hamming distance between any pair of rows is at most δ, for some integer $\delta > 0$. In this original definition, repeated rows are allowed. In Definition 7, an (n, M, d, δ)−anticode does not have repeated codewords.

We note that $\delta \geq d$. And any binary code is a binary anticode. However, the notion of binary anticode captures the maximum distance of a code.

Example 9.

- $C = \{ 01, 10 \}$ is a binary $(2, 2, 2, 2)$−anticode.
- $C = \{ 001, 011, 111 \}$ is a binary $(3, 3, 1, 2)$−anticode.
- An n−repetition code is a binary $(n, 2, n, n)$−anticode.
- A binary parity-check code with length n is a binary $(n, 2^{n-1}, 2, n)$−anticode if n is even. And it is a binary $(n, 2^{n-1}, 2, n-1)$−anticode if n is odd.

Next, let us look at the computation of `XOR` between two bits:

$$\begin{aligned} \texttt{XOR} : \mathbb{F}_2 \times \mathbb{F}_2 &\to \mathbb{F}_2 \\ (a, b) &\mapsto a \oplus b. \end{aligned}$$

We would like to encode the inputs and output of this operation. The operation will be implemented with a lookup table. We further require that the all zero vector $\mathbf{0}$ is not a codeword and output $\mathbf{0}$ indicates an error has happened.

Suppose we choose to use $\{ 01, 10 \}$, a binary $(2, 2, 2, 2)$−anticode. Let 01 be the codeword for 0 and 10 be the codeword for 1. For the decoding, we will map the codewords to the corresponding value (0 or 1), but when the input is not a codeword, we output 00 to indicate error. Then the lookup table will be as in Table 1.

	00	01	10	11
00	00	00	00	00
01	00	01	10	00
10	00	10	01	00
11	00	00	00	00

Table 1: Lookup table for `XOR` computation with 01 as the codeword for 0 and 10 as the codeword for 1.

For example, when the input is 0 and 1, the corresponding codewords are 01 and 10. The output should be $0 \oplus 1 = 1$, which corresponds to the codeword 10. Thus the entry for row 01 and column 10 is 10.

We note that with this design, any 1−bit flip will be detected: if the fault is injected in input 01, with 1−bit flip, we get either 00 or 11, both will give output 00. Similarly, if 1−bit flip is injected in input 10, we will have 00 or 11 and output will again be 00.

On the other hand, a 2−bit flip will be undetected. For example, suppose we would like to compute $0 \oplus 0$. Then the inputs for the table lookup will be 01 and 01, the output will be 01, which corresponds to 0. If a 2−bit flip is injected in the first input, we get 10 and 01 for table lookup. The result will be 10. Such a fault will not be detected and can successfully change the output of the operation.

In general, if the minimum distance of the binary code used is d, then by Theorem 1, the code is an exactly $(d-1)$−error detecting code. Consequently, an `XOR` lookup design described above can detect up to $(d-1)$−bit-flips.

Question 6. Suppose we choose to use $\{ 010, 101 \}$ for encoding the `XOR` operation inputs and outputs. Take 010 to be the codeword for 0 and 101 to be the codeword for 1. Following a similar design as above.

a) (2 marks) Provide the lookup table for implementing `XOR`, formatted analogously to Table 1.

b) (0.5 marks) Suppose an $m-$bit flip fault is injected in one of the inputs. What is the maximum value of m such that the fault injection can be detected?

Now, we consider a different way of designing the lookup table. Instead of having one specific word for indicating error, we follow the minimum distance complete decoding rule. If the binary code used has minimum distance d, then by Theorem 2, it is an exactly $\lfloor (d-1)/2 \rfloor-$error correcting code. Thus, such a lookup table design allows us to correct up to $\lfloor (d-1)/2 \rfloor-$bit flips.

Question 7. (2 marks) For example, let us consider the $3-$repetition code $\{ 000, 111 \}$, which has minimum distance 3. Using this code we will be able to correct errors caused by $1-$bit flip attacks. Let 000 be the codeword for 0 and 111 be the codeword for 1. Construct and present the complete lookup table for this scenario.

From the table we can see that any $1-$bit flips will be corrected. However, if there are more bit flips, the faulty output might be corrected to a wrong codeword.

What to submit.

- The submission should include detailed solution written in latex
- PDF to be submitted in AIS
- Add full name in both the file name and inside the file

When to submit: by Week 10 Thursday 8 am

References

[Far70] PG Farrell. Linear binary anticodes. *Electronics Letters*, 13(6):419–421, 1970.