## Contents

1 Introduction and Basic Ideas 3
   1.1 Introduction .......................................................... 3
   1.2 Terminology ............................................................ 4
   1.3 Basic Assumptions About the Channel ............................... 5

2 Detecting and Correcting Errors 7

3 Linear Codes 13
   3.1 Introduction .......................................................... 13
   3.2 The Generator and Parity Check Matrices ............................ 16
   3.3 Cosets ................................................................. 20
      3.3.1 Maximum Likelihood Decoding (MLD) of Linear Codes ........ 21

4 Bounds for Codes 25

5 Hamming Codes 31
   5.1 Introduction .......................................................... 31
   5.2 Extended Codes ...................................................... 33

6 Golay Codes 37
   6.1 The Extended Golay Code : $C_{24}$ ................................ 37
   6.2 The Golay Code : $C_{23}$ ........................................... 40

7 Reed-Muller Codes 43
   7.1 The Reed-Muller Codes $RM(1,m)$ ................................ 46

8 Decimal Codes 49
   8.1 The ISBN Code ....................................................... 49
   8.2 A Single Error Correcting Decimal Code ............................ 51
   8.3 A Double Error Correcting Decimal Code ............................ 53
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4 The Universal Product Code (UPC)</td>
<td>56</td>
</tr>
<tr>
<td>8.5 US Money Order</td>
<td>57</td>
</tr>
<tr>
<td>8.6 US Postal Code</td>
<td>58</td>
</tr>
<tr>
<td>9 Hadamard Codes</td>
<td>61</td>
</tr>
<tr>
<td>9.1 Background</td>
<td>61</td>
</tr>
<tr>
<td>9.2 Definition of the Codes</td>
<td>66</td>
</tr>
<tr>
<td>9.3 How good are the Hadamard Codes?</td>
<td>67</td>
</tr>
<tr>
<td>10 Introduction to Cryptography</td>
<td>71</td>
</tr>
<tr>
<td>10.1 Basic Definitions</td>
<td>71</td>
</tr>
<tr>
<td>10.2 Affine Cipher</td>
<td>73</td>
</tr>
<tr>
<td>10.2.1 Cryptanalysis of the Affine Cipher</td>
<td>74</td>
</tr>
<tr>
<td>10.3 Some Other Ciphers</td>
<td>76</td>
</tr>
<tr>
<td>10.3.1 The Vigenère Cipher</td>
<td>76</td>
</tr>
<tr>
<td>10.3.2 Cryptanalysis of the Vigenère Cipher : The Kasiski Examination</td>
<td>77</td>
</tr>
<tr>
<td>10.3.3 The Vernam Cipher</td>
<td>79</td>
</tr>
<tr>
<td>11 Public Key Cryptography</td>
<td>81</td>
</tr>
<tr>
<td>11.1 The Rivest-Shamir-Adleman (RSA) Cryptosystem</td>
<td>82</td>
</tr>
<tr>
<td>11.1.1 Security of the RSA System</td>
<td>85</td>
</tr>
<tr>
<td>11.2 Probabilistic Primality Testing</td>
<td>86</td>
</tr>
<tr>
<td>12 The Rabin Cryptosystem</td>
<td>89</td>
</tr>
<tr>
<td>12.1 Security of the Rabin Cryptosystem</td>
<td>94</td>
</tr>
<tr>
<td>13 Factorisation Algorithms</td>
<td>97</td>
</tr>
<tr>
<td>13.1 Pollard’s $p - 1$ Factoring Method (ca. 1974)</td>
<td>97</td>
</tr>
<tr>
<td>14 The ElGamal Cryptosystem</td>
<td>99</td>
</tr>
<tr>
<td>14.1 Discrete Logarithms (a.k.a indices)</td>
<td>99</td>
</tr>
<tr>
<td>14.2 The system</td>
<td>99</td>
</tr>
<tr>
<td>14.3 Attacking the ElGamal System</td>
<td>101</td>
</tr>
<tr>
<td>15 Elliptic Curve Cryptography</td>
<td>105</td>
</tr>
<tr>
<td>15.1 The ElGamal System in Arbitrary Groups</td>
<td>105</td>
</tr>
<tr>
<td>15.2 Elliptic Curves</td>
<td>106</td>
</tr>
<tr>
<td>15.3 The Menezes-Vanstone Cryptosystem</td>
<td>110</td>
</tr>
</tbody>
</table>
CONTENTS

16 The Merkle-Hellman “knapsack” System 113
17 The McEliece Cryptosystem 117
A Assignments 121
Part I Coding
Chapter 1

Introduction and Basic Ideas

1.1 Introduction

A model of a communication system is shown in Figure 1.1.

Figure 1.1: Model of a Communications System

The function of each block is as follows

\textbf{Source} : This produces the output that we would like to transmit over the channel. The output can be either continuous, for example voice, or discrete as in voltage measurements taken at set times.

\textbf{Source Encoder} : The source encoder transforms the source data into binary digits (bits). Further, the aim of the source encoder is to minimise the number of bits required to represent the source data. We have one of two options here in that we could require that the source data be perfectly reconstructible (as in computer data) or we may allow a certain level of errors in the reconstruction (as in images or speech).

\textbf{Encryption} : Is intended to make the data unintelligible to all but the intended receiver. That is, it is intended to preserve the secrecy of the messages in the presence of unwelcome monitoring of the channel. Cryptography is the science of maintaining secrecy of data from both passive intrusion (eavesdropping) and active intrusion (introduction or alteration of messages). This introduces the following distinctions
Authentication: is the corroboration that the origin of a message is as claimed.

Data Integrity: is the property that the data has not been changed in an unauthorised way.

Channel Encoder: Tries to maximise the rate at which information can be reliably transmitted on the channel in the presence of disruptions (noise) that can introduce errors. Error correction coding, located in the channel encoder, adds redundancy in a controlled manner to messages to allow transmission errors to be detected and/or corrected.

Modulator: Transforms the data into a format suitable for transmission over the channel.

Channel: The medium that we use to convey the data. This could be in the form of a wireless link (as in cell phones), a fiber optic link or even a storage medium such as magnetic disks or CD’s.

Demodulator: Converts the received data back into its original format (usually bits).

Channel Decoder: Uses the redundancy introduced by the channel encoder to try and detect/correct any possible errors that the channel introduced.

Decryption: Converts the data back into an intelligible form. At this point one would also perform authentication and data integrity checks.

Source Decoder: Adds back the original (naturally occurring) redundancy that was removed by the source encoder.

Receiver: The intended destination of the data produced by the source.

1.2 Terminology

Definition 1.2.1 (Alphabet). An alphabet is a finite, nonempty set of symbols. Usually the binary alphabet \( K = \{0, 1\} \) is used and the elements of \( K \) are called binary digits or bits.

Definition 1.2.2 (Word). A word is a finite sequence of elements from an alphabet.

Definition 1.2.3 (Code). A code is a set \( C \) of words (or codewords).
Example 1.2.4. \( C_1 = \{00, 01, 10, 11\} \) and \( C_2 = \{000, 001, 01, 1\} \) are both codes over the alphabet \( K \).

Definition 1.2.5 (Block Code). A block code has all codewords of the same length; this number is called the length of the code.

In our example \( C_1 \) is a (binary) block code of length 2, while \( C_2 \) is not a block code.

Definition 1.2.6 (Prefix code). A prefix code is a code such that there do not exist distinct words \( w_i \) and \( w_j \) such that \( w_i \) is a prefix (initial segment) of \( w_j \).

\( C_2 \) is an example of a prefix code.

Prefix codes can be decoded easily since no codeword is a prefix of any other. We simply scan the word from left to right until we have a codeword, we then continue scanning from this point on until we reach the next codeword. This is continued until we reach the end of the received word.

Example 1.2.7. Suppose the words 000, 001, 01 and 1 correspond to N, S, E and W respectively. If 00101110000101011 is received, the message is S E W W N S E E W.

Since \( C_2 \) is a prefix code, there is only one way to decode this message.

Prefix codes can be used in a situation where not all data are equally likely — shorter words are used to encode data that occur more frequently and longer words are used for infrequent data. A good example of this is Morse code. In English, the letter E occurs quite frequently and so in Morse code it is represented by a single dot \( . \). On the other hand the letter Z rarely occurs and it is assigned the sequence \( - - \cdot \cdot \) of dots and dashes.

1.3 Basic Assumptions About the Channel

We make the following three assumptions about the channel

1. If \( n \) symbols are transmitted, then \( n \) symbols are received — though maybe not the same ones. That is, nothing is added or lost.

2. There is no difficulty identifying the beginning of the first word transmitted.

3. Noise is scattered randomly rather than being in clumps (called bursts). That is the probability of error is fixed and the channel stays constant over time. Most channels do not satisfy this property with one notable exception — the deep space channel (used to transmit images and other data between spacecraft and earth).
almost all other channels the probability of error varies over time. For example on a CD, where certain areas are more damaged than others, and in cell phones where the radio channel varies continually as the users move about their terrain.

A binary channel is called *symmetric* if 0 and 1 are transmitted with equal accuracy. The reliability of such a channel is the probability $p$, $0 \leq p \leq 1$, that the digit sent is the digit received. The error probability $q = 1 - p$ is the probability that the digit sent is received in error.

**Figure 1.2: The Binary Symmetric Channel**

*Example 1.3.1.* If we are transmitting codewords of length 4 over a binary symmetric channel (BSC) with reliability $p$ and error probability $q = 1 - p$, then

- Probability of no errors: $\binom{4}{0} p^4 q^0$,
- Probability of 1 error: $\binom{4}{1} p^3 q^1$,
- Probability of 2 errors: $\binom{4}{2} p^2 q^2$,
- Probability of 3 errors: $\binom{4}{3} p^1 q^3$,
- Probability of 4 errors: $\binom{4}{4} p^0 q^4$. 

$\blacksquare$
Chapter 2

Detecting and Correcting Errors

Errors can be detected when the received word is not a codeword. If a codeword is received, then it could be that no errors, or several errors (changing one codeword into another) have occurred.

Consider $C_1 = \{00, 01, 10, 11\}$. Every received word is a codeword, so no errors can be detected (and none can be corrected). On the other hand consider

$$C_2 = \{000000, 010101, 101010, 111111\},$$

obtained from $C_1$ by repeating each word in $C_1$ three times — this is known as a repetition code. Here we can detect 2 or fewer errors as we need 3 or more errors to change one codeword into another.

Example 2.1.2. Let $C_2 = \{000000, 010101, 101010, 111111\}$ be the code defined above. Assume that 010101 is sent, but that 110111 is received. By examining $C_2$ we conclude that at least 1 or 2 errors could have occurred (more is also possible). Keep in mind that we as the receiver only know the received word.

We can correct 1 error by using a majority rule: examine each pair of bits starting at the left and decode the received word to a word that has its repeating bits that pair that occurred the majority of times. So if 101010 is sent and 111010 is received, we decode this to 101010. The idea is that if 1 error occurs, then there exists a unique codeword that differs in 1 place from the received word; we decode to this word. If 2 errors occur, the possibility exists that we may decode to the wrong codeword: if 010101 is sent and 110111 is received we will incorrectly decode this to 111111. Therefore this code cannot, in general, correct 2 errors (specific cases may be possible, but not all possible patterns of two errors are correctable).

Consider $C_3 = \{000, 011, 101, 110\}$, formed from $C_1$ by adding a third digit to each codeword such that the total number of ones in the resulting word is even. This is called

7
Chapter 2. Detecting and Correcting Errors

A parity check code and the digit that was added is called the parity check digit. This code can detect 1 error as no two codewords differ in exactly one place, but it cannot detect 2 errors since two errors may change one codeword into another codeword.

Example 2.1.3. Let \( C_3 = \{000, 011, 101, 110\} \) be the parity check code from above. If 000 is sent and 001 is received we immediately detect an error as the received word has an odd number of ones. If we had been in a position whereby we knew that only one error has occurred (in general we won’t know how many errors will have occurred) this won’t even help us in correctly decoding the word above as 011, 101 and 000 all could have been possible sent words.

If 011 is sent and 110 is received (i.e two errors occurred) we will not detect an error since the received word has the required number of ones.

If \( u \) and \( v \) are binary words of the same length, we define \( u + v \) to be the binary word obtained by component-wise addition modulo 2. That is \( 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1 \) and \( 1 + 1 = 0 \).

Example 2.1.4. \( 01101 + 11001 = 10100 \)

Definition 2.1.5 (Hamming weight). Let \( v \) be a binary word of length \( n \). The Hamming weight of \( v \), denoted by \( wt(v) \), is the number of times the digit 1 occurs in \( v \).

Definition 2.1.6 (Hamming distance). Let \( u \) and \( v \) be a binary words of length \( n \). The Hamming distance between \( u \) and \( v \), denoted by \( d(u, v) \), is the number of places where \( u \) and \( v \) differ.

Note that \( d(u, v) = wt(u + v) \) — in the places where \( u \) and \( v \) differ a 1 will appear in \( u + v \) and in the places where \( u \) and \( v \) are the same a 0 will appear in \( u + v \) — from the way that + was defined.

Example 2.1.7. \( d(01011, 00111) = 2 = wt(01011 + 00111) = wt(01100) \) and \( d(10110, 10110) = 0 = wt(10110, 10110) = wt(00000) \).

Let \( C \) be a binary code of length \( n \). If \( v \in C \) is sent and \( w \) is received, then the error pattern is \( u = v + w \). That is the error pattern indicates those positions where an error occurred by a 1. Also since \( u = v + w, w = v + u \) by the addition defined earlier (addition and subtraction are the same under this rule). Therefore, the received word equals the transmitted word plus the error pattern.

Definition 2.1.8 (Detecting an error pattern). A code \( C \) detects the error pattern \( u \) if \( u + v \notin C \forall v \in C \).
Example 2.1.9. Let $C = \{001, 101, 110\}$. Then $C$ detects the error pattern $u = 010$ because none of the codewords added to the error pattern is again a codeword: $001 + 010 = 011 \notin C$, $101 + 010 = 111 \notin C$ and $110 + 010 = 100 \notin C$. On the other hand $C$ does not detect $u = 100$ because the codeword 001 added to $u$ is again a codeword: $001 + 100 = 101 \in C$.

Definition 2.1.10 (Minimum distance of a code). For a code $C$ with $|C| \geq 2$, the minimum distance of $C$, $d_{\text{min}}(C)$, is the smallest distance between two distinct codewords. That is

$$d_{\text{min}} = \min_{u, v \in C, u \neq v} d(u, v).$$

Example 2.1.11. Let $C = \{0000, 1010, 0111\}$. Then

$$d(0000, 1010) = \text{wt}(0000 + 1010) = \text{wt}(1010) = 2,$$
$$d(0000, 0111) = \text{wt}(0000 + 0111) = \text{wt}(0111) = 3,$$
$$d(1010, 0111) = \text{wt}(1010 + 0111) = \text{wt}(1101) = 3.$$

Therefore the minimum distance of $C$ is 2.

Theorem 2.1.12. A code $C$ can detect all nonzero error patterns of weight at most $d - 1$ if and only if $d_{\text{min}}(C) \geq d$.

Proof.
\(\Leftarrow\): Suppose $C$ has $d_{\text{min}}(C) \geq d$. Let $u$ be a nonzero error pattern with $\text{wt}(u) \leq d - 1$ and $v \in C$. Suppose $v$ is sent and $w = v + u$ is received. Then $d(v, w) = \text{wt}(v + w) = \text{wt}(v + v + u) = \text{wt}(u) \leq d - 1$. Since $C$ has minimum distance at least $d$, $w \notin C$ as the codeword closest to $v$ (in terms of Hamming distance) is at least a distance $d$ from $v$. Therefore $C$ detects the error pattern $u$.

\(\Rightarrow\): Suppose $C$ can detect all nonzero error patterns of weight at most $d - 1$. Let $v, w \in C$ and $v \neq w$. Then $u = v + w$ is not a detectable error pattern as $v + u = v + v + w = w \in C$. Thus $\text{wt}(u) = 0$ or $\text{wt}(u) \geq d$. Since $v \neq w$, $\text{wt}(u) \neq 0$. Therefore $\text{wt}(u) \geq d$, so that $d(v, w) = \text{wt}(v + w) = \text{wt}(u) \geq d$. This shows that $d_{\text{min}}(C) \geq d$ since $v$ and $w$ were two arbitrary, distinct codewords. \(\square\)
Definition 2.1.13 (*t*-error detecting code). A code \( C \) is a *t*-error detecting code if

1. \( C \) detects all nonzero error patterns of weight at most \( t \).
2. \( C \) fails to detect some error pattern of weight \( t + 1 \).

So, by Theorem 2.1.12 a code \( C \) is *t*-error detecting if and only if \( C \) has minimum distance \( t + 1 \).

Definition 2.1.14 (Correcting an error pattern). A code \( C \) is said to correct the error pattern \( u \) if \( \forall v \in C, v + u \) (the received word) is closer (in the sense of Hamming distance) to \( v \) than to any other \( w \in C \).

Example 2.1.15. Let \( C = \{0000, 1010, 0111\} \). Then \( C \) corrects the error pattern \( u = 0100 \):

- If 0000 is sent, then (with \( u \) as error pattern) 0000 + \( u = 0100 \) is received. This received word is closer to 0000 than to any other codeword.

- If 1010 is sent, then 1010 + \( u = 1110 \) is received. This received word is closer to 1010 than to any other codeword.

- If 0111 is sent, then 0111 + \( u = 0011 \) is received. This received word is closer to 0111 than to any other codeword.

On the other hand \( C \) does not correct the error pattern 1000: If 0000 is sent, then 0000 + 1000 = 1000 is received and \( d(1000, 0000) = 1 = d(1000, 1010) \). Thus there is more than one codeword that is closest to the received word.

Theorem 2.1.16. A code \( C \) will correct all error patterns of weight at most \( t \) \( \iff \) \( d_{\min}(C) \geq 2t + 1 \).

Proof.

\( \Rightarrow \): Suppose that \( C \) corrects all error patterns of weight at most \( t \), but that \( d_{\min}(C) \leq 2t \). Let \( v, w \in C \) such that \( d(v, w) = d_{\min}(C) = d \leq 2t \). Let \( u \) by any error pattern obtained from \( v + w \) by replacing 1’s by 0’s until only \( \lceil d/2 \rceil \) 1’s remain. If \( v \) is sent and the error pattern \( u \) occurs, then

\[
\begin{align*}
d(v, v + u) &= wt(v + v + u) = wt(u) = \left\lceil \frac{d}{2} \right\rceil, \\
d(w, v + u) &= wt(w + v + u) = d_{\min}(C) - \left\lceil \frac{d}{2} \right\rceil \leq \left\lceil \frac{d}{2} \right\rceil.
\end{align*}
\]
The second to last step, \( wt(w + v + u) = d_{\text{min}}(C) - \lceil d/2 \rceil \), follows from the fact that \( wt(w + v) = d_{\text{min}}(C) \) and that \( u \) has its 1’s in exactly the same locations where \( w + v \) has some of its 1’s. So by computing \( w + v + u \), \( u \) will cancel exactly \( \lceil d/2 \rceil \) of \( (w + v) \)'s 1’s. Therefore the received word, \( v + u \), is at least as close to \( w \) as to \( v \). This implies that \( C \) does not correct \( u \), but \( u \) is an error pattern of weight \( \lceil d/2 \rceil \) and \( \lceil d/2 \rceil \leq t \), so \( C \) was supposed to be able to correct \( u \), a contradiction.

\( \Leftarrow: \)
Suppose \( C \) has \( d_{\text{min}}(C) \geq 2t + 1 \). Let \( u \) be a nonzero error pattern of weight at most \( t \). If \( v \in C \) is sent and \( v + u \) is received, then \( d(v, v + u) = wt(v + v + u) = wt(u) \leq t \). Now for any other \( w \in C \), \( w \neq v \), \( d(w, v + u) = wt(w + v + u) \geq 2t + 1 - wt(u) \geq 2t + 1 - 2t = t + 1 \). As above, \( wt(w + v + u) \) is bounded by realizing that \( wt(w + v) = d(w, v) \geq 2t + 1 \) since \( w \) and \( v \) are distinct codewords and \( d_{\text{min}}(C) \geq 2t + 1 \). Further \( u \) will be able to cancel at most \( wt(u) \leq t \) of \( (w + v) \)'s 1’s. Therefore the received word \( v + u \) is closer to \( v \) than to any other codeword \( w \), \( w \neq v \). Thus \( v + u \) can be correctly decoded showing that \( C \) corrects all error patterns of weight at most \( t \).

\[ \Box \]

**Definition 2.1.17 (t-error correcting code).** A code \( C \) is a **t-error correcting code** if

1. \( C \) corrects all error patterns of weight at most \( t \).

2. Fails to correct at least one error pattern of weight \( t + 1 \).
Chapter 3

Linear Codes

3.1 Introduction

Definition 3.1.1 (Linear code). A code $C$ is called a linear code if $u + v \in C$ whenever $u, v \in C$.

Example 3.1.2. Let $C_1 = \{000, 001, 101\}$. $C_1$ is not a linear code since $001 + 101 = 100 \notin C_1$. Let $C_2 = \{0000, 1001, 0110, 1111\}$, then $C_2$ is linear: If $v \in C_2$, then $v + v = 0000 \in C$ and also $0000 + u = u \forall u \in C$. Therefore we only have to consider the addition of two nonzero, distinct words:

\[
\begin{align*}
1001 + 0110 &= 1111 \in C, \\
1001 + 1111 &= 0110 \in C, \\
0110 + 1111 &= 1001 \in C.
\end{align*}
\]

Theorem 3.1.3. The minimum distance of a linear code is the smallest weight of a nonzero codeword.

Proof.

See assignment 1 in the appendix.

Recall that we let $K = \{0, 1\}$ be the binary alphabet. If we define $K^n$ to be the set of words (vectors) of length $n$ over $K$, then $K^n$ together with the addition defined earlier and scalar multiplication by elements of $K$ is a vector space.

Let $S \subseteq K^n$, say $S = \{v_1, v_2, \ldots, v_k\}$, then the subspace spanned by $S$ (or generated by $S$) is

\[\langle S \rangle = \{w \in K^n \mid w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k, \alpha_i \in K\},\]
if $S \neq \emptyset$. If $S = \emptyset$, then $\langle S \rangle = \{\overline{0}\}$. Note that a linear code can also be thought of as a subspace generated by some set $S$.

**Example 3.1.4.** Let $S = \{0100, 0011, 1100\}$. The the subspace (code) generated by $S$, $C = \langle S \rangle$, is

$$C = \{ w \mid w = \alpha_1(0100) + \alpha_2(0011) + \alpha_3(1100) \mid \alpha_i \in K \}.$$ 

<table>
<thead>
<tr>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$w$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1100</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0011</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1111</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0100</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0111</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1011</td>
</tr>
</tbody>
</table>

Recall that two vectors $u = \{u_1, u_2, \ldots, u_n\}$ and $v = \{v_1, v_2, \ldots, v_n\}$ in $K^n$ are **orthogonal** if their dot product $u \cdot v = u_1v_1 + u_2v_2 + \cdots + u_nv_n = 0$.

**Example 3.1.5.** If $u = 11001$ and $v = 01101$, then

$$u \cdot v = 1 \times 0 + 1 \times 1 + 0 \times 1 + 0 \times 0 + 1 \times 1 = 0 + 1 + 0 + 0 + 1 = 0,$$

so $u$ and $v$ are orthogonal, keeping in mind that addition is done mod 2.

If $S \subseteq K^n$, a vector $v \in K^n$ is **orthogonal to $S$** if $v \cdot x = 0 \forall x \in S$. The set of vectors orthogonal to $S$ is called the orthogonal complement of $S$ and is denoted by $S^\perp$.

**Theorem 3.1.6.** If $V$ is a vector space and $S \subseteq V$ (note a **subset**, not necessarily a subspace), then $S^\perp$ is a **subspace** of $V$.

If $S \subseteq K^n$ and $C = \langle S \rangle$ (the linear code generated by $S$), we write $C^\perp = S^\perp$ and call $C^\perp$ the dual code of $C$.

**Example 3.1.7.** Let $S = \{0100, 0101\}$. Then $C = \langle S \rangle = \{0000, 0101, 0100, 0001\}$. To find $C^\perp$ we must find all words $v = v_1v_2v_3v_4$ such that $v \cdot 0100 = 0$ and $v \cdot 0101 = 0$. It is enough to ensure that $v$ is orthogonal to the two basis vectors as $v$ will also be orthogonal to any linear combination of these vectors. This translates to

$$0v_1 + 1v_2 + 0v_3 + 0v_4 = 0,$$
$$0v_1 + 1v_2 + 0v_3 + 1v_4 = 0.$$
3.1. INTRODUCTION

From the first equation we find that $v_2 = 0$ and then from the second equation we see that $v_4 = 0$. Thus as long as $v_2 = v_4 = 0$, $v$ will be orthogonal to $C$; $v_1$ and $v_3$ may assume arbitrary values. This implies that

$$C^\perp = \{0000, 0010, 1000, 1010\} = S^\perp.$$ 

Definition 3.1.8 (Dimension). The dimension of a linear code $C = \langle S \rangle$, is the dimension of $\langle S \rangle$. We denote this by $\dim(C)$.

The next result follows from linear algebra.

Theorem 3.1.9. If $C$ is a linear code of length $n$, then $\dim(C) + \dim(C^\perp) = n$.

Proposition 3.1.10. A linear code $C = \langle S \rangle$ of dimension $k$ contains exactly $2^k$ codewords.

Proof. Suppose $C = \langle S \rangle$ has dimension $k$ and let $\{v_1, v_2, \ldots, v_k\}$ be a basis for $\langle S \rangle$. Then each codeword $w \in C$ can be uniquely written as: $\alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k$; $\alpha_i \in K = \{0, 1\}$. Since there are 2 choices for each $\alpha_i$, there are $2^k$ choices for $(\alpha_1, \alpha_2, \ldots, \alpha_k)$. Each such choice gives a different word.

Definition 3.1.11 (Information rate). The information rate of a code $C$ is defined as

$$i(C) = \frac{\log_2 |C|}{n}.$$ 

Corollary 3.1.12. The information rate of a linear code $C$ of dimension $k$ and length $n$ is $k/n$.

Proof.

$$i(C) = \frac{\log_2 |C|}{n} = \frac{\log_2 2^k}{n} = \frac{k}{n}.$$ 

\[]
3.2 The Generator and Parity Check Matrices

Let $C = \langle S \rangle$ be a linear code and \{v_1, v_2, \ldots, v_k\} be a set of basis vectors for $C$ (as in the proof of Proposition 3.1.10). Further, let $G$ be the $k \times n$ matrix whose rows are $v_1, v_2, \ldots, v_k$. Then the codeword $w = \alpha_1 v_1 + \alpha_2 v_2 + \cdots + \alpha_k v_k = [\alpha_1, \alpha_2, \ldots, \alpha_k]G$. Every codeword arises in this way for some choice of $[\alpha_1, \alpha_2, \ldots, \alpha_k]$. If we therefore let the binary words of length $k$ correspond to our data, we can encode a word of length $k$ simply by computing $[\alpha_1, \alpha_2, \ldots, \alpha_k]G$.

**Example 3.2.1.** Let $S = \{11101, 10110, 01011, 11010\}$ and $C = \langle S \rangle$. To be able to use $C$ for encoding we first need a basis for $\langle S \rangle$. We do this by writing the elements of $S$ into a matrix (as its rows) and reducing this matrix to row echelon form:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$  

Therefore a basis for $\langle S \rangle$ is $\{11101, 01011, 00111\}$. We now let

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}.$$  

Then $G$ is what is known as a generator matrix for $C$ (see Definition 3.2.2).

Suppose we have established the following correspondence:

<table>
<thead>
<tr>
<th>A</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>001</td>
</tr>
<tr>
<td>H</td>
<td>010</td>
</tr>
<tr>
<td>K</td>
<td>011</td>
</tr>
<tr>
<td>L</td>
<td>100</td>
</tr>
<tr>
<td>M</td>
<td>101</td>
</tr>
<tr>
<td>P</td>
<td>110</td>
</tr>
<tr>
<td>X</td>
<td>111</td>
</tr>
</tbody>
</table>

Then the message HELP is encoded as a sequence of four codewords by computing

Definition 3.2.2 (Generator matrix). A generator matrix for a linear code $C = \langle S \rangle$ is a matrix $G$ whose rows are a basis for $C$.

The generator matrix has the following properties:

- The rows of $G$ are linearly independent.
- The number of rows of $G$ is equal to $\dim(C)$.
- Any linear code $C$ has a generator matrix in row echelon form or reduced row echelon form.
- Any matrix $G$ whose rows are linearly independent is the generator matrix for some linear code.

Definition 3.2.3 (Parity check matrix). A parity check matrix for a linear code $C$ is a matrix whose columns are a basis for the dual code $C^\perp$. Therefore $H$ is a parity check matrix for $C$ if and only if $H^T$ is a generator matrix for $C^\perp$. Also, if $C$ has length $n$ and $\dim(C) = k$, then $C^\perp$ has length $n$ and $\dim(C^\perp) = n - k$. Thus $H$ has $n$ rows and $n - k$ columns.

Theorem 3.2.4. Let $C$ be a linear code of length $n$, dimension $k$ and let $H$ be a parity check matrix for $C$. Then $C = \{ w \in K^n \mid wH = 0 \}$.

Proof.
Let $H = \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_{n-k} \end{bmatrix}$, where the columns of $H$, $x_1, x_2, x_3, \ldots, x_{n-k}$, are a basis for $C^\perp$. Then $wH$ is the vector $[w \cdot x_1, w \cdot x_2, w \cdot x_3, \ldots, w \cdot x_{n-k}]$. Now $w \in K^n$ has $wH = 0$ if and only if each $w \cdot x_i = 0$. That is $wH = 0$ if and only if $w$ is orthogonal to each $x_i$ if and only if $w \in (C^\perp)^\perp = C$. \qed

Note that if $G$ is a generator matrix for $C$ and $H$ a parity check matrix for $C$, $GH = 0$. This follows from the fact that the rows of $G$ are a basis for $C$, while the columns of $H$ are a basis for $C^\perp$.

We now consider the following special case: Let $C = \langle S \rangle$ be a linear code of length $n$ and dimension $k$. If $A$ is the matrix whose rows are the words in $S$ and if $A$ can be put in reduced row echelon form (which is not always possible, hence the special case), that is

$$A \rightarrow \begin{bmatrix} I_k | X \\ 0 | 0 \end{bmatrix},$$
then \( G = [I_k \mid X] \) is a generator matrix for \( C \). Note the dimensions of the sub matrix \( X \):

\[
k \text{ rows } \rightarrow [I_k \mid X].
\]

Let

\[
H = \begin{bmatrix} X \\ I_{n-k} \end{bmatrix},
\]

then

\[
GH = [I_k \mid X] \begin{bmatrix} X \\ I_{n-k} \end{bmatrix},
\]

\[
= I_k X + X I_{n-k},
\]

\[
= X + X = 0.
\]

This implies that every column of \( H \) lies in \( C^\perp \), furthermore there are \( n - k \) linearly independent columns (because of the identity matrix) and so the columns of \( H \) form a basis for \( C^\perp \). Thus \( H \) is a parity check matrix for \( C \).

So in this special case we can transform between the matrices \( H_{C^\perp}, G_{C^\perp}, G_C \) and \( H_C \), which are the parity check and generator matrices for the codes \( C \) and \( C^\perp \), as shown in the following diagram.

\[
\begin{array}{cccc}
H_{C^\perp} & \xrightarrow{*} & G_{C^\perp} \\
\uparrow T & & \uparrow T \\
G_C & \xrightarrow{*} & H_C
\end{array}
\]

Here \( T \) denotes transpose and \( * \) denotes the operation described above.

**Example 3.2.5.** (This example is continued from the previous one.)

In the previous example we had \( S = \{11101, 10110, 01011, 11010\} \) and \( C = \langle S \rangle \). We found a basis for \( C \) by writing the elements of \( S \) into a matrix \( a \) (as its rows) and reducing \( A \) to row echelon form. We now take this one step further and reduce \( A \) to reduced row echelon form.

\[
A = \begin{bmatrix}
1 & 1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\rightarrow
\begin{bmatrix}
I_3 \mid X
\end{bmatrix}.
\]
Therefore

\[ G = \begin{bmatrix}
I_3 & 0 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}, \]

is a (different) generator matrix for \( C \) and

\[ H = \begin{bmatrix}
0 & 1 \\
1 & 1 \\
1 & 1 \\
1 & 0 \\
0 & 1 \\
\end{bmatrix}, \]

is a parity check matrix for \( C \). Note also that \( GH = 0 \).

**Definition 3.2.6 (Equivalent codes).** Let \( C_1 \) and \( C_2 \) be block codes of length \( n \). If there is a permutation of the \( n \) digits which, when applied to codewords in \( C_1 \) gives \( C_2 \), then \( C_1 \) and \( C_2 \) are called equivalent.

**Example 3.2.7.** Let \( C_1 \) be the linear code with generator matrix

\[ G = \begin{bmatrix}
1 & 0 & 0 \\
0 & 0 & 1 \\
\end{bmatrix}, \]

so that \( C_1 = \{000, 001, 100, 101\} \) (this can easily be seen, as \( C_1 \) is just all possible sums of the rows of \( G \)).

Let \( C_2 \) be obtained from \( C_1 \) by exchanging the last two digits of every codeword of \( C_1 \). Then \( C_2 = \{000, 010, 100, 110\} \), so that \( C_1 \) and \( C_2 \) are equivalent.

Equivalent codes are exactly the same, except for the order of the digits.

**Proposition 3.2.8.** Any linear code \( C \) is equivalent to a (linear) code \( C' \) having a generator matrix in the standard form \( G' = [I_k \mid X] \).

**Example 3.2.9.** (Continued from previous example.)
\( C_1 \)'s codewords all have a 0 in their second position. Therefore \( C_1 \) can not have a generator matrix in the standard form \([I_2 \mid X]\), as such a generator matrix will generate codewords that have a nonzero second digit: The second row of this matrix as a 1 in the second digit (because of the \( I_2 \)) and this row itself is a codeword.
On the other hand

\[
G_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix},
\]

is a generator matrix for \( C_2 \) in standard form and by the previous example we know that \( C_1 \) and \( C_2 \) are equivalent.

