Information

This course deals with the transmission of information from a sender A to a receiver B. To be transmitted via some channel, the information needs to be encoded in the correct form, and possibly encrypted, if it is to remain confidential.

Definition

An alphabet $A$ is a finite set of symbols.

Examples

$\{a, b, c, \ldots, z, \text{"space"}\}$ or $\{0, 1\}$.

Symbols from $A$ are called letters. A message or word is a finite sequence of letters from $A$, and is usually denoted by juxtaposition of letters (e.g. $m = a_1 a_2 \ldots a_n a_1$). The set of all words from $A$ is denoted $A^*$.

To convey a message $m$ successfully, $A$ needs to transform it into a suitable form. So $A$ encodes the message into a new alphabet $B$ via a coding function $c : A \rightarrow B^*$, that codes each letter from alphabet $A$ into a finite sequence from $B$. These finite sequences are codewords ($cw$s).

The entire message $m$ is then encoded as

$c^*(m) = c(a_1) c(a_2) \ldots c(a_{n-1}) c(a_n) \in B^*$

This is then transmitted via a channel to $B$ who decodes it to recover $m$. 


If \(|A| = n\), \(|B| = a\), we say that \(c\) is an \(a\)-ary code of \(n\).

The function \(c\) may be chosen so that only the intended recipient can decode it (called encryption). But it could be encoded simply to compress the message or reduce the likelihood of errors being introduced in the transmission.

Example

- A sends \(m = \text{"Call at 2pm"}\)

- Encode as binary strings using ASCII, e.g. \(c(c) = 1000011\)
  \(c^*(m) = 1000011...1...\) 11 blocks of length 7

- Transmit via the internet

- Decoded by B.

Examples

- Morse code, audio CDs (error correction), SMS (text compression),
  zip files (compression), PIN numbers (cryptography), jpeg files (compression and detail)

Prerequisites

- \(1A\) Probability (expectation, Law of Large Numbers)
- Numbers and Sets (modular arithmetic, CRT, FLT)
- Linear Algebra (vector spaces)
- CRM (finite fields)
Books
Goldie - Pinch, Welch, Keith, Carr's notes, Körner's notes

Plan
1. Noiseless Coding
2. Error correcting codes
3. Shannon's Theorems
4. Linear codes, cyclic codes
5. Cryptography "The science of encipherment"

1. Noiseless Coding

Prefix-free Codes

Encode \( m = a_1 \ldots a_n \) using function \( c: A \rightarrow B^* \) to obtain \( c^*(m) := c(a_1) \ldots c(a_n) \). Here \( c^*: A^* \rightarrow B^* \). We need to decode \( c^*(m) \) and recover the original \( m \). Hence we want \( c^* \) injective, in which case we say that \( c \) is decodable (decipherable).

Remark
For \( c \) to be decodable, \( c: A \rightarrow B^* \) must be injective, but this is not sufficient.

\( A = \{1, 2, 3, 4\} \), \( B = \{0, 1\} \)
\( c(1) = 0 \), \( c(2) = 1 \), \( c(3) = 00 \), \( c(4) = 01 \)
\( c^*(114) = 0001 = c^*(312) \), no \( c \) is injective but not decodable.

1.1 Definition
Take \( w \in B^* \). Another word \( w' \) is a prefix of \( w \) if \( w' \) is
the concatenation of $w$ with another word $w''$ such that $w = w'w''$

A code $c: A \rightarrow B^*$ is prefix-free if no codeword $c(a)$ is a prefix of a different codeword $c(a')$.

Example

$c: \{0,1,2,3\} \rightarrow \{0,1,3\}$
$c: \frac{2}{3} \rightarrow \{1,0\}$
$c: \frac{3}{1} \rightarrow \{1,1\}$

is prefix-free.

If we reversed each codeword this is not prefix-free.

Assuming that $c$ is injective, the following codes are always decodable:

- Prefix-free code is decodable (if we receive $w \in B^*$ we can decode by examining prefixes of $w$. One of these must be $c(a_1)$ for some $a \in A$. Since the code is prefix-free, $a$ is unique. Delete this process and repeat).

Notice that we can decode a message as it is received; we do not require the whole message to decode the first letter.

\[
\begin{array}{cccc}
0 & 1 & 0 & 1 & 1 & 1 & 1
\end{array}
\begin{array}{cccc}
0 & 1 & 0 & 1
\end{array}
\]

Prefix-free codes are sometimes called instantaneous/self-punctuating.

- Block codes (all codewords are the same length)
- Comma codes (reserve a special letter from $B$ to signal the end of the word).
Prefixes of $W$ are all vertices of the tree in the path from the base vertex to $W$. This is prefix free $\iff$ the path from any codeword to $\phi$ contains no other codeword.

**The Kraft Inequality**

(13) $C : A \to B^*$ be a prefix-free code with codeword $c(a)$ having length $l(a)$. Then $\sum_{a \in A} D^{-l(a)} \leq 1$ where $D = |B|$.

**Proof**

Let $L = \max \{ l(a) : a \in A \}$. Consider the tree constructed from $C$. Vertices at level $L$ correspond to words of length $L$.

There are $D^L$ vertices.

A codeword $c(a)$, length $l(a)$ is a prefix of $D^{-l(a)}$ such vertices.

Since $C$ is prefix-free, no level $L$ vertex can have two codewords as prefixes. Hence the total number of level $L$ vertices with codewords as prefixes is $\sum_{a \in A} D^{-l(a)}$ and this can be at most

the number of level $L$ vertices $= D \cdot \sum_{a \in A} D^{-l(a)} \leq 1$. 


Conversely,

**Proposition 1.4**

Let \( A, B \) be finite alphabets with \( |B| = D \). If \( D(a) \in N \) for each \( a \in A \) satisfying \( \sum_{a \in A} D^{-1}(a) \leq 1 \), then \( A \) a prefix-free code \( c : A \rightarrow B^* \) with each codeword \( c(a) \) having length \( l(a) \).

**Proof**

Reorder letters in \( A : a_1, a_2, \ldots, a_k \) such that \( l_{ij} = l(aj) \) satisfying \( l_1 \leq l_2 \leq \ldots \leq l_k \). We define \( c \) inductively.

Choose any word of length \( l_1 \) as \( c(a_1) \). Suppose that \( c(a_j) \) are all chosen for \( j < k \) so that no \( c(a_i) \) is a prefix of \( c(a_j) \) for \( i < j < k \). Consider words of length \( l_k \). There are \( D_{l_k} \) of them and of these \( D_{l_k-1} \) have \( c(a_j) \) as prefix.

By assumption \( 1 + \sum_{j=1}^{k-1} D_{l_k-1-j} = \sum_{j=1}^{k-1} D_{l_k-j} \leq D_{l_k} \).

So we can choose a word \( c(a_k) \) of length \( l_k \) which does not have any \( c(a_i) \), \( i < k \), as prefix. Hence, the result follows by induction.

**1.5 Proposition (McMillan's Inequality)**

Let \( c : A \rightarrow B^* \) be a decodable code with codeword \( c(a) \) with lengths \( l(a) \). Then \( \sum_{a \in A} D^{-1}(a) \leq 1 \), where \( D = |B| \).

**Proof**

Let \( L = \max \sum_{a \in A} l(a) \). Consider \( \left( \sum_{a \in A} D^{-1}(a) \right)^n \) for some \( n \in N \).
Expanding, \( \sum D^{-(c(a_1) + \ldots + c(a_N))} \), summed over all sequences \( a_1, a_2, \ldots, a_N \) of length \( N \).

The word \( w = c(a_1) \ldots c(a_N) \in \mathcal{B}^* \) has length \( L(a_1) + \ldots + L(a_N) \leq NL \).

Since \( c \) is decodable, every word \( w \in \mathcal{B}^* \) of length \( m \) can come from at most one sequence \( a_1, a_2, \ldots, a_N \).

Hence \( \left( \sum_{a \in \mathcal{A}} D^{-c(a)} \right)^N \leq \sum_{m=1}^{NL} n(m) D^{-m} \).

where \( n(m) = \# \) words of length \( m \) of the form \( c(a_1) \ldots c(a_N) \).

Now, the total number of words of length \( m \) is \( \leq D^m \), so \( n(m) \leq D^m \).

\[ \therefore \left( \sum_{a \in \mathcal{A}} D^{-c(a)} \right)^N \leq \sum_{m=1}^{NL} D^m D^{-m} = NL \]

Take the \( N \)th root.

\[ \sum_{a \in \mathcal{A}} D^{-c(a)} \leq (NL)^{1/N} \quad \text{and} \quad \text{let } N \to \infty . \]

Combining these results:

**Corollary 1.6**

If \( \mathcal{C} \) a decodable code of \( n \) words \( c_i \) of length \( l_i \), then there exists a prefix free code consisting of \( n \) words \( c'_i \) of length \( l'_i \).

2 Entropy

We will use \( (P, \Omega) \), (discrete) probability on a sample space.
Definition 2.1
For each $A \subseteq \Omega$ the information of $A$ is $I(A) = -\log_2 P(A)$
So $I(A) \geq 0$, with equality if and only if $P(A) = 1$.

Example
$(X_n)$ a family of independent Bernoulli random variables, each taking values 0, 1 with probability $\frac{1}{2}$. Given a sequence $x_1, x_2, \ldots, x_n$, the set $A = \{x_1 = x_1, \ldots, x_n = x_n\}$ has $P(A) = \left(\frac{1}{2}\right)^n$,
thus $I(A) = N$ (=$\#$ binary digits, or bits, that are specified).

Definition 2.2
Let $X$ be a discrete random variable on $\Omega$. For each $x$, the set $\{X = x\}$ has information $I(X = x) = -\log_2 P(X = x)$
The expected value of this is the entropy of $X$.
$H(X) := -\sum_{x \in X} P(X = x) \log P(X = x)$

Remarks
1. $H(X)$ depends only on the probability distribution of $X$ and not on the individual values of $X$.
2. The definition fails when one of the $\{X = x\}$ has probability 0, so note that $h(p) = -p \log p > 0$ as $p \uparrow 0$, so we interpret $-0 \log 0$ as 0 for this case.
3. $H(X) \geq 0$ with equality if and only if $X$ is almost surely constant.

Example: $X: \mathbb{Q} \rightarrow \{0, 1\}$, Bernoulli r.v. taking value 1 with
0 with probability \( q = 1 - p \).

\[
H(X) = -p \log p - q \log q = H(p, 1-p)
\]

Max at \( p = \frac{1}{2} \)

Entropy is the average amount of information gained by knowing a value of \( X \). We will use it to measure the amount of information in a message, and hence decide how much it can be compressed.

**Lemma 2.3 (Gibb's Inequality)**

Let \( X \) be a discrete random variable taking different values with probabilities \( (p_i)_{i=1}^N \), so \( \sum_{i=1}^N p_i = 1 \). If \( (q_i)_{i=1}^N \) is another set of positive numbers whose sum is equal to 1, then

\[
- \sum_{i=1}^N p_i \log p_i \leq - \sum_{i=1}^N p_i \log q_i \quad \text{with equality if } p_i = q_i \text{ for some ordering}
\]

Shannon-entropy Mixed Entropy
Joint Entropy

Let X, Y be random variables. \((X, Y)\) is a vector-valued random variable with entropy \(H(X, Y)\), the joint entropy of \(X, Y\).

We can prove using 2.3, that

Corollary 2.4

If \(X, Y\) are discrete, then \(H(X, Y) \leq H(X) + H(Y)\) with equality \(\iff\) \(X, Y\) are independent.

Example

\(X\) takes \(D\) values, each with probability \(\frac{1}{D}\).

Then \(H(X) = \log_2 D\). Suppose that \(X_1, \ldots, X_n\) are iid (to X).

Then \(H(X_1, \ldots, X_n) = \sum_{i=1}^{n} H(X_i) = n \log_2 D\)

Conditional Entropy

Suppose that \(x\) is a value taken by \(X\) with non-zero probability.

Then the conditional distribution of another random variable \(Y\) given that \(X = x\) is \(P(Y = y \mid X = x) = \frac{P(X = x, Y = y)}{P(X = x)}\).

Then we have \(H(Y \mid X = x) = -\sum_{xy} P(X = x, Y = y) \log_2 P(X = x, Y = y)\)

\(H(Y) = -\sum_{xy} P(X = x, Y = y) \log_2 P(X = x, Y = y)\)

\(H(X) = -\sum_{xy} P(X = x, Y = y) \log_2 P(X = x)\)

Then \(H(Y \mid X = x) = H(Y) - H(X)\), the conditional entropy \(H(Y \mid X = x) = H(X, Y) - H(X)\).
Now, for each $x$, $H(Y | X = x) \geq 0$ with equality iff 
$(Y | X = x)$ is constant (almost surely).

$\therefore H(X, Y) - H(X) = \sum_{x} H(Y | X) \geq 0$

with equality $\iff Y$ is a function of $X$ (almost surely).

**Proposition 2.5**

$H(X) + H(Y) \geq H(X, Y) \geq H(X)$

**Proof**

Apply the above results.

**3 Efficient Codes**

We try to find codes where the expected length of codewords is as short as possible. Such codes are called optimal (best).

**Shannon-Fano Codes**

Let $c : A \rightarrow \mathbb{B}^*$ be a code, assumed prefix-free (and thus decodable). Given $a \in A$, the codeword $c(a)$ will have length $|c(a)|$. We seek codes where the expected value of length is as small as possible.

**3.1 Definition**

The expected length of a codeword $c(a)$ is

$$E \sum_{a \in A} p(a) |c(a)|.$$ Let $A$ be a random variable taking values in $A$, where $P(A = a) = p(a)$. Then the expected length of a codeword is $E(1c(A)|).$
1.5 shows that the lengths \( l(a) = \left| c(a) \right| \) satisfy

\[
\sum_{a \in A} D^{-l(a)} \leq 1 \quad (\ast)
\]

Kraft II (1.4) shows that if we have positive integers \( l(a) \)

satisfying \((\ast)\) then there is a prefix-free code with codeword length \( l(a) \). We solve

\[
\min \sum_{a \in A} p(a) \cdot l(a) \quad \text{such that} \quad \sum_{a \in A} D^{-l(a)} \leq 1, \quad \text{each} \quad l(a) \in \mathbb{N}.
\]

3.2 Example

We will try to solve this without insisting that \( l(a) \in \mathbb{N} \).

\[
\min \sum_{a \in A} p(a) \cdot l(a) \quad \text{such that} \quad \sum_{a \in A} D^{-l(a)} \leq 1.
\]

Solution: We claim that the minimum occurs when

\[
l(a) = -\log_b p(a) = -\frac{\log_2 p(a)}{\log_2 D}
\]

If so then \( \sum_{a \in A} D^{-l(a)} = \sum_{a \in A} p(a) = 1 \) and expected length is

\[
\sum_{a \in A} p(a) \cdot l(a) = -\sum_{a \in A} p(a) \log_2 p(a) = H(A)
\]

We must show that for any values \( l(a) \) such that \( \sum_{a \in A} D^{-l(a)} \leq 1 \)

we have

\[
-\frac{1}{\log_2 D} \sum_{a \in A} p(a) \log_2 p(a) \leq \sum_{a \in A} p(a) \cdot l(a)
\]

Put \( S = \sum D^{-l(a)} \) and \( q(a) = D^{-l(a)} \). Then \( \sum q(a) = 1 \)

by construction, so by Gibb's (2.3)

\[
-\sum_{a \in A} p(a) \log_2 p(a) \leq -\sum_{a \in A} p(a) \log_2 q(a)
\]

\[
= \sum_{a \in A} p(a) \left[ \log_2 S + \log_2 D \right]
\]

\[
= (\sum_{a \in A} p(a) \cdot l(a)) \log_2 D + \log_2 S \leq (\sum_{a \in A} p(a) \cdot l(a)) \log_2 D
\]
Proposition 3.3

Suppose that \( A \) is a random-variable taking values in a finite \( \mathcal{A} \).
For any decodable \( c : \mathcal{A} \to \mathbb{B}^* \)
\[
E(|c(A)|) \geq \frac{H(A)}{\log_2 D} \quad \text{where} \quad D = |\mathcal{B}|.
\]

Proof

1.3 shows that \( L(a) = |c(a)| \) satisfies \( \Sigma D^{-L(a)} \leq 1 \).
Hence (as in 3.2) \( \Sigma p(a)L(a) \geq \frac{H(A)}{\log_2 D} \).

3.4 Proposition (Shannon-Fano Encoding)

Suppose that \( A \) is an rv taking values in a finite alphabet \( \mathcal{A} \). Then there is a prefix-free code \( c : \mathcal{A} \to \mathbb{B}^* \) satisfying
\[
E(K(A)) < 1 + \frac{H(A)}{\log_2 D} \quad \text{where} \quad D = |\mathcal{B}|
\]

Proof

1.4 shows that we need to find integers \( L(a) \) such that
\[
\Sigma p(a)L(a) < 1 + \frac{H(A)}{\log_2 D} \quad \text{and} \quad \Sigma D^{-L(a)} \leq 1.
\]
If the \( L(a) \in \mathbb{N} \), then we can always take \( L(a) = -\log_0 p(a) \).

No set \( L(a) = \lceil -\log_0 p(a) \rceil \)

Then \( -\log_0 p(a) \leq L(a) \), so \( p(a) \geq D^{-L(a)} \). This implies
\[
\Sigma D^{-L(a)} \leq 1, \text{ so there is a prefix-free code with these word lengths. Moreover}\]
\[
\Sigma p(a)L(a) \leq \Sigma p(a)(1 - \log_0 p(a))
\]
\[
= 1 - \Sigma p(a)\log_0 p(a) = 1 + \frac{H(A)}{\log_2 D}
\]
Given word lengths (a) we can apply 1.4 to construct the denied c.

Such codes are called Shannon-Fano codes. They aren't quite optimal but they are easy to construct.