Let \( G = [I_k \mid X] \) be a standard form generator matrix for a code \( C \), and let \( y \in K^k \) be encoded as \( w = yG = y[I_k \mid X] = [yI_k \mid yX] = [y \mid yX] \). Then from the last expression it is clear that the first \( k \) digits of \( w \) carry the information being encoded (i.e. \( y \)) and the last \( n - k \) digits are there to detect/correct errors. In this case the first \( k \) digits are known as the information digits and the last \( n - k \) digits are known as the parity check digits.

### 3.3 Cosets

Recall from Algebra that \((K^n, +)\) may be regarded as a group and that a linear code \( C \subseteq K^n \) can be thought of as a subgroup of \((K^n, +)\).

**Definition 3.3.1 (Coset).** The (right) coset of \( C \) determined by a word \( u \in K^n \) is the set

\[
C + u = \{v + u \mid v \in C\}.
\]

Furthermore, from Algebra it also follows that distinct cosets partition \( K^n \). The following result is well known.

**Theorem 3.3.2.** Let \( C \subseteq K^n \) be a linear code and \( u, v \in K^n \). Then

1. \( u \in C + u \).
2. \( u \in C + v \Rightarrow C + u = C + v \).
3. \( u + v \in C \Rightarrow C + u = C + v \).
4. \( u + v \notin C \Rightarrow C + u \neq C + v \).
5. Either \( C + u = C + v \) or \((C + u) \cap (C + v) = \emptyset\).
6. \(|C + u| = |C|\).
7. \( \dim(C) = k \Rightarrow |C + u| = 2^k \ \forall u \in K^n \). That is there exists \( 2^{n-k} \) different cosets.
3.3. COSETS

8. $C = C + 0$ is a coset.

Example 3.3.3. Let $C = \{0000, 1011, 0101, 1110\}$. Then the cosets (in this case left and right cosets are the same since addition is commutative) of $C$ are:

<table>
<thead>
<tr>
<th>$C$</th>
<th>$C + 1000$</th>
<th>$C + 0100$</th>
<th>$C + 0010$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>1000</td>
<td>0100</td>
<td>0010</td>
</tr>
<tr>
<td>1011</td>
<td>0011</td>
<td>1111</td>
<td>1001</td>
</tr>
<tr>
<td>0101</td>
<td>1101</td>
<td>0001</td>
<td>0111</td>
</tr>
<tr>
<td>1110</td>
<td>0110</td>
<td>1010</td>
<td>1100</td>
</tr>
</tbody>
</table>

The cosets are determined as follows. By Theorem 3.3.2 number 8, $C$ itself is always a coset. We therefore write down its elements. Next, Theorem 3.3.2 number 5 guarantees that the cosets partition $K^n$. We now look for a word in $K^n$ that does not appear in any of the cosets that we have already found and add it to the elements of $C$. This generates a new coset as all elements in this coset does not appear in any other coset (by number 5 of the previous Theorem). We keep on repeating this process until we have accounted for all the elements of $K^n$.

3.3.1 Maximum Likelihood Decoding (MLD) of Linear Codes

Suppose the word $w$ is received, if there exists a unique codeword $v$ such that $d(w, v)$ is minimum, then decode $w$ as $v$. If more than one such a codeword exists we have one of two options.

- **Complete** Maximum Likelihood Decoding (CMLD) — we arbitrarily choose one of the candidate codewords.

- **Incomplete** Maximum Likelihood Decoding (IMLD) — we request that the codeword be sent again.

If $C$ is a linear code and $v \in C$ is sent and $w \in K^n$ is received, the error pattern is $u = v + w$ (remember that addition and subtraction is the same operation). Then $w + u = w + w + v = v \in C$. By Theorem 3.3.2 number 3, $w$ and $u$ are in the same coset. Now the received word makes it possible to determine the coset and from the coset we should therefore be able to extract the error pattern.

Assuming that errors of small weight are more likely (since the channel is hopefully not too bad), we choose as the error pattern, $u$, a word of least weight in the coset that
contains the received word \((C + w)\) (the error pattern cancels itself).

**Example 3.3.4. (Continued from previous example.)**
Suppose \(w = 1101\) is received. Then \(w \in C + 1000\) and the word of least weight in \(C + 1000\) is \(u = 1000\). Therefore we decode \(w\) as \(w + u = 1101 + 1000 = 0101\). Note that the sum of two words in the same coset is always a codeword.

As a second example suppose \(w = 1111\) is received. Then \(w \in C + 0100\) and in this case there are two words of least weight present in this coset. They are 0100 and 0001. If we are using IMLD we will ask for a retransmission, otherwise using CMLD we arbitrarily choose between 0100 and 0001 as the possible error pattern.

Two steps arise in the decoding of linear codes.

- Find the coset containing the received word.
- Find the word of least weight in this coset.

The need to perform these operations as quickly as possible prompts the following use of the parity check matrix.

**Definition 3.3.5 (Syndrome).** Let \(C \subseteq K^n\) be a linear code and suppose \(\dim(C) = k\). Let \(H\) be a parity check matrix for \(C\). For \(w \in K^n\), the syndrome of \(w\) is the word \(wH\) (\(\in K^{n-k}\)).

**Example 3.3.6. (Continued from previous example.)**
A parity check matrix for \(C\) is

\[
H = \begin{bmatrix}
1 & 1 \\
0 & 1 \\
1 & 0 \\
0 & 1
\end{bmatrix}.
\]

The syndrome for \(w = 1101\) is \(wH = 11\). In fact all \(x \in C + w\) have syndrome 11 (see next Theorem).

**Theorem 3.3.7.** Let \(C \subseteq K^n\) be a linear code with parity check matrix \(H\). Then for \(u, w \in K^n\)

1. \(wH = 0 \iff w \in C\).

2. \(wH = uH \iff w\) and \(u\) are in the same coset of \(C\).
3.3. COSETS

Proof.

1. See Theorem 3.2.4.

2.

\[ wH = uH \iff (w + u)H = 0, \]
\[ \iff w + u \in C \text{ (by 1.),} \]
\[ \iff C + w = C + u. \]

We note the following.

- We can identify each coset by its syndrome. If \( C \) has dimension \( k \), there are \( 2^{n-k} \) cosets and \( 2^{n-k} \) syndromes. The fact that there are \( 2^{n-k} \) syndromes follows from considering the form of the parity check matrix: it has an identity matrix of size \( n - k \) in its bottom part and all possible words of length \( n - k \) can be formed by summing these rows (which is what the product \( wH \), that arises when calculating the syndrome, amounts to) the appropriate way.

- If \( C \) is a linear code, \( H \) its parity check matrix, \( v \in C \) is sent and the error pattern \( u \) occurs. Then \( w = v + u \) will be received. The syndrome of \( w \) will be \( wH = (v + u)H = vH + uH = 0 + uH = uH \) (since \( v \) is a codeword, \( vH = 0 \)). Therefore the syndrome of \( w \) will be the sum of those rows of \( H \) that correspond to the positions in \( v \) where the errors occurred.

In summary, if \( C \subseteq K^n \) is a linear code with parity check matrix \( H \) and \( w \in K^n \) is received we decode as follows.

1. Compute syndrome \( wH \).

2. Identify the coset containing \( w \) from the syndrome.

3. Find the word of least weight, say \( u \), in this coset.

4. Decode \( w \) as \( w + u \).

Definition 3.3.8 (Standard decoding array, Coset leader). A standard decoding array (SDA) is a table that matches each syndrome with a word of least weight — the coset leader — in the corresponding coset.
Example 3.3.9.  (Continued from previous example.)
The standard decoding array from the last example is.

<table>
<thead>
<tr>
<th>Syndrome</th>
<th>Coset Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0000</td>
</tr>
<tr>
<td>01</td>
<td>0100</td>
</tr>
<tr>
<td>10</td>
<td>0010</td>
</tr>
<tr>
<td>11</td>
<td>1000</td>
</tr>
</tbody>
</table>

Here \( \Diamond \) serves to indicate that the coset contains more than one word of least weight. Therefore we either ask for a retransmission (IMLD) or choose one of the two words arbitrarily (CMLD).  \hfill \blacksquare
Chapter 4

Bounds for Codes

In this chapter we consider a variety of bounds on the parameters of a code. Note that we do not restrict our attention only to linear codes. We will explicitly indicate when a code is linear. We begin with the following Proposition.

Proposition 4.0.10. If $0 \leq t \leq n$ and $v \in K^n$, then the number words $w \in K^n$ such that $d(v, w) \leq t$ is

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}.$$

Proof. Note that $\binom{n}{i}$ counts the number of words that differ from $v$ in exactly $i$ places. The final result follows by summing over all possible values for $i$.

Theorem 4.0.11 (Hamming bound [?]). If $C \subseteq K^n$ is a code with minimum distance $d_{\text{min}} = 2t + 1$ or $d_{\text{min}} = 2t + 2$ then

$$|C| \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t} \right] \leq 2^n,$$

or

$$|C| \leq \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}}.$$

Proof. For $v \in C$ define the set

$$B_t(v) = \{ w \in K^n \mid d(v, w) \leq t \},$$

25
to be the ball of radius $t$ centred at $v$. If $u, v \in C$ and $u \neq v$, then $B_t(u) \cap B_t(v) = \emptyset$.

Assume some $w \in K^n \in B_t(u) \cap B_t(v)$. That is $d(u, w) \leq t$ and $d(v, w) \leq t$. This implies that $d(u, v) \leq d(u, w) + d(w, v) \leq t + t = 2t$. This contradicts $d_{\text{min}} \geq 2t + 1$.

The above implies that every $w \in K^n$ lies in at most one ball $B_t(v)$ for some $v \in C$ (it might not lie in any ball). Therefore

$$\sum_{v \in C} |B_t(v)| \leq |K^n| = 2^n.$$

By Proposition 4.0.10

$$|B_t(v)| = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t},$$

so that

$$|C| \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t} \right] \leq 2^n.$$

\[ \square \]

**Example 4.0.12.** A code $C$ of length 6 and minimum distance 3 ($d_{\text{min}} = 2t + 1 = 3 \Rightarrow t = 1$) has

$$|C| \leq \frac{2^6}{\binom{6}{0} + \binom{6}{1}} = \frac{64}{7} = 9\frac{1}{7}.$$  

Since $|C| \in \mathbb{N} \cup \{0\}$, we have $|C| \leq 9$. If $C$ is a linear code then $|C|$ is a power of two (Proposition 3.1.10). In this case $|C| \leq 8$ and $\dim(C) \leq 3$.

**Theorem 4.0.13 (Gilbert-Varshamov bound [?, ?]).** If

$$\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2} < 2^{n-k},$$

then there exists a linear code of length $n$, dimension $k$ and minimum distance at least $d$.

**Proof.**

Suppose the inequality holds. We construct a parity check matrix $H$ for the proposed code. Such a matrix $H$ would have $n$ rows and $n - k$ linearly independent columns. To
ensure a minimum distance of $d$ we need to construct $H$ in such a way that any $d - 1$
rows are linearly independent (and some collection of $d$ rows linearly dependent) — see Exercises.

In the last $n - k$ rows of $H$, put the identity matrix $I_{n-k}$. This guarantees that the columns of $H$ are linearly independent. The remainder of the proof is by induction. Suppose the last $l$ rows of $H$ have been completed, where $n - k \leq l \leq n - 1$. We claim that another row can be added:

Among the $2^{n-k}$ possibilities for the rows of $H$ we cannot select the following

- The zero row.
- Any row that is a sum of $1, 2, \ldots, d - 2$ rows already chosen.

The reason is that any such choice would create a linearly dependent set of size at most $d - 1$ while we are trying to construct linearly independent sets of size $d - 1$. The number of rows that are not allowed is

\[
\begin{align*}
&\binom{0}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2}, \\
&\leq \binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2}, \\
&< 2^{n-k}.
\end{align*}
\]

Here we used the fact that if $l \leq n - 1$, then $\binom{l}{i} \leq \binom{n-1}{i}$. Therefore another row is available and so by induction $H$ can be constructed.

Since the columns of $H$ are linearly independent this implies that the dual code has dimension $n - k$ which in turn implies that the code itself has dimension $k$. By selecting the last row to be the sum of some $d - 1$ rows of $H$, we guarantee that the minimum distance equals $d$.

**Corollary 4.0.14.** If $n \neq 1$ and $d \neq 1$ (since we are assuming $d \leq n$ — a code of length $n$ cannot have minimum distance greater than $n$), then there exists a linear code $C$ of length $n$ and minimum distance at least $d$ with

\[
|C| \geq \frac{2^{n-1}}{\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2}}.
\]
Proof.
Let $k$ be the largest integer less than or equal to $n$ such that
\[
\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2} < 2^{n-k}.
\]
Such an integer $k$ exists since the inequality holds in at least one case namely $k = 0$. For this $k$ we have by the Proposition a linear code $C$ with $|C| = 2^k$ and
\[
2^k \left[ \binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2} \right] < 2^k 2^{n-k} = 2^n.
\]
Since we chose $k$ to be the largest integer such that the inequality holds, the inequality will be reversed for $k + 1$. Therefore
\[
2^k < \frac{2^n}{\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2}} \leq 2^{k+1},
\]
or written differently
\[
|C| = 2^k \geq \frac{2^{n-1}}{\binom{n-1}{0} + \binom{n-1}{1} + \binom{n-1}{2} + \cdots + \binom{n-1}{d-2}}.
\]

Definition 4.0.15 ([n, k, d]-code). An $[n, k, d]$-code is a linear code with length $n$, dimension $k$ and minimum distance $d$.

Example 4.0.16.
(a) Is there a $(9, 2, 5)$-code ? According to the Gilbert-Varshamov bound such code will exist if
\[
\binom{8}{0} + \binom{8}{1} + \binom{8}{2} + \binom{8}{3} = 1 + 8 + 28 + 56 = 93,
\]
is less than $2^{9-2} = 2^7 = 128$. Since the inequality holds we know such a code exists.

(b) Does there exist a $(15, 7, 5)$-code ? Using the Gilbert-Varshamov bound again we find that
\[
\binom{14}{0} + \binom{14}{1} + \binom{14}{2} + \binom{14}{3} = 1 + 14 + 91 + 364 = 470,
\]
is not less than $2^{15-7} = 2^8 = 256$ and so we cannot reach any conclusion based on this bound alone.
Find bounds on the size and dimension of a linear code $C$ with $n = 9$ and $d_{\text{min}}(C) = 5$.

Using the Hamming bound we get

$$|C| \leq \frac{2^9}{\binom{9}{0} + \binom{9}{1} + \binom{9}{2}}, \quad (d_{\text{min}}(C) = 2t + 1 = 5 \Rightarrow t = 2),$$

$$= \frac{512}{1 + 9 + 36},$$

$$= \frac{512}{46},$$

$$\approx 11.1.$$  

Since $C$ is linear, $|C| = 2^k$ so that in fact $|C| \leq 8$ and $k \leq 3$.

The Corollary to the Gilbert-Varshamov bound says that

$$|C| \geq \frac{2^8}{\binom{8}{0} + \binom{8}{1} + \binom{8}{2} + \binom{8}{3}},$$

$$= \frac{256}{93},$$

$$\approx 2.8.$$  

Since $C$ is linear, $|C| = 2^k$ so that $|C| \geq 4$ and $k \geq 2$.

**Definition 4.0.17 (Perfect code).** A code $C$ of length $n$ is called *perfect* if equality holds in the Hamming bound. That is if

$$|C| = \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}},$$

where $d_{\text{min}}(C) = 2t + 1$ or $d_{\text{min}}(C) = 2t + 2$.

In other words the balls of radius $t$ centred at each codeword partition the space $K^n$ — every word in $K^n$ is in exactly one such ball. It actually turns out that a perfect code cannot have an even minimum distance: Let $C$ be a code with minimum distance $d_{\text{min}}(C) = 2t + 2$. Let $v \in C$ and change $v$ in $t + 1$ places to obtain a new word $z$. Therefore $d(v, z) = t + 1$. Let $u \in C$, with $u \neq v$. Then $d(u, v) \leq d(u, z) + d(z, v)$ or $d(u, z) \geq d(u, v) - d(z, v) \geq 2t + 2 - (t + 1) = t + 1$. This implies that $z$ is a distance of at least $t + 1$ from every codeword in $C$. Therefore $z$ is in no ball of radius $t$ centred at the codewords in $C$. Thus $C$ is not perfect.

**Example 4.0.18.**
(a) $K^n$ is perfect: $d_{\text{min}}(K^n) = 1 = 2(0) + 1 \Rightarrow t = 0$ and

$$2^n = |K^n| = \binom{n}{0}.$$

(b) The code $C = \{000 \cdots 0, 111 \cdots 1\}$ with length $n = 2t + 1$ and $d_{\text{min}}(C) = 2t + 1$ is perfect:

$$2 = |C| = \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t}} = \frac{2^n}{2^n} = 2.$$

Here we used the fact that if $n = 2t + 1$, then $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t}$ only includes half of the terms in $\binom{n}{0} + \binom{n}{1} + \cdots + \binom{n}{t} + \binom{n}{t+1} + \cdots + \binom{n}{n-1} + \binom{n}{n} = 2^n$.

The two examples above — $K^n$ and the repetition code — are called the trivial perfect codes.

**Theorem 4.0.19 (Tietäväinen and van Lint [?, ?]).** If $C$ is a nontrivial perfect code of length $n$ and minimum distance $d_{\text{min}}(C) = 2t + 1$ then either

1. $n = 2^r - 1$ for some $r \geq 2$ and $d_{\text{min}}(C) = 3$ or

2. $n = 23$ and $d_{\text{min}}(C) = 7$.

Note the following.

- If $C$ is a perfect code of length $n$ and $d = 2t + 1$, then every $w \in K^n$ is within a distance $t$ of some codeword. Thus $C$ can correct all error patterns of weight at most $t$ and no others.

- By Theorem 4.0.19, a nontrivial perfect code is either

  1. of length $2^r - 1$ ($r \geq 2$) and 1-error correcting or

  2. of length 23 and 3-error correcting.
Chapter 5

Hamming Codes

5.1 Introduction

In this chapter we focus our attention on one of the first classes of codes that was discovered, namely the Hamming codes. We do this by describing their parity check matrices.

Definition 5.1.1 (Hamming codes [?]). Let $r \geq 2$ and let $H$ be the $(2^r - 1) \times r$ matrix whose rows are the nonzero binary words of length $r$. The linear code that has $H$ as its parity check matrix is called the Hamming code of length $2^r - 1$.

Note that $H$ has linearly independent columns since $r$ of its rows contain those binary words with only a single 1. In the example below (for the case $r = 3$) these rows are the last three rows.

Example 5.1.2. If we let $r = 3$ in the definition above, we find that the Hamming code of length 7 has the following parity check matrix.

$$H = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}.$$

In the example above we see that the matrix is in standard form. It will always be possible to place it in this form by ensuring that the appropriate rows are at the bottom.
of the matrix. If

\[ H = \begin{bmatrix} X \\ I_r \end{bmatrix}, \]

then the generator matrix will be of the form

\[ G = \left[ I_{2^r-1-r} \mid X \right]. \]

Therefore the Hamming code has dimension \( 2^r - r - 1 \), so that it has \( 2^{2^r-r-1} \) codewords.

Since \( H \) has no row equal to zero and no two identical rows, every set of 2 rows is linearly independent, so that the minimum distance is at least three. On the other hand there is a linearly dependent set of three rows, for example \( \{100 \cdots 00, 0100 \cdots 00, 1100 \cdots 00\} \). This implies that the minimum distance of the Hamming code is exactly three, i.e. it is a 1-error correcting code.

Let’s Consider the Hamming bound in this case, here \( n = 2^r - 1 \) and \( d_{\text{min}} = 3 = 2t + 1 \Rightarrow t = 1 \). Therefore

\[
2^{2^r-r-1} = |C| \leq \frac{2^{2^r-1}}{\binom{2^r-1}{0} + \binom{2^r-1}{1}},
\]

\[
= \frac{2^{2^r-1}}{1 + (2^r - 1)},
\]

\[
= \frac{2^{2^r-1}}{2^r},
\]

\[
= 2^{2^r-r-1}.
\]

Therefore Hamming codes are perfect.

It’s easy to make up a standard decoding array for a Hamming code: The coset leaders are the words of weight at most 1 (see the exercises). All we therefore need is to compute the syndromes for each one.

**Example 5.1.3.** (Continued from previous example.)

<table>
<thead>
<tr>
<th>Syndrome</th>
<th>Coset Leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>0000000</td>
</tr>
<tr>
<td>111</td>
<td>1000000</td>
</tr>
<tr>
<td>110</td>
<td>0100000</td>
</tr>
<tr>
<td>101</td>
<td>0010000</td>
</tr>
<tr>
<td>011</td>
<td>0001000</td>
</tr>
<tr>
<td>100</td>
<td>0000100</td>
</tr>
<tr>
<td>010</td>
<td>0000010</td>
</tr>
<tr>
<td>001</td>
<td>0000001</td>
</tr>
</tbody>
</table>
5.2. EXTENDED CODES

Recall that equivalent codes have exactly the same parameters. Consider then the Hamming code whose parity check matrix are the nonzero binary words of length \( r \) in numerical order. For the example above this would mean that

\[
H' = \begin{bmatrix}
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1
\end{bmatrix},
\]

would be the new parity check matrix.

Let \( C \) be the Hamming code of length \( 2^r - 1 \) that has its parity check matrix’s rows in numerical order. Let \( v \in C \) be sent and let \( u \) be an error pattern in \( K^{2^r-1} \). Then \( w = u + v \) will be received and the syndrome will be \( wH = (u + v)H = uH + vH = uH \) since \( v \) is a codeword and has zero syndrome. Thus if \( u \) is an error pattern of weight 1 (and \( C \) can only correct these) the syndrome will be equal to some row of \( H \). By the ordering of the rows we know in fact that if \( u \) had a 1 in the \( i \)'th position then \( uH \) will be the \( i \)'th row of \( H \) which will equal the number \( i \) expressed in binary. We can therefore decode \( w \) by merely complementing the \( i \)'th bit.

5.2 Extended Codes

Recall that if a code \( C \) has a generator matrix \( G = [I_k \mid X] \), the the first \( k \) bits of a codeword are the information bits and the last \( n - k \) bits are the parity check bits.

Example 5.2.1. (Hamming code.)

Recall that the Hamming code of length 7 has the following generator matrix

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1
\end{bmatrix}.
\]

Take \( x_1x_2x_3x_4 \in K^4 \) and multiply it against \( G \) to get the codeword \( x_1x_2x_3x_4p_1p_2p_3 \). We
then see that

\begin{align*}
p_1 &= x_1 + x_2 + x_3, \\
p_2 &= x_1 + x_2 + x_4, \\
p_3 &= x_1 + x_3 + x_4.
\end{align*}

\begin{definition}[Extended code] Let $C$ be a linear code of length $n$ and $C^\ast$ a code obtained from $C$ by adding one extra digit to each codeword so that each word in $C^\ast$ has even weight. Then $C^\ast$ is called the extended code of $C$.
\end{definition}

If $C$ is a linear code with parity check matrix $H$, consider the following matrix

$$H^\ast = \begin{bmatrix} H & j \\ 0 & 1 \end{bmatrix},$$

obtained from $H$ by adding a column of 1’s to $H$ and adding a row that has zeros everywhere except in its last entry to this. Thus the final column of $H^\ast$ is made up of 1’s. Then $H^\ast$ has $n - k + 1$ linearly independent columns and for every $v \in C^\ast$, $vH^\ast = 0$. Furthermore, $\dim(C^\ast) = k$: We can use the same set of basis vectors for $C^\ast$ that was used for $C$ by adding a digit to each one of the basis vectors such that each new basis vector has even weight. Each new codeword is still a sum of these new basis vectors. Therefore $C^\ast$ is the null space of $H^\ast$, implying that $C^\ast$ is a linear code. As discussed above the generator matrix, $G^\ast$, for $C^\ast$ can be obtained from the generator matrix of $C$ by adding a bit to every row such that each row has even weight. We then again have

$$G^\ast H^\ast = [G \mid i] = \begin{bmatrix} H & j \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} GH \\ 0 \end{bmatrix} [G \mid i] \begin{bmatrix} j \\ 1 \end{bmatrix} = 0,$$

where $i$ represents the column that was added to $G$ so that each row of $G^\ast$ now has an even number of 1’s. The product in the last column is zero since each entry is just the sum of the corresponding row of $[G \mid i]$, but since each row has an even number of 1’s such a sum equals 0. Of course we also have $GH = 0$.

\begin{example} Extending the Hamming code of length 7 we get the following parity

\begin{align*}
p_1 &= x_1 + x_2 + x_3, \\
p_2 &= x_1 + x_2 + x_4, \\
p_3 &= x_1 + x_3 + x_4.
\end{align*}

\end{example}
check and generator matrices

\[
H^* = \begin{bmatrix}
1 & 1 & 1 & 1 \\
1 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 \\
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}, \\
G^* = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
\end{bmatrix}.
\]

If \( v \in C \) and \( v^* \) is the corresponding codeword in \( C^* \) then

\[
wt(v^*) = \begin{cases}
wt(v) & \text{if } wt(v) \text{ is even}, \\
wt(v) + 1 & \text{if } wt(v) \text{ is odd}.
\end{cases}
\]

Therefore

\[
d_{\text{min}}(C^*) = \begin{cases}
d_{\text{min}}(C) & \text{if } d_{\text{min}}(C) \text{ is even}, \\
(d_{\text{min}}(C) + 1) & \text{if } d_{\text{min}}(C) \text{ is odd}.
\end{cases}
\]

Thus if \( d_{\text{min}}(C) \) is odd, then \( C^* \) will detect one more error than \( C \). This implies that extending \( C \) is only useful when \( d_{\text{min}}(C) \) is odd.

The extended Hamming code of length 8, \( C_e \), has a minimum distance of 4 (it was 3 before it was extended). Looking back at the example above in which the generator and parity check matrices were given for this code, we see that any two rows of \( G^* \) are orthogonal (even if the rows aren’t distinct). Therefore

\[
G^*(G^*)^T = 0.
\]

This implies that the rows of \( G^* \) are all in the dual code, \( C_e^\perp \). The dual code has dimension 4 and \( G^* \) has 4 linearly independent rows. Therefore the rows of \( G^* \) are (also) a basis for \( C_e^\perp \). This shows that the extended Hamming code of length 8 is self-dual: \( C_e = C_e^\perp \).
Chapter 6

Golay Codes

6.1 The Extended Golay Code: $C_{24}$.

The extended Golay code, $C_{24}$, is a $[24, 12, 8]$-code. We describe this code by giving its generator matrix. The construction of the generator matrix proceeds in three steps.

Step 1: Let $B_1$ be the $11 \times 11$ matrix whose first row is 11011100010 and each subsequent row is obtained by cyclically shifting its predecessor one position left. That is

\[
B_1 = \begin{bmatrix}
1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1
\end{bmatrix}.
\]

Step 2: Let $B$ be the $12 \times 12$ matrix

\[
B = \begin{bmatrix}
B_1 & j^T \\
\end{bmatrix},
\]

where $j$ is the first column of the identity matrix.
where \( j \) is the all 1’s vector. Therefore

\[
B = \begin{bmatrix}
1 1 0 1 1 1 0 0 0 1 0 1 & 1 \\
1 0 1 1 1 0 0 0 1 0 1 1 & 1 \\
0 1 1 1 0 0 0 1 0 1 1 1 & 1 \\
1 1 1 0 0 0 1 0 1 1 0 1 & 1 \\
1 1 0 0 0 1 0 1 1 0 1 1 & 1 \\
1 0 0 0 1 0 1 1 0 1 1 1 & 1 \\
0 0 1 0 1 1 0 1 1 1 0 1 & 1 \\
0 1 0 1 1 0 1 1 1 0 0 1 & 1 \\
1 0 1 1 0 1 1 1 0 0 0 1 & 1 \\
0 1 1 0 1 1 1 0 0 0 1 1 & 1 \\
1 1 1 1 1 1 1 1 1 1 1 0 & 0
\end{bmatrix}.
\]

Note the following.

1. \( B = B^T \) (\( B_1 \) is symmetric.)
2. \( BB^T = BB = I \).

**Step 3:** The extended Golay code, \( C_{24} \), is the linear code with generator matrix

\[
G = [I_{12} \mid B] = \begin{bmatrix}
1 0 0 0 0 0 0 0 0 0 0 0 & 1 1 0 1 1 1 0 0 0 1 0 1 \\
0 1 0 0 0 0 0 0 0 0 0 0 & 1 0 1 1 1 0 0 0 1 0 1 1 \\
0 0 1 0 0 0 0 0 0 0 0 0 & 0 1 1 1 0 0 0 1 0 1 1 1 \\
0 0 0 1 0 0 0 0 0 0 0 0 & 1 1 1 0 0 0 1 0 1 1 0 1 \\
0 0 0 0 1 0 0 0 0 0 0 0 & 1 1 0 0 0 1 0 1 1 0 1 1 \\
0 0 0 0 0 1 0 0 0 0 0 0 & 0 1 0 0 1 0 1 1 0 1 1 1 \\
0 0 0 0 0 0 1 0 0 0 0 0 & 0 0 1 0 1 1 0 1 1 0 1 1 \\
0 0 0 0 0 0 0 1 0 0 0 0 & 0 1 0 1 1 0 1 1 1 0 0 1 \\
0 0 0 0 0 0 0 0 1 0 0 0 & 1 0 1 1 0 1 1 0 0 0 1 1 \\
0 0 0 0 0 0 0 0 0 1 0 0 & 0 1 1 0 1 1 1 0 0 0 1 1 \\
0 0 0 0 0 0 0 0 0 0 1 0 & 1 1 1 1 1 1 1 1 1 1 1 0
\end{bmatrix}.
\]

We now have

1. \( C_{24} \) has length 24 and dimension 12.
2. The matrix

\[
\begin{bmatrix}
B \\
I_{12}
\end{bmatrix},
\]

is a parity check matrix for $C_{24}$.

3. Furthermore

\[
H = \begin{bmatrix}
I_{12} \\
B
\end{bmatrix},
\]

is also a parity check matrix for $C_{24}$ (with respect to the same generator matrix $G$) :

\[
GH = [I_{12} \mid B] \begin{bmatrix}
I_{12} \\
B
\end{bmatrix} = I_{12}^2 + BB = I + I = 0,
\]

and $H$ has 12 linearly independent columns ($\dim(C^\perp) = 12$). Therefore $H$ is a parity check matrix.

4. Another generator matrix for $C_{24}$ is $[B \mid I_{12}]$

5. $C_{24}$ is self-dual :

\[
GG^T = GH = 0,
\]

therefore every basis vector for $C_{24}$ belongs to $C^\perp_{24}$. Further, $\dim(C_{24}) = \dim(C^\perp_{24}) = 12$ so that the basis vectors for $C_{24}$ are also basis vectors for $C^\perp_{24}$. This implies that $C_{24} = C^\perp_{24}$.

6. $C_{24}$ has minimum distance equal to 8. We prove this in three steps.

(a) The weight of any codeword is a multiple of 4 :

Note that every row of $G$ has weight 8 or 12. Let $v \in C_{24}$ be a sum of two different rows, $r_i$ and $r_j$, of $G$. Since any two different rows of $G$ are orthogonal ($GG^T = 0$), this implies that when calculating their inner-product all of the 1’s in one word get multiplied against zeros in the other or else the two words have to have an even number of 1’s in the same positions (so that their product can sum to zero). Therefore $r_i$ and $r_j$ have an even number of 1’s in common, say $2x$. Then

\[
wt(v) = wt(r_i + r_j) = wt(r_i) + wt(r_j) - 2(2x),
\]
which is a multiple of four.

Suppose that any word that is a sum of \( t \) or fewer rows of \( G \) has weight that is a multiple of 4. Let \( v \in C_{24} \) be a sum of \( t + 1 \) rows of \( G \). Then \( v = r + w \), where \( r \) is a row of \( G \) and \( w \) is a sum of \( t \) rows of \( G \). By hypothesis \( wt(w) \) is a multiple of 4. As before \( r \) and \( w \) have an even number of 1’s in common, say \( 2y \), so that

\[
wt(v) = wt(r + w) = wt(r) + wt(w) - 2(2y),
\]

which is a multiple of 4. By induction any \( v \in C_{24} \) has a weight that is a multiple of 4.

(b) 

\[
d_{\min}(C_{24}) = \min_{\substack{v \in C_{24} \backslash \{0\}} } wt(v),
\]

\[
= 4 \text{ or } 8.
\]

The last step follows from the fact that \( G \) has rows of weight 8 and each of these rows are also codewords of \( C_{24} \).

(c) \( C_{24} \) has no codewords of weight 4:

Suppose \( v \in C_{24} \) has weight 4. We noted earlier that both \([I_{12} \mid B]\) and \([B \mid I_{12}]\) are generator matrices for \( C_{24} \). Therefore \( v = w_1[I_{12} \mid B] = w_2[B \mid I_{12}] \) for some \( w_1, w_2 \neq 0 \). Now neither of the two halves of \( v \) can be identically zero. This is because of the identity matrices in the generator matrices and also since \( w_1, w_2 \neq 0 \). Further, if either half of \( v \) contained only one 1, this would imply that \( v \) equalled a row of either \([I_{12} \mid B]\) or \([B \mid I_{12}]\), but each row has weight at least eight. Therefore each half of \( v \) must contain exactly two 1’s. This implies that \( wt(w_1) = wt(w_2) = 2 \), but the sum of two rows of \( B \) has weight at least 4. Therefore \( wt(v) = wt(w_1B) + wt(w_2B) > 2 + 4 > 4 \) — a contradiction.

Thus no \( v \in C_{24} \) has weight 4. Therefore \( d_{\min}(C_{24}) = 8 \).

### 6.2 The Golay Code: \( C_{23} \)

This section discusses the original Golay code that appeared in [?].

**Definition 6.2.1** (Puncturing a code). Puncturing a code means removing a digit (the same one) from every codeword.
Recall, that the extended Golay code, $C_{24}$, has generator matrix $[I_{12} \mid B]$, where
\[
B = \begin{bmatrix} B_1 & j^T \end{bmatrix}.
\]
The Golay code $C_{23}$ is obtained by removing the last digit from every codeword in $C_{24}$. Therefore it has the generator matrix $[I_{12} \mid B']$, where
\[
B' = \begin{bmatrix} B_1 & j \end{bmatrix}.
\]
Then $G$ is a $12 \times 23$ matrix with linearly independent rows. Therefore $C_{23}$ has length $n = 23$ and dimension $k = 12$.