Combining 3.3, 3.4, we get

Theorem 3.6 (Shannon's Noiseless Coding Theorem)

Suppose that $A$ is a random variable taking values in a finite alphabet $A$. Then an optimal code $c: A \rightarrow B^*$ for $D = |B|$ satisfies

$$\frac{H(A)}{\log_2 D} < \mathbb{E}(1c(A)) < 1 + \frac{H(A)}{\log_2 D}$$
Example 3.6

\[ A = \{1, 2, 3, 4, 5\} \] with probabilities \[ 0.4, 0.2, 0.2, 0.1, 0.1 \]
\[ B = \{0, 1\} \]

<table>
<thead>
<tr>
<th>( i )</th>
<th>( p_i )</th>
<th>(-\log_2 p_i)</th>
<th>( S_i = \Gamma - \log_2 p_i)</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.4</td>
<td>1.32</td>
<td>2</td>
<td>00</td>
</tr>
<tr>
<td>2</td>
<td>0.2</td>
<td>2.32</td>
<td>3</td>
<td>010</td>
</tr>
<tr>
<td>3</td>
<td>0.2</td>
<td>2.32</td>
<td>3</td>
<td>011</td>
</tr>
<tr>
<td>4</td>
<td>0.1</td>
<td>3.32</td>
<td>4</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td>0.1</td>
<td>3.32</td>
<td>4</td>
<td>1001</td>
</tr>
</tbody>
</table>

\[ H(1c(A)) = \sum p_i S_i = 2.8 \geq 2.12 = H(p_1, p_2, p_3, p_4, p_5) \]

Huffman Codes

How to construct optimal code? Assume \[ B = \{0, 1\}. \]

Let \( A \) be a finite alphabet with probabilities \( p(a) \).

\[ A = \{a_1, \ldots, a_k\} \] with \( p_i = p(a_i) \), such that \( p_1 \geq p_2 \geq \cdots \geq p_k \).

\( k = 2 \) (Base case) The Huffman code is \( h: \{a_1, a_2\} \rightarrow \{0, 1\} \),

\( h(a_1) = 0 \), \( h(a_2) = 1 \). This is clearly optimal.

Recursively, define a Huffman code on alphabets of size \( k-1 \).

Choose two letters in \( A \) that have the smallest probabilities attached to them, \( \lambda A \lambda A \), i.e. \( p_{k-1}, p_k \). Form \( \hat{A} = \{a_1, a_2, \ldots, a_{k-2}, a_{k-1}, a_k\} \)

by combining \( a_{k-1} \) and \( a_k \) to give the letter \( a_{k-1} \cdot a_k \in \hat{A} \) that is given the probability \( p_{k-1} + p_k \).

Let \( \tilde{h}: \hat{A} \rightarrow \{0, 1\}^* \) be a Huffman code for the new alphabet.

Then the Huffman code for \( A \) is obtained by adding a 0 or 1 to the end of the codewords for \( a_{k-1}, a_k \).
\[ h(a_j) = \begin{cases} 
\hat{h}(a_j) & j = 1, 2, \ldots, r - 1 \\
0 & j = r - 1 \\
1 & j = r 
\end{cases} \]

**Definition 3.7**

This is the (binary) Huffman code. Clearly this is prefix free.

**Theorem 3.8**

A Huffman code is indeed optimal (i.e., the average length of codewords is as small as possible for any decodable code) \( A \rightarrow \{0, 1\}^* \)

**Example 3.9**

\[
\begin{array}{c|c|c|c}
 i & p_i & c(i) & l_i \\
1 & 0.4 & 1 & 1 \\
2 & 0.2 & 01 & 2 \\
3 & 0.2 & 000 & 3 \\
4 & 0.1 & 0010 & 4 \\
5 & 0.1 & 0011 & 4 \\
\end{array}
\]

\[ E(1 h(A)) = 2.2 \]

The proof of 3.8 requires a lemma giving properties of optimal codes.

Let \( A = \{a_1, \ldots, a_r\} \) with probabilities \( p_1 > p_2 > \ldots > p_r \).

The random variable \( A \) takes values \( a_j \) with probability \( p_j \).

Average codeword length \( E(1 c(A)) \).

**Lemma 3.10**

There exists an optimal code \( C: A \rightarrow \{0, 1\}^* \) such that

a) \( l_j = 1 c(a_j) \) are ordered inversely to the probabilities i.e., \( l_1 \leq \ldots \leq l_r \)

b) The codewords \( c(a_{k-1}) \) and \( c(a_k) \) have the same length and differ only in their last bit (or final digit).
Proof

We know that there exists a prefix-free code for \( A \). Let the expected code length = \( L \). Then, \( \exists \) only finitely many codes \( c \) such that \( \sum p_i l_i \leq L \). So \( \exists \) a prefix-free code achieving minimum value for \( \sum p_i l_i \). This is optimal.

a) If \( p_i > p_j \) but \( l_i > l_j \) then we could reduce \( \sum p_i l_i \) by swapping the codewords \( c(a_i) \) and \( c(a_j) \).

Hence, we must have \( l_1 \leq l_2 \leq \ldots \leq l_k \).

b) Let \( L \) be the maximum codeword length, \( L = l_k \). Write

\[
C(a_k) = WB \text{ where } W \text{ is a word of length } L - 1 \text{ and } B \text{ is the final bit. Form a new code by changing } C(a_k) \text{ to } W. \text{ The length reduces, no reduces } \sum p_i l_i. \text{ Since } C \text{ was optimal, this new code cannot be prefix-free. Hence there exists } C(a_j) \text{ with } W \text{ as a prefix. Thus } C(a_j) \text{ and } C(a_k) \text{ both have length } L \text{ and differ only in their last bit. Finally, permute the codewords of length } L \text{ so that } C(a_{k-1}) \text{ and } C(a_k) \text{ differ only in the last bit.}

Proof that Huffman is Optimal: \( \text{ (Huffman, 1952) } \)

Use induction on \( k = 1 \). \( k = 2 \) is simple.

Assume true for all alphabets of size \( k-1 \). Let \( A = f_0, f_1, f_2, f_3 \) be a Huffman code, and \( C : A \rightarrow \{ f_0, f_1, f_2 \} \) an optimal code.

Construction of the Huffman code shows that \( F \) a Huffman code
\[X = \{a_1, a_2, \ldots, a_{k-2}, a_{k-1}, u, v, k\}\] of size \(k-1\). Let \(h\) have probabilities \(p_1, p_2, \ldots, p_{k-2}, p_{k-1} + p_k\) respectively. Write \(X\) for the random variable taking letters in \(X\) with these probabilities.

\[h(a_j) = \begin{cases} 0 & \text{if } j = k-1 \\ \tilde{h}(a_{k-1}, u, v, k) & \text{if } j = k \\ \tilde{h}(a_j) & \text{if } j \leq k-2 \end{cases}\]

So the expected codeword length for \(h\) and \(\tilde{h}\) satisfy

\[E(l_h(A)) = E(l_{\tilde{h}}(\tilde{A})) + (p_{k-1} + p_k)\]

For optimal code \(c\), apply 3.10. Choose \(c\) with \(c(a_{k-1}), c(a_k)\) differing only in their last bit, say \(c(a_{k-1}) = w0, c(a_k) = w1\), for a word \(w \in \{0, 1\}^*\). Define a new code on \(\tilde{A}\) via \(\tilde{c}(a_{k-1}, u, v, k) = w\) and \(\tilde{c}(a_j) = c(a_j) \quad \forall j \leq k-2\). \(\tilde{c}\) is prefix-free and has

\[E(l_{\tilde{c}}(\tilde{A})) = E(l_{\tilde{c}}(\tilde{A})) + (p_{k-1} + p_k)\]

Induction says that \(\tilde{c}\) is optimal. So \(E(l_{\tilde{h}}(\tilde{A})) \leq E(l_{\tilde{c}}(\tilde{A}))\)

Thus \(E(l_h(A)) \leq E(l_{\tilde{c}}(\tilde{A}))\).

But \(E(l_{\tilde{c}}(\tilde{A}))\) is minimal average codeword length, so we must have equality. Hence \(h\) is optimal.
Chapter 2: Error Correcting Codes

4. Noisy Channels and Hamming's Code

Up until now, we have assumed that each coded message is transmitted without error through the channel to B, who then decodes it.

\[ \text{encoder } A \rightarrow A - \text{[Channel]} - B - \text{decoder } B \]

What if there is a small, but appreciable chance of error? We would like to devise codes than can detect or, better still, correct such errors.

Discrete Memoryless Channel

Definition 4.1

We have a coded message \( b_1, b_2, \ldots, b_n \in B^* \) sent through a channel, where each letter of \( b_i \) may be altered. Assume that changes to one letter are independent of others. The channel is memoryless. Let \( B_n \) be a random variable that gives the \( n \)th letter of our message, and \( B_n' \) the random variable giving the \( n \)th letter received, then

\[
\Pr(b_1', b_2', \ldots, b_n' \text{ received} \mid b_1, b_2, \ldots, b_n \text{ sent}) = \prod_{n=1}^{n} \Pr(B_n' = b_n' \mid B_n = b_n)
\]

We have a transition matrix \( \Pr(B_n' = i \mid B_n = j) \), \( i, j \in B \).

As the probability of error is small, the \((i, i)\)th entry should be 'close' to 1. Assume that the channel is time independent, i.e., the chances of a particular alteration do not depend on \( n \).
Example 4.2

a) Binary Symmetric Channel (BSC)

\[ A = B = \{0, 1\} \]. A BSC is a time independent, memory less channel with transition matrix \( \begin{pmatrix} p & q \\ q & p \end{pmatrix} \)

\( p = \) probability of error in a single bit (usually small, \(< \frac{1}{2}\)), \( q = 1 - p \)

If \( p = \frac{1}{2} \), no information is transmitted, and we have all noise. This channel is called useless.

If \( p = 0 \) or 1, there is no noise and no errors to be corrected. We call these channels lossless or perfect.

b) Paper-Tape Codes

8-track paper tape is used. \( A = \{0, 1\} \), with a 1 represented by a hole punched in the tape. The first 7 bits \( x_1, \ldots, x_7 \) carry data, while \( x_8 \) is a check digit, \( \sum_{i=1}^{8} x_i \equiv 0 \) (2) \( (*) \)

We model the chances of error in tape using BSC, with \( p \) the probability of error. \( 1 - p \) (receive \( x_1, \ldots, x_7 \) without error) = \( (1 - p)^7 \)

If there is one error in our 8-bits, \( x_1, \ldots, x_8 \), then the check \( (*) \) fails for the received message and the error is detected. If there are two errors then \( (*) \) is true and we do not detect the error.

Remark

Check digits are widely used to detect errors, see sheet 2.
Definition 4.3

A binary \([m, m]-\text{code}\) is a subset \(C \subseteq \{0, 1\}^m\) comprising codewords \(c_i\) (i.e., a fixed length code of length \(m\)). We say that \(C\) has size \(m = k < 1\). If \(m\) is large, we can send lots of messages (more information) but as \(m\) increases, it is harder to distinguish between different messages, for example when errors occur.

Example

\(m = 1\): errors cause no problem, but no information is transmitted.

\(m = 2\): lots of messages but any error moves \(c_i \Rightarrow c_j\).

Definition 4.4

The information rate of \(C\) is \(r(C) = \frac{\log_{2} m}{n}\).

Note that \(m \leq 2^n\), so \(r(C) \leq 1\); \(r(C) = 0\) if \(m = 1\) and \(r(C) = 1\) if \(m = 2^n\).

Assume that all messages are equally likely and that the errors are independent. B's strategy to decode is to guess that the codeword sent is one which differs in the fewest places from the \(n\)-string received.

Definition 4.5

Let \(x, y \in \{0, 1\}^n\). Write \(d(x, y) = \sum_{j=1}^{n} |x_j - y_j|\). and call it the Hamming Distance between \(x, y\).

Check that \(d\) is a metric on \(\{0, 1\}^n\). \(d(x, y)\) counts the number of coordinates where \(x, y\) differ.
When \( x \in \{0, 1\}^n \) is transmitted through a noisy channel and we receive an altered word \( y \), then \( d(x, y) \) counts the number of letters altered.

**Decoding With Errors**

Suppose that \( b \in \{0, 1\}^n \) is sent through a noisy channel and received as \( v \in \{0, 1\}^n \). Errors mean that the word may not be a codeword at all, or may be the wrong codeword. We have 3 possible decoding rules:

i) **Ideal Observer**

Decode \( v \) with \( P(b \text{ sent} \mid v \text{ received}) \) maximal.

ii) **Maximum Likelihood**

Decode \( v \) with \( P(v \text{ received} \mid b \text{ sent}) \) maximal.

iii) **Minimum Distance**

Decode \( v \) as \( b \) with \( d(b, v) \) minimal.

**Proposition 4.6**

If all codewords are equally likely, then i) and ii) give the same result.

**Proof**

\[
P(v \text{ received} \mid b \text{ sent}) = \frac{P(v \text{ received and } b \text{ sent})}{P(b \text{ sent})} = \frac{P(b \text{ sent} \mid v \text{ received})P(v \text{ received})}{P(b \text{ sent})}
\]

So if all codewords are equally likely, we have the same result.

**Proposition 4.7**

If letters are transmitted through a BSC with error probability \( p < \frac{1}{2} \) then ML and Min-distance rules give the same result.
Proof

\[ P(\text{received } 16 \text{ errors}) = \prod_{i=1}^{n} P(\text{received } 1b_i \text{ errors}) = p^d (1-p)^{n-d} = (1-p)^{(1-p)^d} \]

where \( d = \#\text{ terms where } V_j \neq b_i \). Hence \( d = d(V, b) \).

Since \( 0 \leq p < 1 \), we see that \( P(\text{received } 16 \text{ errors}) \) is maximal when \( d(V, b) \) is minimal. \( \square \)

We generally use minimum distance decoding. This is time-consuming if the size of the code is large.

We must specify some convention in the case where \( d(b, V) = d(V) \).

(make a random choice, resend the message).
Definition 4.8 (closed Hamming ball)

The ball of radius $r \geq 0$ centred on $b \in \{0,1\}^n$ is

$$B(b, r) = \{ v \in \{0,1\}^n : d(b, r) \leq r \}$$

It's volume is the number of points that it contains.

$$V(n, r) = |B(b, r)| = \sum_{i=0}^{r} \binom{n}{i}, \text{ clearly independent of the centre } b. \text{ We will estimate the size of } V(n, r) \text{ in Chapter } 6.$$

Definition 4.9

Let $C$ be a code as in 4.3. $C$ is

i) $d$-error detecting if changing up to $d$-digits in each codeword can never produce another codeword.

ii) $e$-error correcting if, knowing that $v \in \{0,1\}^n$ differs from a codeword in at most $e$-places, we can deduce the codeword.

4.10 Examples

a) Repetition code of length $n$ (an $[n,2]$ code)

$$C = \{ (c, c, \ldots, c) \text{ with } c = 0 \text{ or } 1 \}$$

$C$ is $(n-1)$-error detecting and $\frac{n-1}{2}$-error correcting. The ML-decoder chooses the symbol occurring most often. $p(c) = \frac{1}{n}$ (low

b) Paper-tape code (4.26) (an $[n, 2^{n-1}]$ code)

$$C = \{ (c_1, \ldots, c_n) : c_1 + c_2 + \ldots + c_n = 0 \text{ (2)} \}$$

(We're identifying $\{0,1\}^n$ with $F_2^n$). The code is 1-error detecting

(If $x \in F_2^n$ is obtained from $c \in C$ by a single error we have $x_1 + \ldots + x_n = 1$)
It is not error-correcting, since if \( \sum_{i=1}^{n} x_i = 1 \), then every codeword with distance \((x,y) = 1\) has \( p(c) = \frac{n-1}{n} \).

**Example 4.11 (Hamming's Original Code) \([7, 4]\)**

This is a 1-error correcting \([7, 16]\) code with \( p(c) = \frac{4}{7} \)

\[ C = \mathbb{F}_2^7 \text{ defined by} \]
\[ C = \begin{cases} \{ c \in \mathbb{F}_2^7 & c_1 + c_3 + c_5 + c_7 = 0 \ (2) \\ c_2 + c_3 + c_4 + c_6 + c_7 = 0 \ (2) \\ c_4 + c_5 + c_6 + c_7 = 0 \ (2) \} \end{cases} \]

We may choose \( c_3, c_5, c_6, c_7 \) freely, then \( c_1, c_2, c_4 \) are completely determined.

\[ \therefore |C| = 2^4 = 16 \quad p(C) = \frac{\log_2 |C|}{n} = \frac{4}{7} \]

Suppose that we receive \( \mathbb{F}_2^7 \). Form the syndrome \( Z = (z_1, z_2, z_4) \in \mathbb{F}_2^3 \)
\[ z_1 = x_1 + x_3 + x_5 + x_7 \ (2) \]
\[ z_2 = x_2 + x_3 + x_6 + x_7 \ (2) \]
\[ z_4 = x_4 + x_5 + x_6 + x_7 \]

If \( Z = (0, 0, 0) \), then \( x \in C \) is correct. If not, and \( c \in C \) such that \( d(x, c) = 1 \), then \( x_i, c_i \) differ for \( i = z_1 + z_2 + 4z_4 \) (ordinary addition, \( z \) is interpreted as the binary number which gives the number of the bit which is in error). Simply check case by case the 7 binary sequences \( x \) containing one 1 and six 0's.

Thus

**Proposition 4.12 (Hamming's Binary Code)**

Hamming's Binary \([7, 12]\) code is 1-error-correcting.

**Remark**

If 2 errors occur then the decoder always gives the wrong answer.
5. Error-Detecting and Error-Correcting

How good can error-correcting and error-detecting codes be?

Definition 5.1

The (minimum) distance of a code \( C \subseteq \{0, 1\}^n \) is the minimum value of \( d(c_1, c_2) \) for \( c_1, c_2 \) distinct codewords.

5.2 Notation

A code \( C \) of length \( n \), size \( m \), with distance \( d \) is written as \( [n, m, d] \) code \( (C \subseteq F_2^n, |C| = m, d = \min\{d(x, y) : x, y \in C, x \neq y\}) \)

Lemma 5.3

Let \( C \) have minimum distance \( d \).

i) \( C \) is \( (d-1) \)-error detecting, but cannot detect all sets of \( d \)-errors.

ii) \( C \) is \( \lfloor \frac{d-1}{2} \rfloor \) error-correcting, but cannot correct all sets of \( \lfloor \frac{d-1}{2} \rfloor + 1 \) errors.