Notice that the extended Golay code, $C_{24}$, is the extension (as defined previously) of the Golay code $C_{23}$ (or $C_{23}$ resulted from puncturing $C_{24}$). Since the extended code, $C_{24}$, has minimum distance 8, $C_{23}$ has minimum distance 7 or 8. Since $G$ has rows of weight 7, $d_{\text{min}}(C_{23}) = 7$. Thus $C_{23}$ is a $[23, 12, 7]$-code. This also means that $C_{23}$ is perfect (as the computation below shows) and so corrects all error patterns of weight at most 3 and no others (every word in $K^{23}$ is within distance 3 of some codeword in $C_{23}$).

\[
2^{12} = |C_{23}| = \frac{2^{23}}{{23 \choose 0} + {23 \choose 1} + {23 \choose 2} + {23 \choose 3}} = \frac{1 + 23 + 253 + 1771}{2^{23}} = \frac{23 + 253 + 1771}{2^{11}} = 2^{12}.
\]
Chapter 7

Reed-Muller Codes

The \( r \)th order Reed-Muller \([a, b] \) code (of length \( 2^m \)), denoted by \( RM(r, m) \), \( 0 \leq r \leq m \), is defined recursively as

1. \( RM(0, m) = \{000 \cdots \cdot 0, 11 \cdots \cdot 1\} \). Further, \( RM(m, m) = K^{2^m} \).

2. \( RM(r, m) = \{(x, x + y) \mid x \in RM(r, m - 1) \text{ and } y \in RM(r - 1, m - 1)\} \), where \( 0 < r < m \).

Example 7.0.2.

\[
\begin{align*}
RM(0, 0) & = \{0, 1\}. \\
RM(0, 1) & = \{00, 11\}. \\
RM(1, 1) & = \{00, 01, 10, 11\}. \\
RM(0, 2) & = \{0000, 1111\}. \\
RM(1, 2) & = \{(x, x + y) \mid x \in RM(1, 1) \text{ and } y \in RM(0, 1)\}, \\
& = \{(x, x + y) \mid x \in \{00, 01, 10, 11\} \text{ and } y \in \{00, 11\}\} \\
& = \{0000, 0011, 0101, 0110, 1010, 1001, 1111, 1100\}. \\
RM(2, 2) & = K^4.
\end{align*}
\]

Let \( G(r, m) \) denote the generator matrix for \( RM(r, m) \). These can be defined recursively.

1. 
\[
G(0, m) = [1 \ 1 \cdots \cdot 1].
\]
2. For $0 < r < m$,

$$G(r, m) = \begin{bmatrix} G(r, m - 1) & G(r, m - 1) \\ 0 & G(r - 1, m - 1) \end{bmatrix}.$$  

3.

$$G(m, m) = \begin{bmatrix} G(m - 1, m) \\ 00 \cdots 01 \end{bmatrix}.$$  

Example 7.0.3.

1.

$$G(0, 1) = [1 1].$$

2.

$$G(1, 1) = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$  

3.

$$G(1, 2) = \begin{bmatrix} G(1, 1) & G(1, 1) \\ 0 & G(0, 1) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$  

4.

$$G(2, 2) = \begin{bmatrix} G(1, 2) \\ 00 \cdots 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$  

5.

$$G(1, 3) = \begin{bmatrix} G(1, 2) & G(1, 2) \\ 0 & G(0, 2) \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}.$$
Theorem 7.0.4. The r’th order Reed-Muller code $RM(r, m)$ has the following properties

1. Length $n = 2^m$.
2. Minimum distance, $d_{\text{min}} = 2^{m-r}$.
3. Dimension,
   \[ k = \sum_{i=0}^{r} \binom{m}{i}. \]
4. For $r > 0$, $RM(r-1, m) \subseteq RM(r, m)$.
5. For $r < m$, $RM(r, m) \perp = RM(m-1-r, m)$.

**Proof.**

We only prove 1, 2 and 4. All proofs are by induction. From the previous example we see that all statements are true when $m = 0, 1, 2$.

1. When $r = 0$ or $r = m$ this follows from the definition. If $0 < r < m$ then $RM(r, m)$ is constructed from two Reed-Muller codes both of length $2^{m-1}$. Therefore $RM(r, m)$ will have length $2^{m-1} + 2^{m-1} = 2^m$.

4. The proof uses induction on $r+m$. We see from a previous example that $RM(0, 1) \subseteq R(1, 1)$, therefore the statement is true if $r + m \leq 2$. Assume that if $r + m < t$ and $r > 0$, then $RM(r-1, m) \subseteq RM(r, m)$ and consider the situation where $r + m = t$ and $r > 0$. There are three cases.

   (i) $r = 1$ (so that what we want is $RM(0, m) \subseteq RM(1, m)$).
   
   It follows from the recursive definition of $G(r, m)$ that its first row is all 1’s. Since $G(0, m) = [1 1 \cdots 1]$, this means that $G(0, m)$ is a sub-matrix (the first row) of $G(r, m)$. Therefore $RM(0, m) \subseteq RM(1, m)$.

   (ii) $r = m$.

   This is clear since $RM(m, m) = K2^m$, giving $RM(m-1, m) \subseteq RM(m, m)$.

   (iii) $1 < r < m$.

   We have

   \[
   G(r-1, m) = \begin{bmatrix}
   G(r-1, m-1) & G(r-1, m-1) \\
   0 & G(r-2, m-1)
   \end{bmatrix},
   \]
CHAPTER 7. REED-MULLER CODES

and

\[
G(r, m) = \begin{bmatrix}
G(r, m - 1) & G(r, m - 1) \\
0 & G(r - 1, m - 1)
\end{bmatrix}.
\]

By hypothesis \( G(r - 1, m - 1) \) is a sub-matrix of \( G(r, m - 1) \) and \( G(r - 2, m - 1) \) is a sub-matrix of \( G(r - 1, m - 1) \). Therefore \( G(r - 1, m) \) is a sub-matrix of \( G(r, m) \) and the result follows by induction.

2. Again the proof uses induction on \( r + m \). The result is clear if \( r + m = 0 \) or \( r + m = 1 \). Suppose the minimum distance of \( RM(s, t) = 2^t - s \) whenever \( s + t < l \) and consider the case \( r + m = l \) and the code \( RM(r, m) \). If \( r = 0 \) or \( r = m \), then the result is true, so assume \( 0 < r < m \). We know

\[
RM(r, m) = \{(x, x + y) \mid x \in RM(r, m - 1) \text{ and } y \in RM(r - 1, m - 1)\},
\]

and \( RM(r - 1, m - 1) \subseteq RM(r, m - 1) \) (by number 4). Therefore \( x + y \in RM(r, m - 1) \). If \( x \neq y \) then by the induction hypothesis \( wt(x) \geq 2^{m-1-r} \) and \( wt(x+y) \geq 2^{m-1-r} \). Thus \( wt(x, x+y) = wt(x) + wt(x+y) \geq 2^{m-1-r} + 2^{m-1-r} = 2^{m-r} \).

If \( x = y \), then \( x \in RM(r - 1, m) \) and \( (x, x + y) = (x, 0) \). Therefore \( wt(x, x + y) = wt(x) \geq 2^{m-1-(r-1)} = 2^{m-r} \). This shows that \( d_{\text{min}} \geq 2^{m-r} \). Equality can be seen to hold by choosing \( x \) and \( y \) so that equality holds in \( RM(r, m - 1) \) and \( RM(r - 1, m - 1) \) (i.e. choosing them to be the minimum weight words in their respective codes).

7.1 The Reed-Muller Codes \( RM(1, m) \)

These codes have the following properties

- Length \( n = 2^m \).
- Minimum distance \( d_{\text{min}} = 2^{m-1} \).
- Dimension \( \binom{m}{0} + \binom{m}{1} = m + 1 \), so that the code contains \( 2^{m+1} \) codewords.

Therefore it is a \([2^m, m + 1, 2^{m-1}]\)-code.

As always when receiving a word \( w \) the goal is to find a codeword closest to \( w \) if any such a codeword exists. The general principles that apply to the \( RM(1, m) \) codes in this situation are illustrated by the following example.

**Example 7.1.1.** \( RM(1, 3) \) is an \([8, 4, 4]\)-code. Assume that \( w \) is received. If we can find a codeword \( v \) such that \( d(v, w) < 2 \) we decode \( w \) to \( v \). Also, if \( u \) is a codeword with
7.1. THE REED-MULLER CODES $RM(1, M)$

$d(u, w) > 6$, then $d(w, u + 11111111) < 2$ and so $w$ may be decoded to $u + 11111111$. Therefore no more than half the codewords need to be examined to find a codeword closest to $w$, if one exists. Note that 11111111 is always a codeword so that $u + 11111111$ is a codeword if $u$ is a codeword.

■
Chapter 8

Decimal Codes

8.1 The ISBN Code

The International Standard Book Number is the number that is displayed on the back of books that uniquely identifies the book. An example of such a number is: 0 – 201 – 34292 – 8. It is always made up of 10 digits, say $x_1x_2x_3 \cdots x_{10}$, with the last digit being a parity check digit. The first 9 digits is divided into three groups as follows.

**Group Identifier**
This identifies the group of countries where the book was published. It has more digits if the group produces fewer books. As an example $x_1 = 0$ identifies the English speaking countries, $x_1 = 3$ the German speaking countries and $x_1x_2 = 87$ Denmark.

**Publisher Prefix**
This is anywhere from 2 to 7 digits and identifies the publisher.

**Title Number**
It is 1 to 6 digits in length and is assigned by the publisher.

**Parity Check**
The tenth digit, $x_{10}$, is chosen so that

$$
1x_1 + 2x_2 + 3x_3 + \cdots + 9x_9 + 10x_{10} \equiv 0 \pmod{11}
\equiv -10x_{10} \pmod{11}
\equiv 1x_{10} \pmod{11}.
$$
If $x_{10} = 10$, then $X$ is used in the ISBN.

**Example 8.1.1.** Given $0 - 201 - 34292 - 8$ check that it is a valid ISBN. To do this we compute

\[
1 \cdot 0 + 2 \cdot 2 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 3 + 6 \cdot 4 + 7 \cdot 2 + 8 \cdot 9 + 9 \cdot 2 + 10 \cdot 8 \\
= 0 + 4 + 0 + 4 + 15 + 24 + 14 + 72 + 18 + 80 \\
\equiv 4 + 4 + 2 + 3 + 6 + 7 + 3 \pmod{11} \\
\equiv 0 \pmod{11}.
\]

Therefore it is a valid ISBN. ■

The ISBN code has the following properties.

- The ISBN code detects all single errors.
  Suppose the ISBN is $x_1x_2 \cdots x_{10}$, but that $x_i$ is recorded wrong as $x_i + e$, $e \neq 0$. The sum then becomes

\[
1 \cdot x_1 + 2 \cdot x_2 + \cdots + 10x_{10} + i \cdot e, \\
\equiv 0 + i \cdot e \pmod{11}.
\]

Suppose $i \cdot e \equiv 0 \pmod{11}$. This implies that $11 | i \cdot e$, so that $11 | i$ or $11 | e$, but neither $i$ nor $e$ are congruent to 0 mod 11. Therefore we will have a nonzero sum if an error occurred.

- It is possible to correct an error if you know which digit is in error. We show this by way of an example.

**Example 8.1.2.** Find $x$ if $01383x0943$ is an ISBN. Since it is an ISBN we know that

\[
1 \cdot 0 + 2 \cdot 1 + 3 \cdot 3 + 4 \cdot 8 + 5 \cdot 3 + 6 \cdot x + 7 \cdot 0 + 8 \cdot 9 + 9 \cdot 4 + 10 \cdot 3, \\
\equiv 6x + 9 \equiv 0 \pmod{11}, \\
\therefore 6x \equiv -9 \equiv 2 \pmod{11}, \\
x \equiv 4 \pmod{11}.
\]

- The ISBN code detects any single error arising from transposing two digits (adjacent or not).
8.2. A SINGLE ERROR CORRECTING DECIMAL CODE

Let \( x_1x_2 \cdots x_{10} \) be the correct ISBN number. Assume that \( x_i \) and \( x_j \) get interchanged in the recording of the number (\( i < j \)). The check sum becomes

\[
1 \cdot x_1 + 2 \cdot x_2 + \cdots + ix_j + \cdots + jx_i + \cdots + x_{10},
\]

\[
= 1x_1 + 2x_2 + \cdots + ix_i + \cdots + jx_j + \cdots + 10x_{10} + i(x_j - x_i) + j(x_i - x_j),
\]

\[
\equiv 0 + j(x_i - x_j) - i(x_i - x_j) \pmod{11},
\]

\[
\equiv (x_i - x_j)(j - i) \pmod{11}.
\]

We are implicitly assuming that \( x_i \neq x_j \) (otherwise no error occurs) and furthermore \( j - i \neq 0 \). Therefore \((x_i - x_j)(j - i)\) is a product of two nonzero numbers in \( \mathbb{Z}_{11} \) and so is not equal to zero (since \( \mathbb{Z}_{11} \) is a field).

8.2 A Single Error Correcting Decimal Code

Definition 8.2.1 (Distance). If \( u \) and \( v \) are words in any code (binary or otherwise) then the distance between \( u \) and \( v \), \( d(u, v) \), is the number of places where \( u \) and \( v \) differ.

The code that we are considering in this section has the following properties.

- It is made up of 10 digits : \( x_1x_2 \cdots x_9x_{10} \).
- \( x_1 \) through \( x_8 \) are information digits.
- \( x_9 \) and \( x_{10} \) are parity check digits chosen such that

\[
S_1 = \sum_{i=1}^{10} ix_i \equiv 0 \pmod{11},
\]

\[
S_2 = \sum_{i=1}^{10} x_i \equiv 0 \pmod{11}.
\]

The calculation of the parity digits is illustrated in the following example.

Example 8.2.2. Let \( x_1x_2 \cdots x_8 \) be 02013429. We need to choose \( x_9 \) and \( x_{10} \) such that

\[
S_1 = 1 \cdot 0 + 2 \cdot 2 + 3 \cdot 0 + 4 \cdot 1 + 5 \cdot 3 + 6 \cdot 4 + 7 \cdot 2 + 8 \cdot 9 + 9x_9 + 10x_{10} \equiv 0 \pmod{11},
\]

\[
S_2 = 0 + 2 + 0 + 1 + 3 + 4 + 2 + 9 + x_9 + x_{10} \equiv 0 \pmod{11}.
\]

This simplifies to

\[
9x_9 + 10x_{10} \equiv 10 \pmod{11},
\]

\[
x_9 + x_{10} \equiv 1 \pmod{11}.
\]
Adding the two equation together we get \(10x_9 \equiv 0 \pmod{11}\), which implies that \(x_9 = 0\). From this we get \(x_{10} = 1\).

So in general we will find that

\[
9x_9 + 10x_{10} \equiv a \pmod{11}, \\
x_9 + x_{10} \equiv b \pmod{11}.
\]

Adding the equations together leads to

\[
10x_9 \equiv (a + b) \pmod{11}, \\
\therefore x_9 \equiv 10(a + b) \pmod{11}, \\
\therefore x_{10} \equiv a + 2b \pmod{11}.
\]

Suppose now that \(x_1x_2\cdots x_{10}\) is sent and a single error \(e \neq 0\) occurs in the \(i\)'th digit. Then the received word becomes \(x_1x_2\cdots x_{i-1}(x_i + e)x_{i+1}\cdots x_{10}\). Therefore

\[
S_1 = 1x_12x_2 + \cdots ix_i + \cdots 10x_{10} + ie \equiv 0 + ie \pmod{11}, \\
S_2 = x_1 + x_2 + \cdots + x_i + \cdots + 10x_{10} + e \equiv 0 + e \pmod{11}.
\]

So, the second equation gives the magnitude of the error, \(e\). Once we know this we can also calculate the position of the error: \(i \equiv e^{-1}S_1 \equiv S_2^{-1}S_1\).

Therefore the decoding may be summed up as follows.

1. If the received word is \(r\), then compute the syndrome,

\[
rH = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}, \pmod{11},
\]

where

\[
H = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ \vdots & \vdots \\ 10 & 1 \end{bmatrix}.
\]

2. If \(S_1 = S_2 = 0\) then assume \(r\) is a codeword.

3. If \(S_1 \neq 0\) and \(S_2 \neq 0\) then assume an error of magnitude \(e \equiv S_2 \pmod{11}\) has occurred in location \(i \equiv S_2^{-1}S_1 \pmod{11}\) and decode \(r\) as \(x_1x_2\cdots x_{i-1}(x_i - e)x_{i+1}\cdots x_{10}\).
4. If $S_1 = 0$ and $S_2 \neq 0$ or $S_1 \neq 0$ and $S_2 = 0$, then at least two errors have occurred, so request retransmission.

Note that point 4 above always occurs when two different digits are transposed ($S_1 \neq 0$ and $S_2 = 0$). Therefore this code can detect all errors involving the transposition of two digits.

### 8.3 A Double Error Correcting Decimal Code

This code has the following properties.

- Length $n = 10$.
- Information digits are the first 6 digits, $x_1x_2, \ldots x_6$.
- The last four digits, $x_7x_8x_9x_{10}$, are the parity check digits. They are chosen so that

$$
S_1 = \sum_{i=1}^{10} ix_i \equiv 0 \pmod{11},
$$

$$
S_2 = \sum_{i=1}^{10} x_i \equiv 0 \pmod{11},
$$

$$
S_3 = \sum_{i=1}^{10} i^2x_i \equiv 0 \pmod{11},
$$

$$
S_4 = \sum_{i=1}^{10} i^3x_i \equiv 0 \pmod{11},
$$

The digits $x_1$ through $x_6$ will be known since they are information digits. Therefore the equations above will assume the form

$$
7x_7 + 8x_8 + 9x_9 + 10x_{10} \equiv a \pmod{11},
$$

$$
x_7 + x_8 + x_9 + x_{10} \equiv b \pmod{11},
$$

$$
7^2x_7 + 8^2x_8 + 9^2x_9 + 10^2x_{10} \equiv c \pmod{11},
$$

$$
7^3x_7 + 8^3x_8 + 9^3x_9 + 10^3x_{10} \equiv d \pmod{11}.
$$
Here $a, b, c$ and $d$ are elements of $\mathbb{Z}_{11}$ with

\[
\begin{align*}
    a & \equiv - \sum_{i=1}^{i=6} ix_i \pmod{11}, \\
    b & \equiv - \sum_{i=1}^{i=6} x_i \pmod{11}, \\
    c & \equiv - \sum_{i=1}^{i=6} i^2 x_i \pmod{11}, \\
    d & \equiv - \sum_{i=1}^{i=6} i^3 x_i \pmod{11}.
\end{align*}
\]

Suppose that $x_1 x_2 \cdots x_{10}$ are sent and that a single error occurs in location $i$ and is of size $e \neq 0$. This situation is exactly the same as in the previous section and we will be able to correct this single error if we know that only this error occurred. It turns out that it is indeed possible to detect the presence of only one error. To do this we examine the general case of two errors first.

Suppose therefore that two errors occur in locations $i$ and $j$ with sizes $e_1 \neq 0$ and $e_2 \neq 0$. The presence of these errors affects the parity check equations above as follows.

\[
\begin{align*}
    S_1 & \equiv ie_1 + je_2 \pmod{11}, \\
    S_2 & \equiv e_1 + e_2 \pmod{11}, \\
    S_3 & \equiv i^2 e_1 + j^2 e_2 \pmod{11}, \\
    S_4 & \equiv i^3 e_1 + j^3 e_2 \pmod{11}.
\end{align*}
\]

It can be shown that $i$ and $j$ are the roots (in $\mathbb{Z}_{11}$) of

\[ax^2 + bx + c,\]

where

\[
\begin{align*}
    a &= S_1^2 - S_2 S_3, \\
    b &= S_2 S_4 - S_1 S_3, \\
    c &= S_3^2 - S_1 S_4.
\end{align*}
\]

This quadratic can be solved using a formula analogous to the one used for real numbers. The derivation of this formula is done below. Note that all calculations are in $\mathbb{Z}_{11}$ and
8.3. A DOUBLE ERROR CORRECTING DECIMAL CODE

that the square-root used here may not exist. The root represents the number in \( \mathbb{Z}_{11} \) that
upon squaring yields the number under the root sign.

\[
ax^2 + bx + c = 0, \quad a \neq 0,
\]
\[
a^{-1}ax^2 + a^{-1}bx + a^{-1}c = 0,
\]
\[
x^2 + a^{-1}bx + (2^{-1}a^{-1}b)^2 - (2^{-1}a^{-1}b)^2 = -a^{-1}c,
\]
\[
(x + 2^{-1}a^{-1}b)^2 = (2^{-1}a^{-1}b)^2 - a^{-1}c,
\]
\[
x + 2^{-1}a^{-1}b = \pm \sqrt{2^{-2}a^{-2}b^2 - a^{-1}c},
\]
\[
x = -2^{-1}a^{-1}b \pm 2^{-1}a^{-1}\sqrt{b^2 - (a^{-1}c)(2^{-2}a^{-2})^{-1}},
\]
\[
x = [-b \pm \sqrt{b^2 - 4ac}(2a)^{-1}]
\]

So, using this formula we can find \( i \) and \( j \) and then solve

\[
S_1 \equiv ie_1 + je_2,
\]
\[
S_2 \equiv e_1 + e_2,
\]

for \( e_1 \) and \( e_2 \).

The decoding can be summed up as follows. Let

\[
H = \begin{bmatrix}
1 & 1 & 1^2 & 1^3 \\
2 & 1 & 2^2 & 2^3 \\
3 & 1 & 3^2 & 3^3 \\
4 & 1 & 4^2 & 4^3 \\
\vdots & \vdots & \vdots & \vdots \\
10 & 1 & 10^2 & 10^3 \\
\end{bmatrix}
\]

If \( r \) is the received word then

1. Compute the syndrome

\[
rH = \begin{bmatrix}
S_1 \\
S_2 \\
S_3 \\
S_4 \\
\end{bmatrix} = S.
\]

2. If \( S = 0 \) then a codeword was received.

3. If \( S \neq 0 \) and \( a = b = c = 0 \) then a single error occurred in digit \( i = S_1S_2^{-1} \) and with
   size \( e = S_2 \). This may be seen to hold be realizing that a single error implies that
   \( S_1 = ie_1, S_2 = e_1, S_3 = i^2e_1 \) and \( S_4 = i^3e_1 \).
CHAPTER 8. DECIMAL CODES

4. If $S \neq 0$ and $a \neq 0, c \neq 0$ and $b^2 - 4ac$ is a square in $\mathbb{Z}_{11}$, then there are two errors $e_1$ and $e_2$ in locations $i$ and $j$ as before.

5. Otherwise, at least three errors are detected, so retransmit.

**Example 8.3.1.** Suppose the received word is 3254571396. Then

$$
S_1 = 1 \cdot 3 + 2 \cdot 2 + 3 \cdot 5 + 4 \cdot 4 + \cdots + 10 \cdot 6 \equiv 2 \pmod{11}, \\
S_2 = 3 + 2 + 5 + \cdots + 6 \equiv 1 \pmod{11}, \\
S_3 = 1^2 \cdot 3 + 2^2 \cdot 2 + \cdots 10^2 \cdot 6 \equiv 10 \pmod{11}, \\
S_4 = 1^3 \cdot 3 + 2^3 \cdot 2 + \cdots + 10^3 \cdot 6 \equiv 3 \pmod{11}.
$$

So that

$$
a = S_1^2 - S_2S_3 = 2^2 - 1 \cdot 10 \equiv 5 \pmod{11}, \\
b = S_2S_4 - S_1S_3 = 1 \cdot 3 - 2 \cdot 10 \equiv 5 \pmod{11}, \\
c = S_3^2 - S_1S_4 = 10^2 - 2 \cdot 3 \equiv 6 \pmod{11}.
$$

Then $b^2 - 4ac = 5^2 - 4 \cdot 5 \cdot 6 = 25 - 120 \equiv 4$ is a square. Thus

$$
i, j = (-5 \pm 2)(2 \cdot 5)^{-1}, \\
\quad = (-5 \pm 2)10 \text{ since } 10^{-1} = 10, \\
\quad = 3, 7.
$$

Now

$$
S_1 = 3e_1 + 7e_2 \equiv 2 \pmod{11}, \\
S_2 = e_1 + e_2 \equiv 1 \pmod{11}.
$$

Adding 4 times $S_2$ to $S_1$ gives $7e_1 \equiv 6$, implying that $e_1 = 4$ and $e_2 = 8$. Thus the decoded word is 3214574396.

8.4 The Universal Product Code (UPC)

This code has the following properties

- All codewords have length 12: $x_1x_2x_3 \cdots x_{12}$.
- The first digit specifies the type of product.
8.5. US MONEY ORDER

- $x_2x_3 \cdots x_6$ specifies the manufacturer.
- $x_7 \cdots x_{11}$ is a product number.
- The check digit, $x_{12}$, is determined by

$$3(x_1 + x_3 + x_5 + x_7 + x_9 + x_{11}) + x_2 + x_4 + x_6 + x_8 + x_{10} + x_{12} \equiv 0 \pmod{10}.$$  

Example 8.4.1. A 50 cents off Kellogg’s Rice Krispies coupon has UPC 53800051150$x_{12}$. Therefore

$$3(5 + 8 + 0 + 5 + 1 + 0) + 3 + 0 + 0 + 1 + 5 + x_{12} \equiv 7 + 9 + x_{12} \equiv 0 \pmod{10}.$$ 

Thus $x_{12} = 4$.  

Suppose that a single error of size $e \neq 0$ occurs in the $i$’th digit. The check sum either changes by $e$ or by $3e$ depending on $i$ being even or odd. Since $\gcd(10, 3) = 1$, $10 \mid 3e \iff 10 \mid e$. Further $1 \leq e \leq 9$, so that 10 does not divide $e$. Thus the error is detectable since we have a nonzero checksum. Note that the error cannot be corrected unless you know which digit is in error.

If adjacent digits $x_i$ and $x_j$ are transposed then the change to the check sum is $-3x_i + x_i - x_j + 3x_j = 2(x_j - x_i)$. This type of error is detected unless $(x_j - x_i) = \pm 5$.

8.5 US Money Order

The code employed for US money orders has the following properties.

- The codewords have length 11: $x_1x_2 \cdots x_{11}$.
- The check digit, $x_{12}$, is defined by

$$x_1x_2x_3 \cdots x_{10} \equiv x_{11} \pmod{9}.$$ 

Here $x_1x_2x_3 \cdots x_{10}$ represents the number with the digits $x_1, x_2, \ldots, x_{10}$ and is not the product of the numbers $x_1, x_2, \ldots, x_{10}$. Also $0 \leq x_{11} \leq 8$.

- The check equation is equivalent to

$$x_1 + x_2 + x_3 + \cdots + x_{10} \equiv x_{11} \pmod{9}.$$ 

The proof of this is in assignment 3 in the appendix.
Should an error occur in $x_{11}$ then it will be detected since $x_1 + x_2 + x_3 + \cdots x_{10}$ will not be equivalent to $x_{11} + e \pmod{9}$.

If an error occurs in the first 10 digits then it will be detected unless a 0 is changed into a 9 or 9 is changed into a 0. To estimate the probability of this we proceed as follows. The probability that the $i$'th digit is a 0 is $10^9/10^{10} = 0.1$. The probability of the error being a 9 is 1/9. Therefore the probability of changing a 0 into a 9 is 1/90. Similarly we find that the probability of changing a 9 into a 0 is also 1/90. So, the probability of an error occurring in the $i$'th $(1 \leq i \leq 10)$ digit is 2/90.

### 8.6 US Postal Code

The code used by the US postal service is the following.

- Each codeword is of length 10: $x_1x_2\cdots x_{10}$.
- The code is printed in a bar-coded format on the envelope. The bar-code is delimited by two long bars on either side. Each decimal digit is represented by a group of 5 long and short bars.
- Each such group of 5 bars has exactly 2 long and 3 short bars. Thus there are $\binom{5}{2} = 10$ such bars — one for each decimal digit.
- We can think of a long bar as representing a 1 and a short bar representing a 0. Then the correspondence between the bars and the decimal digits is given by: if $abcde$ is the binary sequence of bars then $7a + 4b + 2c + d$ is the decimal digit associated with it. In tabular form this is

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>00011</td>
<td>1</td>
</tr>
<tr>
<td>00101</td>
<td>2</td>
</tr>
<tr>
<td>00110</td>
<td>3</td>
</tr>
<tr>
<td>01001</td>
<td>4</td>
</tr>
<tr>
<td>01010</td>
<td>5</td>
</tr>
<tr>
<td>01100</td>
<td>6</td>
</tr>
<tr>
<td>10001</td>
<td>7</td>
</tr>
<tr>
<td>10010</td>
<td>8</td>
</tr>
<tr>
<td>10100</td>
<td>9</td>
</tr>
<tr>
<td>11000</td>
<td>0</td>
</tr>
</tbody>
</table>
Therefore the following sequence of bars $\text{1111}$ represents 10010 which in turn represents 8.

- A parity check digit $x_{10}$ is added to the bar-code so that $x_1 + x_2 + \cdots x_{10} \equiv 0 \pmod{10}$.

Suppose that the scanner makes a single error when reading the code (i.e. it reads a 1 as 0 or a 0 as 1). Then some block of length 5 doesn’t have 2 ones — so the block itself can be identified. If one error occurs in the $i$’th block, then the decimal digit associated with it will be undefined. This digit can be recovered from the parity check equation since we now know which digit is in error.
Chapter 9

Hadamard Codes

Suppose we want a 2-error correcting code, \( C \), of length 11. Then according to the Hamming bound we have

\[
|C| \leq \binom{11}{0} + \binom{11}{1} + \binom{11}{2} \approx 30.57.
\]

Therefore \( C \) can have at most 30 codewords. Since \( 2^4 < 30 < 2^5 \), the largest binary, linear 2-error correcting code of length 11 will have at most \( 2^4 = 16 \) codewords.

On the other hand there exists a nonlinear code of length 11 that is 2-error correcting and that has 24 codewords — a definite improvement. This is a Hadamard code and its construction is presented at the end of this chapter. To be able to construct these codes we need a fair amount of information on Hadamard matrices. After we have constructed these matrices we use their rows as the codewords for the Hadamard code.

**Definition 9.0.1 \((n, M, d)-code\).** An \((n, M, d)-code\) is a blockcode (linear or nonlinear) that has length \( n \), \( M \) codewords and minimum distance \( d \).

### 9.1 Background

**Definition 9.1.1 (Hadamard Matrix).** A Hadamard matrix, \( H \), of order \( n \) is an \( n \times n \) matrix of +1’s and −1’s such that

\[
HH^T = nI.
\]

Therefore any two distinct rows of a Hadamard matrix are orthogonal (using the real inner product). Furthermore, multiplying any row or column of a Hadamard matrix by −1
changes it into another Hadamard matrix. This allows us to put a given Hadamard matrix into the so-called normalised form where the first row and first column only contains 1’s.

Example 9.1.2. Normalised Hadamard matrices of orders 1, 2, 4 and 8 are shown below. Note that +1’s are only shown as +, while −1 are shown as −.

\[
\begin{align*}
\text{n = 1} & : H_1 = [+] , \\
\text{n = 2} & : \begin{bmatrix}
+ & + \\
+ & -
\end{bmatrix} , \\
\text{n = 4} & : H_4 = \begin{bmatrix}
+ & + & + & + \\
+ & - & - & - \\
+ & + & - & - \\
+ & - & + & -
\end{bmatrix} , \\
\text{n = 8} & : H_8 = \begin{bmatrix}
+ & + & + & + & + & + & + & + \\
+ & - & - & - & - & - & - & - \\
+ & + & - & - & - & - & - & - \\
+ & - & + & - & - & - & - & - \\
+ & + & + & - & - & - & - & - \\
+ & - & + & - & - & - & - & -
\end{bmatrix} .
\end{align*}
\]

\[\blacksquare\]

Theorem 9.1.3. If a Hadamard matrix of order \(n\) exists, then \(n\) is 1, 2 or a multiple of 4.

Proof. By the previous example we know that Hadamard matrices of order 1 and 2 exist.

Let \(H\) be a Hadamard matrix of order \(n > 2\). Put \(H\) in normalised form — this does not affect \(n\), therefore the first row of \(H\) is made up entirely of +1’s. The innerproduct of the first row with any other row has to be zero, but this innerproduct is just the other row. Therefore all rows of \(H\) (except the first) have an equal amount of +1’s and −1’s. By permuting the columns of \(H\) we can change the second row into one that has as its first \(n/2\) entries +1’s and its second \(n/2\) entries all −1’s. Therefore the first two rows of \(H\) are as follows.

\[
\begin{bmatrix}
\underbrace{+ + \cdots +}_{n/2} \\
\underbrace{+ + \cdots +}_{n/2}
\end{bmatrix} \quad \begin{bmatrix}
\underbrace{+ + \cdots +}_{n/2} \\
\underbrace{- - \cdots -}_{n/2}
\end{bmatrix}
\]

Let \(u\) be any row of \(H\) except the first two. Again \(u\) has \(n/2\) +1’s and \(n/2\) −1’s. Say \(u\) has \(x\) +1’s in its first \(n/2\) positions (so that it has \((n/2 - x)\) −1’s in its first \(n/2\) positions)
and \( y \) +1’s in its second \( n/2 \) positions (and therefore \( (n/2 - y) -1 \)'s in its second \( n/2 \) positions). Since the innerproduct between \( u \) and the second row is zero we have

\[
x - (n/2 - x) - y + (n/2 - y) = 2x - 2y = 0,
\]

\[
\therefore x - y = 0.
\]

Further the innerproduct between \( u \) and the first row is also zero and this leads to

\[
x - (n/2 - x) + y - (n/2 - y) = 2x + 2y - n = 0,
\]

\[
\therefore x + y = n/2.
\]

Adding these two equations we find \( 2x = n/2 \) or that \( n = 4x \).

We now present two constructions of Hadamard matrices that prove to be useful from a coding theory standpoint.

**Construction 1**

If \( H_n \) is a Hadamard matrix of order \( n \), then

\[
H_{2n} = \begin{bmatrix}
H_n & H_n \\
H_n & -H_n
\end{bmatrix},
\]

is a Hadamard matrix of order \( 2n \).

To see that this construction works note that

\[
H_{2n}H_{2n}^T = \begin{bmatrix}
H_n & H_n \\
H_n & -H_n
\end{bmatrix} \begin{bmatrix}
H_n^T & H_n^T \\
H_n^T & -H_n^T
\end{bmatrix},
\]

\[
= \begin{bmatrix}
H_nH_n^T + H_nH_n^T & H_nH_n^T - H_nH_n^T \\
H_nH_n^T - H_nH_n^T & H_nH_n^T + H_nH_n^T
\end{bmatrix},
\]

\[
= \begin{bmatrix}
2nI & 0 \\
0 & 2nI
\end{bmatrix} = 2nI.
\]

Starting with \( H_1 = [1] \), this gives \( H_2, H_4, H_8, \ldots \). These are Hadamard matrices of orders a power of two and are known as *Sylvester matrices*.

For the second construction we need some facts about quadratic residues.

**Definition 9.1.4 (Quadratic residue).** Let \( p \) be an odd prime. The nonzero squares modulo \( p: 1^2, 2^2, 3^2, \ldots \) reduced \((\text{mod } p)\), are called quadratic residues \((\text{mod } p)\) (or just residues \((\text{mod } p)\)).

To find the residues \((\text{mod } p)\) it is enough to consider

\[
1^2, 2^2, \ldots, \left(\frac{p-1}{2}\right)^2 \pmod{p},
\]
since \((p-a)^2 = p^2 - 2pa + a^2 \equiv a^2 \pmod{p}\). These are all distinct for if \(i^2 \equiv j^2 \pmod{p}\) with \(1 \leq i, j \leq (p-1)/2\) then \((i-j)(i+j) = i^2 - j^2 \equiv 0 \pmod{p}\). That is \(p|(i-j)\) or \(p|(i+j)\). In the first case \(i \equiv j \pmod{p}\) and in the latter case \(i \equiv -j \pmod{p}\), but \((p-1)/2 + 1 \leq -j \leq p-1\) and \(1 \leq i \leq (p-1)/2\). So \(i^2 \equiv j^2 \pmod{p}\) is only possible if \(i \equiv j \pmod{p}\). Therefore there are \((p-1)/2\) quadratic residues \(\pmod{p}\). The remaining \((p-1)/2\) numbers are called nonresidues.

**Example 9.1.5.** For \(p = 11\), the quadratic residues are

\[
\begin{align*}
1^1 &= 1, \\
2^2 &= 4, \\
3^2 &= 9, \\
4^2 &= 16 \equiv 5 \pmod{11} \text{ and} \\
5^2 &= 25 \equiv 3 \pmod{11}.
\end{align*}
\]

We need the following properties of quadratic residues.

1. The product of two quadratic residues or of two nonresidues is a residue, while the product of a residue and a nonresidue is a nonresidue.

2. If \(p\) is of the form \(4k+1\), \(-1\) is a quadratic residue \(\pmod{p}\). If \(p\) is of the form \(4k+3\), \(-1\) is a nonresidue \(\pmod{p}\).

3. We define the following function on \(\mathbb{Z}_p\), also known as the Legendre symbol (but we use a different notation to simplify later expressions).

\[
\chi(i) = \begin{cases} 
0 & \text{if } i \equiv 0 \pmod{p}, \\
1 & \text{if } i \text{ is a residue,} \\
-1 & \text{if } i \text{ is a nonresidue.}
\end{cases}
\]

Note that \(\chi(x)\chi(y) = \chi(xy)\), where \(x, y \in \mathbb{Z}_p\).

We also need the following theorem.

**Theorem 9.1.6.** For \(c \in \mathbb{Z}_p\) and \(c \neq 0\),

\[
\sum_{b=0}^{p-1} \chi(b)\chi(b+c) = -1.
\]
Construction 2 (The Paley Construction)

This construction produces a Hadamard matrix of order \( n = p + 1 \), where \( p \) is a prime of the form \( 4k + 3 \) (i.e. \( n \) is a multiple of four).

(i) First construct the Jacobsthal matrix \( Q = (q_{ij}) \). This is a \( p \times p \) matrix whose rows and columns are labelled \( 0, 1, 2, \ldots, p - 1 \) and \( q_{ij} = \chi(j - i) \).

Note that \( q_{ij} = \chi(j - i) = \chi(-1)\chi(i - j) = -q_{ji} \) since \( p \) is of the form \( 4k + 3 \) and by property 2 \(-1\) is a nonresidue. That is \( Q \) is skew-symmetric: \( Q^T = -Q \).

(ii) We need the following Lemma.

**Lemma 9.1.7.** \( QQ^T = pI - J \) and \( QJ = JQ = 0 \), where \( J \) is the matrix all of whose entries are 1.

(iii) Let

\[
H = \begin{bmatrix} 1 & j \\ j^T & Q - I \end{bmatrix},
\]

where \( j \) is a vector of all 1's. Therefore \( H \) is a \( (p + 1) \times (p + 1) \) matrix. We now have

\[
HH^T = \begin{bmatrix} 1 & j \\ j^T & Q - I \end{bmatrix} \begin{bmatrix} 1 & j \\ j^T & Q^T - I \end{bmatrix} = \begin{bmatrix} (p + 1) & 0 \\ 0 & J + (Q - I)(Q^T - I) \end{bmatrix}.
\]

From the Lemma it follows that

\[
J + (Q - I)(Q^T - I) = J + QQ^T - QI - IQ^T + I^2,
\]

\[
= J + pI - J - Q - 0 + I,
\]

\[
= pI - Q + Q^T + I,
\]

\[
= (p + 1)I - Q + Q^T,
\]

\[
= (p + 1)I + Q^T - Q^T = (p + 1)I.
\]

Therefore

\[
HH^T = \begin{bmatrix} (p + 1) & 0 \\ 0 & (p + 1)I \end{bmatrix} = (p + 1)I.
\]

Thus \( H \) is a normalised Hadamard matrix of order \( p + 1 \) which is said to be of Paley type.
9.2 Definition of the Codes

We now (finally) come to the definition of the codes themselves.

Let $H_n$ be a normalised Hadamard matrix of order $n$. Replace $+1$’s by 0’s and $-1$’s by 1’s, in $H_n$. Then $H_n$ is changed into the binary Hadamard matrix $A_n$. Since the rows of $H_n$ are orthogonal, any two rows of $H_n$ (and therefore of $A_n$) agree in $n/2$ places and also differ in $n/2$ places. Therefore any two rows of $A_n$ are a Hamming distance $n/2$ apart.

$A_n$ gives rise to three Hadamard codes.

1. An $(n - 1, n, n/2)$-code, $A_n$, consisting of the rows of $A_n$ with the first column deleted.

2. An $(n - 1, 2n, (n/2) - 1)$-code, $B_n$, consisting of $A_n$ together with the complements of all its codewords.

3. An $(n, 2n, n/2)$-code $C_n$ consisting of the rows of $A_n$ and their complements.

Example 9.2.1. Using a $12 \times 12$ Hadamard matrix (constructed using Paley’s method) and deleting the first column we find the following codes.

\[
\begin{array}{cccccccccccc}
0&0&0&0&0&0&0&0&0&0&0&0 \\
1&1&0&1&1&1&0&0&0&1&0&0 \\
0&1&1&0&1&1&1&0&0&0&1&0 \\
1&0&1&1&0&1&1&1&0&0&0&0 \\
0&1&0&1&1&0&1&1&1&0&0&0 \\
0&0&1&0&1&1&0&1&1&1&0&0 \\
0&0&0&1&0&1&1&1&0&1&1&1 \\
1&0&0&0&1&1&1&1&0&1&1&1 \\
1&1&0&0&0&1&0&1&1&0&1&1 \\
1&1&1&0&0&0&1&0&1&1&0&1 \\
0&1&1&1&0&0&0&1&0&1&1&1 \\
1&0&1&1&1&0&0&0&1&0&1&1 \\
0&0&1&0&0&0&1&1&1&0&1&1 \\
1&0&0&1&0&0&0&1&1&1&0&1 \\
1&1&0&1&0&0&1&1&1&0&1&1 \\
1&1&1&0&1&0&0&1&1&1&0&1 \\
0&1&1&1&0&1&0&0&1&0&0&0 \\
0&0&1&1&1&0&1&0&0&1&0&0 \\
0&0&0&1&1&1&0&1&0&0&1&0 \\
1&0&0&0&1&1&1&0&1&0&0&1 \\
0&1&0&0&0&1&1&1&0&1&0&1 \\
1&1&1&1&1&1&1&1&1&1&1&1 \\
\end{array}
\]
9.3. HOW GOOD ARE THE HADAMARD CODES?

The first twelve rows form the \((11, 12, 6)\) Hadamard code \(A_{12}\) (this is the code referred to at the start of the chapter). All 24 rows form the \((11, 24, 5)\) Hadamard code \(B_{12}\). □

9.3 How good are the Hadamard Codes?

In trying to determine how good the Hadamard codes are we turn to the following bound known as Plotkin’s bound.

**Theorem 9.3.1 (Plotkin bound).** For any \((n, M, d)\) code \(C\) with \(d_{\text{min}} = d > n/2\) we have

\[
M \leq 2 \left\lfloor \frac{d}{2d - n} \right\rfloor.
\]

**Proof.**

We will assume throughout that \(M \geq 2\), which is what is needed for a code to be useful.

Consider the sum

\[
S = \sum_{u \in C} \sum_{v \in C} d(u, v).
\]

Since \(d(u, v) \geq d\) if \(u \neq v\), \(S \geq M(M - 1)d\).

Let \(A\) be the \(M \times n\) matrix whose rows are the codewords of \(C\). Suppose the \(i\)'th column contains \(x_i\) 0's and \(M - x_i\) 1's. This column contributes \(2x_i(M - x_i)\) to \(S\). Therefore

\[
S = \sum_{i=1}^{n} 2x_i(M - x_i).
\]

Now \(x_i(M - x_i)\) is maximised if \(x_i = M/2\), so \(S\) is maximised if all \(x_i = M/2\).

Suppose first that \(M\) is even, then \(S \leq (nM^2)/2\). In this case then

\[
M(M - 1)d \leq \frac{n}{2} M^2,
\]

\[
\therefore M^2d - Md - \frac{n}{2}M^2 \leq 0,
\]

\[
\therefore \left(\frac{2d - n}{2}\right) M^2 - dM \leq 0,
\]

\[
\therefore \left(\frac{2d - n}{2}\right) M \leq d,
\]

\[
\therefore M \leq \frac{2d}{2d - n}.
\]
The last step being possible since \( d > n/2 \). We have that \( \lfloor 2x \rfloor \leq 2\lfloor x \rfloor + 1 \), so

\[
M \leq 2 \left\lfloor \frac{d}{n-2d} \right\rfloor + 1,
\]

but since \( M \) is even

\[
M \leq 2 \left\lfloor \frac{d}{n-2d} \right\rfloor.
\]

Suppose now that \( M \) is odd. Then \( S \) is maximised if all \( x_i = (M - 1)/2 \). In this case

\[
S \leq n \left( 2 \frac{M - 1}{2} \right) \left( 2 \frac{M - M + 1}{2} \right) = n \left( (M - 1) \left( \frac{M + 1}{2} \right) \right) = \frac{n}{2} (M^2 - 1).
\]

Therefore

\[
M(M - 1)d \leq \frac{n}{2} (M^2 - 1),
\]

\[
\therefore M(M - 1)d \leq \frac{n}{2} (M - 1)(M + 1),
\]

\[
\therefore Md \leq \frac{n}{2} (M + 1),
\]

\[
\therefore Md - \frac{n}{2} M \leq \frac{n}{2},
\]

\[
\therefore M \left( \frac{2d - n}{2} \right) \leq \frac{n}{2}.
\]

\[
\therefore M \leq \frac{n}{2d - n} = \frac{2d}{2d - n} - 1.
\]

Again using \( \lfloor 2x \rfloor \leq 2\lfloor x \rfloor + 1 \), we find

\[
M \leq 2 \left\lfloor \frac{d}{2d - n} \right\rfloor + 1 - 1 = 2 \left\lfloor \frac{d}{2d - n} \right\rfloor.
\]

If we denote by \( A(n, d) \) the maximum number of codewords in an \((n, d)\) code, then

\[
A(n, d) \leq 2 \left\lfloor \frac{d}{2d - n} \right\rfloor,
\]

if \( d > n/2 \).

By considering the code \( A_n \) from above (for which this bound holds) we see that \( A(n - 1, n/2) \geq n \) provided a Hadamard matrix of order \( n \) exists. Also, from the theorem

\[
A(n - 1, n/2) \leq 2 \left\lfloor \frac{n/2}{n - (n - 1)} \right\rfloor = n.
\]

Therefore in this case the Hadamard codes satisfy the bound (if the appropriate Hadamard matrix exists).
Part II Cryptography
Chapter 10

Introduction to Cryptography

To begin this chapter we define a few basic terms.

Cryptography for our purposes will be considered communication in the presence of adversaries. Cryptanalysis is the study of the mathematical methods for defeating cryptosystems. These two fields fall under the heading of Cryptology.

10.1 Basic Definitions

Definition 10.1.1 (Alphabet, Message space). An alphabet, \( A \), is a finite set. \( A^n \) is the set of all strings of length \( n \) over \( A \). \( A^* \) is the set of all strings of finite length over \( A \). The message space, \( M \), is the set of strings over \( A \).

Definition 10.1.2 (Encryption scheme). An Encryption scheme (or cipher) consists of

- Messages spaces \( M \) and \( C \) — the plain-text and cipher-text spaces.
- A key set, \( K \) (also called the key space). Often keys are in a (key) pair \( k = (e, d) \), where \( e \) is used for encryption and \( d \) is used for decryption.
- Encryption function \( E_k : M \to C \).
- Decryption function \( D_k : C \to M \), such that \( D_k(E_k(m)) = m \) for all \( m \in M \).

We distinguish between the following two classes of adversaries:

1. Passive adversary: This adversary relies on eavesdropping only.
2. Active adversary: This type of adversary may insert or block transmissions.
Our goal is to provide secrecy and the ability to detect altered or forged messages. Although the delivery of messages cannot always be guaranteed (meaning that the message may not arrive at all or may be not the original, intended message) we can send messages regularly to discover communications disruptions.

“Breaking” a cryptosystem will be taken to mean that the plain-text can be discovered from the cipher-text. This leads to various levels of security being defined.

A cryptosystem is

- **Unconditionally secure** if the adversary can’t gain any knowledge about the plain-text (except maybe the length) regardless of the amount of cipher-text available and the amount of computing resources available.

- **Computationally secure** if it is “computationally infeasible” to discover the plain-text using the best known methods and some specified amount of computing power.

- **Provably secure** if breaking the system is at least as hard as solving a “difficult” computational problem.

Kerckhoff (1883) gave the following guidelines for choosing a cipher:

1. The system should be unbreakable in practice.
2. The cipher-text should be easy to transmit.
3. The apparatus should be portable and be of such a nature that one person can operate it.
4. It should be “easy” to operate.

We also have Kerckhoff’s principle:

The security of the cipher should rest with the keys alone. That is security is maintained even if the adversary has the encryption scheme.

The possible attacks on a cryptosystem are:

- **Cipher-text only** attack. Here the adversary attempts to recover plain-text by observing some amount of cipher-text.

- **Known plain-text** attack. The adversary has some quantity of plain-text and the corresponding cipher-text.

- **Chosen plain-text** attack. Cipher-text corresponding to plain-text chosen by the adversary is available.
• **Chosen cipher-text** attack. The plain-text corresponding to cipher-text chosen by the adversary is available.

One of the earliest ciphers is the **simple substitution cipher**. In this system the keys are just permutations of \( A \). The following example illustrates this.

**Example 10.1.3. (The shift cipher).**
Number \( A, B, C, \ldots , Z \) by \( 0, 1, 2, \ldots , 25 \). The encryption key \( e \) is a fixed shift \( \text{mod } 26 \). That is
\[
\alpha \mapsto \alpha + e \pmod{26}.
\]
The decryption key \( d = 26 - e \); the additive inverse \( \text{mod } 26 \). If \( e = 3 \) then \( d = 23 \), \( E_3(\text{HOCKEY}) = \text{KRFNHB} \) and \( D_{23}(\text{KRFNHB}) = \text{HOCKEY} \).

This system is completely insecure against chosen plain-text attacks — choose as text \( A, B, C, \ldots , Z \).

**Definition 10.1.4 (Symmetric key encryption).** A **symmetric key encryption** scheme is one in which the effort required to find \( D_d \) from \( e \) is about as much as finding \( E_e \) from \( d \).

Two examples of symmetric key encryption schemes are the simple substitution cipher where the keys are permutations of the alphabet and the shift cipher where the key specifies the amount by which a letter is shifted \( \text{mod } 26 \).

### 10.2 Affine Cipher

To describe this cipher we label the letters of the alphabet, \( A, B, \ldots , Z \), with the numbers \( 0, 1, \ldots , 25 \). The encryption function is then given by
\[
E_k(x) = ax + b \pmod{26}.
\]
Note that for decryption to be possible, \( E_k(x) \), needs to be one-to-one. This implies that we need \( \gcd(a, 26) = 1 \); any \( b \) may be used.

Therefore the encryption key \( e \) is a pair, \((a, b)\), with \( \gcd(a, 26) = 1 \). The number of encryption keys thus is \( \phi(26) \cdot 26 = 12 \cdot 26 = 312 \), where \( \phi \) is Euler’s phi function.

As far as decryption is concerned, note that if \( y \equiv ax + b \pmod{26} \), then \( y - b \equiv ax \pmod{26} \), so that \( x \equiv a^{-1}(y - b) \pmod{26} \). Here \( a^{-1} \) is the multiplicative inverse of \( a \) among the elements relatively prime to 26. We can determine \( a^{-1} \) as follows: since
\( \gcd(a, 26) = 1 \) there exists integers \( r \) and \( s \) such that \( ar + 26s = 1 \). These integers can be found using the Euclidean algorithm. Therefore

\[
\begin{align*}
ar & \equiv 1 + 26(-s) \pmod{26}, \\
ar & \equiv 1 \pmod{26}, \\
\therefore a^{-1} & \equiv r \pmod{26}.
\end{align*}
\]

Therefore the decryption key is also a pair \((a^{-1}, b)\) and the decryption function is \( D_k(y) = a^{-1}(y - b) \).

**Example 10.2.1.** As an example of the affine cipher let our key pair be \( k = (e, d) = ((7, 3), (15, 3)) \). Therefore

\[
\begin{align*}
E_k(x) &= 7x + 3 \pmod{26}, \\
D_k(y) &= 15(y - 3) \pmod{26}.
\end{align*}
\]

Also, \( D_k(E_k(x)) = D_k(7x + 3) = 15(7x + 3 - 3) = 15(7x) \equiv x \pmod{26} \).

The encryption of the word GARY would then proceed as follows

\[
\begin{align*}
G : 7 \cdot 6 + 3 &= 45 \equiv 19 \rightarrow T, \\
A : 7 \cdot 0 + 3 &= 3 \equiv 3 \rightarrow D, \\
R : 7 \cdot 17 + 3 &= 122 \equiv 18 \rightarrow S, \\
Y : 7 \cdot 24 + 3 &= 171 \equiv 15 \rightarrow P,
\end{align*}
\]

\[\blacksquare\]

### 10.2.1 Cryptanalysis of the Affine Cipher

The cryptanalysis of the affine cipher is based on examining the frequency with which cipher-text occurs. As such we require the frequency with which the letters occur in everyday usage. Below is a table that shows the letters, their associated decimal number and their frequency of occurrence in everyday English (plain-text).
Consider the following block of text that was obtained from an affine cipher.

F M X V E D K A P H F E N D R B
N D K R X R S R E F M O R U D S
D K D V S H V U F E D K A P R K
D L Y E V L R H H R H

Counting the number of occurrences of the letters above we find that the most frequent characters are: R (8 times), D (8 times), E, H, K (5 times) and F, V (4 times). Based on this we guess that one of the most frequent characters in the cipher-text, R, represents

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>A</td>
<td>0.082</td>
</tr>
<tr>
<td>1</td>
<td>B</td>
<td>0.015</td>
</tr>
<tr>
<td>2</td>
<td>C</td>
<td>0.028</td>
</tr>
<tr>
<td>3</td>
<td>D</td>
<td>0.043</td>
</tr>
<tr>
<td>4</td>
<td>E</td>
<td>0.127</td>
</tr>
<tr>
<td>5</td>
<td>F</td>
<td>0.022</td>
</tr>
<tr>
<td>6</td>
<td>G</td>
<td>0.020</td>
</tr>
<tr>
<td>7</td>
<td>H</td>
<td>0.061</td>
</tr>
<tr>
<td>8</td>
<td>I</td>
<td>0.070</td>
</tr>
<tr>
<td>9</td>
<td>J</td>
<td>0.002</td>
</tr>
<tr>
<td>10</td>
<td>K</td>
<td>0.008</td>
</tr>
<tr>
<td>11</td>
<td>L</td>
<td>0.040</td>
</tr>
<tr>
<td>12</td>
<td>M</td>
<td>0.024</td>
</tr>
<tr>
<td>13</td>
<td>N</td>
<td>0.067</td>
</tr>
<tr>
<td>14</td>
<td>O</td>
<td>0.075</td>
</tr>
<tr>
<td>15</td>
<td>P</td>
<td>0.019</td>
</tr>
<tr>
<td>16</td>
<td>Q</td>
<td>0.001</td>
</tr>
<tr>
<td>17</td>
<td>R</td>
<td>0.060</td>
</tr>
<tr>
<td>18</td>
<td>S</td>
<td>0.063</td>
</tr>
<tr>
<td>19</td>
<td>T</td>
<td>0.091</td>
</tr>
<tr>
<td>20</td>
<td>U</td>
<td>0.028</td>
</tr>
<tr>
<td>21</td>
<td>V</td>
<td>0.010</td>
</tr>
<tr>
<td>22</td>
<td>W</td>
<td>0.023</td>
</tr>
<tr>
<td>23</td>
<td>X</td>
<td>0.001</td>
</tr>
<tr>
<td>24</td>
<td>Y</td>
<td>0.020</td>
</tr>
<tr>
<td>25</td>
<td>Z</td>
<td>0.001</td>
</tr>
</tbody>
</table>
the most frequent character in plain-text, E. Next, we guess that the most frequent letter after R in the cipher-text, D, represents the second most frequent letter in plain-text, T. Therefore our guess is that E → R and T → D. This is the same as $E_k(4) = 17$ and $E_k(19) = 3$ or $a \cdot 4 + b = 17$ and $a \cdot 19 + b = 3$. This implies that $15a = -14 \equiv 12 \pmod{26}$, so that $7 \cdot 15 \cdot a \equiv 7 \cdot 12 \equiv 6 \pmod{26}$ and $b = 19$. This cannot be a valid key as $\gcd(a, 26) \neq 1$.

After this we might guess that E → R and T → E, but this leads to the same sort of problem as does E → R and T → H.

Our next best guess would be that E → R and T → K. Therefore $E_k(4) = 17$ and $E_k(19) = 10$. This implies that $4a + b \equiv 17 \pmod{26}$ and $19a + b \equiv 10 \pmod{26}$. Thus $15a \equiv -7 \equiv 19 \pmod{26}$, so that $a \equiv 3 \pmod{26}$ and $b \equiv 5 \pmod{26}$. Therefore our encryption key would be $(3, 5)$, which is a valid key since 3 is relatively prime to 26. The decryption key corresponding to this is $(3^{-1}, 5) = (9, 5)$ and then $D_k(y) = 9(y - 5)$. Applying this to the cipher-text we find the following.

\[
\begin{align*}
\text{ALGORITHMS ARE EQUI} \\
\text{TE GENERAL DEFINIT} \\
\text{IONS OF ARITHMETIC} \\
\text{PROCESSES}
\end{align*}
\]

This corresponds to the text algorithms are quite general definitions of arithmetic processes. Seeing that we have a piece of plain-text that “makes sense” we can assume that we have the key.

### 10.3 Some Other Ciphers

**Definition 10.3.1 (Block cipher).** A block cipher transforms the message by encrypting blocks of a fixed number of characters.

**Definition 10.3.2 (Polyalphabetic cipher).** In a polyalphabetic cipher each source symbol can map to one of several cipher-text symbols.

An example of a polyalphabetic cipher is the following.

### 10.3.1 The Vigenère Cipher

This cipher has the following properties.

- The symbols from $A$ are identified with $0, 1, \ldots, |A| - 1$. 

• A key(word) \( k = k_0 k_1, \ldots, k_{n-1} \) is a string in \( A^n \).

• Encryption is performed on \( n \) character blocks of source by adding the keyword \((\text{mod } |A|)\) to the source characters.

**Example 10.3.3. (Vigenère cipher)**

\( A = \{a, b, \ldots, z\} \) and \( k = \text{golf} \). The encryption of a piece of plain-text \((\text{double eagle})\) would then proceed as follows.

\[
\begin{align*}
double\ eagle & \\
golf & = g\ o\ l\ f
\end{align*}
\]

\[
\begin{align*}
\text{jcfgrspfmzp} & \quad \text{Therefore the plain-text \textit{double eagle} is encrypted as \textit{jcfgrspfmzp}.}
\end{align*}
\]

To decrypt one subtracts the key from the cipher-text.

\[
\begin{align*}
\text{ubpragekuzwtchswuir} & \\
\text{golfgolfgolfgolf} & \quad \text{In this case the cipher-text \textit{ubpragekuzwtchswuir} corresponds to the plain-text \textit{one must follow through}.}
\end{align*}
\]

### 10.3.2 Cryptanalysis of the Vigenère Cipher: The Kasiski Examination

We now discuss a method that can be used to attack the Vigenère cipher. It was proposed by Friedrich Kasiski in 1863. The discussion is based on the following piece of cipher-text that is the result of a Vigenère cipher applied to a piece of English text. The plain-text was first processed to remove all punctuation and spaces. The spaces in the cipher-text were added to improve readability.

<table>
<thead>
<tr>
<th>Offset</th>
<th>Cipher-text</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UPVZB BVUPN KKFOK OGAKU FTBKJ LFUJX VIPZV KFXZO FIDLO ONLUP</td>
</tr>
<tr>
<td>50</td>
<td>KKFOZ OMQFQ MQXKU AFJUP VVVK KFDFL DMTIU PVVFI ZVTMU XDBZY</td>
</tr>
<tr>
<td>100</td>
<td>FFVYF ZTHBA ZQHEY LTXVU JVXFM IDRSG EJNCI PVZZQ HQEYJ BZQHB</td>
</tr>
<tr>
<td>150</td>
<td>YHTWL OUNW OLUJ VREZA JHTWW VPTZW VLVDM TROPV XWIMN KJBE</td>
</tr>
<tr>
<td>200</td>
<td>FITKW XQGEL FZQBY HSVND TVFOJ DQHBY YLOOZ QTQXK UIJLS LNUP</td>
</tr>
<tr>
<td>250</td>
<td>RESWB HOEZQ HERFC MRFJW XXIMR LSIR WMIHF TZQHN CXUBV UJVX</td>
</tr>
<tr>
<td>300</td>
<td>JZTOJ VXGJA REMMU GPEEG PEWP BYHXI KHS</td>
</tr>
</tbody>
</table>
The idea of the Kasiski examination is to try and recover the key-length used in the Vigenère cipher. This is accomplished by analysing the distances between identical pieces of cipher-text: occasionally identical portions of plain-text will align with the same fragment of the key producing exactly the same cipher-text. The possibility also exists for “accidental” matches. That is identical fragments of cipher-text which did not result from the same plain-text, but these tend to be rare especially with longer match lengths. In the case of a “correct” match the key-length has to divide the difference in the positions where these pieces of identical cipher-text occurred. Since accidental matches may also occur it may not be enough to only examine the greatest common divisor of all distances.

In our example the fragment of cipher-text ZQH occurs in a number of places which are underlined in the table above. The corresponding offsets are: 110, 138, 146, 226, 258, 286. It is therefore likely that the keyword length, \( l \), divides the difference of any of these. The differences \( 138 - 110 = 2^27 \) and \( 226 - 110 = 2^229 \) suggest that \( l \) divides \( \gcd(2^27, 2^229) = 2^2 \). The situation with \( l = 1 \) corresponds to a simple shift.

At this point we may now examine the frequency distributions for each candidate key-length. The idea being that at these fixed lengths the plain-text was combined with the same letter in the keyword and so will give an accurate distribution. The table below shows the frequencies that were found at each position of the key, that is for \( l = 2 \) we examine the letters in the odd and even positions separately.

<table>
<thead>
<tr>
<th>( l )</th>
<th>Offsets</th>
<th>Frequencies of the letters in the sub-message (sorted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>0, 2, 4, \ldots</td>
<td>19 16 12 12 11 10 9 8 8 7 6 6 5 5 5 5 4 4 3 2 2 2 1 0 0</td>
</tr>
<tr>
<td></td>
<td>1, 3, 5, \ldots</td>
<td>14 14 13 10 10 9 9 8 8 7 6 6 5 5 5 3 3 3 2 2 2 1 0</td>
</tr>
<tr>
<td>4</td>
<td>0, 4, 8, \ldots</td>
<td>12 10 10 7 6 6 6 5 5 4 4 2 2 2 1 1 1 1 1 1 1 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>1, 5, 9, \ldots</td>
<td>10 9 8 7 5 5 5 5 5 4 4 3 3 2 2 2 1 1 1 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>2, 6, 10, \ldots</td>
<td>12 11 8 8 7 5 5 5 4 4 2 2 2 2 2 1 1 1 1 0 0 0 0 0 0 0 0 0</td>
</tr>
<tr>
<td></td>
<td>3, 7, 11, \ldots</td>
<td>9 9 8 8 7 5 5 5 4 3 3 3 3 3 2 2 2 1 1 0 0 0 0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

For a fixed source distribution each line of the correct key-length should reflect the source frequencies. In this example, the frequencies suggest that \( l = 4 \) is more likely than \( l = 2 \).

Since we know that the letter ‘e’ is the most frequent letter in everyday English, we suspect that one of the higher-frequency cipher-text letters, in each line of the \( l = 4 \) case, corresponds to the plain-text letter ‘e’. Now each line in the \( l = 4 \) case represents a letter of the keyword. So if the highest frequency cipher-text letter in each row corresponded to ‘e’, then all we would have to do is to subtract ‘e’ from the cipher-text letter to get the corresponding keyword letter and so retrieve the keyword. Since we are not absolutely
10.3. SOME OTHER CIPHERS

100% sure that the highest-frequency cipher-text letter in each line corresponds to ‘e’ our best strategy would be to consider the 3 or 4 most frequent letters in each row (‘e’ would definitely be among the highest frequency ones). The table below shows the four most frequent letters in each row as well as the required keyword letter that maps these cipher-text letters back to ‘e’.

<table>
<thead>
<tr>
<th>Cipher-text</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>F J U P</td>
<td>B F Q L</td>
</tr>
<tr>
<td>B V M I</td>
<td>X R I E</td>
</tr>
<tr>
<td>V Z R X</td>
<td>R V N T</td>
</tr>
<tr>
<td>K Q H L</td>
<td>G M D H</td>
</tr>
</tbody>
</table>

At this point one would do an exhaustive search using keywords that are built from the likely letters shown above. That is one chooses a letter from the first row as the possible first letter of the keyword, a letter from the second row for the second possible letter of the keyword and so on. This requires $4^4 = 256$ decryptions. If the keyword was chosen from a dictionary this simplifies things a great deal. In our example this would be words such as LEND and BIRD. Using BIRD on the cipher-text yields the following.

The water of the Gulf stretched out before her, gleaming with the million lights of the sun. The voice voice of the sea is seductive, never ceasing, whispering, clamoring, murmuring, inviting, the soul to wander in abysses of solitude. All along the white beach, up and down, there was no living thing in sight. A bird with a broken wing was beating the air above, reeling fluttering, circling disabled down, down to the water.

The ZQH in the cipher-text used in the Kasiski examination corresponds to ‘ing’ in the plain-text.

10.3.3 The Vernam Cipher

The Vernam cipher is an example of a stream cipher.

**Definition 10.3.4 (Stream cipher (state cipher)).** In a *stream cipher* the mapping of a block may depend on its position in the message.

This cipher operates as follows.

- The alphabet $A = \{0, 1\}$.
• Keys are the same length as messages over $A$.

• A message $m$ is encrypted as $m \oplus k$. Where $\oplus$ is (mod 2) addition.

• To decrypt the cipher $c$ we add the key $k$ to it: $c \oplus k = (m \oplus k) \oplus k = m$.

If the key digits are the result of independent Bernoulli trials with probability $1/2$ (and used only once), then the cipher is known as a one-time-pad and is unconditionally secure against a cipher-text only attack. The reason for this is that every message of the same length as the cipher-text maps to the cipher-text for some choice of key, and all keys are equally likely.
Chapter 11

Public Key Cryptography

This chapter is devoted to one of the most widely used forms of cryptography, namely public key cryptography. The RSA cryptosystem, which is an example of this, is used by millions of people around the world.

**Definition 11.0.5 (Private key system).** In a private key cryptosystem $D_k$ is either the same as $E_k$, or easy to get from it. If $E_k$ is known the system is insecure. Therefore $D_k$ and $E_k$ must be kept private.

**Definition 11.0.6 (Public key system).** In a public key system if $E_k$ is known it is (believed to be) computationally infeasible to use it to determine $D_k$. Therefore $E_k$ can be made public (for instance in a directory) and anyone can lookup $E_k$ when they want to send a message.

The believed computational infeasibility of determining $D_k$ makes the exchange of keys unnecessary.

The idea of a public key system arose in 1976 in the work of Diffie and Hellman [?]. The first public key system was discovered in 1977 by Rivest, Shamir and Adleman [?] (the RSA system).

Public key systems can never be unconditionally secure. If the adversary has the cipher-text, they can use $E_k$ to encrypt every possible piece of plain-text until a match is found. This process might not be feasible in practice, but is nonetheless possible in principle.

**Definition 11.0.7 (One-way function).** A function $f : M \rightarrow C$ is one-way if

- $f(m)$ is “easy” to compute for all $m \in M$.
- for all (or most) $c \in C$ it is “computationally infeasible” to find $m \in M$ such that $f(m) = c$. 