Proof

i) \( d(c_1, c_2) \geq d \) for all distinct \( c_1, c_2 \subseteq C \); \( C \) is \( (d-1) \)-error detecting.

But \( d(c_1, c_2) = d \) for some \( c_1, c_2 \subseteq C \); \( C \) cannot detect all sets of \( d \)-errors.

ii) Recall (4.8), \( B(x, r) = \{ y \in F_2^n, d(x, y) \leq r \} \). \( C \) is \( e \)-error correcting iff for all distinct \( c_1, c_2 \subseteq C \), \( B(c_1, e) \cap B(c_2, e) = \emptyset \).

But \( x \in B(c_1, e) \cap B(c_2, e) \) then \( d(c_1, c_2) \leq d(c_1, x) + d(c_2, x) \), \( d(c_1, c_2) \leq 2e \). So if \( d \geq 2e + 1 \) then \( C \) is \( e \)-error-correcting.

Take \( e = \lfloor \frac{d-1}{2} \rfloor \). Let \( c_1, c_2 \subseteq C \), \( d(c_1, c_2) = d \). Let \( x \in F_2^n \).
differ from \( c_1 \) in \( e \)-digits where \( c_1, c_2 \) differ. Then \( d(x, c_1) = e \) and \( d(x, c_2) = d - e \) and if \( d \leq 2e \) then \( B(c_1, e) \cap B(c_2, e) \neq \emptyset \). i.e. \( C \) cannot correct all sets of \( e \)-errors. Take \( e = \frac{d}{2} \).

5.4 Examples

1. Repetition code of length \( n \) in an \([n, 2, n] \) code

2. Paper tape code of length \( n \) is an \([n, 2^{n-1}, 2] \) code (check!)

3. Hamming code \((4, 11)\) as a \([7, 16, 3]\) code (1-error-correcting)

\[ d \geq 3, \text{ but also 0000000 and 1110000 are both codewords, so } d = 3. \]

It is 2-error-detecting.

6. Bounds on Codes

**Theorem 6.11** (Hamming's Bound)

Let \( C \) be \( e \)-error correcting of length \( n \). Then \( |C| \leq \frac{2^n}{v(n, e)} \)

**Proof**

\( C \) is \( e \)-error-correcting \( \Rightarrow \) \( B(c, e), c \in C \) are pairwise disjoint.

Then \( \sum_{c \in C} |B(c, e)| \leq |H_2^n| = 2^n \)

\[ \Rightarrow |C| v(n, e) \leq 2^n \]
Definition 6.2
A code $c$ of length $n$, size $m$ which is $e$-error-correcting is called perfect if $m = 2^{\lceil n/e \rceil}$

$(\Rightarrow) \forall x \in \mathbb{F}_2^n \exists! c_c \in C$ with $d(x, c) \leq e$

$(\Leftarrow) \mathbb{F}_2^n = \bigcup_{c \in B(c, e)} \text{i.e. any } e+1 \text{ errors will make you decode wrongly}$

Example
The Hamming Code $(4, 11)$ is 1-error-correcting, $n = 7$, $m = 2^4$,

$N(7, 1) = \begin{pmatrix} 7 \\ 0 \end{pmatrix} + \begin{pmatrix} 7 \\ 1 \end{pmatrix} = 8 = 2^3 \therefore c_1 = 2^4 = 2^{7/3}$

Plancher Lemma 6.3
Hamming’s Original $[7, 16, 3]$ code is perfect.

Remark
If $2^{\lceil n/e \rceil} \notin \mathbb{Z}$ then no perfect $e$-error-correcting code of length $n$ can exist. Sheet 3, q.8, 17 deal with the converse, a Golay code.

Now a lower bound:

Definition 6.4
$A(n, d) = \max \{ m : \exists [n, m, d] \text{ code} \}$

Examples
$A(n, n) = 2$, $A(n, 1) = 2^n \geq A(n, 2) = 2^{n-1}$. In fact,

Lemma 6.5
$A(n, d + 1) \leq A(n, d)$
Proof
Let $m = A(n, d+1)$. So $E$ is a code $G$ with parameters $[n, m, d+1]$. Let $c_1, c_2 \in C$ with $d(c_1, c_2) = d+1$. Let $c_1'$ be obtained from $c_1$ by switching one of the digits where $c_1, c_2$ differ.

$d(c_1', c_2) = d$. If $c \in C \setminus \{c_1, c_2\}$ then $d(c, c_1) \leq d(c, c_1') + d(c_1', c_2) + 1 = d(c_1', c_2) + 1 \Rightarrow d(c, c_1) \geq d$.

Replacing $c_1$ by $c_1'$ gives $[n, m, d]$ code, so $A(n, d) > m$.

Corollary 6.6

$A(n, d) = \max \{m : \exists [n, m, d']$ code for some $d' \geq d\}$

Theorem 6.7

Hamming Bound  Sphere Packing Bound

$2^n / V(n, d-1) \leq A(n, d) \leq 2^n / V(n, \lfloor d-1 \rfloor)$

Gilbert-Varshamov (GV) Bound  Sphere Covering Bound

Proof
Let $m = A(n, d)$. $\exists [n, m, d]$ code, $C$, say, i.e. no word can be added to $C$ without decreasing the minimum distance $d$.

Now $\forall x \in F_2^n$ such that $d(x, c) \geq d$ for $c \in C$, otherwise replace $C$ by $C' = C \cup \{x\}$, an $[n, m+1, d]$ code.

So $F_2^n = \bigcup_{C \subseteq C} B(c, d-1) \Rightarrow 2^n \leq \sum_{C \subseteq C} |B(c, d-1)| = \sum_{C \subseteq C} |C| V(n, d-1) \Rightarrow m = |C| \geq 2^n / V(n, d-1)$

Example

$n = 10, d = 3, V(10, 2) = \binom{10}{0} + \binom{10}{1} + \binom{10}{2} = 56$, $V(10, 1) = 10$

$6.7 \Rightarrow \frac{2^{10}}{56} \leq A(10, 3) \leq \frac{2^{10}}{10}$, i.e. $A(10, 3) \leq 9.3$.
The best that we can say is that $72 \leq A(10, 3) \leq 79$ but $A(10, 3)$ is unknown.

Asymptotics for $V(n, r)$

We study $\frac{1}{n} \log A(n, \lfloor n \sigma \rfloor)$ as $n \to \infty$ to see how large the information rate can be for a given error rate.

**Proposition 6.8**

Let $\sigma \in (0, \frac{1}{2})$

1. $\log V(n, \lfloor n \sigma \rfloor) \leq n H(\sigma)$

2. $\frac{1}{n} \log A(n, \lfloor n \sigma \rfloor) \geq 1 - H(\sigma)$

where $H : [0, 1] \to \mathbb{R}$, $p \mapsto \begin{cases} -\log p - q \log \frac{q}{1-p} & p \in (0, 1) \ \ q=1-p \\ 0 & p \in \{0, 1\}$

**Proof**

1) $\Rightarrow$ 2)

**GUV:** $A(n, \lfloor n \sigma \rfloor) \geq \frac{2^n}{V(n, \lfloor n \sigma \rfloor)}$

$\frac{1}{n} \log A(n, \lfloor n \sigma \rfloor) \geq 1 - \frac{1}{n} \log V(n, \lfloor n \sigma \rfloor) \geq 1 - H(\sigma)$

1) $H(\sigma)$ is increasing for $\sigma \leq \frac{1}{2}$. WLOG, assume $n \sigma \in \mathbb{Z}$.

\[
1 = (\sigma + (1-\sigma))^n = \sum_{j=0}^{n} \binom{n}{j} \sigma^j (1-\sigma)^{n-j} \geq \sum_{j=0}^{\lfloor n \sigma \rfloor} \binom{n}{j} \sigma^j (1-\sigma)^{n-j} \geq (1-\sigma)^n \sum_{j=0}^{\lfloor n \sigma \rfloor} \binom{n}{j} (\frac{\sigma}{1-\sigma})^j \geq (1-\sigma)^n \sum_{j=0}^{\lfloor n \sigma \rfloor} \binom{n}{j} (\frac{\sigma}{1-\sigma})^j = \sigma^{\lfloor n \sigma \rfloor} (1-\sigma)^{n(1-\sigma)} V(n, n \sigma)
\]

Taking logs, base, $0 \geq n \sigma \log \sigma + n(1-\sigma) \log (1-\sigma) + \log V(n, n \sigma)$

i.e. $0 \geq -n H(\sigma) + \log V(n, n \sigma)$

\[\square\]

Actually, the constant $H(\sigma)$ in 6.8.i) is the best possible.
Theorem 6.9

For $8 \in (0, \frac{1}{2})$, \( \frac{1}{n} \log V(n, \lfloor 8n \rfloor) \to H(8) \) as \( n \to \infty \).

Proof

Let \( \delta \in (0, \frac{1}{2}) \). Let \( r \in [0, \frac{\delta}{2}] \), and recall that \( V(n, r) = \sum_{i=0}^\infty \binom{n}{i} \leq V(r) \leq \binom{n}{r+1} \frac{r^n}{(r+1)^n} \) \( \ast \) \n
Recall Stirling's Formula: \( \ln n! = n \ln n - n + O(n) \) \( \ast \)

and by substituting for \( \log_2 n! \) in \( \log_2 n! \)
\( \ln(n!) = (n \ln n - n) - (n \ln r - r) - (n-r) \ln(n-r) - (n-r) + O(n) \)
\( \Rightarrow \log_2(n!) = n \log_2 n - n \log_2 \frac{r}{n} - (n-r) \log_2 \frac{n-r}{n} + O(\log_2 n) = n H(\frac{r}{n}) + O(\log_2 n) \)
\( \Rightarrow H(\frac{r}{n}) + O(\frac{1}{n \log_2 n}) \leq \frac{n}{\log_2 n} V(n, r) \leq H(\frac{r}{n}) + O(\frac{1}{n \log_2 n}) \)
\( \therefore \lim_{n \to \infty} \frac{1}{n} V(n, \lfloor 8n \rfloor) = H(8) \)

Theorem 6.10

For \( 8 \in (0, \frac{1}{2}) \), define the "asymptotic bound" \( \alpha(8) = \liminf_{n \to \infty} \frac{1}{n \log_2(n \lfloor 8n \rfloor)} \)

Then \( 1 - H(8) \leq \alpha(8) \leq 1 - H(\frac{8}{2}) \). Use a combination of

the Hamming bound, GSV bound, and 6.9. \( \square \)

The Hamming bound and 6.10 suggest that it is not possible to have an information rate \( > 1 - H(8) \) for an error rate \( 8 < \frac{1}{2} \). But the GSV bound and 6.10 suggests that it is always possible to have an information rate \( > 1 - H(25) \) for an error rate \( 8 < \frac{1}{2} \). So Shannon realized how to improve the information rate while retaining the error rate low.
3. Shannon's Theorem

7. Capacity

Given a time-independent memory-less channel (4.1, 4.2), a random letter \( B \in B \) channel \( B \rightarrow B' \) is sent and received. A noisy channel means that maybe \( B' \neq B \). The average amount of information given by \( B \in H(B) \), and received by \( B' \) is \( H(B') \). The conditional entropy \( H(B' | B) \) is the info due to noise in the channel. The remaining info is \( H(B') - H(B' | B) \), the part due to the letter being sent.

\[
H(Y | X) = H(X, Y) - H(X) = \sum_{x} p(x = x) H(Y | X = x)
\]

Definition 7.1

The mutual information of \( B, B' \) is \( I(B', B) = H(B') - H(B' | B) \), the amount of uncertainty about \( B' \) that is removed by knowing \( B \).

Lemma 7.2

a) \( I(B', B) = H(B') + H(B) - H(B', B) = I(B', B) \)

b) \( I(B', B) \geq 0 \) with equality \( \iff B, B' \) independent

c) \( I(B', B) \leq H(B') \) with equality \( \iff B' \) is a function of \( B \)

d) \( I(B', B) \leq H(B) \) with equality \( \iff B \) is a function of \( B' \).

Proofs

a) \( H(B' | B) = H(B', B) - H(B) \)

\[
\therefore I(B', B) = H(B') - H(B' | B) = H(B') - H(B', B) + H(B)
\]

So this is symmetric in \( B, B' \).
b) in (2.4)

c) \( H(B' | B) \geq 0 \), equal iff \( B' \) is a function of \( B \).

d) From c), a)

We know that \( P(B = b) \), and transition probability \( P(B' = b' | B = b) \). We can calculate both

\[
P(B = b) = \sum_{b' \in B} P(B' = b' | B = b) P(B = b)
\]

and hence obtain \( H(B), H(B'), H(B', B), I(B', B) \). If we have no noise then \( B' = B \) and (7.2) implies that \( I = H(B) \).

**Definition 7.3**

The *information capacity* is map \( \{ I(B', B) \} \) over all probability distributions for \( B \). The distribution \( P = (P_1(a_1), \ldots, P_k(a_k)) \)

\( \in [0, 1]^k \) (compact) and \( I \) is a continuous function of \( P \).

Therefore, the map is attained for some distribution. So capacity depends only on the transition probabilities \( P(B' | B) \).

**Theorem 7.4** (Information Capacity for BSC)

If a BSC has error probability \( p \) then its info capacity is

\[
C(p) = 1 - H(p) = 1 + p \log p + q \log q
\]

**Proof**

\[
P(B = 1) = t, \ P(B = 0) = 1 - t \quad \text{distribution of } B
\]

\[
H(B) = H(1-t, t) = -t \log t - (1-t) \log (1-t)
\]

entropy of \( B \).
\[ H(B' | B) = P(B = 1) H(B' | B = 1) + P(B = 0) H(B' | B = 0) \]

\[ = t H(1-p, p) + (1-t) H(p, 1-p) \]

\[ \begin{align*}
P(B' = 1) &= t(1-p) + (1-t)p = t + p - 2tp \\
P(B' = 0) &= t p + (1-t)(1-p) = (1-p) - t + 2tp
\end{align*} \]

Distribution of \( B' \)

Entropy of \( B' \): \( H(B') = h(t + p - 2tp) \)

Mutual info: \( I(B', B) = H(B') - H(B' | B) = h(t + p - 2tp) - h(p) \)

We can choose any distribution for \( B \) i.e. any \( t \in (0, 1) \).

\[ \max \left[ h(t + p - 2tp) \right] = 1 \text{, attained when } t = \frac{1}{2}, t + p - 2tp = \frac{1}{2} \]

So the info capacity = \( 1 - h(p) \)

\[ p = 0 \text{ or } 1, \quad c(0) = c(1) = 1 \quad (\text{perfect transmission}) \]

\[ p = \frac{1}{2},\quad c(\frac{1}{2}) = 0 \quad (\text{perfect scrambler}) \]

The Noisy Coding Theorem

Given any code \( C \) and any decoding scheme for \( C \), then

\[ e(C) = \text{error probability} = \frac{1}{k} \sum_{i=1}^{k} P(\text{error} \mid \text{c}_2 \text{ transmitted}) \]

where there are \( k \) codewords \( \text{c}_1, \ldots, \text{c}_k \) in \( C \) (i.e. \( e(C) \) is the average probability of error on the assumption that all codewords are equally likely to have been sent).

For binary codes, we assume that ML (max distance) decoding rule is used and refer to the "error probability" of code without mentioning the decoding rule. We seek codes with small \( e(C) \), requiring

Definition 7.5

The maximum error probability \( \tilde{e}(C) = \max_\text{to h small} \{ e(C) \} \).
Recall that the rate of transmission for $C$ is $R = p(C)$ which we want to make as large as possible. We aim to relate this to the information capacity of the channel.

For the BSC, capacity $C(p) = 1 + p \log_2 p + q \log_2 q$.

**Theorem 7.6 (Shannon, 1948)**

Given a BSC of capacity $C$ and any $R$ with $0 < R < C$, then, if

$\left\{ M_n : 1 \leq n < \infty \right\}$ is any sequence of integers satisfying

$1 \leq M_n \leq 2^{nR}$ \hspace{1cm} (1 \leq n < \infty) and any $\epsilon > 0$, then there exists a sequence of codes $\left\{ C_n : 1 \leq n < \infty \right\}$ and an integer $N_0(\epsilon)$ with $C_n$ having

$m_n$ codewords of length $n$, and with max error probability

$\hat{e}(C_n) \leq \epsilon$ \hspace{1cm} $\forall n \geq N_0(\epsilon)$.

So provided the transmission rate is below the channel capacity, we achieve arbitrarily high reliability.
Lemma 7.7 (Chebyshev’s Inequality)
If $X$ is a real valued random variable, mean $\mu$, finite variance $\sigma^2$, then for any $a > 0$, $P(|X - \mu| \geq a) \leq \frac{\sigma^2}{a^2}$

Lemma 7.8 (Tail Inequality)
Given $s, 0 \leq s \leq \frac{1}{2}$, $\sum_{j=0}^{\infty} \binom{n}{j} \leq 2^{n h(s)}$
where $h(p) = \frac{-p \log p - q \log q}{2}$, see 6.8 i).

Proof of 7.6 (Welsh)
We use the method of "random coding". Fix $n \in \mathbb{N}$ and work with binary codes in the space $V_n = \{0,1\}^n$ over $\mathbb{F}_2$. We seek a code with $m$ codewords $b_i \in V_n$. Choose these $b_i$ by just selecting a vector at random from $V_n$ independently for each $b_i$.

Decide as follows: Fix $r > 0$ and consider $B(v, r)$; if we receive $v$ we decode as the codeword $b_i$ if $b_i$ is the unique codeword in $B(v, r)$, otherwise decode $v$ as some arbitrary codeword, say $b_i$.

Notation
Send a codeword $b \rightarrow$ receive $v$ (i.e., an error has occurred)

Let $E = \{v \text{ is decoded as a codeword other than } b_i\}$

This can happen if, either

A) $d(b, v) > r$, i.e., too far from any other codeword

B) $d(b, v) \leq r$ and $d(b', v) \leq r$, for some other codeword $b'$

$E = A \cup B$, so $P(E) = P(A \cup B) \leq P(A) + P(B)$
Now because if both

\[ B_1 \leq r \text{ errors are made in transmission, and} \]

\[ B_2 \text{ one of the codewords other than } b \text{ is within } r \text{ of } v. \]

\[ B = B_1 \cap B_2 \quad \therefore P(B) \leq P(B_2) \quad 1 \]

For \( B_2 \), codewords are chosen randomly \( \therefore P(b \text{ within } r \text{ of } v) = \frac{V(n, r)}{2^n} \)

Hence the probability that at least one of the other \( m-1 \) codewords (\( \neq b \)) is within distance \( r \) of \( v \) satisfies \( P(B_2) \leq \frac{m-1}{2^n} V(n, r) \quad 2 \)

So \( \forall \varepsilon > 0 \), taking \( r = \ln \frac{1}{np + n\varepsilon} \), from \( 1 \) and \( 7.8 \)

\[ P(B) \leq \frac{m}{2^n} 2^{-n(h(p + \varepsilon))} \quad 3 \]

For \( A \), let \( U = \# \text{ symbols in error when sending codeword } b \)

\[ P(A) = P(U > r) \text{ and } U \text{ is Binomial } (n, p). \]

Hence, by Chebyshev (7.7),

\[ P(A) = P(U > (p + \varepsilon)n) \leq P(|U - np| > n\varepsilon) \leq \frac{\text{Var}(U)}{n^2 \varepsilon^2} \]

Now \( \text{Var}(U) = npq \), so

\[ P(E) \leq m \frac{2^{-n(1 - h(p + \varepsilon))}}{n\varepsilon^2} + \frac{pq}{n^2 \varepsilon^2} \text{ for sufficiently large } n. \]

But, from 7.4, \( C(p + \varepsilon) = 1 - h(p + \varepsilon) \) so this reduces to

\[ P(E) \leq \frac{pq}{n\varepsilon^2} + m 2^{-nC(p + \varepsilon)} \]

Since \( \varepsilon > 0 \), this error probability is arbitrarily small for

\( n \) large, provided that \( m = m(n) \) grows at a rate no faster than \( 2^{nC(p)} \).

This only bounds the average error probability. \( \Box \)

Claim

There exist codes \( C_n \) with \( m_n \) codewords where \( m_n \leq 2^{R_n} \), \( \varepsilon(n) < \varepsilon \)
Let $E' = \frac{\varepsilon}{2}$ and $m'_n = 2^m$.

Since $m_n / 2^n$ and $R < C$, there exists $R'$ such that $R < R' < C$ and $N'_0$ such that $m_n \geq N'_0$, $m'_n \leq 2^n R'$, and a sequence of codes $C_n'$ such that $C_n'$ has $m'_n$ codewords and average error probability $< \varepsilon'$ for $n > N'_0$.

But if $x_1, \ldots, x_{m'_n}$ are codewords of length $C_n'$, this means that

$$\sum_{i=1}^{m'_n} P(E | x_i) \leq \varepsilon' m'_n$$

Hence at least half of the $x_i$ must satisfy $P(E | x_i) \leq 2\varepsilon' \leq \varepsilon$.

Let $C_n$ be any $m_n$ of these codewords satisfying (4). We now have the required code and $\varepsilon' \leq \varepsilon$.

7. Remarks and Examples

i) 7.6 tells us that such codes exist, but not how to construct them.

ii) 7.6 extends to more general noisy channels (e.g., when noise is generated by a Markov chain).

iii) The order of magnitude for the rate at which error probabilities approach 0 are known (Shannon 1957).

iv) Shannon's Theorem has a converse:

For a discrete, memoryless channel of capacity $C$, and any $R < C$, there cannot exist a sequence of codes $C_n$ with the property that $C_n$ has $2^n$ codewords of length $n$ and error probabilities $\varepsilon(C_n) \rightarrow \varepsilon$ as $n \rightarrow \infty$. (Actually, Wolfowitz (1961) showed that the maximum error probabilities $\rightarrow 1$ as $n \rightarrow \infty$).
Every day at noon, you make a bet with me for any amount $k$ that you choose. Whatever happens, I keep the stake.

I toss a coin, and if Heads, I pay you $k u$. If Tails, I pay you nothing.

$P(\text{H}) = p$. What is the best strategy?

*Expected Winnings*: $-k + pk u$

If $pu < 1$, $E(\text{winning}) < 0$, don't bet.

If $pu > 1$, you bet. But how much? If you bet your entire fortune (bankroll) and win, then you are better off than if you had bet a smaller sum; but if you lose, you are bankrupt.
What proportion, \( w \), to bet each time? Choice of \( w \) is always the same. Suppose that your fortune after \( n \) plays is \( X_n \), and initial bankroll is \( X_0 = 1 \).

Consider \( Z_n = \frac{X_n}{X_n} \), independent identically distributed random variables, say to \( Z \), where \( Z_{n+1} = \frac{uw + (1-w)(n+1)^{th \text{ toss}}} {1 - \frac{w}{n+1}} \). Now \( \frac{X_n}{X_0} = \prod_{i=0}^{n-1} Z_i \), so \( \log X_n = \sum_{i=0}^{n-1} \log Z_i \).

\[ \Rightarrow \frac{\log X_n}{n} = \frac{\sum_{i=0}^{n-1} \log Z_i}{n} \rightarrow \mathbb{E}(\log(Z)) \text{, meaning, by the law of large numbers.} \]

**Lemma 8.1**

Suppose that \( Z_1, Z_2, \ldots \) are iid taking values in \( [a, b] \), \( 0 < a < b \). If \( X_n = \prod_{i=0}^{n-1} Z_i \), then

\[ \mathbb{P}\left( \left| \frac{\log X_n}{n} - \mathbb{E}(\log(Z)) \right| > \epsilon \right) \rightarrow 0 \text{ as } n \rightarrow \infty. \]

We want to choose \( w \) to maximise

\[ \mathbb{E}(\log Z_n) = \text{expected value of the random variable taking value } \log(1-w) \text{ if we lose, and } \log((1-w) + uw) \text{ if you win.} \]

\[ = (1-p) \log(1-w) + p \log((1-w) + uw) =: f(w) \]

One method: \( f'(w) = \frac{(pu - 1) + (w - 1)w}{(w - 1)(w + 1)(1-w)} \)

So if \( u \leq 1 \), obviously we don't bet. So assume \( u > 1 \). If \( pu \leq 1 \), then \( f(w) \) is decreasing for the \( w \), so don't bet.

If \( pu > 1 \), \( f \) takes a maximum, so we bet when \( w = w_0 = \frac{pu - 1}{u - 1} \).
Then \( E(\log Z_n) = (1-p) \log (1-W_0) + p \log (W_0 + (1-W_0)) \)
\[ = p \log W_{up} + q \log W_{down} - q \log (n-1) \]
\[ = \log n + p \log p + q \log q - q \log (n-1) \]

**Exercise:** Show that if you bet < optimum, your bankroll increases, but more slowly than optimum, but if you bet more than some proportion \( W_0 \), your bankroll decreases.

We have seen \(-(p \log p + q \log q)\) before as the Shannon entropy for BSC. Kelly (1956) gave interpretations in gambling, and the stock market by which an agent receives info over a noisy channel, about, for example, which horse will win.

In Shannon’s World, information can be transmitted through a noisy channel at a rate close to the capacity with negligible error, provided that the messages are long enough.

In Kelly’s World, the agent can, with arbitrarily high probability, increase her bankroll at a certain optimum rate provided she continues long enough.

---

**Definition 8.2:**

The idea that agents making a long sequence of bets should aim to maximise the expectation of the log is called the Kelly Criterion. (Ed Thorp: Beat the Market, Beat the Dealer)

*Chapter 4: Linear and Cyclic Codes*

*9 Linear Codes*

*Polynomials and Fields*

What follows are a few results from Number + Sets, Linear Algebra, GEM
- A ring $R$ is a set with operations $+ , \cdot$ satisfying the usual axioms.
- A field $k$ is a ring such that every non-zero element has a multiplicative inverse, e.g. $\mathbb{Q}, \mathbb{R}, \mathbb{C}, \mathbb{F}_p = \mathbb{Z}/p \mathbb{Z}$, prime.
- Every field is either an extension of $\mathbb{F}_p$, or of $\mathbb{Q}$.
- $k$ has $q$ elements $\iff q = p^n$, prime. These are unique up to isomorphism and will be denoted $\mathbb{F}_q$.

Examples

- $p$ prime, $\mathbb{F}_p$ a finite field, $\mathbb{F}_p$

- $n > 2$, note that $\mathbb{F}_{p^n} \neq \mathbb{F}_{p^2}$ (e.g. $x \in \mathbb{F}_2^*$, $x \neq 0, 1$, $x + 1 \neq 0, 1$, $x^2 = x + 1$, $\mathbb{F}_2^* = \{0, 1, x, x^2\}$)

- The non-zero elements of $\mathbb{F}_q$ form an abelian group under $\cdot$, $\mathbb{F}_q^*$, which is actually cyclic, of order $q - 1$. Any generator of $\mathbb{F}_q^*$ is a primitive element of $\mathbb{F}_q$.

If $\alpha$ is a primitive element, then $\mathbb{F}_q^* = \{\alpha^j \mid 1 \leq j \leq q - 1\}$. Note that $\alpha (\alpha^j) = \frac{q - 1}{\gcd(j, q - 1)}$

$\# \text{primitive elements} = \varphi(q - 1) = \{j: j = 1, \ldots, q - 1, (q - 1, j) = 1\}$

A primitive $r$th root of unity is an element of $k$ of order $r$ in $k^*$.

We want to work with alphabets that are finite dimensional vector spaces over $\mathbb{F}_q$, (e.g. $\mathbb{F}_q^d$ is a $d$-dimensional vector space over $\mathbb{F}_q$). Any $d$-dimensional vector space has $q^d$ vectors. We have a scalar product in $\mathbb{F}_q^d$. 
Every linear map $\phi : F_q^d \to F_q$ has the form $x \mapsto x \cdot y$ (for some $y \in F_q^d$).

So (*) allows an identification of $F_q^d \times (F_q^d)^* \cong F_q^d$. If $|S| < \infty$

$V = F_q^S = \{ f : S \to F_q \}$ is $|S|$-dimensional as a vector space over $F_q$.

A polynomial over $F_q$ is a formal expression $P(x) = \sum_{j=0}^{d} a_j x^j$ with $a_j \in F_q$ and $x$ an indeterminate.

Its degree $\deg(P)$ is the largest $m$ with $a_m \neq 0$. By convention, $\deg(0) = -\infty$.

Manipulate polynomials in $x$ using the standard rules but beware of anomalies.

Example

Over $F_2$, $x^2 + x$ is a non-zero polynomial but $t^2 + t = 0 \forall t \in F_2$.
Finite Fields

The multiplicative group of a finite field is cyclic. For a field $H$, $x \in H$, $|H| = n$, we have $o(x) | n$.

Therefore, for $H = F$ finite, $x \in H$, $x^n = 1$.

Suppose that there are $H[d]$ elements of order $d$.

For each element, $x^d = 1$, and they are solutions to $x^d - 1$. These generate the subgroup of solutions to $x^d - 1$ in the field, so there can be at most $\varphi(d)$ of them.

Thus,

$\sum_{d \mid n} \varphi(d) \leq \sum_{d \mid n} H[d] = n$

So $H[d] = \varphi(d)$ for all $d \mid n$.

Then, $H[\varphi(n)] = \varphi(n) > 0$.

So there is an element of order $\varphi(n)$, and this must generate $H$. 


Coding and Cryptography

$(\mathbb{F}_q[x], +, \cdot)$ is a ring, and the 'degree' is a Euclidean function for this ring $\Rightarrow$ Euclidean domain. So for $P, Q \in \mathbb{F}_q[x], D \neq 0$, $\exists ! Q, R \in \mathbb{F}_q[x]$ with $P = QD + R$, $\deg(R) < \deg(D)$.

**Remainder Theorem**

i) Given $P \in \mathbb{F}_q[x], a \in \mathbb{F}_q, \exists Q \in \mathbb{F}_q[x], r \in \mathbb{F}_q$ such that $P(x) = (x-a)Q(x) + r$

ii) If $P(x) \in \mathbb{F}_q[x]$ and $a \in \mathbb{F}_q$ is such that $P(a) = 0$, then $\exists Q$ such that $P(x) = (x-a)Q(x)$

- A polynomial $P$ is reducible if $\exists$ non-constant $Q, \deg(Q) < \deg(P)$ which divides $P$. A non-constant $P$ that is not reducible is irreducible.

- $\{x \in \mathbb{F}_q[x] \mid \text{irreducible} \} \Rightarrow \mathbb{F}_q[x]/\langle f(x) \rangle$ is a field.

If $d = \deg(f)$, this is a field of order $q^d$.

(e.g. $f(x) = x^2 + x + 1 \in \mathbb{F}_2[x], \frac{\mathbb{F}_2[x]}{(x^2 + x + 1)} = \mathbb{F}_4$)

- $\mathbb{F}_q[x]$ is a PID: if $I \triangleleft \mathbb{F}_q[x]$ then either $I = 0$, or $\exists 0 \neq D \in I$ of smallest degree. Given any $P \in I$, $P = QD + R$, $\deg(R) < \deg(D)$. But $R \in I \Rightarrow R = 0$, so $P = QD$. Hence $I = DF_q[x]$ for some $D, I = (D)$.

- Given $I = (D)$, the quotient $\frac{\mathbb{F}_q[x]}{I}$ is a ring, and $\theta : \mathbb{F}_q[x] \to \mathbb{F}_q[x]/I = R$.

- We will take $D = x^q - x$ and consider $R$. Divide any $P \in \mathbb{F}_q[x]$ by $x^q - x$ to obtain a remainder of degree $\leq n$. 

1
So each coset \( P + (x^n - 1)[F_2, [x]] \) contains a unique polynomial of degree \( n - 1 \). Hence, we can represent \( R \) by
\[
[a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} : a_0, a_1, \ldots, a_{n-1} \in F_2] \]
Multiplication in \( R \) corresponds to multiplying polynomials, and then reducing mod \((x^n - 1)\). In this case, \( R \) is a vector space of dimension \( n = \deg(f) \), i.e. \( R = F_2^n \).

**Linear Codes**

**Definition 9.1**

A **linear code** \( C \subseteq F_2^n \) is such that

i) \( 0 \in C \)

ii) \( x, y \in C \implies xy \in C \)

i.e. \( \subseteq C \) is an \( F_2 \)-subspace of \( F_2^n \).

The **rank** of \( C \) is \( k = \dim_{F_2} C \). A linear code of length \( n \), rank \( k \), is an \((n, k)\)-code, and if the minimum distance is \( d \), it is an \((n, k, d)\)-code.

If \( e_1, \ldots, e_k \) is a basis of \( C/F_2 \) then \( C = \bigoplus_{i=1}^{k} F_2 e_i \).

\[ |C| = 2^k \]. Thus an \((n,k)\) code is an \([n, 2^k]\)-code.

The information rate is \( \frac{k}{n} = R(C) \).

**Examples of Linear Codes**

i) **Repetition Code** \( C = \{ x : x = (x, x, \ldots, x), x \in F_2 \} \)

ii) **Parity Code** \( C = \{ x : \sum_{i=0}^{n-1} x_i = 0 \} \)

iii) **Hamming’s Original Code**
Lemma 9.2

The weight of \( x \in \mathbb{F}_2^n \) is \( \text{wt}(x) = d(x, 0) = 1 \{ i : x_i \neq 0 \} \)

The minimum distance of a linear code

is the minimum weight of the non-zero codewords.

Proof:

If \( x, y \in C \), then \( d(x, y) = d(x+y, 0) = \text{wt}(x+y) \)

Since \( x+y \in C \), \( \min \{ d(x, y) : x, y \in C, x \neq y \} \)

is the minimum non-zero weight of codewords.

Codes \( C_1, C_2 \) are equivalent if reordering each codeword of \( C_1 \)

using the same permutation gives the codewords of \( C_2 \).

Definition 9.3

Let \( C \) be an \( (n, k) \) code. A generator matrix for \( C \) is a \( k \times n \) matrix \( G \) whose rows form a basis for \( C \). If \( G \) is a generator matrix for \( C \) and \( G' \) is any other matrix obtained from

\( G \) by any finite sequence of elementary row or column operations

of the form:

i) Permuting rows or permuting columns

ii) Multiplying a row or column by a non-zero scalar

iii) Adding to a row a multiple of another row,

then the following is routine:

Lemma 9.4

\( C' \) is the generator matrix of \( C' \) equivalent to \( C \).
Lemma 9.5

Let $G$ be any $k \times n$ matrix whose rows are linearly independent. Then, by applying a sequence of operations $(\ast)$ to $G$, it is possible to transform $G$ to a matrix of type $[I_k | B]$, $I_k$ the $k \times k$ identity.

Proof

Gaussian Elimination transforms $G$ into row-echelon form

$G_{ij} = \begin{cases} 0 & s < u(i) \\ 1 & s = u(i) \end{cases}$ for some $u(1) < \ldots < u(k)$. Permuting columns replaces code with an equivalent code. So WLOG $u(i) = i$ $(1 \leq i \leq k)$. Thus $G = \begin{pmatrix} 1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & 1 \\ \end{pmatrix}$. Further row operations give $G = [I_k | B]$.

Theorem 9.6

If $C$ is a linear $(n,k)$-code then $\exists$ an equivalent code $C'$ with generator matrix $[I_k | B]$. Combine 9.4, 9.5.

Remark

Theorem 9.6 says that any codeword can be written as $(y, z) = (y | yB)$ where $y = (y_1, \ldots, y_k)$ can be considered the message and $z = yB$ of length $n-k$ may be considered as check digits. Any code whose codewords can be split in this way is called systematic.

Definition 9.7

Let $P \subseteq F_2^n$. Then the parity check code, denoted by $P$ is

$C = \{ x \in F_2^n : p \cdot x = 0 \ \forall p \in P \}$
Examples

- $P = \{ (1,1,\ldots,1) \}$ gives the simple parity check code.
- $P = \{ 1010101, 0110101, 0001111 \}$ gives Hamming's $[7,16,3]$-code.

Lemma 9.8

Every parity-check code is linear.