81
It is not known whether one-way functions exist. A number of functions have been identified that seem to be one-way.

Example 11.0.8. An example of a one-way function is found in the UNIX operating system. Here the password to a user’s account is stored in a file that is readable to all users of the system. The passwords are encrypted using a (what is believed to be) one-way function.

Definition 11.0.9 (Trapdoor function). A one-way function is a trapdoor function if it has the property that given some extra information, it becomes feasible to find \( m \in M \) such that \( f(m) = c \) for a given \( c \in C \).

So, in public key systems the extra information in a trapdoor function makes it possible to find \( D_k \).

11.1 The Rivest-Shamir-Adleman (RSA) Cryptosystem

This system consists of the following.

- An integer \( n = pq \), where \( p \) and \( q \) are large, distinct primes.
- The message space and cipher space are the same : \( M = C = \mathbb{Z}_n \).
- The key pair \( k = (a, b) \) where \( ab \equiv 1 \pmod{\phi(n)} \) with \( \phi(n) = \phi(pq) = (p-1)(q-1) \) the Euler phi function.
- The encryption function \( E_k(x) = x^b \pmod{n} \).
- The decryption function \( D_k(y) = y^a \pmod{n} \).
- \( n \) and \( b \) are public, while \( p, q \) and \( a \) are private.

Let’s compute \( D_k(E_k(x)) \) to see that we can recover an encrypted message. To do this we need the following facts.

\[ ab \equiv 1 \pmod{\phi(n)} \iff ab = 1 + t\phi(n). \]

\[ \star \] Euler’s Theorem : If \( \gcd(x, n) = 1 \), then \( x^{\phi(n)} \equiv 1 \pmod{n} \).
\[ \text{If } n = pq, \text{ then } x_1 \equiv x_2 \pmod{n} \iff x_1 \equiv x_2 \pmod{p} \text{ and } x_1 \equiv x_2 \pmod{q} : \]

**Proof.**

\[ \implies : \]

\[ x_1 \equiv x_2 \pmod{pq} \text{ implies that } x_1 = x_2 + t \cdot pq. \]

Therefore \( x_1 = x_2 + (tq)p = x_2 + (tp)q, \) so that \( x_1 \equiv x_2 \pmod{p} \) and \( x_1 \equiv x_2 \pmod{q}. \)

\[ \Leftarrow : \]

\( x_1 \equiv x_2 \pmod{p} \) and \( x_1 \equiv x_2 \pmod{q} \) implies that \( x_1 - x_2 = l_1 p \) and \( x_1 - x_2 = l_2 q. \)

Therefore \( q \mid l_1 p \) so that \( q \mid l_1 \) (\( p \) and \( q \) are different primes). Thus \( x_1 - x_2 = l_3 pq \) or in otherwords \( x_1 \equiv x_2 \pmod{pq}. \)

\[ \square \]

We have \( D_k(E_k(x)) = D_k(x^b) = (x^b)^a \pmod{n}. \) Here \( x^{ab} = x^{1+t\phi(n)} = x \cdot x^{t\phi(n)} \pmod{n}. \) Now there are two cases to consider.

1. \( \gcd(x, n) = 1. \)
   
   By Euler’s Theorem \( x^{\phi(n)} \equiv 1 \pmod{n}, \) so that \( x^{t\phi(n)} = (x^{\phi(n)})^t \equiv 1^t \equiv 1 \pmod{n}. \)
   
   Therefore \( D_k(E_k(x)) = x \cdot x^{t\phi(n)} \equiv x \pmod{n}. \)

2. \( \gcd(x, n) > 1. \)
   
   Here \( x^{t\phi(n)} = s \pmod{n} \iff x^{t\phi(n)} = s \pmod{p} \) and \( x^{t\phi(n)} = s \pmod{q}. \)
   
   We know that \( \phi(n) = (p - 1)(q - 1), \phi(p) = p - 1 \) and \( \phi(q) = q - 1. \)
   
   Thus \( t \cdot \phi(n) = t_1 \cdot \phi(p) = t_2 \cdot \phi(q). \)
   
   Therefore \( x^{t\phi(n)} \equiv x^{t_1 \phi(p)} \equiv 1 \pmod{p} \) and \( x^{t\phi(n)} \equiv x^{t_2 \phi(q)} \equiv 1 \pmod{q}. \)
   
   By Euler’s Theorem then \( x^{t\phi(n)} \equiv 1 \pmod{pq} \). This gives \( D_k(E_k(x)) = x \cdot x^{t\phi(n)} \equiv x \pmod{n}. \)

In summary then the RSA system operates as follows.

1. Generate two large distinct primes \( p \) and \( q. \)

2. Compute \( n = pq \) and \( \phi(n) = (p - 1)(q - 1). \)

3. Choose a random \( b \) with \( 0 < b < \phi(n) \) and \( \gcd(b, \phi(n)) = 1. \) The last requirement ensures that \( b^{-1} \) exists, which is needed in the next step.

4. Compute \( a = b^{-1} \pmod{\phi(n)}. \)

5. Publish \( b \) and \( n. \)

The choice of the \( b \) in step 3 is made using the using the Euclidean algorithm to check whether \( \gcd(b, \phi(n)) = 1. \) The probability that a randomly chosen \( b \) is relatively prime to \( \phi(n) \) is \( \phi(\phi(n))/\phi(n). \)

Therefore we need to try about \( \phi(n)/\phi(\phi(n)) \) different \( b \)’s before one that is relatively prime to \( \phi(n) \) is found.
The computation of $a$ in step 4 can be done as a byproduct of the Euclidean algorithm (using it in reverse).

**Example 11.1.1.** Suppose Bob choose his two primes to be $p = 101$ and $q = 113$. Then $n = 11413$ and $\phi(n) = (p - 1)(q - 1) = (100)(112) = 11200 = 2^55^27$. Since the prime divisors of $\phi(n)$ are 2, 5 and 7 any $b$ not divisible by 2, 5 and 7 is relatively prime to $\phi(n)$.

Let’s say Bob chooses $b = 3533$.

We now need to find $a$ such that $ab \equiv 1 \pmod{11200}$. The Euclidean algorithm yields

\[
\begin{align*}
11200 &= 3533 \times 3 + 601, \\
3533 &= 601 \times 5 + 528, \\
601 &= 528 \times 1 + 73, \\
528 &= 73 \times 7 + 17, \\
73 &= 17 \times 4 + 5, \\
17 &= 5 \times 3 + 2, \\
5 &= 2 \times 2 + 1.
\end{align*}
\]

The last nonzero remainder is the gcd (=1). We now use the results of the algorithm in reverse to express 1 (the gcd) in terms of 11200 and 3533.

\[
\begin{align*}
1 &= 5 - 2 \times 2, \\
&= 5 - 2(17 - 5 \times 3) = 7 \times 5 - 2 \times 17, \\
&= 7(73 - 17 \times 4) - 2 \times 17 = 7 \times 73 - 30 \times 17, \\
&= 7 \times 73 - 30(528 - 73 \times 7) = 217 \times 73 - 30 \times 528, \\
&= 217(601 - 528) - 30(528) = 217 \times 601 - 247 \times 528, \\
&= 217 \times 601 - 247(3533 - 601 \times 5) = 1452 \times 601 - 247 \times 3533, \\
&= 1452(11200 - 3533 \times 3) - 247 \times 3533 = 1452 \times 11200 - 4603 \times 3533, \\
&= 1452 \times 11200 + (-4603) \times 3533.
\end{align*}
\]

Therefore $a = b^{-1} \equiv -4603 \equiv 6597 \pmod{11200}$.

Bob will publish $n = 11413$ and $b = 3533$.

To send plain-text message 9726 to Bob, Alice computes

\[9726^{3533} \pmod{11413} = 5761.\]

Bob decrypts this as

\[5761^{6597} \pmod{11413} = 9726,\]
11.1. THE RIVEST-SHAMIR-ADLEMAN (RSA) CRYPTOSYSTEM

which equals the original message.

For the RSA system to be effective we need fast methods to do the exponentiation. The method we describe is known as “square and multiply.”

Suppose we want to compute \( x^b \pmod{n} \). Let the binary representation of \( b \) be equal to \( b_l b_{l-1} \cdots b_0 \). That is \( b = b_0 + 2b_1 + 2^2b_2 + \cdots 2^lb_l \) (here \( l = \lceil \log_2 b \rceil - 1 \)). Therefore
\[
x^b = x^{b_0+2b_1+2^2b_2+\cdots+2^lb_l} = x^{b_0}x^{2b_1}x^{2^2b_2}\cdots x^{2^lb_l} = x^{b_0}(x^{2^{b_1}}(x^{2^{b_2}}\cdots(x^{2^l})^{b_l})
\]

Thus we compute \( x, x^2, x^4, \ldots, x^{2^l} \) and multiply together the terms corresponding to the bits \( b_i \) that equal 1.

**Example 11.1.2.** Compute \( 57^{26} \pmod{91} \).

\[
26 = 11010.
\]

We need
\[
57^1 \equiv 57 \pmod{91}, \\
57^2 = 3249 \equiv 64 \pmod{91}, \\
57^4 = (57^2)^2 = 64^2 = 4096 \equiv 1 \pmod{91}, \\
57^8 = (57^4)^2 = 1^2 \equiv 1 \pmod{91}, \\
57^{16} = (57^8)^2 = 1^2 \equiv 1 \pmod{91}.
\]

Therefore \( 57^{26} = 1 \cdot 1 \cdot 64 \equiv 64 \pmod{91} \).

11.1.1 Security of the RSA System

One of the most straightforward ways to attack the RSA system would be to try and compute \( \phi(n) \) — from which we would be able to compute \( b \). In this section we show that the level of difficulty involved in doing this is no higher than trying to factor \( n \).

Assume then that a cryptanalyst has both \( n \) and \( \phi(n) \). Then by solving the set of equations
\[
n = pq, \\
\phi(n) = (p-1)(q-1),
\]
for the unknowns \( p \) and \( q \) we can factor \( n \). If we substitute \( q = n/p \) into the second equation, we obtain a quadratic in the unknown \( p \)
\[
p^2 - (n - \phi(n) + 1)p + n = 0.
\]
The two roots of this equation will be $p$ and $q$.

Therefore if a cryptanalyst knows $\phi(n)$ this can be used to factor $n$. Thus computing $\phi(n)$ is no easier than factoring $n$.

## 11.2 Probabilistic Primality Testing

In setting up an RSA system we need two large (on the order of 100 decimal digits) primes. This section discusses a method for identifying candidate primes.

**Definition 11.2.1 (Decision problem).** A decision problem is a problem with a “yes” or “no” answer.

**Definition 11.2.2 (Probabilistic algorithm).** A probabilistic algorithm is any algorithm that uses random numbers.

**Definition 11.2.3 (Yes-based Monte Carlo algorithm).** A yes-based Monte Carlo algorithm, for a decision problem, is a probabilistic algorithm in which a “yes” answer is always correct but a “no” answer may be incorrect.

A yes-based Monte Carlo algorithm has an error probability of $\epsilon$ if the algorithm gives an incorrect answer “no” (when the answer is really “yes”) with a probability at most $\epsilon$. Here the probability is computed over all possible random choices made by the algorithm when it is run with a given input.

The basic idea behind the identification of possible primes is as follows. Generate random odd numbers and use an algorithm that answers “yes” or “no” to the question “is $n$ composite?” A “yes” answer is always correct, but a “no” answer maybe incorrect with some probability $\epsilon < 1$. Run the algorithm $k$ times. If it ever answers “yes” then $n$ is composite. Otherwise $n$ is prime with probability $1 - \epsilon^k$.

**Theorem 11.2.4 (The Prime Number Theorem (1899)).**

$$\lim_{n \to \infty} \frac{\pi(n)}{n/\ln(n)} = 1,$$

where $\pi(n)$ is the number of primes less than or equal to $n$.

From the Prime Number Theorem we see that the probability that a randomly chosen integer $k$ is prime is approximately

$$\frac{k/\ln(k)}{k} = \frac{1}{\ln(k)}.$$
If only odd integers are considered, the probability is approximately $2/\ln(k)$. So on average one could expect to test about $\ln(k)/2 = \ln(\sqrt{k})$ odd integers before a prime is found.

If we need to find a prime with about 100 digits, then $k \approx 10^{100}$ above, so that we need to test about $\ln(\sqrt{k}) \approx 115$ integers on average before a prime is found.

The algorithm for identifying possible primes that we will be discussing is known as the Miller-Rabin algorithm. We introduce this algorithm by way of the following theorem.

**Theorem 11.2.5 (Fermat’s Little Theorem).** If $n$ is prime and $\gcd(a, n) = 1$, then $a^{n-1} \equiv 1 \pmod{n}$.

Suppose that $n$ is an odd prime, then $n - 1$ is even, so that $n - 1$ can be written as $n - 1 = 2^k m$, where $k \geq 1$ and $m$ is odd. Let $a$ be an integer relatively prime to $n$.

By Fermat’s little Theorem then $a^{2^k m} \equiv 1 \pmod{n}$. That is $(a^{2^{k-1} m})^2 \equiv 1 \pmod{n}$. Therefore $a^{2^{k-1} m} \equiv \pm 1 \pmod{n}$. If it is congruent to $+1$, we can repeat the argument to find that $a^{2^{k-2} m} \equiv \pm 1 \pmod{n}$.

If at each step $a^{2^{k-1} m} \equiv 1 \pmod{n}$ then eventually $a^{m} \equiv \pm 1 \pmod{n}$.

The algorithm works in reverse by looking at the values $a^m$, $a^{2m}$, $a^{2^2 m}$, $\ldots$, $a^{2^{k-1} m}$ and seeing whether they are congruent to $-1 \pmod{n}$.

The Miller-Rabin algorithm is given as input an odd integer $n$ and it answers the question “is $n$ composite?” A “yes” answer is correct and a “no” answer maybe wrong. The algorithm consists of six steps.

1. Write $n - 1 = 2^k m$, where $m$ is odd.

2. Choose a random $a$ with $1 \leq a \leq n - 1$.

3. Compute $b = a^m \pmod{n}$.

4. If $b \equiv 1 \pmod{n}$ then answer “prime” and stop.

5. For $i = 0, 1, 2, \ldots, k - 1$
   
   If $b \equiv -1 \pmod{n}$, then answer “prime” and stop,
   
   else replace $b$ by $b^2$, $i$ by $i + 1$ and go to step 5.

6. Answer “composite” and stop.

**Theorem 11.2.6.** The Miller-Rabin algorithm is a yes-based Monte-Carlo algorithm for testing if $n$ is composite.
CHAPTER 11. PUBLIC KEY CRYPTOGRAPHY

**Proof.**

Assume the algorithm answers “composite” for some prime integer \( n \). We will obtain a contradiction.

Since the answer is “composite”, it must be the case that \( a^m \not\equiv 1 \pmod{n} \). In step 5 of the algorithm, the sequence of values tested is

\[
b = a^m, b^2 = a^{2m}, b^4 = a^{4m}, b^8 = a^{8m}, \ldots, b^{2^{k-1}} = a^{2^{k-1}m}.
\]

Since the algorithm answers “composite” we know that

\[
a^{2im} \not\equiv -1 \pmod{n} \quad i = 0, 1, 2, \ldots, k - 1.
\]

The fact that \( n \) is prime implies that \( a^{n-1} = a^{2^m} \equiv 1 \pmod{n} \), so that \((a^{2k-1}m)^2 \equiv 1 \pmod{n} \). Thus \( a^{2^{k-1}m} \equiv \pm 1 \pmod{n} \). We know that \( a^{2^{k-1}m} \not\equiv -1 \pmod{n} \) so \( a^{2^{k-1}m} \not\equiv 1 \pmod{n} \). Repeating the argument \( k - 1 \) more times we find that \( a^m \equiv 1 \pmod{n} \), a contradiction.

**Fact** The error probability in the Miller-Rabin Algorithm is less than or equal to 0.25.
Chapter 12

The Rabin Cryptosystem

The second example of a public key cryptosystem that we will consider is the Rabin Cryptosystem [?]. This system has the following features.

- It uses two distinct primes $p$ and $q$ each of which is congruent to 3 (mod 4). Let $n = pq$.
- The message space and cipher space are the same: $M = C = \mathbb{Z}_n$.
- Further, it has an encryption key $B$, chosen so that $0 \leq B \leq n - 1$.
- The encryption function,
  \[ E_k(x) = x(x + B) \pmod{n}. \]
- The decryption function,
  \[ D_k(y) = \sqrt{\frac{B^2}{4} + y - \frac{B}{2}} \pmod{n}, \]
  where $B/4 = 4^{-1}B$ and $\sqrt{t}$ is any number $s$ such that $s^2 = t$.
- $n$ and $B$ are made public, $p$ and $q$ are private.

For this system to be of use we need to know that there are infinitely many primes of the form $4k + 3$ (i.e., primes that are congruent to 3 (mod 4)). This can be seen in one of two ways. The first is Dirichlet’s Theorem and the second is a direct proof, both of which are shown below.

**Theorem 12.0.7 (Dirichlet’s Theorem).** The sequence $ak + b$, $k = 0, 1, 2, \ldots$, contains infinitely many primes if and only if $\gcd(a, b) = 1$. 

89
CHAPTER 12. THE RABIN CRYPTOSYSTEM

Since \( \gcd(4, 3) = 1 \), we know that there are infinitely many primes of the form \( 4k + 3 \).

We now show directly that there are infinitely many primes congruent to 3 (mod 4): Assume that there are only finitely many primes congruent to 3 (mod 4), say \( p_0 = 3 \), \( p_1 = 7 \), \( p_2 = 11 \), \ldots, \( p_n \). Consider \( N = 4p_1p_2 \cdots p_n + 3 \) (note that \( p_0 \) is not included in this product). We know that if \( p \) and \( q \) are both congruent to 1 (mod 4), then so is their product. Therefore a number congruent to 3 (mod 4) must have a prime divisor that is congruent to 3 (mod 4) (if all were congruent to 1 (mod 4), the number itself would be congruent to 1 (mod 4)). We have \( N \equiv 3 \) (mod 4), so that \( N \) has a prime divisor congruent to 3 (mod 4). On the other hand if 3|\( N \), then 3|\( 4p_1p_2 \cdots p_n \) and since 3 is prime this implies that 3|\( 4 \) or 3|\( p_i \) for some \( i \geq 1 \), a contradiction. Thus \( 3 \not| N \). Also, if \( i \geq 1 \) then \( p_i|N \) and \( p_i|4p_1p_2 \cdots p_n \) implies that \( p_i|3 \), a contradiction. That is there exists a prime congruent to 3 (mod 4) not among \( p_0, p_1, \ldots, p_n \), a contradiction. Therefore the number of primes congruent to 3 (mod 4) is infinite.

It turns out that the encryption function is not one-to-one. To see this we need the following two results.

**Theorem 12.0.8 (Chinese Remainder Theorem).** The system of congruences

\[
x \equiv a_1 \pmod{n_1}, \\
x \equiv a_2 \pmod{n_2}, \\
\vdots \\
x \equiv a_t \pmod{n_t},
\]

where \( \gcd(n_i, n_j) = 1 \) if \( i \neq j \), has a unique solution \( \pmod{n_1n_2 \cdots n_t} \). This solution is \( x = a_1M_1y_1 + a_2M_2y_2 + \cdots + a_tM_ty_t \), where \( M_i = (n_1n_2 \cdots n_t)/n_i \) and \( y_i = M_i^{-1} \pmod{n_i} \), \( 1 \leq i \leq t \).

**Proposition 12.0.9.** If \( p \) and \( q \) are distinct odd primes, then the congruence \( x^2 \equiv 1 \pmod{pq} \) has exactly four solutions \( \pmod{pq} \).

**Proof.**

\[
x^2 \equiv 1 \pmod{pq} \iff x^2 - 1 \equiv 0 \pmod{pq} \iff x^2 - 1 = kpq, \ k \in \mathbb{Z} \iff p|(x^2 - 1) \text{ and } q|(x^2 - 1) \iff x^2 \equiv 1 \pmod{p} \text{ and } x^2 \equiv 1 \pmod{q}.
\]

Now \( x^2 - 1 = (x - 1)(x + 1) \), so \( p|(x^2 - 1) \iff p|(x - 1)(x + 1) \Rightarrow p|(x - 1) \text{ or } p|(x + 1) \). That is \( x \equiv 1 \pmod{p} \) or \( x \equiv -1 \pmod{p} \). Similarly \( x \equiv \pm 1 \pmod{q} \).
Consider the four systems of congruences.

\[
\begin{align*}
  x &\equiv 1 \pmod{p} \quad \text{and} \quad x \equiv 1 \pmod{q}, \\
  x &\equiv 1 \pmod{p} \quad \text{and} \quad x \equiv -1 \pmod{q}, \\
  x &\equiv -1 \pmod{p} \quad \text{and} \quad x \equiv 1 \pmod{q}, \\
  x &\equiv -1 \pmod{p} \quad \text{and} \quad x \equiv -1 \pmod{q}.
\end{align*}
\]

By the Chinese Remainder Theorem each system has a unique solution \((\pmod{pq})\) and each solution leads to a solution of \(x^2 \equiv 1 \pmod{pq}\). Since \(p\) and \(q\) are odd primes, \(1 \not\equiv -1\), so the four solutions are all distinct.

\(\square\)

**Example 12.0.10.** If \(n = 15 = 3 \times 5\), then \(1 \equiv 1, -1 \equiv 14, 4 \equiv 4\) and \(-4 \equiv 11\) are the square roots of \(1 \pmod{15}\).

We are now in a position to show that the encryption function is not one-to-one. Let \(x \in \mathbb{Z}_n\) and \(\omega\) be one of the four square roots of \(1 \pmod{n}\), where \(n = pq\). Then

\[
E_k\left(\omega\left(x + \frac{B}{2}\right) - \frac{B}{2}\right),
\]

\[
= \left[\omega \left(x + \frac{B}{2}\right) - \frac{B}{2}\right] \left[\omega\left(x + \frac{B}{2}\right) + \frac{B}{2}\right],
\]

\[
= \omega^2 \left(x + \frac{B}{2}\right)^2 - \frac{B^2}{4}; \quad \omega^2 = 1,
\]

\[
= x^2 + xB = x(x + B) = E_k(x).
\]

This implies that there are four different plain-texts that encrypt to the same cipher-text as \(x\).

Next we treat the decryption process. The receiver is given a cipher-text \(y\) and wants to determine \(x\) such that \(x^2 + Bx \equiv y \pmod{n}\). To simplify notation let \(x_1 = x + B/2\), that is \(x = x_1 - B/2\). The congruence then becomes

\[
y \equiv \left(x_1 - \frac{B}{2}\right)^2 + B \left(x_1 - \frac{B}{2}\right),
\]

\[
= x_1^2 + \frac{B^2}{4} - \frac{B^2}{2},
\]

\[
= x_1^2 - \frac{B^2}{4} \pmod{n}.
\]

Therefore

\[
x_1^2 \equiv y + \frac{B^2}{4} \pmod{n}.
\]
By letting $C = y + B^2/4$ we get

$$x_1^2 \equiv C \pmod{n}.$$  

This is equivalent to the system

$$x_1^2 \equiv C \pmod{p},$$
$$x_1^2 \equiv C \pmod{q}.$$  

Each congruence in the system has zero or two solutions. These can be combined, as before, to get up to four solutions (mod $pq$).

To determine the solutions to the congruences above we need the following concept and theorem.

**Definition 12.0.11 (Quadratic residue).** Let $m$ be a prime number. A number $a \not\equiv 0 \pmod{m}$ is called a quadratic residue (mod $m$) if there exists and $x$ such that $x^2 \equiv a \pmod{m}$. Otherwise $a$ is a quadratic non-residue (mod $m$).

**Example 12.0.12.** If $m = 7$, the quadratic residues are

$$1^2 \equiv 1,$$
$$2^2 \equiv 4,$$
$$3^2 \equiv 2,$$
$$-2^2 \equiv 4,$$
$$-3^2 \equiv 2,$$
$$-1^2 \equiv 1.$$  

As an aside we note that if $m$ is prime then there exists $(m - 1)/2$ quadratic residues and $(m - 1)/2$ quadratic non-residues.

One way to determine if $a$ is a quadratic residue (mod $m$) is to use Euler’s criterion.

**Theorem 12.0.13 (Euler’s Criterion).** Let $m$ be prime. Then a number $a \not\equiv 0 \pmod{m}$ is a quadratic residue (mod $m$) $\iff a^{(m-1)/2} \equiv 1 \pmod{m}$.

Note that if $a$ is a nonzero quadratic residue then $a^{(m-1)/2} \equiv -1 \pmod{m}$.

Euler’s Criterion only answers yes or no as to whether there exists an $x$ such that $x^2 \equiv a \pmod{m}$. It does not say how to find this $x$. If our prime, $m$, is of a specific form then the determination of this $x$ becomes easy.
If \( m \) is prime and \( m \equiv 1 \pmod{4} \), then there is no known, efficient algorithm to find square roots, i.e. the \( x \pmod{m} \).

On the other hand when \( m \) is a prime and \( m \equiv 3 \pmod{4} \), the square roots are easy to find. It is just \( \pm a^{(m+1)/4} \). To check this we compute

\[
\left( \pm a^{(m+1)/4} \right)^2 \equiv a^{(m+1)/2},
\]

\[\equiv a \cdot a^{(m-1)/2},
\]

\[\equiv a \pmod{m},
\]

as long as \( a \) is a quadratic residue \( \pmod{m} \).

Therefore the square roots of \( a \) are \( \pm a^{(m+1)/4} \) \( \pmod{m} \). Note that we are using the fact that \( m \equiv 3 \pmod{4} \) when we calculate the exponent \( (m + 1)/4 \).

Returning to our original problem we have

\[
x_1^2 \equiv C \pmod{p},
\]

\[
x_2^2 \equiv C \pmod{q}.
\]

Using the above procedure we find the two square roots of \( C \pmod{p} \) and the two square roots \( \pmod{q} \). These may then be combined using the Chinese Remainder Theorem to find the square roots of \( C \pmod{n} \).

Recall that we have

\[
x = x_1 - \frac{B}{2},
\]

\[
C = y + \frac{B^2}{4}.
\]

We know that

\[
x_1 = \sqrt{C} = \sqrt{y + \frac{B^2}{4}},
\]

so that

\[
x = \sqrt{y + \frac{B^2}{4}} - \frac{B}{2} = D_k(y).
\]

The four square roots of \( C \pmod{n} \) leads to the four possibilities for \( x \).

**Example 12.0.14.** \( n = 7 \times 11 = 77 \) and \( B = 9 \).

The encryption function is \( E_k(x) = x^2 + Bx = x^2 + 9x \pmod{77} \).
The decryption function is
\[ D_k(y) = \sqrt{y + 4^{-1}B^2} - 2^{-1}B, \]
\[ = \sqrt{y + 4^{-1} \cdot 81} - 2^{-1} \cdot 9, \]
\[ = \sqrt{y + 4^{-1} \cdot 4} - 39 \cdot 9, \]
\[ = \sqrt{y + 1} - 43. \]

Suppose the cipher-text 22 is received. Then \( D_k(22) = \sqrt{23} - 43 \) (mod 77). We therefore need to find \( \sqrt{23} \) (mod 77). Both 7 and 11 are congruent to 3 (mod 4). Therefore

\[ 23^{(7+1)/4} \equiv 23^2 \equiv 4 \pmod{7}, \]
\[ 23^{(11+1)/4} \equiv 23^3 \equiv 1 \pmod{11}. \]

Thus the square roots of 23 (mod 7) are ±4 and the square roots of 23 (mod 11) are ±1. We use this to get the four square roots of 23 (mod 77).

From 4 \( \equiv \sqrt{23} \) (mod 7) and 1 \( \equiv \sqrt{23} \) (mod 11) we get \( x_1 \equiv \sqrt{23} \) (mod 77), where
\[ x_1 = 4 \times 11 \times y_1 + 1 \times 7 \times y_2 \pmod{77}, \]
and
\[ y_1 \equiv 11^{-1} \equiv 2 \pmod{7}, \]
\[ y_2 \equiv 7^{-1} \equiv 8 \pmod{11}. \]

Therefore \( x_1 = 4 \times 11 \times 2 + 1 \times 7 \times 8 \equiv 144 \equiv -10 \pmod{77}. \)

Similarly, the other square roots of 23 (mod 77) can be found to be 10, ±32.

The four possible plain-texts are found by subtracting \( B/2 \equiv 43 \) (mod 77). They are
\[ 10 - 43 \equiv 44 \pmod{77}, \]
\[ 67 - 43 \equiv 24 \pmod{77}, \]
\[ 32 - 43 \equiv 66 \pmod{77}, \]
\[ 45 - 43 \equiv 2 \pmod{77}. \]

12.1 Security of the Rabin Cryptosystem

The aim of this section is to show that being in possession of an algorithm that decrypts (maybe only part of the) cipher-text from a Rabin cryptosystem is equivalent to factoring
12.1. SECURITY OF THE RABIN CRYPTOSYSTEM

Since it is suspected that factorisation is “difficult,” one can expect that the design of such a decryption algorithm will be at least as hard as designing an algorithm that factors $n$.

So suppose that the attacker has a decryption algorithm $A$. We may then proceed as follows.

1. Choose a random $r$, $1 \leq r \leq n - 1$.
2. Compute $y = r^2 - B^2/4 \pmod{n}$.
3. Use $A$ to obtain a decryption $x$.
4. Let $x_1 = x + B/2$.
5. If $x_1 \equiv \pm r \pmod{n}$ the algorithm fails, else $\gcd(x_1 + r, n) = p$ or $q$ — a success.

We will see below that this algorithm factors $n$ with probability at least $1/2$. First we explain its working.

In step 2 we have

$$E_k \left( \frac{r - B}{2} \right) = \left( r - \frac{B}{2} \right) \left( r - \frac{B}{2} + B \right),$$

$$= \left( r - \frac{B}{2} \right) \left( r + \frac{B}{2} \right),$$

$$\equiv r^2 - \frac{B^2}{4} \pmod{n},$$

$$= y.$$ 

So $y$ is the encryption of $x = r - B/2$ or $x$ will be the decryption that we obtain in step 3.

In step 4, $x_1^2 \equiv (x + B/2)^2 \equiv r^2 \pmod{n}$. That is $x_1 \equiv \pm r \pmod{n}$ or $x_1 \equiv \pm \omega r \pmod{n}$, where $\omega \neq \pm 1$ is a square-root of 1 (mod $n$). This also says that $x_1^2 - r^2 \equiv 0 \pmod{n}$, that is $n|(x_1 - r)(x_1 + r)$.

From this we see why the algorithm fails if $x_1 \equiv \pm r \pmod{n}$: if this happens, then one of the two factors that $n$ divides is zero and this does not give us any useful information. On the other hand if $x_1 \equiv \pm \omega r \pmod{n}$, then $n$ does not divide either of the factors $(x_1 - r)$ or $(x_1 + r)$. Thus computing $\gcd(x_1 + r, n)$ (or $\gcd(x_1 - r, n)$) must give $p$ or $q$.

We now turn to estimating the probability of success of the algorithm. To do this we consider all possible $n - 1$ choices for the random $r$ in step 1.

Define an equivalence relation on $\mathbb{Z}_n \setminus \{0\}$ by $r_1 \sim r_2 \iff r_1^2 \equiv r_2^2 \pmod{n}$. The equivalence classes all have 4 elements: The equivalence class of $r$ is $\{\pm r, \pm \omega r\}$ and these
are all distinct by Proposition 12.0.9. Note that any two values in the same equivalence class give the same value for $y$ in step 2. In step 3, given $y$ the decryption algorithm returns $x$. By the calculation in step 4, $x_1$ is a member of the equivalence class of $r$. If it is $\pm r$, the algorithm fails. If it is $\pm \omega r$ the algorithm factors $n$ as explained above.

Since $r$ is chosen at random it is equally likely to be any of the four members of its equivalence class. Two of the four members leads to success, therefore the probability of success is at least $1/2$. 


Chapter 13

Factorisation Algorithms

The purpose of this chapter is to discuss some of the algorithms available for attempting to factor a given integer.

13.1 Pollard’s $p - 1$ Factoring Method (ca. 1974)

Suppose the odd number $n$ is to be factored and let $p$ be a prime divisor of $n$. This method uses the following result.

**Proposition 13.1.1.** If for any prime power $q|(p - 1)$ we have $q \leq B$, then $(p - 1)|B!$.

**Proof.**
Suppose $p - 1$ has the prime factorisation

$$p - 1 = p_1^{a_1}p_2^{a_2}\cdots p_k^{a_k},$$

$$= q_1q_2\cdots q_k,$$

where $q_i = p_i^{a_i}$. Since $p_i \neq p_j$ if $i \neq j$, we have $\gcd(q_i, q_j) = 1$ if $i \neq j$.

By hypothesis, $q_i \leq B$ for $i = 1, 2, \ldots, k$. Therefore $q_1q_2\cdots q_k$ are all distinct terms in $B! = 1 \cdot 2 \cdot 3 \cdots B$. Thus $p - 1 = q_1q_2\cdots q_k|B!$.

At this point we assume that we have found the $B$ in the proposition above.

Let $a \equiv 2^{B!} \pmod{n}$, then $a \equiv 2^{B!} \pmod{p}$. By Fermat’s Theorem we know $2^{p-1} \equiv 1 \pmod{p}$. Since $(p - 1)|B!$, $a \equiv 2^{B!} \equiv (2^{p-1})^t \equiv 1^t \equiv 1 \pmod{p}$. Therefore $p|(a - 1)$, furthermore $p|n$. These two statements together imply $p|\gcd(a - 1, n)$.

We now present the algorithm. It has as input an integer $n$ (the integer to be factored) and an integer $B$ (the “bound” on the size of the prime divisors of $p - 1$).