Proof:

- for $p \in P$, $p \cdot 0 = 0 \forall p \in P$
- $x,y \in C \Rightarrow p \cdot (x+y) = p \cdot x + p \cdot y = 0 \forall p \in P$
Definition 9.9
Let \( C \subseteq \mathbb{F}_2^n \), linear. Define the dual code
\[
C^\perp = \{ x \in \mathbb{F}_2^n : x \cdot y = 0 \ \forall y \in C \}
\]
This is a parity check code, hence linear. Note that
\( C \cap C^\perp \neq \{0\} \) is possible.

Theorem 9.10
\[
\text{rank } C + \text{rank } C^\perp = n
\]

Proof
Note that \( C^\perp \) is the annihilator of \( C \), and use linear algebra.
OR, WLOG, \( C \) has generator matrix \( G = [I_k, B] \). \( C \) has
\( k \) linearly independent columns, so the linear map
\( \alpha : \mathbb{F}_2^k \rightarrow \mathbb{F}_2^n \), \( x \rightarrow Gx \) is injective with \( \ker \alpha = C^\perp \),
hence \( \dim \mathbb{F}_2^k = \dim \ker \alpha + \dim \im \alpha \) by rank-nullity,
so
\[
n = \text{rank } C^\perp + \text{rank } C
\]

Lemma 9.11
Let \( C \) be linear. Then \((C^\perp)^\perp = C\). In particular, \( C \) is a parity
check code.

Proof
Let \( x \in C \). Then \( x \cdot y = 0 \ \forall y \in C^\perp \). So \( x \in (C^\perp)^\perp \),
hence \( C \subseteq (C^\perp)^\perp \). By 9.10,
\[
\text{rank } C = n - \text{rank } C^\perp
= n - (n - \text{rank}(C^\perp)^\perp) = \text{rank } (C^\perp)^\perp.
\]
So \( C = C^\perp \)
Definition 9.12
A parity check matrix $H$ for $C$ is a generator matrix for $C^\perp$.
It is an $(n-k) \times n$ matrix. Note that codewords in $C$ can now be
thought of as dependence relations between columns of $H$, i.e.
$C = \{ x \in F_2^n : Hx = 0 \}$

9.13 Syndrome Decoding
The syndrome of $x \in F_2^n$ is $Hx$. Suppose that you receive
$x = c + z$ where $c$ is the codeword, and $z$ is the error.
Then $Hx = Hc + Hz = Hz$. If $C$ is known to be $e$-error
correcting, we precompute $Hz$ for all $z$ with $wt(z) \leq e$.
On receiving $x \in F_2^n$, we look for $Hz$ in our list.
$Hx = Hz$, i.e. $H(x - z) = 0$, so $c = x - z \in C$, with
d$(x, c) = wt(z) \leq e$ (c.f. 4.11 with $e = 1$).

Example
$G = (0,0,1,0), H = (0,1,0,1)$
Codewords: 0000, 1010, 0111, 1101

Syndrome Lookup Table

<table>
<thead>
<tr>
<th>Syndrome</th>
<th>Corrector = code leader</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0000</td>
</tr>
<tr>
<td>01</td>
<td>0010</td>
</tr>
<tr>
<td>11</td>
<td>0100</td>
</tr>
</tbody>
</table>

Receive 1111, Syndrome 10. The leader = 0010, decode as 1101.
But, the leader of 10 might have been 1000, then we would
decode as 0111.
Finally, we try to exploit the fact that if $C$ is linear and $a \in C$, then $a + C \subseteq C$. Recall that $\text{wt}(x) = d(x, 0)$ ($x \in \mathbb{F}_2^n$).

**Weight Enumeration for Linear Codes**

**Lemma 9.14**

i) $\text{wt}(x) \geq 0$ , and $\text{wt}(x) = 0 \iff x = 0$

ii) $\text{wt}(\lambda x) = \text{wt}(x)$ ($\lambda \in \mathbb{F}_2^*$)

iii) $\text{wt}(x + y) \leq \text{wt}(x) + \text{wt}(y)$

This is immediate, since $d$ defines a metric on $\mathbb{F}_2^n$.  

Recall 9.2: \textit{the non-zero, min-distance of a linear code}

\textit{is the minimum non-zero weight}.

**Definition 9.15**

The \textit{weight enumeration polynomial} of a linear code $C$ is the homogeneous, 2-variable polynomial $W_C(s, t) = \sum_{i=0}^{\text{rank}
\text{C}} A_i s^i t^{n-i}$  

where $A_i = \#$ elements of $C$ of weight $i$.

**Lemma 9.16**

i) $W_C(s, t)$ is homogeneous of degree $n$.

ii) If rank $C = k$, then $W_C(1, 1) = 2^k$.

iii) $W_C(0, 1) = 1$

iv) $W_C(1, 0) = 0$ or 1

v) $W_C(s, t) = W_C(t, s)$ \forall $s, t$ \iff $W_C(1, 0) = 1$

**Proof**

i) Clear from the definition.
ii) \( W_c(1, 1) = \sum A_i = \# \text{codewords in } C = 2^k \)

iii) \( W_c(0, 1) = A_0 = 1 \) (0 is the unique codeword of weight 0)

iv) \( W_c(1, 0) = A_n = \left\{ \begin{array}{ll}
0 & (1, ..., 1) \in C \\
1 & \text{otherwise}
\end{array} \right. \)

v) \( W_c(s, t) = W_c(t, s) \Rightarrow 1 = W_c(0, 1) = W_c(1, 0) \)

(\Rightarrow) \( W_c(1, 0) = 1 \Rightarrow (1, ..., 1) \in C \)

\( \Rightarrow (x + (1, ..., 1)) \in C \Rightarrow x \in C \)

\( \Rightarrow ((1-x), ..., (1-x_n)) \in C \Rightarrow (x_1, ..., x_n) \in C \)

\( \Rightarrow A_i = A_{n-i} \Rightarrow W_c(s, t) = W_c(t, s) \)

9.17 Examples

i) BSC, probability \( p \) of error. Let \( C \) be a linear code of length \( n \).

\( W_c(p, 1-p) = P(\text{receive correct code word syndrome transmitted}) = W_c(p, 1-p) - (1-p) \)

P (receive incorrect code word transmitted) = \( W_c(p, 1-p) - (1-p) \)

ii) \( C \) the repetition code of length \( n \), \( W_c(s, t) = s^n + t^n \)

iii) \( C \) the parity check code, length \( n \). \( W_c(s, t) = \frac{1}{2} ((s+t)^n + (t-s)^n) \)

To see i), \( W_c(p, 1-p) = \sum A_i p^i (1-p)^{n-i} \)

\( = \sum (\# \text{words length } i) \times P(\text{we make } i \text{ mistakes transmitting } (0, ..., 0)) \)

\( = P(\text{codeword received } | (0, ..., 0) \text{ sent}) \)

\( = P(\text{codeword received } | \text{codeword sent}) \)

i.e. \( P(\text{receive incorrect code word code transmitted}) = P(\text{receive code word code sent}) - P(\text{no mistakes}) \)

\( = W_c(p, 1-p) - (1-p)^n \)

ii), iii) are special cases of the following theorem.
Theorem 9.18 (MacWilliams Identity)

If $C \subseteq F_2^n$ a linear code, and $C^\perp$ is its dual, then

$$W_{C^\perp}(s,t) = 2^{-\dim C} W_C(t-s, s+t)$$

Proof

Sheet 3, question 9

10 Constructing Codes from other codes

Hamming Codes

Definition 10.1

For $d \geq 1$, let $n = 2^d - 1$. Let $D = F_2^d$, the (column) vector space of dimension $d$ over $F_2$. Let $H$ be the $d \times n$ matrix whose columns are the $n$ distinct, non-zero vectors of $D$.

The Hamming $(n, n-d)$-code is the linear code $C$ of length $n$ defined by the parity check matrix $H$.

(This is only defined up to equivalence.)

Example

Hamming's $[7, 16]$ example is one, with $d = 3$.

Lemma 10.2

The Hamming $(n, n-d)$ code is a linear code $C$ of length $n$, rank $n-d$, having minimum weight 3, and is a perfect 1-error-correcting code ($n = 2^d - 1$).
Example

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

Proof of 10.2

By 9.12, the rows of \( C \) are dependence relations between columns of \( H \). Any two columns of \( H \) are linearly independent, so there are no non-zero codewords of weight \( \leq 2 \). Hence \( d(C) \geq 3 \).

If \( H = \begin{pmatrix} 1 & 1 & 1 & \ldots & 1 \\ 0 & 0 & 0 & \ldots & 0 \end{pmatrix} \) then \( 1110 \ldots 0 \in C \), hence \( d(C) = 3 \).

By 5.3, \( C \) is 1-error-correcting.

\[ \frac{2^n}{\binom{n}{k}} = \frac{2^n}{n+1} = 2^{n-1} = \frac{1}{4} \]

Definition 10.3

If \( C \) is an \([n, m, d]\) code

a) the parity check extension \( C' \) of \( C \) is the code of length \( n+1 \) given by \( C' = \{ x \in F_2^{n+1} : (x_1, \ldots, x_n, x_{n+1}) \in C, \sum_{i=1}^{n+1} x_i = 0 \} \).

b) the truncation \( C^- \) of \( C \) is the code of length \( n-1 \) given by \( C^- = \{ (x_1, \ldots, x_{n-1}) : (x_1, \ldots, x_n) \in C \text{ for some } x_n \in F_2 \} \).

c) the shortening (puncturing) \( C' \) of \( C \) by the symbol \( \alpha \in \{0, 1\} \) is the code of length \( n-1 \) given by \( C' = \{ (x_1, \ldots, x_{n-1}) : (x_1, \ldots, x_{n-1}) \} \).

This is an \([n-1, m', d']\) code, \( d' \geq d \), \( m' \geq m \) for suitable \( \alpha \).

Lemma 10.4

If \( C \) is linear, so are \( C^+, C^- \), \( C' \) (for \( \alpha = 0 \)).

d) The repetition \( C^{(m)} \) is the code of length \( mn \) obtained by repeating each word of \( C \) \( m \) times.
Combining Two Linear Codes $C_1, C_2$

The direct sum $C_1 \oplus C_2 = \{(x, y), \quad x \in C_1, \quad y \in C_2\}$ is of little interest because

**Lemma 10.5**

$$d(C_1 \oplus C_2) = \min \left\{ d(C_1), d(C_2) \right\}$$

**Proof**

$(x, 0) \in C_1 \oplus C_2$, $\text{wt}(x, 0) = \text{wt}(x)$

$(0, y) \in C_1 \oplus C_2$, $\text{wt}(0, y) = \text{wt}(y)$

$$\Rightarrow \min \text{wt}(C_1 \oplus C_2) \leq \min \left\{ \min \text{wt}(C_1), \min \text{wt}(C_2) \right\}$$

However, if $(x, y) \in C_1 \oplus C_2$, $(x, y) \neq 0$, then

$x \neq 0$, $\text{wt}(x, y) \geq \text{wt}(x)$, and/or $y \neq 0$, $\text{wt}(x, y) \geq \text{wt}(y)$

$$\min \text{wt}(C_1 \oplus C_2) = \min \left\{ \min \text{wt}(C_1), \min \text{wt}(C_2) \right\}$$

Hence we have equality.

Finally, $(0, 0) \in C_1 \oplus C_2$, $(0, y) + (0, y') = (0, x + y + y') \in C_1 \oplus C_2$

$\therefore C_1 \oplus C_2$ linear $\therefore \text{min weight} = \text{min distance}$ (9.2)

$\therefore \text{min dist}(C_1 \oplus C_2) = \min \left\{ d(C_1), d(C_2) \right\}$

Note $rk(C_1 \oplus C_2) = rk(C_1) + rk(C_2)$ and length $(C_1 \oplus C_2) = \text{length}(C_1) + \text{length}(C_2)$

**Definition 10.6**

$C_1, C_2$ linear, length $n$, $C_2 \subseteq C_1$. The bar product $C_1 \overline{C_2}$ is the code of length $2n$ with words $(x, x + y)$, $x \in C_1, y \in C_2$

**Lemma 10.7**

$C_1 \overline{C_2}$ is linear, rank $= \text{rank}(C_1) + \text{rank}(C_2)$

$$d(C_1 \overline{C_2}) = \min\left\{ d(C_1), d(C_2) \right\}$$
Proof

Let $C_1$, have a basis $x_1, \ldots, x_k$, $C_2$ a basis $y_1, \ldots, y_l$. $C_1 \cap C_2$ has basis $\{ (x_i, 1x_i) | 1 \leq i \leq k \} \cup \{ (0, y_j) | 1 \leq j \leq l \}$.

Let $Z = (x, x+y) \in C_1 \cap C_2$. Then $wt(Z) = wt(x) + wt(x+y)$.

Suppose that $Z \neq 0$, so that $x, x+y$ may not both be 0.

If $x = 0$, then $y \neq 0$, $wt(Z) = wt(y) \geq d(C_2)$.

If $x \neq 0$, $y = -x$, $wt(Z) = wt(x) \geq d(C_2)$.

since in this case $x = -y \in C_2$.

If $x \neq 0$, $y \neq -x$, then $x+y \in C_1$, so $wt(Z) = wt(x) + wt(x+y) \geq 2d(C_1)$

Now, $\exists x \in C_1$ with $d(C_1)$. Then, $d(C_1 \cap C_2) \leq wt(x) = 2d(C_1)$.

$\exists y \in C_2$ with $wt(y) = d(C_2)$. Then $d(C_1 \cap C_2) \leq wt(0y) = d(C_1 \cap C_2)$.

$\therefore d(C_1 \cap C_2) \leq \min \{ 2d(C_1), d(C_2) \}$

Decoding rule:

Knowing decoders for $C_1$, $C_2$.

Send $(x, x+y) \rightarrow (u_1u_2)$ received, with $e$-errors, $e \leq \frac{1}{2}d(C_1)$

See Goldie + Pinch, page 157.

Reed-Muller Codes

These are linear codes suitable when the error-rate is high and we are less interested in the information rate. These were famously used by NASA's early planetary probes.

Preliminaries

Let $X$ be a set, $X = \{ x_1, \ldots, x_n \}$, $|X| = n$. We have a correspondence
\[ P(x) \leftrightarrow \mathbb{F}_2^n, \text{ namely } P(x) \leftrightarrow \{ f: x \rightarrow \mathbb{F}_2 \} \leftarrow \rightarrow \mathbb{F}_2^n \]

\[ A \rightarrow 2^A, \text{ then } F \rightarrow (f(p_0), \ldots, f(p_{n-1})) \]

Symmetric difference \( A \Delta B \leftrightarrow x + y \) vector addition

Intersection \( A \cap B \leftrightarrow \text{ wedge product } \)

\[ \emptyset \leftrightarrow (0, 0, \ldots, 0), \quad V_0 = (1, \ldots, 1) \leftrightarrow 1_x \]

Take \( X = \mathbb{F}_2^d \) and list the points of this \( d \)-dimensional space with some \( P_0, \ldots, P_{n-1} \). So \( 1 \times 1 = 2^d \).

\[ V_0 = (1, \ldots, 1) \leftrightarrow 1_x \]

\[ V_i = 1_{H_i} \text{ for } 1 \leq i \leq d \text{ where } H_i = \{ P \in X : i^{th} \text{ coordinate of } P = 0 \} \]

**Definition 10.8**

The Reed-Muller code of order \( r \), \( 0 \leq r \leq d \), length \( n = 2^r \), denoted \( RM(d, r) \), is the linear code spanned by \( V_0 \), and all wedge products of \( r \) or fewer of the \( V_i \). (Convention: the empty wedge is \( V_0 \).)
Example 10.9

\( d = 3, \ X = \mathbb{F}_2^3, \ P_0 = 000, \ P_1 = 001, \ldots, \ P_7 = 111 \)

\( H_1 = \) element of \( X \) with 0 in the first coordinate

\[
\begin{array}{cccccccc}
X & 000 & 001 & 010 & 011 & 100 & 101 & 110 & 111 \\
V_0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
V_1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
V_2 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
V_3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
V_0 \land V_2 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
V_1 \land V_3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
V_2 \land V_3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
V_0 \land V_1 \land V_3 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\
\end{array}
\]

These are the generator of \( RM(3, r) \)

\( RM(3, 0) = \) repetition code of length \( 8 = \langle V_0 \rangle \)

\( RM(3, 1) = \) equivalent to a parity check extension of Hamming's original code

\( RM(3, 2) = \) paper tape code of length 8 - simple parity check

\( RM(3, 3) = \) trivial code = all elements of \( \mathbb{F}_2^3 \)

Proposition 10.10

i) The vectors \( V_i, \ldots, V_i \) for \( 1 \leq i_1 < \ldots < i_s \leq d \) and \( 0 \leq s \leq d \) are a basis of \( \mathbb{F}_2^v \)

ii) Rank \( RM(d, r) = \sum_{s=0}^{d} (\binom{s}{r}) \)

Proof:

i) We have listed in total \( \sum_{s=0}^{d} (\binom{s}{r}) = (1+1)^d = 2^d \) \( = \) \( n \) vectors so it is enough to check that \( RM(d, d) = \mathbb{F}_2^v \).

Let \( P \in X \) and \( y_i = \begin{cases} V_i & \text{if the } i \text{-th coordinate of } P \text{ is 1} \\ 0 & \text{otherwise} \end{cases} \)

Then \( L \sum_{j=1}^{n} y_j = y_1 \land \ldots \land y_d \). Expand using the distributive law
to see that $1_{F_2^3} \in RM(d,d)$, but $1_{F_2^3}$ for $P \in X$ spans $F_2^d$. The given vectors form a basis.

ii) $RM(d,r) = \text{span } \{ V_i \wedge \ldots \wedge V_i_s \mid 1 \leq i_1 < \ldots < i_s \leq d, 0 \leq r \leq d \}$

These vectors are linearly independent by i), so they form a basis. Hence $\text{rank } RM(d,r) = 1 + \binom{d}{r} + \ldots + \binom{d}{r}$.

Proposition 10.11

$RM(d,r) = RM(d-1,r) \cap RM(d-1,r-1), 0 < r < d$

Proof

$RM(d-1,r-1) \subseteq RM(d-1,r)$, so the bar product (10.6) makes sense.

Arrange vectors in $X = F_2^d$ so that $V_d = (0, \ldots, 0, 1, 1, \ldots, 1)$ with $2^{d-1}$ 0 digits then $2^{d-1}$ 1 digits.

Let $Z \in RM(d,r)$, a sum of wedges $V_i \wedge \ldots \wedge V_i_s$, some of which involve $V_d$. Write the sum of these terms as $y \wedge V_d$ and the sum of the remaining terms in $Z$ as $Z'$.

$(x, y)$ do not involve $V_d$.

$Z = Z' + (y \wedge V_d) \ (x \in RM(d,r), y \in RM(d,r-1))$

Let $x'$ be the vector containing the first $2^{d-1}$ components of $x$ and $y'$ containing the first $2^{d-1}$ components of $y$.

Now $x = (x' | x'), y = (y' | y')$ with $x' \in RM(d-1,r)$ and $y' \in RM(d-1,r-1)$.

We now have $y \wedge V_d = \{0 \} \cdot 0^2 \cup \ldots \cup \{0 \} \cdot 0^2 \cup \ldots \cup \{0 \} \cdot 0^2$.

$Z = (x' | x' + y') = (x' | x' + y')$.
Proposition 10.12
The minimum weight of an RM(d, r) code is $2^{d-r}$ ($0 \leq r \leq d$).

Proof
Clearly $WT(V, \ldots, V_r) = 2^{d-r} \Rightarrow \min \text{ weight} \leq 2^{d-r}$

$r = d \Rightarrow \text{RM is the trivial code, min weight} \ 1 = 2^0$

$r = 0 \Rightarrow \text{RM is the repetition code, min weight} \ n = 2^d$

For $0 < r < d$ we use induction on $d$.

$d = 1$ is clear. Suppose that $d > 1$, and the result is true for
RM codes of length $< 2^d$.

By 10.11, $RM(d, r) = RM(d-1, r) \cup RM(d-1, r-1)$.

By induction, min weights of $RM(d-1, r), RM(d-1, r-1)$ are

$2^{d-1-r}$ and $2^{d-r}$ respectively. By 10.7,

$\min \text{ weight } RM(d, r) = \min \{2 \cdot 2^{d-1-r}, 2^{d-r}\} = 2^{d-r}$  \ \Box$

Example 10.13

$RM(5, 1)$ is a $(32, 64, 16)$ code used by NASA for the Mariner

II Cyclic Codes

Definition II.1

A linear code $C \subseteq F_2^n$ is called cyclic if

$(a_0, a_1, \ldots, a_{n-1}) \in C \Rightarrow (a_{n-1}, a_0, a_1, \ldots, a_{n-2}) \in C$

Examples

Repetition, parity check. Hamming's $(7, 4)$ code are all
cyclic codes.
Establish a correspondence between $\mathbb{F}_2^n$ and $\mathbb{F}_2[x]/(x^n-1)$ as follows: $\mathbb{F}_2^n \iff \{ f \in \mathbb{F}_2[x] : \deg(f) < n \} \iff \mathbb{F}_2[x]/(x^n-1)$

vector $a = (a_0, \ldots, a_{n-1}) \iff a(x) = \sum_{i=0}^{n-1} a_i x^i$ polynomial

Lemma 11.2

A code $C \subseteq \mathbb{F}_2[x]/(x^n-1)$ is cyclic $\iff$

i) $0 \in C$

ii) $f, g \in C \Rightarrow f + g \in C$

iii) $f(x) \in \mathbb{F}_2[x], \ g(x) \in C \Rightarrow f(x)g(x) \in C$

(i.e. $C$ is an ideal in $\mathbb{F}_2[x]/(x^n-1)$)

Proof

If $g(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \mod (x^n-1)$

$xg(x) = a_{n-1} + a_0 x + \ldots + a_{n-2} x^{n-1} \mod (x^n-1)$

$C$ is cyclic $\iff$ i), ii) hold (C is linear) and

iii) $g \in C \Rightarrow xg \in C$ (special case $f(x) = x$ of iii)

In general $f(x) = \sum b_i x^i = \sum b_i x^i g(x) \in C$ by iii)

The converse is immediate.

Problem

Classify all cyclic codes of length $n$.

$\{\text{cyclic codes of length } n\} \iff \{\text{ideals of } \mathbb{F}_2[x]/(x^n-1)\}$

$\iff \{\text{ideals in } \mathbb{F}_2[x] \text{ containing } x^n-1\}$

$\iff \{\text{polynomials in } \mathbb{F}_2[x] \text{ dividing } x^n-1\}$
Note on 13.11 Proof

\[ V_0 = (0, \ldots, 0, 1, \ldots, 1), \quad V_i = (V_i', |V_i'|) \quad |1 \leq i \leq d-1| \\
Z = (y \wedge V_0) + x, \text{ where } x, y \text{ are sums of wedges of } V_1, \ldots, V_{d-1}, \text{ so } Z = (x', |x'|) + (y', |y'|) \wedge V_0 \]

**Theorem 11.3**

Let \( C \subseteq F_2[x]/(x^n-1) \) be a cyclic code. Then \( \exists ! \ g(x) \in F_2[x] \) such that

i) \( C = \{ f(x)g(x) \mod x^n-1 : f(x) \in F_2[x] \} \)

ii) \( g(x) \mid x^n-1 \quad (p(x) \in F_2[x] \text{ represents a codeword } \Rightarrow g(x) \mid p(x)) \)

**Definition 11.4**

\( g(x) \) is called the generator polynomial of \( C \).

**Proof**

i) Consider \( g(x) \), a non-zero polynomial of least degree representing a non-zero codeword. Note \( \deg(g) < n \). \( C \) is cyclic, so \( \exists ! \) holds.

Let \( p(x) \in F_2[x] \) represent a codeword. By the division algorithm, \( p(x) = q(x)g(x) + r(x) \) for some \( q, r \in F_2[x] \), \( \deg(r) < \deg(g) \). Hence \( r(x) = p(x) - q(x)g(x) \in C \)

\( \Rightarrow r(x) = 0 \quad \Rightarrow \deg(r) < \deg(g) < n \). Hence \( g(x) \mid p(x) \)

ii) Take \( p(x) = x^n - 1 \) in the above to get ii)

**Uniqueness**: If \( g_1, g_2 \) are two generator polynomials then \( g_1 \mid g_2, g_2 \mid g_1 \), \( \Rightarrow g_1 = ug_2 \) for \( u \) \in \( F_2^* = \{1, 3\} \).
11.3 tells us that we should seek generators $g(x)$ from amongst the factors of $x^n - 1$. With no conditions on $n$, life is dull, e.g. in $\mathbb{F}_2[x]$, $x^2 - 1 = (x-1)^2$. To avoid this, we only consider the separable cyclic codes, i.e. when $n$ is odd.

**Lemma 11.5**

Let $n \in \mathbb{N}$ be odd, and $k = \mathbb{F}_2$, in which $x^n - 1 = \prod_{j=1}^{\frac{n}{2}} f_j(x)$ with $f_j$ linear (i.e. $k = \text{spcl}(x^n - 1)$). Then $f_1, \ldots, f_n$ are all distinct. Hence $x^n - 1$ has $n$ distinct roots which form a cyclic group.

**Note**

1. In particular there are $2^5$ cyclic codes of length $n$.
2. This is false if $n$ is even: $(x^2 - 1) = (x-1)^2$ over $\mathbb{F}_2$.

**Proof**

Suppose that $x^n - 1$ has a repeated factor. Then find an extension $k \supset \mathbb{F}_2$ such that $x^n - 1 = (x-\alpha)^2 \cdot p(x)$, $\alpha \in k$, $p(x) \in k[x]$. Taking formal derivatives,

$$nx^n - 1 = 2(x-\alpha)p(x) + (x-\alpha)^2p'(x)$$

$$= (x-\alpha)[2p(x) + (x-\alpha)p'(x)]$$

Since $n$ is odd, $n \alpha \neq 0$. But $n\alpha^{n-1} = 0$.

$\therefore \alpha = 0$ since $n$ is odd. Hence $0 = \alpha^n = 1$.

If $S = \{\text{roots of } x^n - 1 \text{ in } k\}$, $|S| = n$.

Clearly $S \leq k^*$, and $k^*$ is cyclic. Hence $S$ is cyclic.
Lemma 11.6
Let \( C \) be a cyclic code of length \( n \) with generator polynomial \( g(x) = g_0 + g_1 x + \ldots + g_k x^k \) of degree \( k \) (i.e. \( g_k \neq 0 \)).
Then a basis for \( C \) is \( \{g(x), xg(x), \ldots, x^{n-k} g(x)\} \) and in particular rank \( C = n - k \).

Proof.
\( i) \) Linear Independence: Suppose that \( f(x)g(x) \equiv 0 \pmod{x^n-1} \) for some \( f(x) \in \mathbb{F}_2[x] \), \( \deg(f) < n - k \). Then \( \deg(fg) < n \), so \( f(x)g(x) = 0 \), hence \( f(x) = 0 \), i.e. every dependence relation is trivial.

\( ii) \) Spanning: Let \( p(x) \in \mathbb{F}_2[x] \) represent a codeword.
WLOG \( \deg(p) < n \). Since \( g(x) \) is generator polynomial, \( g(x) \mid p(x) \) i.e. \( p(x) = f(x)g(x) \) for some \( f(x) \). Now \( \deg(f) = \deg(p) - \deg(g) \) so \( \deg(f) < n - k \), i.e. \( p(x) \in \text{span} \{g(x), xg(x), \ldots, x^{n-k} g(x)\} \).

Corollary 11.7
A generator matrix for \( C \) is
\[
G = \begin{pmatrix}
g_0 & g_1 & \cdots & g_k & 0 & \cdots & 0 \\
g_k & 0 & \cdots & 0 & g_k & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
g_{n-k} & g_{n-k+1} & \cdots & g_{n-1} & g_k & \cdots & 0 \\
g_{n-1} & g_{n-2} & \cdots & g_{n-k} & 0 & \cdots & 0
\end{pmatrix}
\]

Definition 11.8
The parity check polynomial \( h(x) \in \mathbb{F}_2[x] \) is defined as
\( h(x)g(x) = x^n - 1 \). If \( h(x) = h_0 + h_1 x + \ldots + h_{n-k} x^{n-k} \) then the parity check matrix (9.12) is
\[
H = \begin{pmatrix}
h_{n-k} & h_{n-k-1} & \cdots & h_0 & 0 & \cdots & 0 \\
h_{n-1} & h_{n-2} & \cdots & h_0 & 0 & \cdots & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
h_k & 0 & \cdots & 0 & h_k & \cdots & 0
\end{pmatrix}
\]
Proof.
Take the dot product of the $i$th row of $G$ and the $j$th row of $H$ and compare coefficients.

Lemma 11.9
Suppose $X^n - 1 = g(x) h(x)$. Then $g$ is a generator of a cyclic code (of length $n$) and $h$ is a generator for another cyclic code which is the reverse of a dual code $C^\perp$. In particular, the dual of a cyclic code is cyclic.

Proof.
Write $X^n - 1 = \sum_{i=1}^{\ell} f_i(x)$ and let $\beta_i$ be a root of $f_i$ for some extension of $GF_2$. Then $f_i(x)$ is the minimal polynomial of $\beta_i$, and the corresponding code is just the set of polynomials $a(x)$ such that $a(\beta_i) = 0$.

Consider the set of elements in some extension of $GF_2$ and let $g(x) = \text{lcm}$ of their min. poly. over $GF_2$. Then the cyclic code with gen. poly. $g(x)$ consists of polys. $a(x)$ such that $a(x) = 0$.

\[\therefore\] prescribing a cyclic code is the same as prescribing common roots of its codewords.

Definition 11.10
A defining set for cyclic code $C$ over $GF_2$ is a set $A$ of elements in some field $K \supset GF_2$ such that $a(x) \in C$.

$\iff$ Each element of $A$ is a root of $a(x)$. 
Equivalently, \( C \) is generated by the least common multiples of the min. poly. of the elements of \( A \).

12 BCH Codes

Let \( n \) be odd. Pick \( r \geq 1 \) such that \( 2^r \equiv 1 \pmod{n} \). \( k = \mathbb{F}_{2^r} \).

Let \( \mathcal{M}_n(k) = \{ x \in k : x^r = 1 \} \leq k^* \), a cyclic group of order \( n \) (since \( n | 2^r - 1 \)).

So \( \mathcal{M}_n(k) = \{ 1, \alpha, \ldots, \alpha^{n-1} \} \).

12.1 Definition

The BCH (Bose-Chaudhuri-Hocquenghem) code with design distance \( s \) is the cyclic code \( C \) of length \( n \) with defining set \( A = \{ \alpha, \alpha^2, \ldots, \alpha^{5-1} \} \).

Here \( s \in \mathbb{N} \) such that \( 1 \leq s \leq n \).
BCH Codes

Theorem 12.2 (BCH Bound)
A BCH code $C$ with design distance $s$ has $d(C) \geq s$.

Lemma 12.3

Work over any field $k$. The vander Monde determinant

$$\begin{vmatrix}
1 & \alpha_1 & \alpha_1^2 & \ldots & \alpha_1^{s-1} \\
1 & \alpha_2 & \alpha_2^2 & \ldots & \alpha_2^{s-1} \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
1 & \alpha_n & \alpha_n^2 & \ldots & \alpha_n^{s-1}
\end{vmatrix} = \prod_{1 \leq i < j \leq n} (\alpha_j - \alpha_i)$$

Proofs of 12.2

Let $P(x) = b_0 + b_1 x + \ldots + b_{n-1} x^{n-1}$ be a non-zero codeword.

$$H = \begin{pmatrix}
\alpha_0 & \alpha_1 & \ldots & \alpha_0^{s-1} \\
\alpha_1 & \alpha_2 & \ldots & \alpha_1^{s-1} \\
\vdots & \vdots & \ddots & \vdots \\
\alpha_{n-1} & \alpha_0 & \ldots & \alpha_{n-1}^{s-1}
\end{pmatrix}$$

$H(b_i) = 0$

By 12.3 any $s-1$ columns of $H$ are linearly independent over $F_2$ since no two of the $n$ powers of $\alpha$ are equal or 0 (11.5).

But any codeword of $C$ is a dependence relation between columns of $H$. Hence $w_T(P(x)) \geq s$, so $d(C) \geq s$.

Remark

$H$ is not a parity check matrix as in 9.12 since the entries do not lie in $F_2$.

Decoding BCH Codes

Let $C$ be a BCH code, length $n$, over $F_2$, design distance $s$, defining set $A = \{\alpha, \alpha^2, \ldots, \alpha^{s-1}\}$ where $\alpha$ is the $n^{\text{th}}$ root of unity lying in $k \supseteq F_2$. We know by 12.2 that $C$ is $t$-error correcting, for $t = \lfloor \frac{s}{2} \rfloor$. Suppose that $C = (c_0, \ldots, c_{n-1})$ is sent and we receive $r + e$ where
The error vector $e$ has at most $t$ non-zero entries, i.e. $|E| \leq t$ where $E = \{0 \leq j \leq n-1: e_j = 1\}$

**Problem**

Find $e$, hence recover $c$. Under $\mathbb{F}_2^A \subseteq \mathbb{F}_2[\alpha]/(\alpha^n-1)$, $e(x) = \sum_{j \geq 0} e_j x^j$, $r(x) = \sum_{j \geq 0} r_j x^j$, $e(x) = \sum_{j \geq 0} e_j x^j$ polynomials of degrees $< n$, corresponding to $c, r, e$.

We know that $c(x)$ satisfies $c(x) = c(x^2) = \ldots = c(x^{s-1}) = c$, $r(x) = e(\alpha)$, $r(x^2) = e(\alpha^2)$, $\ldots$, $r(x^{s-1}) = e(\alpha^{s-1})$.

**Idea:** Calculate $r(x^j) (1 \leq j \leq s-1)$. If all zero then $r(x)$ is a codeword and there are no errors (or $\leq s-1$ errors).

Otherwise, assume $0 < |E| < t$

**Definition 12.4**

The error-locator polynomial is $\sigma(x) = \prod_{j \in E} (1 - \alpha^j x) \in k[x]$.

If we know $\sigma(x)$, we can find which $\alpha^j$ are roots of $\sigma(x)$ and hence find the indices where errors have occurred.

Correct the errors by changing the entries. How do we find $\sigma(x)$?

**Theorem 12.5**

Assume $\sigma(0) = |E| \leq t$. Then $\sigma(x)$ is the unique polynomial over $k$, of least degree, such that:

i) $\sigma(0) = 1$

ii) $\exists \sigma(x) \in k[x], \sigma(x) \leq t$, such that

$$\sigma(x) \times \sum_{j=1}^{2s} r(\alpha^j) x^j \equiv c(x) \pmod{x^{2s+1}}$$
The Algorithm

1. Input: \( r(x) \) (received)

2. Compute: \( \sum_{j=0}^{2t} r(\alpha^j) x^j \) \( \equiv \sigma(x) \) \( \mod x^{2t+1} \) with \( \sigma(0) = 1 \)

3. Put \( \sigma(x) = \sum_{j=0}^{2t} \sigma_j x^j \) and compute the coefficients of \( x^i \) for \( t+1 \leq i < 2t \) to obtain \( t \) linear equations for the \( \sigma_i \) \((1 \leq i \leq t)\)

4. Solve these equations over \( k \) and keep the solutions of least degree.

5. Compute \( E = \{ 0 \leq i \leq n-1 \mid \sigma(\alpha^{-i}) = 0 \} \) and check that \( |E| = \bar{e}(t) \)

6. Put \( a(x) = \sum_{i \in E} x^i \), \( c(x) = r(x) + e(x) \) and check that \( c(x) \) is a codeword.

Proof of 12.5

Existence: Let \( \omega(x) = -x \sigma'(x) = \sum_{i \in E} x^i \prod_{j \in E, j \neq i} (1-\alpha^j x) \)
( the error-co-locator polynomial).