1. Compute $a \equiv 2^{B!} \pmod{n}$. 


2. Set \( d = \gcd(a - 1, n) \).

3. If \( 1 < d < n \) then \( d \mid n \) (success) else no factor of \( n \) is found.

**Example 13.1.2.** Let’s apply the algorithm to \( n = 36259 \) and \( B = 5 \). We need to calculate \( 2^B \pmod{36259} \). This can be done in five steps.

\[
\begin{align*}
2^1 & \equiv 2 \pmod{36259}, \\
2^2 & = (2^1)^2 \equiv 4 \pmod{36259}, \\
2^3 & = (2^2)^3 = 4^3 \equiv 64 \pmod{36259}, \\
2^4 & = (2^3)^4 = 64^4 \equiv 25558 \pmod{36259}, \\
2^5 & = (2^4)^5 = 25558^5 \equiv 22719 \pmod{36259}.
\end{align*}
\]

Therefore \( a \equiv 22719 \pmod{36259} \). Now we need \( \gcd(a - 1, n) = \gcd(22718, 36259) = 1 \). Since \( d = 1 \) in the algorithm we have failed to find a factor of \( n \).

Next we try a bigger \( B \), namely \( B = 10 \). So we need \( 2^{10} \pmod{36259} \). We calculate in the same manner as above and find that

\[
\begin{align*}
2^6 & \equiv 34839 \pmod{36259}, \\
2^7 & \equiv 7207, \pmod{36259} \\
2^8 & \equiv 21103, \pmod{36259} \\
2^9 & \equiv 25536, \pmod{36259} \\
2^{10} & \equiv 25251. \pmod{36259}
\end{align*}
\]

That is \( a \equiv 25251 \pmod{36259} \). So \( \gcd(a - 1, n) = \gcd(25250, 36259) = 101 \). From this we find that \( n = 36259 = 101 \times 359 \).

Note that in the example we have \( p = 101 \) so that \( p - 1 = 100 = 2^25^2 \). By the proposition we need \( B \) to be at least 25 for the algorithm to be guaranteed to work. In our case \( B = 10 \) worked, so the algorithm apparently does not need \( B \) to be as big as required by the proposition. The proposition just says that if \( B \) is this big the algorithm will work.
Chapter 14

The ElGamal Cryptosystem

In this chapter we introduce yet another public key cryptosystem. It is known as the ElGamal Cryptosystem \[?] and relies on discrete logarithms which we introduce first.

14.1 Discrete Logarithms (a.k.a indices)

Let \( p \) be a prime. Then \( \mathbb{Z}_p \) is a field and \( \mathbb{Z}_p^* = \mathbb{Z}_p \setminus \{0\} \) is a cyclic group under multiplication (mod \( p \)).

**Definition 14.1.1 (Primitive element (primitive root)).** A generator of \( \mathbb{Z}_p^* \) is called a **primitive element** of \( \mathbb{Z}_p^* \) (or also a primitive root of \( p \)).

So if \( a \) is a primitive element of \( \mathbb{Z}_p^* \), then \( a, a^2, a^3, \ldots, a^{p-1} \) (mod \( p \)) are just the elements \( 1, 2, 3, \ldots, p - 1 \) in some order. That is for each \( x \in \mathbb{Z}_p^* \) there is a number \( e \in \{1, 2, 3, \ldots, p - 1\} \) such that \( a^e \equiv x \) (mod \( p \)).

**Definition 14.1.2 (Discrete Logarithm).** We call \( e \) the **discrete logarithm** (or index) of \( x \) with respect to \( a \) if \( a^e \equiv x \) (mod \( p \)) and denote it by \( \log_a(x) \).

The problem of finding \( \log_a(x) \) in \( \mathbb{Z}_p \) is generally regarded as being difficult. Modular exponentiation is easy, but its inverse — discrete logarithm — is not. That is modular exponentiation is believed to be a one-way function.

14.2 The system

The ElGamal cryptosystem has the following features.

\( \times \) It is a public key system.
It is based on the difficulty of finding discrete logarithms.

It is non-deterministic: it involves a random integer \( k \) chosen by the sender.

The details of the system are as follows.

- Choose a prime \( p \) such that the determination of discrete logarithms in \( \mathbb{Z}_p^* \) is difficult.
- Choose a generator \( \alpha \) of \( \mathbb{Z}_p^* \), there are \( \phi(p-1) \) choices.
- The message space \( M = \mathbb{Z}_p^* \).
- The cipher space \( C = \mathbb{Z}_p^* \times \mathbb{Z}_p^* \).
- The keys \( k = (p, \alpha, e, \beta) \), where \( \beta = \alpha^e \pmod{p} \), that is \( \log\alpha(\beta) = e \pmod{p} \).
- \( p, \alpha \) and \( \beta \) are made public while \( e \) is kept private.
- The encryption function

\[
E_k(x) = (y_1, y_2),
\]

where

\[
y_1 = \alpha^k \pmod{p}, \\
y_2 = x\beta^k \pmod{p},
\]

and \( k \) is a random integer in \( \{0, 1, 2, \ldots, p-1\} \).

- The decryption function \( D_k(y_1, y_2) = y_2(y_1^e)^{-1} \pmod{p} \).

Firstly let’s check that decryption works. So with \( y_1 \) and \( y_2 \) as given above we find

\[
D_k(y_1, y_2) = y_2(y_1^e)^{-1} \pmod{p}, \\
= y_2([\alpha^k]^e)^{-1} \pmod{p}, \\
= y_2(\alpha^{ek})^{-1} \pmod{p}, \\
= y_2(\beta^k)^{-1} \pmod{p}, \\
= (x\beta^k)(\beta^k)^{-1} \equiv x \pmod{p}.
\]

Example 14.2.1. Let \( p = 23 \) and \( \alpha = 5 \) in an ElGamal cryptosystem. If \( e = 9 \), then

\[
\beta \equiv \alpha^e \equiv 5^9 \equiv 11 \pmod{23}.
\]

To send the plain-text \( x = 7 \):
14.3. ATTACKING THE ELGAMAL SYSTEM

• Choose a random $k \in \{0, 1, 2, \ldots, 22\}$. Suppose $k = 13$.

• Then compute

\[
y_1 \equiv 5^{13} \equiv 21 \pmod{23},
\]
\[
y_2 \equiv 7 \cdot 11^{13} \equiv 4 \pmod{23},
\]

and send $(21, 4)$.

On the receiving end suppose now that $(21, 4)$ is received. So firstly we compute
\[
y_1^e \equiv 21^9 \equiv 17 \pmod{23}.
\]
Now we need $(y_1^e)^{-1} = 17^{-1}$ in $\mathbb{Z}_{23}^*$. To do this we use the
Euclidean algorithm.

\[
23 = 1 \times 17 + 6,
\]
\[
17 = 2 \times 6 + 5,
\]
\[
6 = 1 \times 5 + 1.
\]

Working backwards through the results of the algorithm we find.

\[
1 = 5 - 6,
\]
\[
= 6 - (17 - 2 \times 6) = 3 \times 6 - 17,
\]
\[
= 3(23 - 17) - 17 = 3 \times 23 - 4 \times 17.
\]

From the last step we find $-4 \times 17 = 1 + (-3) \times 23 \equiv 1 \pmod{23}$. That is $17^{-1} \equiv -4 \equiv 19
(\pmod{23})$.

To complete decryption we compute $y_2(y_1^e)^{-1} = 4 \times 19 \equiv 76 \equiv 7 \pmod{23}$.

---

14.3 Attacking the ElGamal System by Computing Discrete Logarithms

This section will discuss one algorithm that can be used to compute discrete logarithms
and in so doing break the ElGamal system. The algorithm is due to Shanks.

Let $p$ be a prime and $\alpha$ a generator for $\mathbb{Z}_p^*$. In the ElGamal system we are given $\beta$ and
we want to find $e = \log \alpha \beta$. Let $m = \lceil \sqrt{p-1} \rceil$. By the division algorithm we can write
\[
e = mj + i, \; 0 \leq i \leq m - 1.
\]

Since $m = \lceil \sqrt{p-1} \rceil$ and $0 \leq e \leq p - 2, \; 0 \leq j \leq m - 1$. Now $\beta \equiv \alpha^e \pmod{p} \iff
\beta \equiv \alpha^{mj+i} \pmod{p} \iff \beta \alpha^{-i} \equiv \alpha^{mj} \pmod{p}$. This is the basis of Shanks’ algorithm.
1. Compute $\alpha^{mj} \pmod{p}$, $0 \leq j \leq m - 1$ and store the pairs $(j, \alpha^{mj})$ in a list sorted by increasing value of the second coordinate. The reason for storing the numbers in this way is to simplify searching through the list later.

2. Compute $\beta \alpha^{-i} \pmod{p}$, $0 \leq i \leq m - 1$ and store the pairs $(i, \beta \alpha^{-i})$ in a list sorted by increasing second coordinate.

3. Find a pair $(j, y)$ in the list from step 1 and a pair in the list from step 2 having the same second coordinate.

4. $e = \log_{\alpha} \beta = mj + i \pmod{p - 1}$

Note that in the last step we are computing $\pmod{p - 1}$. The reason for this is that the powers of $\alpha$ (the generator) are always going to be in the range $1, 2, 3, \ldots, p - 1$.

**Example 14.3.1.** Let $p = 23$ and $\alpha = 5$. We would like to find $\log_5(11)$.

Let $m = \lceil \sqrt{p - 1} \rceil = \lceil \sqrt{22} \rceil = 5$, so that $m - 1 = 4$.

We compute the following.

$$
\begin{align*}
5^{0.5} & \equiv 1 \pmod{23}, \\
5^{1.5} & \equiv 20 \pmod{23}, \\
5^{2.5} & \equiv 9 \pmod{23}, \\
5^{3.5} & \equiv 19 \pmod{23}, \\
5^{4.5} & \equiv 12 \pmod{23}.
\end{align*}
$$

Therefore list 1 is $(0, 1), (2, 9), (4, 12), (3, 19), (1, 20)$.

Next we need to compute $11 \cdot 5^{-i}$, $0 \leq i \leq 4$. To do this note that $5^{22} \equiv 1 \pmod{23}$. Therefore $5^{-i} \equiv 5^{22-i} \pmod{23}$.

$$
\begin{align*}
11 \cdot 5^{-0} & \equiv 11 \cdot 1 \equiv 11 \pmod{23}, \\
11 \cdot 5^{-1} & \equiv 11 \cdot 5^{21} \equiv 11 \cdot 14 \equiv 16 \pmod{23}, \\
11 \cdot 5^{-2} & \equiv 11 \cdot 5^{20} \equiv 11 \cdot 12 \equiv 17 \pmod{23}, \\
11 \cdot 5^{-3} & \equiv 11 \cdot 5^{19} \equiv 11 \cdot 8 \equiv 8 \pmod{23}, \\
11 \cdot 5^{-4} & \equiv 11 \cdot 5^{18} \equiv 11 \cdot 6 \equiv 20 \pmod{23}.
\end{align*}
$$

This gives list 2 as $(3, 8), (0, 11), (1, 16), (2, 17), (4, 20)$. 

Scanning through the lists we find $(1, 20)$ in the first list and $(4, 20)$ in the second list, giving

$$e = \log_5(11) = 1 \times m + 4 = 1 \times 5 + 4 \equiv 9 \pmod{22}.$$
CHAPTER 14. THE ELGAMAL CRYPTO SYSTEM


Chapter 15

Elliptic Curve Cryptography

This chapter discusses a generalisation of the ElGamal cryptosystem, namely elliptic curve cryptography. This was proposed by Koblitz and Miller [?, ?].

15.1 The ElGamal System in Arbitrary Groups

Let $G$ be a group and $\alpha \in G$. Let $H = \langle \alpha \rangle$, the cyclic subgroup of $G$ generated by $\alpha$.

For $\beta \in H$, we can define $\log_\alpha \beta = k \iff \alpha^k = \beta$ and $0 \leq k \leq |H| - 1$. If the problem of computing $\log_\alpha \beta$ from $G$, $\alpha$ and $\beta$ is “hard” we can set up a cryptosystem as follows.

- Let $(G, \circ)$ be a finite group and $\alpha \in G$.
- Let $H = \langle \alpha \rangle$ such that the problem of computing $\log_\alpha \beta$ ($\beta \in H$) is hard.
- The message space $M = G$.
- The cipher space $C = G \times G$.
- The keys $k = (G, \alpha, e, \beta)$, where $\beta = \alpha^e$, that is $\log_\alpha \beta = e$.
- $\alpha$ and $\beta$ are made public, while $e$ is kept private.
- For $x \in G$ the encryption function is $E_k(x) = (y_1, y_2)$, where

  \[
  y_1 = \alpha^k, \\
  y_2 = x \circ \beta^k,
  \]

  and $k$ is a secret random number with $0 \leq k \leq |G| - 1$. 


• The decryption function is $D_k(y_1, y_2) = y_2 \circ (y_1^e)^{-1}$.

Note that $x^k = x \circ x \circ \cdots \circ x$ ($k$ times).

The proof that encryption is one-to-one is exactly the same as in the case of the original ElGamal system.

An important point to note is that care is needed when choosing the group $G$ and $\alpha \in G$. The following illustrates this point.

Let $G = (\mathbb{Z}_n, +)$ and $\alpha$ a generator of $\mathbb{Z}_n$, that is $\gcd(\alpha, n) = 1$. Since exponentiation is repeated application of the group operation we have $x^k = x + x + \cdots + x = kx$ in this group. Therefore finding $\log_\alpha \beta$ means finding $k$ such that $k \alpha = \beta \pmod{n}$. That is $k = \alpha^{-1} \beta$, $\alpha^{-1}$ exists since $\gcd(\alpha, n) = 1$ and $\alpha^{-1}$ can be found by using the Euclidean algorithm.

**Example 15.1.1.** Let $G = \mathbb{Z}_{30}$ and $\alpha = 3$. Then $H = \{0, 3, 6, 9, \ldots, 27\}$. Suppose $e = 4$ is chosen, then $\beta = \alpha^e = 4 \times 3 = 12$. To send the plain-text 17 we proceed as follows.

- Choose a random $k \in \{0, 1, 2, \ldots, 29\}$, say $k = 5$.
- Compute $E_k(17) = (y_1, y_2)$, where
  $$ y_1 = \alpha^k = 5 \times 3 \equiv 15 \pmod{30}, $$
  $$ y_2 = x \circ \beta^k = 17 + 5 \times 12 \equiv 17 \pmod{30}. $$

We then send $(15, 17)$.

Assume now that $(15, 17)$ is received. The first step in the decoding is to compute $y_1^e$. We find that $y_1^e = 4 \times 15 = 60 \equiv 0 \pmod{30}$. Secondly we invert this to get $(y_1^e)^{-1} = -0 \equiv 0 \pmod{30}$. Lastly we compute $y_2 \circ (y_1^e)^{-1} = 17 + 0 \equiv 17 \pmod{30}$.

Suppose that Eve intercepts the cipher-text $(9, 16)$ and tries to decrypt it. She knows that $\alpha = 3$ and $\beta = 12$ (these are made public). She needs to find an $e$ such that $e \times \alpha = \beta$ in $\mathbb{Z}_{30}$. That is

$$ 3e \equiv 12 \pmod{30} \iff e \equiv 4 \pmod{30/\gcd(3, 30)} \iff e \equiv 4, 14, 24 \pmod{30}. $$

If Eve knew that $|H| = 10$, then she’d be done. On the other hand by Lagrange’s Theorem we have that $|H|$ divides $|G|$ and therefore $|H| \leq 15$ which rules out 24.

This shows that we might also want to keep order of $H$ secret.

### 15.2 Elliptic Curves

We begin with the definition of elliptic curves.
Definition 15.2.1 (Elliptic Curve). Let $p > 3$ be prime. The elliptic curve

$$y^2 = x^3 + ax + b,$$

over $\mathbb{Z}_p$ is the set of solutions to the congruence

$$y^2 \equiv x^3 + ax + b \pmod{p},$$

where $a, b \in \mathbb{Z}_p$ are constants such that

$$4a^3 + 27b^2 \not\equiv 0 \pmod{p},$$

together with a special point $O$ called the point at infinity.

An elliptic curve $E$ can be made into an Abelian group by using the following operation, where arithmetic is in $\mathbb{Z}_p$.

Let $P = (x_1, y_1)$ and $Q = (x_2, y_2)$ be points on $E$, then

$$P + Q = \begin{cases} O & \text{if } x_1 = x_2 \text{ and } y_1 = -y_2, \\ (x_3, y_3) & \text{otherwise}, \end{cases}$$

where

$$\begin{align*}
    x_3 &= \lambda^2 - x_1 - x_2, \\
    y_3 &= \lambda(x_1 - x_3) - y_1,
\end{align*}$$

and

$$\lambda = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} & \text{if } P \neq Q, \\
\frac{3x_1^2 + a}{2y_1} & \text{if } P = Q. \end{cases}$$

We also take $O$ to be the identity, that is $P + O = O + P = P$.

Note that the inverse of a point $(x, y)$ is the point $(x, -y)$.

We will skip the proof that $(E, +)$ is an Abelian group with identity $O$.

Example 15.2.2. Let $E$ be the elliptic curve $y^2 \equiv x^3 + x + 6$ over $\mathbb{Z}_{11}$.

The first thing we need to do is to find the points on $E$. We do this by taking each $x \in \mathbb{Z}_{11}$, computing $x^3 + x + 6$ and using Euler’s criterion to see whether the result is a
square. If it is a square, then since $11 \equiv 3 \pmod{4}$, $\sqrt{z} \equiv \pm z^{(11+1)/4} \equiv \pm z^3 \pmod{11}$. This produces the following table.

<table>
<thead>
<tr>
<th>$x$</th>
<th>$x^3 + x + 6$</th>
<th>square ?</th>
<th>$y$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>6</td>
<td>$6^5 \equiv 1$</td>
<td>$\times$</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>$8^5 \equiv 1$</td>
<td>$\times$</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>$5^5 \equiv 1$</td>
<td>$\checkmark$</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>$\checkmark$</td>
<td>$\pm 3^3 \equiv \pm 5$</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>$\checkmark$</td>
<td>$\pm 2$</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4</td>
<td>$\checkmark$</td>
<td>$\pm 2$</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>$\checkmark$</td>
<td>$\pm 3$</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>$\times$</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>4</td>
<td>$\checkmark$</td>
<td>$\pm 2$</td>
</tr>
</tbody>
</table>

Therefore $E$ has 13 points (including the point at infinity) and they are:

$O; (2, 4); (2, 7); (3, 5); (3, 6); (5, 2); (5, 9); (7, 2); (7, 9); (8, 3); (8, 8); (10, 2); (10, 9)$.

Since 13 is prime any nonidentity element will generate the group. Note also that $(E, +)$ is cyclic and isomorphic to $\mathbb{Z}_{13}$.

Let $\alpha = (2, 7)$. Then the powers of $\alpha$ are multiples of $\alpha$ in this group and we have the following.

$2\alpha = (2, 7) + (2, 7),$

$\lambda = (3(2^2) + 1)(2 \cdot 7)^{-1} \equiv 2(3^{-1}) \equiv 2 \cdot 4 \equiv 8 \pmod{11},$

$\therefore x_3 = 8^2 - 2 - 2 \equiv 60 \equiv 5 \pmod{11},$

$\therefore y_3 = 8(2 - 5) - 7 \equiv 8(-3) - 7 \equiv 8^2 - 7 \equiv 2 \pmod{11},$

$\therefore 2\alpha = (5, 2).$

Also,

$3\alpha = 2\alpha + \alpha = (5, 2) + (2, 7),$

$\lambda = (7 - 2)(2 - 5)^{-1} \equiv 5(-3)^{-1} \equiv 5(-4) \equiv 5(7) \equiv 2 \pmod{11},$

$\therefore x_3 = 2^2 - 5 - 2 \equiv -3 \equiv 8 \pmod{11},$

$\therefore y_3 = 2(5 - 8) - 2 \equiv 2(-3) - 2 \equiv 2 \pmod{11},$

$\therefore 3\alpha = (8, 3).$
The following table gives the “powers” of $\alpha$.

<table>
<thead>
<tr>
<th>$k$</th>
<th>$k\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(2, 7)</td>
</tr>
<tr>
<td>2</td>
<td>(5, 2)</td>
</tr>
<tr>
<td>3</td>
<td>(8, 3)</td>
</tr>
<tr>
<td>4</td>
<td>(10, 2)</td>
</tr>
<tr>
<td>5</td>
<td>(3, 6)</td>
</tr>
<tr>
<td>6</td>
<td>(7, 9)</td>
</tr>
<tr>
<td>7</td>
<td>(7, 2)</td>
</tr>
<tr>
<td>8</td>
<td>(3, 5)</td>
</tr>
<tr>
<td>9</td>
<td>(10, 9)</td>
</tr>
<tr>
<td>10</td>
<td>(8, 8)</td>
</tr>
<tr>
<td>11</td>
<td>(5, 9)</td>
</tr>
<tr>
<td>12</td>
<td>(2, 4)</td>
</tr>
</tbody>
</table>

In general one would like to be able to know how many points there are on a given elliptic curve over $\mathbb{Z}_p$. This is needed so that one may be able to construct a correspondence between plain-text and the points on the curve. The following theorem gives bounds on the number of points.

**Theorem 15.2.3 (Hasse’s Theorem).** Let $p > 3$ be prime and $E$ an elliptic curve over $\mathbb{Z}_p$. Then the number, $N(E)$, of points on $E$ satisfies

$$p + 1 - 2\sqrt{p} \leq N(E) \leq p + 1 + 2\sqrt{p}.$$

We also have the following interesting result.

**Theorem 15.2.4.** Let $p > 3$ be prime and $E$ an elliptic curve over $\mathbb{Z}_p$. Then there exist integers $n_1$ and $n_2$ such that

$$(E, +) \cong \mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2}.$$ 

Furthermore $n_2|n_1$ and $n_2|(p - 1)$.

The last theorem implies that there exists a cyclic subgroup of $(E, +)$ isomorphic to $\mathbb{Z}_{n_1}$. We may be able to use this in an ElGamal system if we can find it.

**Example 15.2.5.** (Continued)
We use the elliptic curve in the previous example to set up an ElGamal system with $\alpha = (2, 7)$ and an exponent $e = 7$. So $\beta = \alpha^e = 7 \cdot \alpha = (7, 2)$ (from table).

$E_k(x) = (k\alpha, k\beta + x) = (k(2, 7), k(7, 2) + x)$, where $x \in E$ and $k$ is chosen at random from $\{0, 1, 2, \ldots, 12\}$.

$D_k(y_1, y_2) = y_2 - ey_1 = y_2 - 7y_1$.

To encrypt $(10, 9)$ (which is a point on the curve):

1. Choose $k$, say $k = 3$.
2. Compute

\[
\begin{align*}
y_1 &= 3\alpha = (8, 3), \\
y_2 &= 3(7, 2) + (10, 9), \\
     &= 3(7\alpha) + (10, 9), \\
     &= 21\alpha + (10, 9), \\
     &= 8\alpha + (10, 9), \\
     &= (3, 5) + (10, 9), \\
     &= (10, 2).
\end{align*}
\]

Therefore we send the cipher-text $((8, 3), (10, 2))$.

Suppose now that we are the receiver of the cipher-text $((8, 3), (10, 2)) = (y_1, y_2)$ and that we would like to decrypt it. So we would have to compute

\[
D_k(y_1, y_2) = y_2 \circ (y_1^e)^{-1},
\]

\[
\begin{align*}
     &= (10, 2) - 7(8, 3), \\
     &= (10, 2) - 7(3\alpha), \\
     &= (10, 2) - 21\alpha = (10, 2) - 8\alpha, \\
     &= (10, 2) + (-8(2, 7)) = (10, 2) + (-3, 5), \\
     &= (10, 2) + (3, -5) = (10, 2) + (3, 6), \\
     &= (10, 9).
\end{align*}
\]

\[\square\]

## 15.3 The Menezes-Vanstone Cryptosystem

In this section we describe another cryptosystem, the Menezes-Vanstone system [?], that also uses elliptic curves. This system is made up as follows.
• An elliptic curve $E$ over $\mathbb{Z}_p$ with $p > 3$ is used such that $(E, +)$ has a cyclic subgroup $H = \langle \alpha \rangle$ for which computing discrete logarithms is “hard.”

• The message space $M = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.

• The cipher space $C = E \times \mathbb{Z}_p^* \times \mathbb{Z}_p^*$.

• The keys $k = (E, \alpha, e, \beta)$ where $\beta = e\alpha$, that is $\log_\alpha \beta = e$, and $\alpha \in E$.

• $\alpha$ and $\beta$ are made public, while $e$ is kept secret.

• The encryption function $E_k(x_1, x_2) = (y_0, y_1, y_2)$, where

$$
y_0 = k\alpha \text{ where } k \text{ is a random integer,}
$$

$$
y_1 = c_1x_1 \pmod{p},
$$

$$
y_2 = c_2x_2 \pmod{p},
$$

and

$$(c_1, c_2) = k\beta \in E.$$

• The decryption function $D_k(y_0, y_1, y_2) = (x', x'')$, where

$$
x' = y_1c_1^{-1} \pmod{p},
$$

$$
x'' = y_2c_2^{-1} \pmod{p},
$$

and

$$(c_1, c_2) = ey_0.$$

We now show that the decryption function is in fact the inverse of the encryption function. So suppose (as above) that we have encrypted the message $(x_1, x_2)$ as $(y_0, y_1, y_2)$. Then computing $D_k(y_0, y_1, y_2)$ yields the following.

$$
ey_0 = ek\alpha = k\beta = (c_1, c_2),
$$

$$
x' = y_1c_1^{-1} \equiv c_1x_1c_1^{-1} \equiv x_1 \pmod{p},
$$

$$
x'' = y_2c_2^{-1} \equiv c_2x_2c_2^{-1} \equiv x_2 \pmod{p}.
$$

Therefore $D_k(y_0, y_1, y_2) = (x_1, x_2)$, as desired.

Example 15.3.1. Let $E$ be the elliptic curve $y^2 \equiv x^3 + x + 6$ over $\mathbb{Z}_{11}$ — the same one as in the previous example. We also choose $\alpha = (2, 7)$ and $e = 7$. Therefore $\beta = 7\alpha = (7, 2)$.

To encrypt the message $(9, 1)$ (which in this case is an element of $\mathbb{Z}_{11}^* \times \mathbb{Z}_{11}^*$ and not of the curve as in the previous example) we proceed as follows.
1. Choose a random integer \( k \), say \( k = 6 \).

2. Compute

\[
\begin{align*}
y_0 &= k\alpha = 6(2, 7) = (7, 9), \\
k\beta &= 6(7, 2) = 6 \cdot 7 \cdot \alpha = 42\alpha = 3\alpha = (8, 3), \\
\therefore c_1 &= 8 \text{ and } c_2 = 3, \\
y_1 &= c_1 x_1 = 8 \cdot 9 \equiv 6 \pmod{11}, \\
y_2 &= c_2 x_2 = 3 \cdot 1 \equiv 3 \pmod{11}.
\end{align*}
\]

So we send \(((7, 9), 6, 3)\) as the encrypted message.

Suppose now that \(((7, 9), 6, 3)\) is received and that we would like to decrypt it. We therefore compute the following.

\[
\begin{align*}
ey_0 &= 7(7, 9) = 7 \cdot 6\alpha = 42\alpha = 3\alpha = 2(2, 7) = (8, 3), \\
\therefore c_1 &= 8 \text{ and } c_2 = 3, \\
x' &= y_1 c_1^{-1} = 6 \cdot 8^{-1} \equiv 6 \cdot 7 \equiv 42 \equiv 9 \pmod{11}, \\
x'' &= y_2 c_2^{-1} = 3 \cdot 3^{-1} \equiv 1 \pmod{11}.
\end{align*}
\]

Therefore we decrypt this as \((9, 1)\).
This chapter describes a system that was developed around 1978, but that was subsequently broken a few years later. It remains an interesting system in spite of this that can be used in conjunction with other systems.

The Merkle-Hellman \[?] or “knapsack” cryptosystem revolves around the subset sum problem.

**Definition 16.0.2 (Subset sum problem).** Given positive integers \(s_1, s_2, \ldots, s_n\) and \(T\) — the sizes and target — try to find a binary vector \(x = (x_1, x_2, \ldots, x_n)\) such that
\[x_1s_1 + x_2s_2 + \cdots + x_n s_n = T.\]

This problem is known to be NP-complete in general, but there are easy special cases.

**Definition 16.0.3 (Superincreasing list).** A list \((s_1, s_2, \ldots, s_n)\) is called superincreasing if
\[s_j > \sum_{i=1}^{j-1} s_i,\]
for \(2 \leq j \leq n.\)

If the list of sizes in the subset sum problem is superincreasing, then the problem is easy to solve, as demonstrated by the following algorithm (shown in Figure 16) that finds the binary vector in this case.

The reason why the algorithms works is simply because \(s_n\) is greater than the sum of all the other sizes, so if \(s_n \leq T\) it has to be chosen to try and reach the target — all the other sizes put together is not enough to reach the target. Once \(s_n\) is chosen (or discarded
for $i$ from $n$ down to 1
{
    if $T \geq s_i$
    then $x_i = 1$ and replace $T$ by $T - s_i$
    else $x_i = 0$
}

if $\sum x_is_i = T$ then a solution has been found
otherwise no solution exists

Figure 16.1: Algorithm for solving the subset problem for a superincreasing sequence

if $s_n > T$) and the size of $T$ reduced to $T - s_n$ (or left as $T$ if $s_n$ is not chosen) we have a
new subset sum problem with a smaller target (or the same) and $n-1$ sizes that have to
be chosen; we just repeat this procedure. To see the uniqueness, realize that each $s_i$ that
was chosen had to be chosen — there was no possibility of not using it.

The Merkle-Hellman “knapsack” system is made up of the following components.

• $S = (s_1, s_2, \ldots, s_n)$ a superincreasing list of integers.

• $p > \sum s_i$ is a prime number and $a \in \mathbb{Z}_p^*$.

• $t = (t_1, t_2, \ldots, t_n)$ is defined by

$$t_i = as_i \pmod{p}.$$  

That is each $t_i$ is taken to be the least residue of $as_i \pmod{p}$.

• The message space $M = \{0, 1\}^n$.

• The cipher-text space $C = \{0, 1, 2, \ldots, n(p-1)\}$.

• The keys $k = (S, p, a, t)$, where $t$ is made public and $S, p$ and $a$ are kept private.

• The encryption function

$$E_k(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} x_it_i.$$  

• The decryption function $D_k(y) = (x_1, x_2, \ldots, x_n)$, where $x_1, x_2, \ldots, x_n$ equals the
solution to the subset sum problem with target sum $T = a^{-1}y \pmod{p}$ and sizes $S$.  

First let’s show that the decryption function is the inverse of the encryption function. So suppose that the binary vector \( x_1, x_2, \ldots, x_n \) is encrypted as \( y = \sum x_i t_i \), as shown above. We need to show that \( \sum x_i s_i = T = a^{-1} y \) (mod \( p \)).

\[
y = x_1 t_1 + x_2 t_2 + \cdots + x_n t_n,
\]
\[
\therefore a^{-1} y \equiv x_1 a^{-1} t_1 + x_2 a^{-1} t_2 + \cdots + x_n a^{-1} t_n \pmod{p},
\]
\[
\equiv x_1 s_1 + x_2 s_2 + \cdots + x_n s_n \pmod{p},
\]
\[
= x_1 s_1 + x_2 s_2 + \cdots + x_n s_n.
\]

The equality in the last step follows from the fact that \( p > \sum s_i \). Also since \((s_1, s_2, \ldots, s_n)\) is superincreasing the solution is unique.

**Example 16.0.4.** \( S = (2, 5, 10, 25) \) is superincreasing. Choose \( p = 53 \) and \( a = 10 \). The public list of sizes, \( t \), is

\[
t_1 = 20 \pmod{53},
\]
\[
t_2 = 50 \pmod{53},
\]
\[
t_3 = 100 \equiv 47 \pmod{53},
\]
\[
t_4 = 250 \equiv 38 \pmod{53}.
\]

Thus \( t = (20, 50, 47, 38) \).

To encrypt \((1, 0, 1, 1)\) compute the following

\[
1 \cdot 20 + 0 \cdot 50 + 1 \cdot 47 + 1 \cdot 38 = 105.
\]

To decrypt 105 we need \( 10^{-1} \equiv 16 \pmod{53} \). Then \( T = 16 \times 105 \equiv 16 \times -1 \equiv 37 \pmod{53} \). We now solve for the subset sum problem with the list \((2, 5, 10, 25)\) and target sum 37.

\[
25 \leq 37 \Rightarrow x_4 = 1 \text{ new target : } 37 - 25 = 12,
\]
\[
10 \leq 12 \Rightarrow x_3 = 1 \text{ new target : } 12 - 10 = 2,
\]
\[
5 \geq 2 \Rightarrow x_2 = 0 \text{ new target : } 2 - 0 = 2,
\]
\[
2 \leq 2 \Rightarrow x_1 = 1 \text{ new target : } 2 - 2 = 0.
\]

Since \(1 \cdot 2 + 0 \cdot 5 + 1 \cdot 10 + 1 \cdot 25 = 37\) we have achieved the target and the solution is correct. 

\[\blacksquare\]
Chapter 17

The McEliece Cryptosystem

In the previous chapter we saw an example of a cryptosystem that was constructed using an easy instance of a “hard” problem. The system that we present in this chapter is based on the same idea and appeared in [?]. Here the hard problem is that of decoding a binary linear code where the generator matrix is given. As an easy special case we consider the class of Goppa codes (which include the Hamming codes).

The Goppa codes have the following properties.

\( \times \) They are \([2^m, 2^m - mt, 2t + 1]\)-codes.

\( \times \) They have efficient encoding and decoding algorithms.

\( \times \) There exist many inequivalent codes in this family all with the same parameters.

The McEliece cryptosystem has the following components.

\( \bullet \) \( G \) is a generator matrix for a \([2^m, 2^m - mt, 2t + 1]\) Goppa code.

\( \bullet \) \( S \) is a \( k \times k \) matrix that is invertible over \( \mathbb{Z}_2 \) where \( k = 2^m - mt \).

\( \bullet \) \( P \) is an \( n \times n \) permutation matrix where \( n = 2^m \).

\( \bullet \) \( G' = SGP \).

\( \bullet \) The message space \( M = (\mathbb{Z}_2)^k \) \(- k\)-tuples over \( \mathbb{Z}_2 \).

\( \bullet \) The cipher-text space \( C = (\mathbb{Z}_2)^n \).

\( \bullet \) The keys \( k = (G, S, P, G') \), where \( G' \) is made public and \( S, P \) and \( G \) are kept private.