Formally, we work in \( k[X]/(x^{2t+1}) \), \( k(x) \rightarrow \frac{k[X]}{(x^{2t+1})} \), \( k(x) = \text{Frac} k[X] \)
\[
\frac{\omega(x)}{\sigma(x)} = \sum_{i \in E} \frac{x^i}{1-\alpha^i x} = \sum_{i \in E} \sum_{j=1}^{\infty} (\alpha^i x)^j = \sum_{i \in E} \left( \sum_{j=1}^{\infty} (\alpha^i)^j \right) x^j = \sum_{j=1}^{\infty} e(\alpha^i) x^j \Rightarrow \sigma(x) \cdot \sum_{j=1}^{\infty} e(\alpha^i) x^j = \omega(x)
\]

By definition of \( C \), \( c(\alpha^i) = 0 \) \( \forall i \leq t-1 \), and as \( r = c + e \)
\( r(\alpha^i) = e(\alpha^i) \) \( \forall i \leq 2t \)

We have shown that \( \sigma(x) \sum_{j=0}^{2t} r(\alpha^j) x^j \equiv \omega(x) \pmod{x^{2t}} \).

i), ii) hold with \( \omega(x) = -x \sigma'(x) \) and so both \( \sigma(x) \)
and \( \omega(x) \) have degree \( |E| = \bar{e}(t) \).
Uniqueness:
Suppose $\sigma(x), \tilde{\sigma}(x) \in k[x]$ also satisfy i), ii), and $\sigma(\tilde{\sigma}(x)) = \tilde{\sigma}(\sigma(x))$. If $i e E$, $\omega(\alpha^{-i}) = \prod_{i \in E, i \neq j} (1 - \alpha^{j-i}) \neq 0$ so $\sigma(x), \omega(x)$ are coprime. By ii),
$$\tilde{\sigma}(x) \omega(x) = \sigma(x) \tilde{\omega}(x) = \sigma(x) \tilde{\sigma}(x) \sum_{i=1}^{2^e} r(\alpha^i)x^i \mod x^{2^e+1}$$
But $\sigma, \tilde{\sigma}, \omega, \tilde{\omega}$ all have deg $\leq t$, hence
$$\tilde{\sigma}(x) \omega(x) = \sigma(x) \tilde{\omega}(x)$$
Finally, $\sigma(x) \mid \tilde{\sigma}(x)$ since $\sigma(x), \omega(x)$ coprime.
But $e(\tilde{\sigma}) \leq e(\sigma) \implies \tilde{\sigma}(x) = k\sigma(x), k \in K$
By i), $\sigma = \tilde{\sigma}$.

**13 One-Time Pads and Shift Registers**
We are headed towards cryptography starting with Vernam and M-bornsne's.

**Definition 13.1 (one-time pad) 1976**
Let $A$ be an alphabet, and take a "random" sequence $k_1, k_2, \ldots, k_n$ of letters from $A$ "the key" $k_j$ is chosen uniformly from the elements of $A$ independently for each $j$.

We encipher $m = a_1, \ldots, a_n$ as $b_1, \ldots, b_n$ where
$$b_i = a_i + k_i \mod |A|$$
One Time Pad

We have a key-stream \( k_1, k_2, \ldots, k_n \), with \( m = n \), encrypted as \( b_i = a_i + k_i \pmod{m \cdot A} \). Each random \( k_i \) is only used once.

These were used to encrypt White House/Kremlin hotline in the Cold War.

Flaws

1. Generating a truly random string of letters.
2. To decipher, the key must be known (so the key must also be sent in a secure way).

We will be replacing the random key stream, by a suitable pseudo-random sequence, such as:

Shift Register

Definition 13:2

A (general) feedback shift register is a function \( f: \mathbb{F}_2^d \to \mathbb{F}_2^d \) of the form

\[
f: (x_0, x_1, \ldots, x_{d-1}) = (x_1, x_2, \ldots, x_{d-1}, C(x_0, x_1, \ldots, x_{d-1}))
\]

for some map \( C: \mathbb{F}_2^d \to \mathbb{F}_2 \). This register has length \( d \).

If \( f \) is linear (LFSR) if \( C \) is a linear map.

\[
(x_0, \ldots, x_{d-1}) \mapsto \sum_{0 \leq i \leq d-1} a_i x_i
\]

(N.B. if \( a_0 \) was 0, then the value of \( x_0 \) does not appear any later values and we could consider the remaining \( d-1 \) registers as a simpler LFSR).

We will assume that \( a_0 = 1 \) in the definition.
The initial feed \((y_0, \ldots, y_{i-1})\) produces the output stream \((y_n)_{n=0}^{\infty}\) such that
\[y_{n+1} = C(y_n, y_{n+1}, \ldots, y_{n+a-1}) = \sum_{i=0}^{a-1} A_i y_{n+i}\]
i.e., we have a sequence determined by a linear recurrence relation with auxiliary polynomial
\[C(x) = x^d + a_1 x^{d-1} + \ldots + a_d x^0\]
Solving the recurrence relations over \(\mathbb{F}_2\) are similar to solving over \(\mathbb{R}\) or \(\mathbb{C}\), but more complicated over \(\mathbb{F}_2\) (since \(n^2 = n\) (2)).

See Sheet 4, question 20.

**Definition 13.3**

The feedback polynomial is \(\tilde{C}(x) = a_0 x^d + a_1 x^{d-1} + \ldots + a_d x^0\).

Now suppose that we have a sequence of elements \((y_n)_{n=0}^{\infty} \subset \mathbb{F}_2\).

**Definition 13.4**

The generating function \(G(x)\) of \((y_n)_{n=0}^{\infty}\) is \(G(x) = \sum_{i=0}^{\infty} y_i x^i\).

Compare the following with 12.5.

**Theorem 13.5**

The sequence \((y_n)_{n=0}^{\infty} \subset \mathbb{F}_2\) is the output from a LFSR with auxiliary polynomial \(C(x)\)

\[\iff G(x) = \frac{B(x)}{\tilde{C}(x)}, \quad B(x) \in \mathbb{F}_2[x], \quad \text{with } \deg(B) < \deg(C)\]

and \(\tilde{C}(x) = x^{d(C)} \tilde{C}(\frac{1}{x}) \in \mathbb{F}_2[x]\)

**Remark**

If we can recover \(C\) or \(\tilde{C}\) from \(G\), then we have recovered the LFSR from the stream.
Proof
\[ C(x) = \sum_{j=0}^{d} q_j x^j \quad \tilde{C}(x) = a_0 x^d + a_1 x^{d-1} + \ldots + a_{d-1} x + a_d \]
The condition is that \( (\sum_{j=0}^{d} y_j x^j) \tilde{C}(x) \) is a polynomial of degree \( < d \). This holds \( \iff \) \( \sum_{j=0}^{d-1} a_j y_{n+j} = y_n \quad \forall n \geq d \).
\( \iff \sum_{j=0}^{d-1} a_j y_{n+j} = y_{nd} \quad \forall n \geq 0 \)
\( \iff \eta_n \) is the output from a LFSR. \( \square \)

Problem
Suppose that \( \eta_n \) is the output of a LFSR. Can you find \( a_0, \ldots, a_{d-1} \in \mathbb{F}_2 \) such that \( y_{nd} = \sum_{j=0}^{d-1} a_j y_{n+j} \quad \forall n \geq 0 \) ?

13.6 Berlekamp–Massey Method

Observe that \( \sum_{j=0}^{d} a_j y_{n-j} = 0 \quad (a_0 = 1) \quad \forall n \geq d \).
\[ \begin{pmatrix} y_d & y_{d-1} & \cdots & y_1 & y_0 \\ y_{d-1} & y_{d-2} & \cdots & y_2 & y_1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ y_1 & y_0 & \cdots & y_{d-2} & y_{d-1} \end{pmatrix} = 0 \quad (\ast) \]

If we know the register has length \( \geq r \), start at \( i = r \).
Compute \( \det A_i \).

i) If \( \det A_i \neq 0 \), then \( d > i \). Replace \( i \) by \( i + 1 \) and repeat.

ii) If \( \det A_i = 0 \), solve \( (\ast) \) for the \( a_i \), by Gaussian Elimination, and test the solution over as many terms of the sequence as we like. If it fails this test, then \( d > i \) so replace \( i \) by \( i + 1 \) and repeat.

13.7 Secret Sharing (Tappa, Washington)

If Cambridge University is attacked by the Gove virus, all the doses retreat to the CMS. Entry to the CMS requires entering
Each of the \( n \) dons knows a pair of numbers (their "shadow"). We require that in C's absence, any \( k \) dons can reconstruct \( m \) from their shadows but no \( k-1 \) can. Any method to solve this is known as a \((k,n)\) threshold scheme.

Shamir's Threshold Scheme (1979)

Suppose that \( 0 < m < N \) (\( m \) random).

i) C chooses a prime \( p > mN \) then chooses \( k-1 \) integers \( a_1, \ldots, a_{k-1} \) at random and distinct integers \( x_1, \ldots, x_n \) at random such that \( 0 \leq q_i \leq p-1, 1 \leq x_i \leq p-1 \); then sets \( a_0 = m \).

ii) C computes \( P(r) = a_0 + a_1 x_i^0 + a_2 x_i^2 + \ldots + a_{k-1} x_i^{k-1} \pmod{p} \)

choosing \( 0 \leq P(r) < p-1 \) (\( 1 \leq r \leq n \)).

iii) C now gives the \( r \)th don their shadow pair \( (x_r, P(r)) \); secret from everybody else.

iv) \( p \) is known to all participants. C then burns his calculator.

Claim

Any \( k \) participants can recover \( m \), but not \( k-1 \).
Suppose that there are $k$-dons with shadows $(y_i, Q_i) = (x_i, P_i)$. Let $1 \leq i \leq k$, are together. Rewrite the system as
\[
\begin{pmatrix}
1 & y_{1z} & \cdots & y_{1k-1} \\
y_{2z} & 1 & \cdots & y_{2k-1} \\
\vdots & \vdots & \ddots & \vdots \\
y_{kz} & y_{k2} & \cdots & 1
\end{pmatrix}
\begin{pmatrix}
x_1 \\
x_2 \\
\vdots \\
x_{k-1}
\end{pmatrix}
= 
\begin{pmatrix}
Q_1 \\
Q_2 \\
\vdots \\
Q_{k-1}
\end{pmatrix}
\pmod{p}
\]
where $V$ is a Vandermonde matrix and from 12.3 we know that it has a unique solution $\pmod{p}$.

\[
\begin{pmatrix}
1 & y_1 & \cdots & y_{k-1}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_{k-1}
\end{pmatrix}
= \det V \neq 0 \pmod{p}
\]
Hence the system $Z_0 + y_1 Z_1 + \cdots + y_{k-1} Z_{k-1} = Q_1$, ..., $Z_0 + y_k Z_1 + \cdots + y_{k-1} Z_{k-1} = Q_k$ has a unique solution $Z$. But we know that $Z$ is a solution, so $Z = a$, and the secret $M = Z_0$.

On the other hand,
\[
\begin{pmatrix}
y_1 & y_2 & \cdots & y_{k-1}
\end{pmatrix}
\begin{pmatrix}
y_1 \\
y_2 \\
\vdots \\
y_{k-1}
\end{pmatrix}
= \prod_{i=1}^{k-1} (y_i - y_j) \neq 0 \pmod{p}
\]
So the system of equations $Z_0 + y_1 Z_1 + y_2 Z_2 + \cdots + y_{k-1} Z_{k-1} = Q_1$, ..., $Z_0 + y_k Z_1 + \cdots + y_{k-1} Z_{k-1} = Q_{k-1}$ has a solution, whatever the value of $Z_0$ is taken. So $k-1$ dons cannot decide if any value of $M$ is more likely than any other.

**Exercise**
Is the secret $M$ compromised if the $x_i$ become known to everybody?

5. **Cryptography**

14. **Introduction to Cryptosystems**

This is the art of codemaking, where a message is enciphered (or encrypted) so that it can only be read by the intended recipient.
14.1 Terminology

Message $M$ (plaintext), a string from some finite alphabet $A$.

We encrypt $M$ using a key from a finite set $X$ of possible keys.

We have

i) An encrypting function $e_k : A \rightarrow B$ ($k \in X$)

ii) Encrypted plaintext is called ciphertext

iii) Decrypting function $d_k : B \rightarrow A$ with $d_k(e_k(m)) = m$ ($k \in X$)

iv) $<M, X, E>$ is called a cryptosystem

Examples

i) Simple Substitution  "monalphabetic cipher"

key = a permutation $\pi$ of $A$. Each letter $m$ is replaced by $\pi(m)$

e.g. a 26-letter string, UXEB..., $A \rightarrow U$, $B \rightarrow X$, etc

'easy' to decipher/break the code: check the frequency of

letters in English and compare this to the frequencies in the

ciphertext. The content of the message also helps to find the

permutation.

ii) Vigenère Cipher (1523, "la chiffre indéchiffrable")

The key is a string of $D$ letters $k_1, \ldots, k_D$ repeated to give

an infinite sequence $k_1, k_2, \ldots, k_0, k_1, \ldots, k_D$

where $0 \leq a' \equiv a~(0) \Rightarrow k_{a'} = k_a$. The plaintext

$M = a_1 \ldots$ is encrypted as $b_1 \ldots b_n$ such that

$b_i = a_i + k_i \pmod{|A|}$
D = 1: Caesar cipher
   "polyalphabetic"
   e.g. D = 4, key: BEER

For a long key, a given letter is encrypted differently depending on
two places in M, so a simple frequency analysis will not work.
Babbage used a more clever frequency analysis to break it
(Singh, the Code Book).

Breaking a Cryptosystem

E knows possible functions e.g. die but not k. If E intercepts
some ciphertext C, she will try to decode it (or work out the key
so that she can decode all future ciphertexts). 3 forms of attack:

i) Ciphertext only (L1)

E has a single piece of ciphertext

ii) Known Plaintext (L2)

E has a piece of plaintext and matching ciphertext, and
seeks the key.

iii) Chosen plaintext (L3)

E can obtain arbitrary plaintexts of choice, and their corresponding
ciphertexts.

Examples

In the English alphabet, simple substitution and the Vigenère
Cipher are both vulnerable at L1: If E can find the key,
if the ciphertext is long enough.

All ciphers are vulnerable at L3: the system is finite i.e.
only a finite number of letters and keys and E can carry out an exhaustive search. To withstand L3 we want the effect of searching and/or the time it takes to be prohibitively large.
Equivocation

Suppose that we have a random plaintext $M$ chosen from $M$, the set of possible messages, with a certain probability distribution. Then choose some random $K \in X$ independently of $M$. The ciphertext is a random variable $C = e_K(M) \in C$

Definition 14.2

Modelled as above as transmitting $M$ through a channel to produce $C$, with noise coming from the choice of $K$.

$H(M|C)$ is the message equivocation.

$H(K|C)$ is the key equivocation.

Theorem 14.3

$H(K|C) \geq H(M|C)$

Proof

$H(K|C) = H(K,C) - H(C) = H(K,M,C) - H(C)$ (since $M \indep K|C$ is a function of $(K,C)$).

$H(K) = H(K,M,C) - H(M,C) + H(M,C) - H(C)$

$\geq H(K|M,C) + H(M|C) \geq H(M|C)$

Definition 14.4

Shannon defines perfect secrecy for a cryptosystem if the ciphertext gives no information about the plaintext, i.e. $H(M|C) = H(M)$. Thus $H(M,C) = H(M) + H(C)$, i.e. $M, C$ are independent.
Remark

Equivalently, \( I(M,C) = H(C) - H(C|M) = H(C) + H(M) - H(M,C) = C \) \( (7.1) \)

Proposition 14.5

Perfect secrecy \( \Rightarrow |K| \geq |UM| \)

Proof

Fix \( m_0 \in M \), and \( k_0 \in K \), both with strictly positive probability.
Then \( c_0 = e_{k_0}(m_0) \) also has strictly positive probability.

For any \( m \in M \), we have
\[
P(C = c_0 | M = m) = P(k = k_0) = 1 \quad \text{for all } m \in M
\]

So \( \exists k \in K \) with \( c_0 = e_k(m) \). If \( m_1, m_2 \) give the same key \( k \), then \( e_k(m_1) = c_0 = e_k(m_2) \).
\( m_1 = m_2 \)

Hence \( M \rightarrow k \) is injective.

Example 14.6

The One-Time Pad (13.1) has perfect secrecy. Suppose \( |A| = \frac{1}{2} \)
\[
P(M = m, C = c) = P(M = m, k = c - m) \\
= P(M = m)P(k = c - m) = P(M = m) \frac{1}{2n}
\]

\( \therefore M, C \) independent. \( (n \) is the length of string). We might replace the key by a pseudo-random sequence (e.g. using LFSR). But this is vulnerable at L2.

Theorem 14.7

The use of the output sequence from LFSR as a pseudo-random one-time pad is vulnerable at L2.
Proof
Message \( M = M_1, M_2, \ldots, M_i \in \mathbb{F}_2 \), \( x = x_1 x_2 \ldots \), stream from LFSR
Ciphertext \( C_i = M_i + x_i \mod(2) \), \( 1 \leq i < \infty \).

If \( M, C \) are known, the key sequence \( x \) is known \((M+C) \mod 2 \).

If the register has feedback coefficients \( a_0, \ldots, a_d \), as in LFSR 3,
then once \( E \) knows any 2d consecutive \( x \), the output stream, the
\( a \) can be found by solving \( d \) linear equations (BM, 13.6) \( \square \)

Unicity Distance
In attacking the simple substitution cipher, \( E \) can:
- examine letter frequencies
- know that the original message makes sense
- (for reasonable length) \( \exists \) ! \( k \), a key giving a sensible message
  from known ciphertext.

How long a message does \( E \) need for this argument to apply?
Take \( M = C = \Sigma \) say. Send \( n \) messages
\( M^{(n)} = (M_1, \ldots, M_n) \) encrypted as \( C^{(n)} = (C_1, \ldots, C_n) \) using
the same key.

Definition 14.8

The unicity distance is the least \( n \) such that \( H(k \mid C^{(n)}) = 0 \),
( the smallest number of encrypted messages required to
determine the key uniquely).