\( \bullet \) The encryption function is \( E_k(x) = xG' + e \), where \( x \in (\mathbb{Z}_2)^k \) and \( e \in (\mathbb{Z}_2)^n \) is a random error of weight \( t \).
• The decryption function is a four step process that operates on \( y \in (\mathbb{Z}_2)^n \) as follows.

   (i) Compute \( y_1 = yP^{-1} \).

   (ii) Decode \( y_1 \), obtaining \( y_1 = x_1 + e_1 \), where \( x_1 \in C \).

   (iii) Compute \( x_0 \in (\mathbb{Z}_2)^k \) such that \( x_0G = x_1 \).

   (iv) Compute \( x = x_0S^{-1} \).

Let’s show that decryption actually reverses the encryption.
Assume that \( x \) is encrypted as \( E_k(x) = y = xG' + e = x(SGP) + e \). Therefore
\( yP^{-1} = x(SGP)P^{-1} + eP^{-1} = xSG + eP^{-1} = x_1 + e_1 \), where \( x_1 = (xS)G \) that is \( x_0 = xS \).
Thus \( x_0S^{-1} = xS S^{-1} = x \).

\textbf{Example 17.0.5.} As our Goppa code we choose the \([7,4,3]\) Hamming code that has generator matrix
\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 & 1 & 1
\end{bmatrix}.
\]

Furthermore we choose
\[
S = \begin{bmatrix}
1 & 1 & 0 & 1 \\
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0
\end{bmatrix},
\]

to be invertible over \( \mathbb{Z}_2 \) and
\[
P = \begin{bmatrix}
0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0
\end{bmatrix}.
\]
The public generating matrix then is
\[
G' = SGP = \begin{bmatrix}
1 & 1 & 1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 1 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}.
\]
Suppose now that we would like to encrypt the plain-text \( x = (1, 1, 0, 1) \). Since the Hamming code is a single error correcting code our random error vector has to be of weight one. Say we choose \( e = (0, 0, 0, 1, 0, 0) \). The corresponding cipher-text is

\[
y = xG' + e,
\]

\[
= (1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0) + (0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
Appendix A

Assignments

MATH 433D/550
Assignment 1

Due: Monday, January 21, 2002, at the start of class.

1. Recall that the reliability of a binary symmetric channel (BSC) is the probability $p$, $0 \leq p \leq 1$, that the digit sent is the digit received.
   (a) [1] Would you use a BSC with $p = 0$? If so, how? What about a BSC with $p = 1/2$?
   (b) [1] Explain how to convert a BSC with $0 \leq p < 1/2$ into a channel with $1/2 < p \leq 1$.

2. Let $C$ be a binary code of length $n$. We define the information rate of $C$ to be the number $i(C) = \frac{1}{n} \log_2(|C|)$. This quantity is a measure of the proportion of each codeword that is carrying the message, as opposed to redundancy that has been added to help deal with errors.
   (a) [1] Prove that if $C$ is a binary code, then $0 \leq i(C) \leq 1$. In questions (b) and (c), suppose you are using a BSC with reliability $p = 1 - 10^{-8}$ and that digits are transmitted at the rate $10^7$ digits per second.
   (b) [3] Suppose the code $C$ contains all of the binary words of length 11. (i) What is the information rate of $C$? (ii) What is the probability that a word is transmitted incorrectly and the errors are not detected? (iii) If the channel is in constant use, about how many words are (probably) transmitted incorrectly each day?
   (c) Now suppose the code $C'$ is obtained from $C$ by adding an extra (parity check) digit to the words in $C$, so that the number of 1's in each codeword is even.
      (i) What is the information rate of $C''$?
      (ii) What is the probability that a word is transmitted incorrectly and the transmission errors go undetected?
(iii) If the channel is in constant use, about how long do you expect must pass between undetected incorrectly transmitted words? Express your answer as a number of days.

3. [4] Establish the following three properties of the Hamming distance (for binary codes $C$):
   
   (a) $d(u, w) = 0$ if and only if $u = w$.
   (b) $d(v, w) = d(w, v)$.
   (c) $d(v, w) \leq d(v, u) + d(u, w)$, $\forall u \in C$.

4. Let $C$ be the code consisting of all binary words of length 4 that have even weight.
   
   (a) [2] Find the error patterns $C$ detects.
   (b) [2] Find the error patterns $C$ corrects.

5. [2] Prove that the minimum distance of a linear code is the smallest weight of a non-zero codeword.

6. [6] Prove that a code can simultaneously correct all error patterns of weight at most $t$, and unambiguously detect all non-zero error patterns of weight $t + 1$ to $d$ (where $t \leq d$) if and only if it has minimum distance at least $t + d + 1$. (For example, consider $C = \{000, 111\}$. This single error correcting code detects all non-zero error patterns of weight at most 2. But, if 000 is sent and 110 is received, then only one error is detected and the received word is incorrectly decoded as 111. The ambiguity here is that it is not clear whether the error pattern is 110 or 001.)
Question 1

(a) Yes. Since the probability of error equals 1, each bit is received incorrectly. By inverting each bit at the receiver we obtain the original bit. On the other hand a BSC with $p = 1/2$ is completely unreliable. The probability of seeing a specific bit at the receiving end equals $1/2$, this situation is similar to flipping an unbiased coin at the receiver and recording heads as 1 and tails as 0. Thus the channel is not able to carry any information.

(b) Simply invert each bit at the receiving end.

Question 2

(a) We have $1 \leq |C| \leq 2^n$ (one codeword; all words of length $n$). Therefore since log is monotone increasing

$$\frac{\log_2(1)}{n} \leq \frac{\log_2 |C|}{n} \leq \frac{\log_2(2^n)}{n},$$

so that

$$0 \leq i(C) \leq 1.$$

(b) $n = 11$ and $|C| = 2^{11}$. Then

(i)

$$i(C) = \frac{\log_2 |C|}{n} = 1.$$

(ii) This code cannot detect any errors since all words of length 11 are codewords — any codeword will be changed into another codeword by any error pattern. The undetected error probability then is ($q = 1 - p$ the error probability)

$$P_e(C) = \binom{11}{1} q^{10} p + \binom{11}{2} q^9 p^2 + \binom{11}{3} q^8 p^3 + \binom{11}{4} q^7 p^4 + \binom{11}{5} q^6 p^5 + \binom{11}{6} q^5 p^6 + \binom{11}{7} q^4 p^7 + \binom{11}{8} q^3 p^8 + \binom{11}{9} q^2 p^9 + \binom{11}{10} q p^{10} + \binom{11}{11} p^{11},$$

$$= 1.1 \times 10^{-7}.$$
APPENDIX A. ASSIGNMENTS

(iii) $10^7$ bits per second implies $864 \times 10^9$ bits per day, this is approximately 78545454545 words (of length 11) per day. We expect a fraction of $1.1 \times 10^{-7}$ of these to be in error. Therefore about 8640 words per day are in error.

(c) (i) $|C'| = 2^{11}$; $n' = 12$, so that

$$i(C') = \frac{\log_2 |C'|}{n'} = \frac{11}{12}$$

(ii) This parity check code can detect all error patterns of odd weight. On the other hand an error pattern of even weight results in a received word of even weight (either two ones cancel or both ones contribute to the weight of the received word). Thus even weight error patterns are not detectable. Therefore

$$P_e(C') = \sum_{i=0}^{12} \binom{12}{i} q^i p^{12-i},$$

$$= 6.6 \times 10^{-15}.$$ 

(iii) We are transmitting about $72 \times 10^9$ words per day. From part (ii) we know that the undetected error rate is $6.6 \times 10^{-15}$. Therefore about $(72 \times 10^9)(6.6 \times 10^{-15}) = 4.752 \times 10^{-4}$ words per day are in error. This is the same as about 1 word error every 2104 days ($\approx 5.77$ years).

**Question 3**

(a) $d(u, w) = 0 \iff u = w$.

$\Rightarrow$: Let $u, w \in C$ and $d(u, w) = 0$. This means that $u$ and $w$ differ in 0 places, therefore $u = w$.

$\Leftarrow$: $d(u, u) = \text{wt}(u + u) = \text{wt}(0) = 0$.

(b) $d(v, w) = d(w, v)$.

$$d(v, w) = \text{wt}(v + w) = \text{wt}(w + v) = d(w, v).$$

(c) $d(v, w) \leq d(v, u) + d(u, w)$, $\forall u \in C$.

$$d(v, w) = \text{wt}(v + w) = \text{wt}(v + u + u + w) = d(v + u, u + w) \leq \text{wt}(v + u) + \text{wt}(u + w) = d(v, u) + d(u, w).$$

The last inequality follows from the fact that $d(x, y) \leq \text{wt}(x) + \text{wt}(y)$ for any $x, y \in C$ since the ones in $x$ could line up with the zeros in $y$ and vice versa.
Question 4

\[ C = \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\} \]

(a) Any odd weight error pattern is detectable since it changes a codeword (of even weight) into a word of odd weight. Even error patterns are not detectable since they change a codeword into a word if even weight, which is a codeword. Thus the detectable error patterns are

\[ \{0001, 0010, 0100, 0111, 1000, 1011, 1101, 1110\} \]

(b) It is easily verified that this code is linear and so its minimum distance is 2. To be able to correct all error patterns of weight \(t\) we need a minimum distance of at least \(2t + 1\). In our case this implies \(t = 1/2\). This then says that no errors of weight greater than 1 will be correctable but some errors of weight one may still be correctable. Adding a weight one error pattern to any of the codewords produces a received word that is closer to more than one codeword, therefore decoding becomes impossible. Thus the code cannot correct any errors.

Question 5

Let \(C\) be a linear code, then

\[ d_{\text{min}}(C) = \min_{u, v \in C} d(u, v) = \min_{u, v \in C} wt(u + v) = \min_{w \in C} wt(w). \]

The last step follows from the fact that \(C\) is linear so that the sum of two codewords is again a codeword. Further, all nonzero codewords can be formed in this way since if \(v\) is a nonzero codeword, then \(v \neq 0\) and \(v = v + \bar{0}\).

Question 6

A code \(C\) can simultaneously correct all error patterns of weight at most \(t\) and detect all nonzero error patterns of weight \(t + 1\) to \(d (d \geq t) \iff d_{\text{min}}(C) \geq t + d + 1\).

\[ \Leftarrow: \]

Let \(d_{\text{min}}(C) \geq t + d + 1 \geq 2t + 1\), since \(d \geq t\). By a previous theorem we now know that \(C\) can correct all error patterns of weight \(t\). Let \(u \in C\) be sent, \(w\) be received and \(z\) be an error pattern such that \(t + 1 \leq wt(z) \leq d\). Therefore \(t + 1 \leq d(u, w) \leq d\). Let \(v \in C\)
and \( v \neq u \), then

\[
\begin{align*}
  d(u, v) & \leq d(u, w) + d(w, v) \\
  \therefore d(w, v) & \geq d(u, v) - d(u, w) \\
  & \geq t + d + 1 - d = t + 1.
\end{align*}
\]

Therefore \( w \) lies outside the \( t \)-ball around \( v \). Since \( v \) was an arbitrary codeword, \( w \) does not lie inside any of the \( t \)-balls that surround the codewords of \( C \). This implies that \( w \) will not be decoded to any codeword, as only words that lie inside a \( t \)-ball of a codeword will be decoded to that codeword. We are therefore able to detect that more than \( t \), but less than \( d + 1 \) errors have occurred.

\( \Rightarrow \):

Let \( C \) be a code that can simultaneously correct \( t \) and detect \( t + 1 \) to \( d \) \((d \geq t)\) errors. The fact that \( C \) can correct \( t \) errors implies that \( d_{\text{min}}(C) \geq 2t + 1 \). Let \( u, v \in C \) such that \( d(u, v) = d_{\text{min}}(C) \). Place a ball of radius \( d \) around \( u \) and a ball of radius \( t \) around \( v \). By the error detection property of the code, any error pattern of weight \( d \) will not be able to change \( u \) in such a way that it lies inside the \( t \)-ball around \( v \). Therefore the two balls above are disjoint. This implies that \( d(u, v) = d_{\text{min}}(C) \geq t + d + 1 \).
1. Let \( S = \{11000, 01111, 11110, 01010\} \), and \( C = \langle S \rangle \).
   (a) [3] Find a both generator matrix and a parity check matrix for \( C \).
   (b) [1] What is the dimension of \( C \), and of \( C^\perp \)?
   (c) An \((n, k, d)\)-code is a linear code of length \( n \), dimension \( k \), and minimum distance \( d \). Find the parameters \((n, k, d)\) for \( C \) and \( C^\perp \). (You may want to use the result of question 3 below.)

2. Let \( C \) be the linear code with generator matrix
   \[
   G = \begin{bmatrix}
   1 & 0 & 0 & 0 & 1 & 1 & 1 \\
   0 & 1 & 0 & 0 & 1 & 1 & 0 \\
   0 & 0 & 1 & 0 & 1 & 0 & 1 \\
   0 & 0 & 0 & 1 & 0 & 1 & 1
   \end{bmatrix}.
   \]
   Assign data to the words in \( K^4 \) by letting the letters \( A, B, \ldots, P \) correspond to 0000, 0001, \ldots, 1111, respectively.
   (a) [2] Encode the message CALL HOME (ignore the space).
   (b) [5] Suppose the message 0111000, 0110110, 1011001, 1011111, 1100101, 0100110 is received. Decode this message using the CMLD procedure involving syndromes and cosets (syndrome decoding). How many errors occurred in transmission?

3. [4] Let \( H \) be a parity check matrix for a linear code \( C \). Prove that \( C \) has minimum distance \( d \) if and only if any set of \( d - 1 \) rows of \( H \) is linearly independent and some set of \( d \) rows of \( H \) is linearly dependent.

4. [4] Suppose that \( C \) is a linear code of length \( n \) with minimum distance at least \( 2t + 1 \). Prove that every coset of \( C \) contains at most one word of weight \( t \) or less. Use this to show that syndrome decoding corrects all error patterns of weight at most \( t \).

5. (a) [3] Prove the Singleton bound: For an \((n, k, d)\)-code, \( d - 1 \leq n - k \). (Hint: consider a parity check matrix.)
   (b) [4] An \((n, k, d)\)-code is called maximum distance separable (MDS) if equality holds in the Singleton bound, that is, if \( d = n - k + 1 \). Prove that the following statements are equivalent.
(1) $C$ is MDS,
(2) every $n - k$ rows of the parity check matrix are linearly independent,
(3) every $k$ columns of the generator matrix are linearly independent.

(c) [3] Show that the dual of an $(n, k, n + k - 1)$ MDS code is an $(n, n - k, k + 1)$ MDS code.
MATH 550  
Assignment 2  
Solutions

**Question 1**

(a) We write the rows of $S$ into the matrix $A$ and put it in reduced row echelon form:

\[
G = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Add row 1 to row 3

\[
G = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Add row 2 to row 4

\[
G = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 0 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Add row 3 to row 4

\[
G = \begin{bmatrix}
1 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

Therefore the generator matrix is

\[
G = \begin{bmatrix}
1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
\]

and the parity check matrix is

\[
H = \begin{bmatrix}
1 \\
1 \\
1 \\
1 \\
\end{bmatrix}
\]

(b) $\dim(C) = 4$ and $\dim(C^\perp) = 1$.

(c) Using question 3 we see that $d_{\min}(C) = 2$. Further, $G^T$ is a parity check matrix for $C^\perp$ from which we see (using question 3 again) that $d_{\min}(C^\perp) = 5$. Therefore $C$ is a $[5, 4, 2]$-code and $C^\perp$ is a $[5, 1, 5]$-code.
Question 2

(a) The message CALL HOME is encoded as: 0010101 0000000 1011001 1011001 0111000 111100 1100001 0100110.

(b) The parity check matrix for this code is

$$H = \begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1
\end{bmatrix}.$$ 

This is a parity check matrix for the (7, 4, 3) Hamming code. We can also see using question 3 that the minimum distance is 3. Therefore this code can only correct one error and so the coset leaders are the words of weight at most 1. The syndrome for a coset leader of weight 1 will be the row of $H$ corresponding to the position where the coset leader has a one. The received words have the following syndromes:

<table>
<thead>
<tr>
<th>Received</th>
<th>Syndrome</th>
</tr>
</thead>
<tbody>
<tr>
<td>0111000</td>
<td>000</td>
</tr>
<tr>
<td>0110110</td>
<td>101</td>
</tr>
<tr>
<td>1011001</td>
<td>000</td>
</tr>
<tr>
<td>1011111</td>
<td>110</td>
</tr>
<tr>
<td>1100101</td>
<td>100</td>
</tr>
<tr>
<td>0100110</td>
<td>000</td>
</tr>
</tbody>
</table>

The syndrome 101, which is row three of $H$, shows that an error occurred in the third position of the second received word. The syndrome 110, which is the second row of $H$, shows that an error occurred in the second position of the fourth received word. Similarly, the fifth received word has an error in position five. Of course, the zero syndrome indicates no errors. Decoding then produces the following words: 0111 0100 1011 1111 1100 0100. This corresponds to: HELP ME.

Question 3

Let $C$ be a linear code and $H$ its parity check matrix. Then $C$ has minimum distance $d$ if and only if any set of $d - 1$ rows of $H$ is linearly independent and some set of $d$ rows is
linearly dependent.

Proof:

$\leq$:
Let $r_{i_1}, r_{i_2}, \ldots, r_{i_d}$ be the set of linearly dependent rows. That is $r_{i_1} + r_{i_2} + \cdots + r_{i_d} = 0$. Let $x = x_1 x_2 x_3 \ldots x_n \in K^n$ such that $x_j = 1$ if $j \in \{i_1, i_2, \ldots, i_k\}$ and $x_j = 0$ otherwise. Then $xH = r_{i_1} + r_{i_2} + \cdots + r_{i_d} = 0$. This implies that $wt(x) = d$ and $x \in C$. Let $z \in C$ with $z \neq \bar{0}$. Then $zH = 0$. Assume $wt(z) \leq d - 1$. This would imply that a sum of $d - 1$ or fewer rows of $H$ is equal to zero. By hypothesis any set of $d - 1$ rows of $H$ is supposed to be linearly independent (any smaller set will also be), so that their sum will be nonzero — a contradiction. Thus $wt(z) \geq d$, so that $d_{\min}(C) = d$.

$\Rightarrow$:
Let $d_{\min}(C) = d$. Then there exists $x \in C$ such that $wt(x) = d$ and $x \neq \bar{0}$. Further, $xH = 0$, which implies that a sum of $d$ rows of $H$ equals zero meaning that they are linearly dependent. Since $wt(z) \geq d$ for all $z \in C$, this means that if $y \in K^n$ and $wt(y) \leq d - 1$, $yH \neq 0$. Therefore a sum of any set of $d - 1$ or fewer rows of $H$ is linearly independent.

Question 4

Let $C$ be a linear code of length $n$ and $d_{\min}(C) \geq 2t + 1$. One of the cosets of $C$ is $C$ itself. For every $x \in C$ with $x \neq \bar{0}$, $wt(x) \geq 2t + 1$. Therefore in the coset $C$, $\bar{0}$ is the only word of that has weight at most $t$. Let $y \in K^n$ with $wt(y) \leq t$ and consider the coset $y + C$. Let $z \in y + C$, therefore $z = y + c$, $c \in C$. If $c \neq \bar{0}$, then $wt(y + c) \geq t + 1$, since $y$ has only at most $t$ 1’s to cancel the 1’s of $c$ (of which there are at least $2t + 1$). Now $y \in y + C$ and $y = y + \bar{0}$, so $y$ is the only word in $y + C$ that has weight at most $t$. Thus the set of cosets $\{y + C \mid wt(y) \leq t\}$ each has a unique word of weight at most $t$. All cosets are disjoint and any cosets that remain, apart from the ones above (if there are any), will all contain words of weight at least $t + 1$. Therefore every coset has at most one word of weight $t$ or less (some cosets may have none of these words).

Let $u$ be an error pattern of weight at most $t$, $v \in C$ be sent and $w = v + u$ be received. Then $wH = (u + v)H = uH + vH = uH$. Therefore the coset is uniquely determined by the error pattern. If we let the syndrome $uH$ correspond to the coset $u + C$ then $u$ is the unique word of weight at most $t$ in $u + C$ and so $u$ will be chosen as the (correct) error pattern and decoding will be successful.

Question 5

(a) This can be shown in one of two ways:
(i) An \((n, k, d)\)-code is equivalent to a code in standard form that has a parity check matrix of the form
\[
\begin{bmatrix}
X \\
I_{n-k}
\end{bmatrix}.
\]
Assume \(d - 1 > n - k\) (\(d - 1 \geq n - k + 1\)). By question 3, any set of \(d - 1\) rows of \(H\) is linearly independent. Therefore any set of \(n - k + 1\) rows of \(H\) is linearly independent. On the other hand if we take the \(n - k\) rows of \(I_{n-k}\) and any row of \(X\) they form a linearly dependent set — any \(z \in K^{n-k}\) is a linear combination of the rows of \(I_{n-k}\). This gives us the desired contradiction.

(ii) Here again we use the fact that the code is equivalent to one that is in standard form. Specifically we use the fact that the generator matrix is in standard form. Let \(v \in C\) be a codeword that has only one of its information bits being nonzero. Then the remaining \(n - k\) parity bits can have weight at most \(n - k\). Therefore \(v\) has weight at most \(1 + n - k\), so that \(d_{\min}(C) \leq n - k + 1\).

(b) (i) \(C\) is MDS \(\iff\) every \(n - k\) rows of the parity check matrix are linearly independent.
\[\Rightarrow:\]
Follows from question 3.
\[\Leftarrow:\]
We again use question three and assume that \(C\) is in standard form. Note that the linearly dependent set of \(n - k + 1\) rows can be taken as the \(n - k\) rows of \(I_{n-k}\) together with a row from \(X\).

(ii) \(C\) is MDS \(\iff\) every \(k\) columns of the generator matrix are linearly independent.
\[\Rightarrow:\]
Let \(C\) be an MDS code and assume some set of \(k\) columns of its generator matrix, \(G\), are linearly dependent, say \(c_{i_1}, c_{i_2}, \ldots, c_{i_k}\). Consider the square matrix \(M = [c_{i_1}, c_{i_2}, \ldots, c_{i_k}]\). Since dimension of the row space is equal to the dimension of the column space and since \(M\) has \(k\) linearly dependent columns, we know that the \(k\) rows of \(M\) are also linearly dependent. That is they sum to zero. Thus summing the \(k\) rows of \(G\) (which is the same as encoding the all ones word) produces a word with zeros in positions \(i_1, i_2, \ldots, i_k\). Therefore this codeword has at least \(k\) zeros so that it can have at most \(n - k\) ones. Since \(C\) is MDS, \(d_{\min} = n - k + 1\), implying that all nonzero words should have
weight at least $n - k + 1$ giving a contradiction.

\[ \equiv: \]
Let $C$ be a linear code such that every $k$ columns of the generator matrix are linearly independent and assume that $d_{\text{min}} < n - k + 1$. If $w \in C$ is a word of minimum weight then $\text{wt}(w) \leq n - k$. Therefore $w$ has at least $k$ zeros say in positions $i_1, i_2, \ldots, i_k$. Furthermore let $w$ be the sum of rows $j_1, j_2, \ldots, j_l$ with $j_1 < j_2 < \ldots < j_l \leq k$ (so $l \leq k$). Let $M$ be the submatrix of $G$ obtained by taking rows $j_1, j_2, \ldots, j_l$, but only in columns $i_1, i_2, \ldots, i_k$. This submatrix has $l$ linearly dependent rows (because of the zeros in $w$) and so it has $l$ linearly dependent columns. We know that $l \leq k$ which is a contradiction to the fact that every set of $k$ columns is linearly independent so that sets of smaller size are still linearly independent.

(c) Assume $C$ is in standard form. We know that if $G$ and $H$ are the generator and parity check matrices for $C$, then $G^T$ and $H^T$ are the parity check and generator matrices for $C^\perp$. So from part (b) above we know that the parity check matrix for $C^\perp (G^T)$ has every set of $k$ rows linearly independent and we can also find a set of $k+1$ linearly dependent rows (as above). Then by question 3 this shows that $d_{\text{min}}(C^\perp) = k + 1$. Therefore $C^\perp$ is a $[n, n - k, k + 1]$-code and since $k + 1 = n - (n - k) + 1$ it is also MDS.
Math 433D/550
Assignment 3

Due: Thursday, February 28, 2002, in class.

1. [5] Is it true that in a self-dual code all words have even weight?

2. [4] Let $C$ be a Hamming code of length 15. Find the number of error patterns the extended code $C^*$ will detect and the number of error patterns that $C^*$ will correct.

3. [4] Count the number of codewords of weight 7 in the Golay code $C_{23}$. (Hint: Start by proving that every word of weight 4 in $K^{23}$ is distance 3 from exactly one codeword.)

4. [4] Let $G(1,3)$ be the generator matrix for $RM(1,3)$. Decode the following received words: (i) 01011110; (ii) 01100111; (iii) 00010100; (iv) 11001110.

5. [5] If possible, devise a single error correcting code with length 6, 4 information digits, and using the digits 0, 1, 2, 3, 4, 5. Describe the code, an encoding procedure, and a decoding procedure. Prove that your code corrects all single errors. What is its information rate? If not possible, say why not.

6. Codewords $x_1x_2x_3x_4x_5$ with decimal digits are defined by $x_1x_2x_3x_4 \equiv x_5 \pmod{9}$, where $x_5$ is a the check digit.
   (a) [2] Show that the parity check equation is equivalent to $x_1 + x_2 + x_3 + x_4 \equiv x_5 \pmod{9}$.
   (b) [3] Assuming that each single error is equally likely, what percentage of single errors are undetected?
   (c) [2] Which errors involving transposition of digits are detected?

7. Consider the code with 10 decimal digits in which the check digit $x_{10}$ is the least residue of $x_1x_2\cdots x_9 \pmod{7}$ (that is, $0 \leq x_{10} \leq 6$). (As of my last information, this code is used by UPS and Federal Express.)
   (a) [3] Under what conditions are single errors undetected?
   (b) [3] Assuming that each single error is equally likely, what percentage of single errors are undetected?
   (c) [2] Repeat (b) for errors involving transposition of digits.
Question 1

Let \( v \in C \), where \( C \) is a self-dual code. Then since \( C = C^\perp \), \( v \cdot u = 0 \) for all \( u \in C \). Specifically, \( v \cdot v = 0 \). If \( v = v_1v_2\cdots v_n \), then \( v \cdot v = v_1^2 + v_2^2 + \cdots + v_n^2 \). In the binary case this equals \( v_1 + v_2 + \cdots + v_n \) which from the orthogonality equals zero. This implies that \( v \) has an even number of 1’s, so that all \( v \in C \) have even weight.

Question 2

All Hamming codes have a minimum distance of 3. The extended code \( C^* \) has minimum distance 4. Therefore \( C^* \) can detect all errors of weight 1, 2 or 3. Further \( C^* \) can detect all error patterns of odd weight. This can be seen from the parity check matrix for \( C^* \).

\[
H^* = \begin{bmatrix}
0 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 1 \\
0 & 1 & 0 & 1 & 1 \\
0 & 1 & 1 & 0 & 1 \\
0 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 \\
1 & 1 & 0 & 1 & 1 \\
1 & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 1
\end{bmatrix}.
\]

The syndrome associated with an error pattern of odd weight is the sum of an odd number of rows of \( H^* \). Such a sum will always have a 1 in the last digit and will thus be nonzero, enabling us to detect the error. The number of odd weight error patterns are

\[
\binom{16}{1} + \binom{16}{3} + \binom{16}{5} + \cdots + \binom{16}{15} = \frac{1}{2}2^{16} = 2^{15}.
\]
An even weight error pattern that is not detectable takes one codeword into another codeword. Therefore this error pattern is itself a codeword. All codewords of $C^*$ are of even weight and therefore they are all even weight error patterns that $C^*$ cannot detect. $C^*$ has $2^{2^4 - 4 - 1} = 2^{11}$ codewords. So the number of even weight error patterns that are detectable are

$$\binom{16}{0} + \binom{16}{2} + \binom{16}{4} + \cdots + \binom{16}{16} - 2^{11} = \frac{1}{2} 2^{16} - 2^{11} = 30720.$$

Thus the number of detectable error patterns are the number of odd weight error patterns plus the number of even weight detectable error patterns. This is $2^{15} + 30720 = 63488$.

All error patterns of weight one are correctable since the minimum distance is 4. The syndrome corresponding to an error pattern of weight 2 is the sum of two rows of $H^*$. Such a sum is equal to a row of $H^*$ except that the entry on the right will be 0 instead of 1. Each such sum can arise in at least two different ways: use the row itself plus the last row of $H^*$ and row 1 = row 2 + row 3, row 2 = row 8 + row 10, row 3 = row 1 + row 2, row 4 = row 3 + row 1, ... , row 16 = row 9 + row 7. Therefore the syndrome associated with error patterns of weight 2 is not unique. In other words the coset containing an error pattern of weight 2 contains at least one other error pattern of weight 2. Thus no error pattern of weight 2 is correctable.

An error pattern of weight $k \geq 3$ is the sum of an error pattern of weight 2 and an error pattern of weight $k - 2$. Therefore the syndrome associated with this error pattern is the sum of the syndrome of the weight 2 error pattern and the syndrome of the weight $k - 2$ error pattern. Since the first syndrome is not unique, these syndromes (for weight $k$ error patterns) can arise in more than one way. So no error pattern of weight $k \geq 3$ is correctable.

Therefore the number correctable error patterns equals the number of single errors which equals 16.

**Question 3**

The Golay code $C_{23}$ is a perfect $(23, 12, 7)$-code. Therefore every word in $K^{23}$ is in exactly one ball with center a codeword and radius 3. The ball around the zero codeword contains words of weight at most 3, so no word of weight 4 is inside this ball. Let $v \in C_{23}$ be a codeword of weight at least 8. Then the words inside the ball around $v$ differ from $v$ in 1, 2 or 3 places. Therefore the words inside this ball have weight at least 5. Every nonzero codeword in $C_{23}$ has weight at least 7. So consider now a codeword $u \in C_{23}$ of weight 7. The ball around $u$ contains words that differ from $u$ in 1, 2 or 3 places. By changing 3 of
u’s 1’s to 0’s we obtain a word of weight 4 inside this ball. So we have \( \binom{7}{3} = 35 \) words of weight 4 inside this ball. There are \( \binom{23}{4} = 8855 \) words of weight 4 in \( K^{23} \) and they all have to lie inside a ball of radius 3 with center a codeword of weight 7. So we have 35 words of weight 4 in each such ball so that this requires \( 8855/35 = 253 \) such balls. Therefore there are 253 codewords of weight 7.

**Question 4**

\[
G(1, 3) = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1
\end{bmatrix}.
\]

Note that \( RM(1, 3)^\perp = RM(3 - 1 - 1, 3) = RM(1, 3) \). Therefore \( G(1, 3)G(1, 3)^T = 0 \). Also \( G(1, 3)^T \) has 4 linearly independent columns so that \( G(1, 3)^T \) is a parity check matrix for \( RM(1, 3) \). Further \( d_{\min}(RM(1, 3)) = 4 \).

(i) \( 01011110 \cdot G(1, 3) = 1100 + 1110 + 1001 + 1101 + 1011 = 1101 \). This equals the sixth row of \( G(1, 3) \) and therefore we assume an error in the sixth position and decode to 01011010.

(ii) \( 01100111 \cdot G(1, 3) = 1100 + 1010 + 1101 + 1011 + 1111 = 1111 \). This is row 8 so we assume a single error in position 8 and decode to 01100110.

(iii) \( 00010100 \cdot G(1, 3) = 1110 + 1101 = 0011 \). This doesn’t equal a row of \( G(1, 3) \) so more than one error probably occurred. Furthermore this syndrome can arise in two different ways: errors in positions 4 and 6 as well as errors in positions 3 and 5. So the best we can do is ask for a retransmission.

(iv) \( 11001110 \cdot G(1, 3) = 1000 + 1100 + 1001 + 1101 + 1011 = 1011 \). This equals row 7 so we assume a single error in position 7 and decode to 11001100.

**Question 5**

A codeword will be made up of 4 information digits \( x_1, x_2, x_3 \) and \( x_4 \). The two parity digits will be chosen such that

\[
S_1 = \sum_{i=1}^{6} ix_i \equiv 0 \pmod{7},
\]

\[
S_2 = \sum_{i=1}^{6} x_i \equiv 0 \pmod{7}.
\]
Adding twice the second equation the first we find that
\[ 3x_1 + 4x_2 + 5x_3 + 6x_4 + x_6 \equiv 0 \pmod{7}. \]

Therefore
\[ x_6 \equiv -3x_1 - 4x_2 - 5x_3 - 6x_4 \equiv 4x_1 + 3x_2 + 2x_2 + x_4 \pmod{7}. \]

Adding the two check equations together we find
\[ 2x_1 + 3x_2 + 4x_3 + 5x_4 + 6x_5 \equiv 0 \pmod{7}, \]
so that
\[ 6x_5 \equiv -2x_1 - 3x_2 - 4x_3 - 5x_4, \]
\[ \therefore x_5 \equiv 2x_1 + 3x_2 + 4x_3 + 5x_4. \]

Up to now we have not placed any restriction on the digits of the code. Therefore \( x_1 \) through \( x_4 \) could be any digit from 0 to 6 and \( x_5 \) and \( x_6 \) will be any digit from 0 to 6. We will show later how to remedy this. First we show that the code can correct all single errors.

To encode a given word \([x_1, x_2, x_3, x_4]\) we compute the product
\[
[x_1, x_2, x_3, x_4] \begin{bmatrix} 1 & 0 & 0 & 0 & 4 & 2 \\ 0 & 1 & 0 & 0 & 3 & 3 \\ 0 & 0 & 1 & 0 & 2 & 4 \\ 0 & 0 & 0 & 1 & 1 & 5 \end{bmatrix}.
\]

Assume now that a single error of size \( e \) occurs in position \( i \). Then
\[
S_1 = 1 \cdot x_1 + \cdots + i(x_i + e) + \cdots 6x_6 \equiv 0 + i \cdot e \pmod{7},
\]
\[
S_2 = x_1 + \cdots + (x_i + e) + \cdots x_6 \equiv 0 + e \pmod{7}.
\]

Therefore by computing firstly \( S_2 \) and then \( S_1 \) we can find the size and position of the error.

Let
\[
H = \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 3 & 1 \\ 4 & 1 \\ 5 & 1 \\ 5 & 1 \end{bmatrix}.
\]

Then the decoding may be described as follows.
1. Compute \( rH = [S_1, S_2] \), where \( r \) is the received word and \( H \) is defined above.

2. If \( S_1 = 0 \) and \( S_2 = 0 \) assume the word is correct.

3. If \( S_1 \neq 0 \) and \( S_2 \neq 0 \), then assume an error of size \( S_2 \) occurred in position \( S_2^{-1}S_1 \pmod{7} \).

4. If only one of the \( S_i \)'s is nonzero and the other equal to zero, then assume more than 3 errors occurred.

We now modify the code so that it only uses the digits 0 through 5. By simply restricting the possibilities for the digits \( x_1, x_2, x_3 \) and \( x_4 \) to \( \{0, 1, 2, 3, 4, 5\} \) we ensure that they meet the requirement. The possibility still exits that \( x_5, x_6 \) or both could equal 6. So among the \( 6^4 \) codewords that have \( x_1, x_2, x_3 \) and \( x_4 \) in \( \{0, 1, 2, 3, 4, 5\} \), we want to remove those with \( x_5, x_6 \) or both equal 6.

For \( x_5 = 6 \) we have

\[
2x_1 + 3x_2 + 4x_3 + 5x_4 \equiv 6 \pmod{7},
\]

where \( 0 \leq x_i \leq 5 \). This is the same as having

\[
y_1 + y_2 + y_3 + y_4 = 7k + 6,
\]

with \( y_1 = 2x_1, y_2 = 3x_2, y_3 = 4x_3 \) and \( y_4 = 5x_4 \), so that \( y_1 \in \{0, 2, 4, 6, 8, 10\}, y_2 \in \{0, 3, 6, 9, 12, 15\}, y_3 \in \{0, 4, 8, 12, 16, 20\} \) and \( y_4 \in \{0, 5, 10, 15, 20, 25\} \).

The generating function that counts the number of solutions in this case is

\[
(1 + x^2 + x^4 + x^6 + x^8 + x^{10})(1 + x^3 + x^6 + x^9 + x^{12} + x^{15})(1 + x^4 + x^8 + x^{12} + x^{16} + x^{20})(1 + x^5 + x^{10} + x^{15})
\]

We want the coefficient of \( x^i \) for \( i = 6, 13, 20, 27, 34, 41, 48, 55, 62, 69 \) where \( i \) here corresponds to the possible values \( 7k + 6 \) for \( k = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 \). In each case the corresponding coefficients are \( 3, 10, 22, 33, 38, 36, 25, 13, 5, 0 \). So in total there are \( 3 + 10 + 22 + 33 + 38 + 36 + 25 + 13 + 5 + 0 = 185 \) solutions (or codewords then) that have at least \( x_5 = 6 \).

Proceeding in the same manner as above we find that for \( x_6 = 6 \) we get

\[
4x_1 + 3x_2 + 2x_3 + x_4 \equiv 6 \pmod{7},
\]

which is the same as

\[
y_1 + y_2 + y_3 + y_4 = 7k + 6,
\]
with \( y_1 \in \{0, 4, 8, 12, 16, 20\} \), \( y_2 \in \{0, 3, 6, 9, 12, 15\} \), \( y_3 \in \{0, 2, 4, 6, 8, 10\} \) and \( y_4 \in \{0, 1, 2, 3, 4, 5\} \).

Here the generating function is

\[
(1 + x^4 + x^8 + x^{12} + x^{16} + x^{20})(1 + x^3 + x^6 + x^9 + x^{12} + x^{15})(1 + x^2 + x^4 + x^6 + x^8 + x^{10})(1 + x^1 + x^2 + x^3 + x^4 + x^5 + x^6).
\]

The coefficients that we are after are the coefficients of \( x^i \) for \( i = 6, 13, 20, 27, 34, 41, 48, 55, 62, 69 \). In this case they are 8, 27, 46, 51, 36, 15, 2, 0, 0, 0. Therefore there are 8 + 27 + 46 + 51 + 36 + 15 + 2 + 0 + 0 + 0 = 185 codewords that have at least \( x_6 = 6 \).

Lastly we need the case \( x_5 = x_6 = 6 \). In this case we have

\[
2x_1 + 3x_2 + 4x_3 + 5x_4 \equiv 6 \pmod{7} \text{ and } 4x_1 + 3x_2 + 2x_3 + x_4 \equiv 6 \pmod{7}.
\]

Subtracting the second equation form the first gives

\[5x_1 + 2x_3 + 4x_4 \equiv 0 \pmod{7}.\]

This is the same as

\[y_1 + y_2 + y_3 = 7k,\]

where \( y_1 = 5x_1, y_2 = 2x_3 \) and \( y_3 = 4x_4 \) so that \( y_1 \in \{0, 5, 10, 15, 20, 25\}, y_2 \in \{0, 2, 4, 6, 8, 10\} \) and \( y_3 \in \{0, 4, 8, 12, 16, 20\} \). The generating function for counting the number of solutions is

\[
(1 + x^5 + x^{10} + x^{15} + x^{20} + x^{25})(1 + x^2 + x^4 + x^6 + x^8 + x^{10})(1 + x^4 + x^8 + x^{12} + x^{16} + x^{20}).
\]

Here we want the coefficients of \( x^i \) for \( i = 0, 7, 14, 21, 28, 35, 42, 49, 56, 63, 70 \). They are respectively 1, 1, 5, 5, 7, 7, 3, 2, 0, 0, 0. So there are \(1 + 1 + 5 + 5 + 7 + 7 + 3 + 2 + 0 + 0 + 0 = 31\) solutions where \( x_5 = x_6 = 6 \).

Therefore the total number of solutions where \( x_5, x_6 \) or both equal 6 is \(185 + 185 - 31 = 339\). So by removing these codewords from the code we get a code with \(6^4 - 339 = 957\) codewords.

For the rate we notice that there are \(6^4\) possible codewords but that only 957 of them are used, therefore the rate is \(957/(6^4) \approx 0.73\).

**Question 6**
(a) The parity check equation is \(x_1x_2x_3x_4 \equiv x_5 \pmod{9}\) where \(x_1, x_2, x_3, x_4\) is a number with the digits \(x_1, x_2, x_3, x_4\) and not the product of these numbers. Therefore the parity check equation is \(x_1 \cdot 10^3 + x_2 \cdot 10^2 + x_3 \cdot 10^1 + x_4 \cdot 10^0 \equiv x_5 \pmod{9}\). This is the same as \(999x_1 + 99x_2 + 9x_4 + (x_1 + x_2 + x_3 + x_4) \equiv x_5 \pmod{9}\). The first set of terms is congruent to zero mod 9 so that the parity check equation is equivalent to \(x_1 + x_2 + x_3 + x_4 \equiv x_5 \pmod{9}\).

(b) If an error, \(e\), occurs in position 5 and it is to be undetectable, it has to be the case that \(x_1 + x_2 + x_3 + x_4 \equiv x_5 + e \pmod{9}\). This says that \(e \equiv 0 \pmod{9}\). This corresponds to the following two errors: \(0 \rightarrow 9\) and \(9 \rightarrow 0\). Since \(0 \leq x_5 \leq 8\), the only possible undetectable error in position 5 is \(0 \rightarrow 9\). This should also be detectable because if \(x_5 = 0\), then \(x_1 + x_2 + x_3 + x_4 \equiv 0 \pmod{9}\) and should \(x_5\) change to 9 and if this is the only error then \(x_1 + x_2 + x_3 + x_4\) is still congruent to zero mod 9 which is different from 9. So all errors are detectable in position 5. Note however that it is not possible detect this error from the check equation alone.

If an undetectable error, \(e\), occurs in positions 1 through 4, then \(x_1 + x_2 + x_3 + x_4 + e \equiv x_5 \pmod{9}\). Again this says that \(e \equiv 0 \pmod{9}\) which corresponds to the errors \(0 \rightarrow 9\) and \(9 \rightarrow 0\).

Now in total there are \(10 \times 5 \times 9 = 450\) possible single errors: one of 10 digits is in error, in one of 5 positions and it can be changed to one of 9 possible values. Since all errors in position 5 are detectable we only need to consider the errors in positions 1 through 4. Of these we know that the errors \(0 \rightarrow 9\) and \(9 \rightarrow 0\) are undetectable. These two types of errors can occur in any of the positions 1 through 4, so there are 8 possible undetected errors. Therefore the percentage that is undetectable is \(8/450 \approx 1.8\%\).

(c) If two digits from \(x_1, x_2, x_3\) or \(x_4\) are transposed this will not be detected since the parity check equation remains valid. As an example say that \(x_4\) and \(x_5\) are transposed, then the check equation becomes \(x_1 + x_2 + x_3 + x_5 \equiv x_4 \pmod{9}\). This is the same as \(x_1 + x_2 + x_3 + x_4 + x_5 \equiv 2x_4 \pmod{9}\), which in turn is \(2x_5 \equiv 2x_4 \pmod{9}\) implying \(x_4 \equiv x_5 \pmod{9}\). The only way in which this can occur is if \(x_4 = 9\) and \(x_5 = 0\).

Therefore transpositions involving the check digit will be detectable as long as the digits involved are not 9 (for \(x_1\) through \(x_4\)) and 0 (for \(x_5\)).

**Question 7**
The check equation is \( x_1 x_2 \cdots x_9 \equiv x_{10} \pmod{7} \) (as above). Therefore it is the same as \( x_1 \cdot 10^8 + x_2 \cdot 10^7 + \cdots + x_9 \cdot 10^0 \equiv x_{10} \pmod{7} \).

(a) Say a single error of size \( e \) occurs in position \( i \), \( 1 \leq i \leq 9 \). The check equation becomes \( x_1 \cdot 10^8 + x_2 \cdot 10^7 + \cdots + (x_i + e) \cdot 10^{9-i} + \cdots + x_9 \cdot 10^0 \equiv x_{10} + e \cdot 10^{9-i} \pmod{7} \). The error will be undetectable if \( e \cdot 10^{9-i} \equiv 0 \pmod{7} \). This implies that \( e = \pm 7 \). This corresponds to the following errors: \( 0 \to 7, 1 \to 8, 2 \to 9, 7 \to 0, 8 \to 1 \) and \( 9 \to 2 \).

If an undetectable error of size \( e \) occurs in position 10, then it has to be the case (also) that \( e \equiv 0 \pmod{7} \). This corresponds to the same set of errors as above, but now since \( 0 \leq x_{10} \leq 6 \), the only possible undetectable errors in position 10 are \( 0 \to 7, 1 \to 8 \) and \( 2 \to 9 \). Since all these errors change \( x_{10} \) into something bigger than 6, they will be detectable.

(b) Consider first errors occurring in positions 1 through 9. For each of the 9 possible positions there are 10 possible digits for each position and for each digit and position there are 9 possible errors. Therefore in the first nine positions there are \( 10 \times 9 \times 9 = 810 \) possible errors. In the tenth position there can be any one of 7 digits and each digit can be changed into one of 9 possibilities. Therefore there are \( 7 \times 9 = 63 \) single errors involving the tenth position. So in total there are \( 810 + 63 = 873 \) possible single errors.

All errors in position 10 are detectable, so we only concern ourselves with the first nine positions. In these positions there are 6 possible undetectable errors. Each of these 6 errors can occur in one of 9 positions, so there are \( 9 \times 6 = 54 \) undetectable errors. Therefore the percentage of undetectable errors is \( 54/873 \approx 6\% \).

(c) Let \( x_i \) and \( x_j \) be transposed with \( i < j \). If \( i = 1 \), there 9 possible transpositions: \( x_1 \leftrightarrow x_2, x_1 \leftrightarrow x_3, \ldots, x_1 \leftrightarrow x_{10} \). If \( i = 2 \) there are 8 transpositions, \ldots, with \( i = 9 \) there is 1 transposition. In total there are \( 1 + 2 + \cdots + 9 = 45 \) transpositions. Each transposition that is an error is made up of an ordered pair of distinct numbers: there are 90 of these. Each pair can be combined with each of the 45 transpositions. So there are \( 90 \times 45 = 4050 \) transpositions that are errors.

Say digits \( x_i \) and \( x_j \) are transposed where \( 1 \leq i < j \leq 9 \). The checksum becomes

\[
x_1 \cdot 10^8 + \cdots + x_j \cdot 10^{9-i} + \cdots + x_i \cdot 10^{9-j} + \cdots + x_9 \cdot 10^0,
= x_1 \cdot 10^8 + \cdots + x_i \cdot 10^{9-i} + \cdots + x_j \cdot 10^{9-j} + \cdots + x_9 \cdot 10^0 - x_i \cdot 10^{9-i} + x_j \cdot 10^{9-i} - x_j \cdot 10^{9-j} + x_i \cdot 10^{9-j},
\equiv x_{10} + (x_j - x_i) \cdot 10^{9-i} - (x_j - x_i) \cdot 10^{9-j} \pmod{7}.
\]
The error will be undetectable if \((x_j - x_i)(10^9-i - 10^9-j) \equiv 0 \pmod{7}\). That is if \((x_j - x_i) = \pm7\) or if \((10^9-i - 10^9-j) \equiv 0 \pmod{7}\). The last equation is the same as \(10^9-j(10^j-i) \equiv 0 \pmod{7}\). Now \(7 \nmid 10^9-j\), so \(7 \mid 10^j-i - 1\). That is \(10^j-i \equiv 1 \pmod{7}\) implying \(j - i \equiv 0 \pmod{\phi(7)}\). Since \(\phi(7) = 6\) this corresponds to \(j = 9, i = 3; j = 8, i = 2\) and \(j = 7, i = 1\). Therefore if the digits in positions 1 and 7, 2 and 8 or 3 and 9 are transposed an undetectable error occurs (regardless of the digits involved). Now there are \(90 \times 3 = 270\) transpositions involving these positions.

Furthermore if the size of a transposition is \(\pm7\), that is \((x_i - x_j) = \pm7\), the transposition is undetectable. This corresponds to the 6 transpositions \(0 \leftrightarrow 7, 1 \leftrightarrow 8, 2 \leftrightarrow 9, 7 \leftrightarrow 0, 8 \leftrightarrow 1\) and \(9 \leftrightarrow 2\). Transpositions of size \(\pm7\) have already been considered for positions 3 and 9, 2 and 8 and positions 1 and 7, above. Thus we are left with considering transpositions involving all the other positions. There are \((1+2+3+\cdots+8) - 3 = 33\) of these (the total number of transpositions involving the first 9 positions – the 3 considered above). So there are \(33 \times 6 = 198\) transpositions of size 7 in the remaining positions.

The check equation is equivalent to

\[
2x_1 + 3x_2 + x_3 + 5x_4 + 4x_5 + 6x_6 + 2x_7 + 3x_8 + x_9 + 6x_{10} \equiv 0 \pmod{7}.
\]

Therefore \(x_{10}\) and \(x_6\) can always be transposed and it will not be detected. There are 90 such transpositions.

So, there are \(270 + 198 + 90 = 558\) undetectable transpositions. The percentage then is \(558/4050 \approx 14\%\).
Math 433D/550
Assignment 4

Due: Thursday, March 28, 2002, in class.

1. [5] The following text was intercepted from an Affine cipher:
   
   KQEREJEBCPPCJCRKIEACUZBKRVPKRBCIBQCARBJCVFCUP
   KRIOFKACUZQEPBKRXPEIIEABDKPBCPFCDCCAFIEABDKP
   BCPFEQPKAZBKRHAIBKAPCCIBURCCDJKDCCJCIDFUIXPAFF
   ERBICZDFKABICBBENEFCUPJCVKABPCYDCCDPKBCOCPERK
   IVKSCPICBRKIJPKABI

   Determine the plain-text. Give a clear description of the steps you followed.

2. Suppose Bob has an RSA cryptosystem with a large modulus $n$ which can not be factored easily. Alice sends a message to Bob by representing the alphabetic characters A, B, ..., Z as 0, 1, ..., 25, respectively, and then encrypting each character (i.e., number) separately.

   (a) [4] Explain how a message encrypted in this way can easily be decrypted.

   (b) [2] The following cipher-text was encrypted using the scheme described above with $n = 18721$ and $b = 25$:

   365, 0, 4845, 14930, 2608, 2608, 0

   Illustrate your method from (a) by decrypting this cipher-text without factoring $n$.

   This example illustrates a protocol failure in RSA. It demonstrates that a cipher-text can sometimes be decrypted by an adversary if the system is used in a careless way. Thus a secure cryptosystem is not enough to assure secure communication, it must also be used properly.

3. [5] What happens if the RSA system is set up using $p$ and $q$ where $p$ is prime but $q$ is not? Does encryption work (is $E_K(x)$ 1-1)? Can all encrypted messages be uniquely decrypted? Illustrate your points with an example where $p$ and $q$ are two digit numbers.

4. [4] Find a (hopefully small) composite odd integer $n$, and an integer $a$, $1 \leq a \leq n - 1$ for which the Miller-Rabin algorithm answers ”prime”. Demonstrate this by stepping through the algorithm, assuming $a$ is generated in step 2.

5. Suppose $p = 199$, $q = 211$ and $B = 1357$ in a Rabin cryptosystem.

   (a) [2] Determine the four square roots of 1 modulo $n$, where $n = pq$. 
(b) [2] Compute $E_k(32767)$.

(c) [2] Determine the four possible decryptions of this cipher-text $y$.

6. [4] Factor 262063 and 9420457 using the $p - 1$ method. In each case, how big does $B$ have to be to be successful?

7. [6] Gary’s Poor Security (GPS) public key cryptosystem has $E_k(x) = ax \pmod{17575}$, where $a$ the receiver’s public key, and $gcd(a, n) = 1$. The plain-text space and cipher-text space are both $\mathbb{Z}_n$, and each element of $\mathbb{Z}_n$ represents three alphabetic characters as in the following examples:

\[
\text{DOG} \to 3 \times 26^2 + 14 \times 26 + 6 = 2398
\]
\[
\text{CAT} \to 2 \times 26^2 + 0 \times 26 + 19 = 1731.
\]

The following message has been encrypted using Gary’s public key, 1411.

7017, 17342, 5595, 16298, 12285

Explain how to break the system and decrypt the message. Do it. Show your work.

8. [5] The following message was encrypted using the Rabin cryptosystem with $n = 19177 = 127 \times 151$ and $B = 5679$:

2251, 8836, 7291, 6035

The elements of $\mathbb{Z}_n$ correspond to triples of alphabetic characters as in question 7. Decrypt the message. explain how you decided among the four possible plain-texts for each cipher-text symbol.
MATH 550
Assignment 4
Solutions

Question 1

The frequencies of the letters in the cipher-text are shown below.

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>13</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
</tr>
<tr>
<td>C</td>
<td>32</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
</tr>
<tr>
<td>E</td>
<td>13</td>
</tr>
<tr>
<td>F</td>
<td>10</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
</tr>
<tr>
<td>I</td>
<td>16</td>
</tr>
<tr>
<td>J</td>
<td>6</td>
</tr>
<tr>
<td>K</td>
<td>20</td>
</tr>
<tr>
<td>N</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Letter</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>O</td>
<td>2</td>
</tr>
<tr>
<td>P</td>
<td>20</td>
</tr>
<tr>
<td>Q</td>
<td>4</td>
</tr>
<tr>
<td>R</td>
<td>12</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td>U</td>
<td>6</td>
</tr>
<tr>
<td>V</td>
<td>4</td>
</tr>
<tr>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
</tr>
<tr>
<td>Z</td>
<td>4</td>
</tr>
</tbody>
</table>

The most frequent letters turn out to be C, B, K, P and I. Based on this we guess that $E \mapsto C$ and $T \mapsto B$. That is

$$E_k(4) = 2, \quad :a \cdot 4 + b = 2,$$
$$E_k(19) = 1, \quad :a \cdot 19 + b = 1.$$  

From this we find that $a \equiv 19 \pmod{26}$ and $b \equiv 4 \pmod{26}$. Therefore

$$D_k(y) = a^{-1}(y - b) \equiv 11(y + 22) \pmod{26}.$$  

Applying $D_k(y)$ to the cipher-text we find the following.
Therefore the cipher-text becomes.

\[
\begin{array}{c}
D_k(A) = I \\
D_k(B) = T \\
D_k(C) = E \\
D_k(D) = P \\
D_k(E) = A \\
D_k(F) = L \\
D_k(H) = H \\
D_k(I) = S \\
D_k(J) = D \\
D_k(K) = O \\
D_k(N) = V
\end{array}
\quad \begin{array}{c}
D_k(O) = G \\
D_k(P) = R \\
D_k(Q) = C \\
D_k(R) = N \\
D_k(S) = Y \\
D_k(U) = U \\
D_k(V) = F \\
D_k(X) = B \\
D_k(Y) = M \\
D_k(Z) = X
\end{array}
\]

\[
\text{OCANADATERREDOIEUXTONFRONTTESTCEINTDEFLLEUR} \\
\text{ONSGLORIEUXCARTONBRASSAITPORTELPEEILHAITPOR} \\
\text{TERLACROIXTONHISTOIREESTUNEPEEDEPLUSBRILL} \\
\text{ANTSEXPOITSETTAVEURDEFOITREMPEEPOTGERANO} \\
\text{SFOYERSETNOSDROITS}
\]

Which turns out to be the following.

Ô Canada!
Terre de nos âieux.
Ton front est ceint,
De fleurons glorieux.
Car ton bras
Saït porter l’épée,
Il saït porter la croix.
Ton histoire est une épopée,
des plus brillants exploits.
Et ta valeur,
de foi trempée,
protègera nos foyers et nos droits.

**Question 2**
(a) The eavesdropper encrypts the alphabet A, B, . . . , Z. He/She then knows what the cipher-text of each plain-text letter is and since the encryption function is one-to-one we know what the inverse of each cipher-text “letter” is.

(b) Encrypting the alphabet using the given parameters we find the following.

| A ↦ 0       | N ↦ 4845 |
| B ↦ 1       | O ↦ 1375 |
| C ↦ 6400    | P ↦ 13444 |
| D ↦ 18718   | Q ↦ 16 |
| E ↦ 17173   | R ↦ 13663 |
| F ↦ 1759    | S ↦ 1437 |
| G ↦ 18242   | T ↦ 2940 |
| H ↦ 12359   | U ↦ 10334 |
| I ↦ 14930   | V ↦ 365 |
| J ↦ 9       | W ↦ 10789 |
| K ↦ 6279    | X ↦ 8945 |
| L ↦ 2608    | Y ↦ 11373 |
| M ↦ 4644    | Z ↦ 5116 |

Reading off the plain-text from the table we find that the cipher-text message is: VANILLA.

**Question 3**

In general the encryption function will not be one-to-one. To see this let \( p = 11 \) and \( q = 12 \). Then \( n = pq = 132 = 11 \times 3 \times 2^2 \) and \( \phi(n) = [132/(11 \times 2 \times 2)](11 - 1)(3 - 1)(2 - 1) = 40 = 5 \times 2^3 \). Thus if we choose \( b = 3 \) then \( \gcd(b, \phi(n)) = 1 \). Further \( a \equiv b^{-1} \pmod{\phi(n)} \), so that \( a \equiv 27 \pmod{40} \). We now find that \( E_k(3) \equiv 3^b \equiv 3^3 \equiv 9 \pmod{132} \) and \( D_k(9) \equiv 9^a \equiv 9^{27} \equiv 81 \pmod{132} \). So here \( D_k(E_k(3)) \neq 3 \). Also, \( E_k(81) \equiv 81^b \equiv 81^3 \equiv 9 \pmod{26} \). Therefore two different elements, 3 and 81, both encrypt to 9.

**Question 4**

Let \( n = 49 = 7 \times 7 \). Then \( n - 1 = 48 = 2^4 \times 3 \), therefore \( m = 3 \) in the Miller-Rabin algorithm. If we choose \( a = 18 \) or 30, then in step 3 \( b = a^3 \equiv 1 \pmod{49} \). In step 4 the algorithm will answer “prime” contrary to \( n \) being composite.
Question 5

\(p = 199, \ q = 211, \ n = pq = 41989\) and \(B = 1357\).

(a) \(x^2 \equiv 1 \pmod{41989}\) \iff \(x \equiv \pm 1 \pmod{199}\) and \(x \equiv \pm 1 \pmod{211}\).

From \(x \equiv 1 \pmod{199}\) and \(x \equiv 1 \pmod{211}\) the Chinese remainder Theorem gives the solution \(x \equiv 1 \times M_1 \times y_1 + 1 \times M_2 \times y_2\), where

\[M_1 = 211, \quad M_2 = 199,\]
\[y_1 \equiv M_1^{-1} \equiv 83 \pmod{199}, \quad y_2 \equiv M_2^{-1} \equiv 123 \pmod{211}.\]

Therefore \(x \equiv 1 \times 211 \times 83 + 1 \times 199 \times 123 \equiv 1 \pmod{41989}\).

From \(x \equiv 1 \pmod{199}\) and \(x \equiv -1 \pmod{211}\) the Chinese remainder Theorem gives the solution \(x \equiv 1 \times M_1 \times y_1 + (-1) \times M_2 \times y_2 \equiv 1 \times 211 \times 83 - 1 \times 199 \times 123 \equiv 35025 \pmod{41989}\).

The other two solutions will therefore be \(x \equiv -1 \pmod{41989}\) and \(x \equiv -35025 \pmod{41989}\).

(b) \(E_k(x) = x(x + B) \pmod{n}\). Therefore

\[E_k(32767) \equiv (32767)(32767 + 1357) \pmod{41989},\]
\[\equiv 16027 \pmod{41989}.

(c) We know that

\[E_k \left( \omega \left( \frac{x + B}{2} \right) - \frac{B}{2} \right) = E_k(x),\]

where \(\omega\) is a square-root of 1 mod(41989). Also \(2^{-1} \equiv 20995 \pmod{41989}\). Therefore

\[\omega \left( \frac{32767 + 1357}{2} \right) \equiv \omega(12451) - 21673 \equiv \omega(12451) + 20316 \pmod{41989}.

This gives the four possible decryptions of 16027 as

\[\begin{align*}
\omega = 1 & : \quad 1(12451) + 20316 \equiv 32767 \pmod{41989}, \\
\omega = -1 & : \quad -1(12451) + 20316 \equiv 7865 \pmod{41989}, \\
\omega = 35025 & : \quad 35025(12451) + 20316 \equiv 18837 \pmod{41989}, \\
\omega = 6964 & : \quad 6964(12451) + 20316 \equiv 21795 \pmod{41989}.
\end{align*}\]
Question 6

Using the \( p - 1 \) method we find that \( 262063 = 521 \times 503 \) and the first \( B \) for which the method produces an answer is \( B = 13 \). Further \( 9420457 = 2351 \times 4007 \) and the smallest \( B \) that works in this case is \( B = 47 \).

Question 7

We are given that \( E_k(x) = ax \pmod{17575} \), with \( a = 1411 \). From this we see that \( D_k(y) = a^{-1}y \pmod{17575} \). Therefore all we need to do is find \( a^{-1} \pmod{17575} \). We find that \( a^{-1} \equiv 16591 \pmod{17575} \). Decrypting the given cipher-text we get:

\[
\begin{align*}
2247 & \equiv 3 \times 26^2 + 8 \times 26 + 11, \\
797 & \equiv 1 \times 26^2 + 4 \times 26 + 17, \\
13070 & \equiv 19 \times 26^2 + 8 \times 26 + 18, \\
8743 & \equiv 12 \times 26^2 + 24 \times 26 + 7, \\
3160 & \equiv 4 \times 26^2 + 17 \times 26 + 14.
\end{align*}
\]

Reading off the letters we find that the plain-text is: DILBERT IS MY HERO.

Question 8

We are given \( n = 127 \times 151 = 19177 \) and \( B = 5679 \). Therefore

\[
D_k(y) = \sqrt{y + 4^{-1}B^2} - 2^{-1}B.
\]

Now \( B^2 \equiv 14504 \pmod{19177} \) and \( 4^{-1} \equiv 14383 \pmod{19177} \), so that \( 4^{-1}B^2 \equiv 3626 \pmod{19177} \). Also, \( 2^{-1} \equiv 9589 \pmod{19177} \), therefore \( 2^{-1}B \equiv 12428 \pmod{19177} \). We now have

\[
D_k(y) = \sqrt{y + 3626} - 12428 \equiv \sqrt{y + 3626} + 6749 \pmod{19177}.
\]

Decrypting the cipher-text we find the following.

\[
D_k(2251) \equiv \sqrt{5877} + 6749 \pmod{19177} \text{. Now } \sqrt{5877} \equiv 5877^{128/4} \equiv \pm 17 \pmod{127} \text{ and } \sqrt{5877} \equiv 5877^{152/4} \equiv \pm 21 \pmod{151}.\]
From \( x \equiv 17 \pmod{127} \) and \( x \equiv 21 \pmod{151} \) the Chinese remainder Theorem gives 
\[ x \equiv 17 \times 151 \times y_1 + 21 \times 127 \times y_2 \pmod{19177}. \]
Here \( y_1 \equiv 151^{-1} \equiv 90 \pmod{127} \) and 
\( y_2 \equiv 127^{-1} \equiv 44 \pmod{151} \). Therefore 
\[ x \equiv 17 \times 151 \times 90 + 21 \times 127 \times 44 \equiv 3192 \pmod{19177}. \]

From \( x \equiv 17 \pmod{127} \) and \( x \equiv -21 \pmod{151} \), we find 
\[ x \equiv 17 \times 151 \times 90 + (-21) \times 127 \times 44 \equiv 17797 \pmod{19177}. \]

From \( x \equiv -17 \pmod{127} \) and \( x \equiv 21 \pmod{151} \), we find 
\[ x \equiv -17 \times 151 \times 90 + 21 \times 127 \times 44 \equiv 1380 \pmod{19177}. \]

From \( x \equiv -17 \pmod{127} \) and \( x \equiv -21 \pmod{151} \), we find 
\[ x \equiv -17 \times 151 \times 90 + (-21) \times 127 \times 44 \equiv 15985 \pmod{19177}. \]

Therefore 
\[
D_k(2251) \equiv 3192 + 6749 \equiv 9941 \pmod{19177},
\]
\[
D_k(2251) \equiv 17797 + 6749 \equiv 5369 \pmod{19177},
\]
\[
D_k(2251) \equiv 1380 + 6749 \equiv 8129 \pmod{19177},
\]
\[
D_k(2251) \equiv 15985 + 6749 \equiv 3557 \pmod{19177}.
\]

Giving us the following 
\[
9941 = 14 \times 26^2 + 18 \times 26 + 9,
\]
\[
5369 = 7 \times 26^2 + 24 \times 26 + 13,
\]
\[
8129 = 12 \times 26^2 + 0 \times 26 + 17,
\]
\[
9941 = 5 \times 26^2 + 6 \times 26 + 21.
\]

In each case this corresponds to the plain-text OSJ, HYN, MAR, FGV. At this point the plain-text that holds the most promise seems to be MAR.

\( D_k(8836) \equiv \sqrt{12462} + 6749 \pmod{19177} \). Here we find that the four square-roots of 
12462 (mod 19177) are : 18038, 13974, 5203 and 1139. Therefore 
\[
D_k(8836) \equiv 18038 + 6749 \equiv 5610 \pmod{19177},
\]
\[
D_k(8836) \equiv 13974 + 6749 \equiv 1546 \pmod{19177},
\]
\[
D_k(8836) \equiv 5203 + 6749 \equiv 11952 \pmod{19177},
\]
\[
D_k(8836) \equiv 1139 + 6749 \equiv 7888 \pmod{19177}.
\]

We now have 
\[
5610 = 8 \times 26^2 + 7 \times 26 + 20,
\]
\[
1546 = 2 \times 26^2 + 7 \times 26 + 12,
\]
\[
11952 = 17 \times 26^2 + 17 \times 26 + 18,
\]
\[
7888 = 11 \times 26^2 + 17 \times 26 + 10.
\]
Here the corresponding plain-text is IHU, CHM, RRS, LRK. Considered on their own none of these seem to be a better choice than the other. If we combine them with the first set we find that CHM seems to be a good choice as this gives MARCHM.

\[ D_k(7291) = \sqrt{10917} + 6749. \] The four square-roots of 10917 are 12519, 15567, 3610 and 6658. This gives

\[
\begin{align*}
D_k(7291) & \equiv 12519 + 6749 \equiv 91 \pmod{19177}, \\
D_k(7291) & \equiv 15567 + 6749 \equiv 3139 \pmod{19177}, \\
D_k(7291) & \equiv 3610 + 6749 \equiv 10359 \pmod{19177}, \\
D_k(7291) & \equiv 6658 + 6749 \equiv 13407 \pmod{19177}.
\end{align*}
\]

This leads to

\[
\begin{align*}
91 & = 0 \times 26^2 + 3 \times 26 + 13, \\
3139 & = 4 \times 26^2 + 16 \times 26 + 19, \\
10359 & = 15 \times 26^2 + 8 \times 26 + 11, \\
13407 & = 19 \times 26^2 + 21 \times 26 + 17.
\end{align*}
\]

The plain-text is ADN, EQT, PIL, TVR. Out of these four plain-texts the only one that combines with the result so far in a sensible manner is the first one. This combined with the result so far gives MARCHMADN.

\[ D_k(6035) = \sqrt{9661} + 6749. \] The four square-roots of 9661 are 17904, 15618, 3559 and 1273. This gives

\[
\begin{align*}
D_k(6035) & \equiv 17904 + 6749 \equiv 5476 \pmod{19177}, \\
D_k(6035) & \equiv 15618 + 6749 \equiv 3190 \pmod{19177}, \\
D_k(6035) & \equiv 3559 + 6749 \equiv 10308 \pmod{19177}, \\
D_k(6035) & \equiv 1273 + 6749 \equiv 8022 \pmod{19177}.
\end{align*}
\]

Giving us the following

\[
\begin{align*}
5476 & = 8 \times 26^2 + 2 \times 26 + 16, \\
3190 & = 4 \times 26^2 + 18 \times 26 + 18, \\
10308 & = 15 \times 26^2 + 6 \times 26 + 12, \\
8022 & = 11 \times 26^2 + 22 \times 26 + 14.
\end{align*}
\]
The four plain-texts are: ICQ, ESS, PGM and LWO. Here the second one seems to be the only one that fits in with the results so far. This gives MARCHMADNESS $\rightarrow$ March Madness.