Now \( H(k \mid C^{(n)}) = H(k, C^{(n)}) - H(C^{(n)}) = H(k, M^{(n)}) - H(C^{(n)}) \)
\( = H(k) + H(M^{(n)}) - H(C^{(n)}) \)
Assume:
i) That all keys are equally likely, so $H(K) = \log_2 |X|$
ii) $H(M^n) = nH$, for $H$ constant, sufficiently large $n$.
iii) All sequences of ciphertext are equally likely, so $H(C^n) = n \log_2 |\Sigma|$

Then $H(K|C^n) = \log_2 |X| + nH - n \log_2 |\Sigma| \geq 0$

$\iff n \leq \frac{\log_2 |X|}{\log_2 |\Sigma| - H}$

Now $0 \leq H \leq \log_2 |\Sigma|$. To make $U$ large, make $|X|$ large, or use a message source with little redundancy.

Example

The English entropy per letter is roughly 1-2 bits. Take a substitution cipher with key in $S_{27}$.

$|X| = 27$, $\log_2 |X| \approx 4.75$, $|\Sigma| = 27$.

$U = \frac{\log(27)}{\log(27) - 1.2} \approx 26.2$; we expect to determine the key uniquely if the ciphertext is $> 26$ letters.

15 Stream Ciphers

Recall that a stream is a sequence in $F_2$. For a plaintext stream $p_0, p_1, p_2, \ldots$ and a key stream $k_0, k_1, \ldots$, we set ciphertext stream to be $z_0, z_1, \ldots$, where $z_n = p_n + k_n \pmod{2}$.

Remark

This is an example of a private key / symmetric system (both $A$ and $B$ need the key $k$ in order to encrypt or decrypt).

Example

Without $k$, deciphering is impossible ($U = \infty$) for the one-time pad: key stream is a random sequence $k_i = k_0$ where $k_i$ are iid with $P(k_i = 0) = P(k_i = 1) = \frac{1}{2}$

Ciphertext $z_i = p_i + k_i$. If $z_i$ are iid with $P(z_i = 0) = P(z_i = 1) = \frac{1}{2}$, $\frac{1}{2}$
Lemma 15.1
Let \((x_n)_{n=0}^\infty\) be a stream produced by a (general) feedback shift register of length \(d\). Then \(\exists N, M \leq 2^d\) such that \(x_{N+r} = x_r\), \(\forall r > M\).

Proof
Let \(f: \mathbb{F}_2^d \rightarrow \mathbb{F}_2^d\) be a register. Let \(V_j = (x_j, x_{j+1}, \ldots, x_{j+d-1})\) so \(V_{j+1} = f(V_j)\). Since \(|\mathbb{F}_2^d| = 2^d\), \(V_0, V_1, \ldots, V_{2^d}\) are not all distinct. So \(\exists 0 \leq a < b \leq 2^d\) such that \(V_a = V_b\). Let \(M = a\) and \(N = b - a\). So \(V_M = V_{M+N}\) and \(V_r = V_{r+N}\) \(\forall r > N\) (by induction, applying \(f\)), so \(x_r = x_{r+N}\) \(\forall r > M\) \(\square\)

Remarks
1. The maximum period of a feedback shift register of length \(d\) is \(2^d\).
2. The maximum period of a LFSR of length \(d\) is \(2^d - 1\) \(\text{(Fact 1, Q18)}\)
3. Stream ciphers using LFSR of length \(d\) are vulnerable at \(L_2\) due to Berlekamp-Massey.

However, they have advantages:
- Cheap (computationally), fast, easy to use
- Messages can be encrypted/decrypted "on the fly"
- Error tolerant

Lemma 15.2
If \((x_n), (y_n)\) are outputs from LFSR of lengths \(M, N\) respectively. Then

a) \((x_n + y_n)\) is the output from a LFSR of length \(M + N\).
b) \((x_n y_n)\) is the output from a LFSR of length MN

Proof (Sketch, complete using Galois Theory)

Assume that the auxiliary polynomials \(P, Q\) each have distinct roots \(\alpha_1, \ldots, \alpha_M\) and \(\beta_1, \ldots, \beta_N\) in some extension \(K/\mathbb{F}_2\).

Then \(x_n = \sum_{i=1}^{M} \lambda_i \alpha_i^n\), \(y_n = \sum_{i=1}^{N} \mu_i \beta_i^n\) where \(\lambda_i, \mu_i \in K\).

\(x_n + y_n = \sum_{i=1}^{M} \lambda_i \alpha_i^n + \sum_{i=1}^{N} \mu_i \beta_i^n\) is produced by a LFSR with auxiliary polynomials \(P(x) Q(x)\).

\(x_n y_n = \prod_{i=1}^{M} \prod_{j=1}^{N} (x - \alpha_i \beta_j)\), produced by a LFSR with auxiliary polynomial \(\prod_{i=1}^{M} \prod_{j=1}^{N} (x - \alpha_i \beta_j)\) lying in \(\mathbb{F}_2[x]\) (which must be proved)

\(\Box\)

i) Adding outputs of 2 LFSRs is no more economical than producing the same stream with a single LFSR.

ii) Multiplying streams looks more promising until we realise that \(x_n y_n = 0\) about 75% of the time (see Könen).

iii) Non-linear feedback shift registers are hard to analyse, and E may understand them better than we do.

Lemma 15.3

Suppose that \((x_n)\) is a periodic stream with period N, and \((y_n)\) similarly with period M. Then \((x_n + y_n), (x_n y_n)\) are periodic with periods dividing \(\operatorname{LCM}(N, M)\).

Proof

Easy exercise.
10 Public Key Cryptography

An example of an asymmetric cryptosystem (i.e. we have 2 keys, one for encrypting and one for decrypting). We have the private key (for decrypting) and the public key (for encrypting).

Anyone can encrypt the plaintext message and send it to B, but only B knows the private key to allow decryption.

Easy: encrypt \( m \in M \Rightarrow c = e_k(m) \) using known \( e_k \).

Hard: given \( c \in E \), find \( m \in M \) such that \( c = e_k(m) \).

\( e_k \) is a "one way function": it is easy to find \( e_k(m) \), difficult to find \( e_k^{-1}(c) \). We aim to be secure at L3.

This relies on two problems which are hard (to solve quickly).

i) Factoring: Let \( N = pq \), \( p \), \( q \) large primes. Given \( N \), find \( p, q \).

ii) Discrete Logarithm: Let \( p \) be a large prime, \( g \) a primitive root \( \mod p \) (i.e. \( \mathbb{F}_p^* = \langle g \rangle \)). Given \( x \in \mathbb{F}_p^* \), find a \( a \) such that \( x = g^a \mod p \).

Definition 16.1

An algorithm runs in polynomial time if

\# operations \( \leq c \) (input size) \( ^d \) for constants \( c, d \).

Examples:

i) Integer arithmetic \((+, -, \times, \div, \text{ division algorithm})\)

ii) Computation of gcd by the Euclidean Algorithm

iii) Primality Testing (Agrawal et al., 2002)

iv) Modular Exponentiation, computing \( x^a \mod N \) using repeated squaring.
Polynomial time algorithms are not known for factoring and discrete logarithms

- Trial division (properly organized) takes time $O((n)$

- Baby Step Giant Step algorithm (set $m = \lceil \log p \rceil$, $a = q_m r$, $0 \leq q, r < m$. Then $x^p \equiv g^{q_m r} (p) \Rightarrow g^{r x} \equiv g^{q_m x} (p)$ List $g^{r x} (p)$ for $q = 0, \ldots, m-1$ and $g^{q_m x} (p)$ for $r = 0, \ldots, m-1$. Sort the two lists and look for a match.

We can find discrete logs in time and storage $O(\log p \log \log p)$

- Factor base (number field sieve), $O(e^{(\log n)^{1/3}(\log \log n)^{1/3}})$ known

RSA Labs offered prizes for factoring large numbers.

<table>
<thead>
<tr>
<th>Challenge No.</th>
<th>#Decimal Digits</th>
<th>Factored</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSA-576</td>
<td>174</td>
<td>Dec 2003</td>
<td>$10k</td>
</tr>
<tr>
<td>RSA-640</td>
<td>193</td>
<td>Nov 2003</td>
<td>$20k</td>
</tr>
<tr>
<td>RSA-704</td>
<td>212</td>
<td>?</td>
<td>$30k</td>
</tr>
</tbody>
</table>

Recall $\phi(N) = \left\lfloor \frac{1}{2} \right\rfloor | \{1 \leq a \leq N \mid (a, N) = 1\} | = \left(\frac{N}{2}\right)^*$

Euler-Fermat

$\phi(N) = 1 \ (N)$ for each $\varepsilon$ with $(a, N) = 1$

Fermat's Little Theorem

$a^p \equiv a \ (p)$ for all $a$ with $(a, p) = 1$

**Lemma 16.2**

Let $p = 4k - 1$ prime, $d \in \mathbb{Z}$. If $x^2 \equiv d \ (p)$ is soluble (i.e. $(\frac{d}{p}) = 1$) then solutions are $x \equiv \pm d^{k-1} (p)$.

**Proof**

Let $x_0$ be a solution. WLOG $x_0 \neq 0 (p)$.

$d^{2k-1} = x_0^{2(2k-1)} \equiv x_0^{p-1} \equiv 1 \ (p) \Rightarrow (d^{k})^2 \equiv d \ (p)$
i) Choose two large distinct primes $p, q \equiv 3 \pmod{4}$

ii) Private key $(p, q)$, Public key $N = pq$

iii) $M = C = \{0, 1, \ldots, N-1\}$

iv) $m \in M$ is encrypted as $c \equiv m^2 \pmod{N}$

v) We restrict the alphabet so that $(m, N) = 1$, $m > \sqrt{N}$

If $B$ receives $c \equiv m^2 \pmod{N}$, we use 16.2 to solve

$m = S c^k (p), \quad M = E c^k (q), \quad S, E = \pm 1.$

Find $u, v$ with $up + vq = 1$. CRT:

$m = Sc^k + up (Ec^k - Sc^k) \pmod{N}$

[ CRT: $x \equiv c(p), x \equiv d(q), (kp + qq = 1) \iff x \equiv c + ap(d - c) \pmod{pq}$]

We claim that all 4 possible values can occur and are distinct.

To decipher $c$, we find all 4 square roots mod $N$ and choose the one making sense. Messages should contain enough redundancy for only one of the four choices to make sense.

Lemma 16.4

Let $N = pq$, a product of distinct primes. $(\frac{m}{N}) = 1$. Then

i) Either $(m, N) = 1$ and $\exists$ exactly 4 square roots of $m \pmod{N}$,

ii) Or $(m, N) = p$, or $q$, and $\exists$ exactly 2 square roots

iii) $m \equiv 0 \pmod{N}$, so 0 is the unique square root of 0 $\pmod{N}$.

Proof:

$x^2 \equiv m \pmod{N} \iff x^2 \equiv m(p), x^2 \equiv m(q) \pmod{N}

(m, N) = 1 \Rightarrow 2$ solutions to each of $x^2 \equiv m(p), x^2 \equiv m(q)$.
\[ \text{CRT} \Rightarrow 4 \text{ solutions to } x^2 \equiv m \ (N) \ , \ \text{giving i).} \]
\[ p \mid m \Rightarrow \text{unique solution to } x^2 \equiv m \ (p) \ , \ \text{giving ii), iii)} \]

**Theorem 16.5**

An algorithm to decipher Robin gives an algorithm to factorize \( N \).

**Proof**

An algorithm to decipher Robin must give one of the 4 square roots of a ciphertext \( C \mod N \) which we obtain as \( C = x^2 \mod (N) \).

Choose \( m \in \left( \mathbb{Z}/n\mathbb{Z} \right)^* \) at random. The algorithm gives a particular square root of \( C = x^2 \mod (N) \). So \( x^2 \equiv m^2 \mod (N) \). If 4 distinct choices of \( m \) giving the same value of \( C \). Two give \( x \equiv \pm m \) and the others do not.

For the latter two \( x^2 - m^2 \equiv (x - m)(x + m) \equiv 0 \mod (N) \) but neither \( x + m, x - m \) are divisible by \( N \). Thus \( (N, x - m) \neq 1 \). With probability \( \frac{1}{2} \) we find a non-trivial factor \( (N, x - m) \) of \( N \).

Repeating \( r \) times means that the probability of finding a factor of \( N \) is \( \geq 1 - \left( \frac{1}{2} \right)^r \).


i) Choose two large distinct primes \( p, q \). Set \( N = pq \).

ii) Pick \( e \) randomly with gcd \( (e, \varphi(N)) = 1 \) (e.g. \( e \) prime, \( > pq \) )

"encrypting exponent" \( e \)

iii) Find \( d \in \mathbb{Z} \) such that \( de - k \varphi(N) = 1 \)

iv) Public key \( (N, e) \), encrypt with \( m \mapsto m^e \mod (N) \)
v) Private key $d$, decrypt with $c \rightarrow c^d \pmod{N}$

Euler-Fermat: $(m^e)^d = m^{de} \equiv m^{k_p(N) + 1} \equiv m \pmod{N}$

Theorem 16.6

Finding the RSA private key from public key $(N, e)$ is essentially as difficult as factoring $N$.

Proof

Assume that there is an algorithm giving $d$ in terms of $N, e$.
Then $m^{de} \equiv m \pmod{N}$, $\forall m \in \mathbb{Z}/N\mathbb{Z}$. Write $de - 1 = 2^s r, r \neq 0$.
Let $|x^r|_p =$ (order of $x$ in $(\mathbb{Z}/p\mathbb{Z})^*$). Let $X = \{x \in (\mathbb{Z}/N\mathbb{Z})^* \mid |x^r|_p \neq |x^{-r}|_p\}$

Lemma 16.7

If $x \in X$, we can factorise $N$.

Proof

Let $x \in X$. Let $y = x^r$. This satisfies $y^{2^s} = x^{2^s r} = x^{de - 1} = 1$.

$\therefore |y|_p, |y|_q$ are powers of 2. Suppose $|y|_p = 2^t < |y|_q$.
Then $y^{2^t} \equiv 1 \pmod{p}$ but $y^{2^t} \equiv 1 \pmod{q}$ so $(y^{2^t} - 1, N) = p$ and we could use Euclid to find one factor of $N$. To factorise $N$ when we have $x \in X$, compute $(y^{2^t} - 1, N)$ for $0 \leq t \leq s$.

It must be $p$ for one of these choices of $t$, so we factor $N$.

Lemma 16.8

$|\{x \in (\mathbb{Z}/p\mathbb{Z})^* : |x|_p = c\}| \leq \frac{1}{2}(p-1)$ for every possible order $c$. 


Proof
Let \( x \) be a primitive root \( \mod p \). \( \left( \mathbb{Z}_p / \mathbb{Z} \right)^* = \langle x \rangle \), \( x^{2^s} \equiv 1 \mod p \)
so certainly \( 1x^r \pmod{p} \mid 2^s \). Let \( 1x^r \pmod{p} = 2^t \), \( 0 \leq t \leq s \).
If \( x = x^k \) then \( x^r = (x^r)^k \implies 1x^r \pmod{p} = \frac{2^b}{(2^s, k)} \).
Hence \( 1x^r \pmod{p} = 2^b \implies k \) is odd. There are \( \frac{1}{2}(p-1) \) such values for \( x \in \left( \mathbb{Z}_p / \mathbb{Z} \right)^* \).
The remaining \( \frac{1}{2}(p-1) \) values for \( x \) have \( 1x^r \pmod{p} < 2^t \).
Hence there are no more than \( \frac{1}{2}(p-1) \) elements from \( \left( \mathbb{Z}_p / \mathbb{Z} \right)^* \) with \( 1x^r \pmod{p} < 2^t \).

Lemma 16.9
\( 1x1 \geq \frac{1}{2}(p-1)(q-1) \) (\( = \frac{1}{2} \varphi(N) = \frac{1}{2} \left| \left( \mathbb{Z}_N / \mathbb{Z} \right)^* \right| \))

Proof
CRT gives \( \left( \mathbb{Z}_N / \mathbb{Z} \right)^* \to \left( \mathbb{Z}_p / \mathbb{Z} \right)^* \oplus \left( \mathbb{Z}_q / \mathbb{Z} \right)^* \), a multiplicative group isomorphism
\( x \mapsto (x \pmod{p}, x \pmod{q}) \). So \( 1x1 = \left\{ (x, y) \in \left( \mathbb{Z}_p / \mathbb{Z} \right)^* \oplus \left( \mathbb{Z}_q / \mathbb{Z} \right)^*: 1x^r \pmod{p} = 1y^r \pmod{q} \} \).
For each \( y \in \left( \mathbb{Z}_q / \mathbb{Z} \right)^* \) we know \( \left| \{ x \in \left( \mathbb{Z}_p / \mathbb{Z} \right)^*: 1x^r \pmod{p} \neq 1y^r \pmod{q} \} \right| > \frac{1}{2}(p-1) \).
Hence the result. \( \square \)

To complete the proof of Theorem 16.6, choose \( x \in \left( \mathbb{Z}_N / \mathbb{Z} \right)^* \) at random. \( p(x \in X) \geq \frac{1}{2} \). If \( x \in X \) we use 16.7 to find the factors of \( N \). If \( x \notin X \), choose another random value for \( x \).
After \( k \) such random choices, we will find a factor of \( N \) with probability \( \geq 1 - \left( \frac{1}{2} \right)^k \). \( \square \)

Remark
It is unknown as to whether decrypting RSA messages
without knowledge of \( d \) is essentially as hard as factoring \( N \).
**17. Discrete Logarithm Ciphers**

i) Take $p$, a large prime, and $g$, a primitive root mod $p$. These are both fixed and public.

### 17.1 Diffie–Hellman Key Exchange (1976)

ii) $A, B$ wish to agree a secret key $k \in \mathbb{Z}_p^\times$ to use. $A$ chooses random $a \in (\mathbb{Z}_{p-1})^\times$, computes $g^a = \alpha$, then publishes $\alpha$.

iii) $B$ chooses random $b \in (\mathbb{Z}_{p-1})^\times$, computes $g^b = \beta$, then publishes $\beta$.

iv) $A$ knows $\alpha$ and $\beta$ and computes $\beta^a (p)$.

$v$) $B$ knows $b$ and $\alpha$ and computes $\alpha^b (p)$.

$\beta^a = (g^b)^a = (g^a)^b = \alpha^b (p)$ so $A, B$ use the common value $k$ of $\beta^a, \alpha^b$, for their key.

vi) $E$ wants to compute $k = g^{ab}$ from $p, g, \alpha, \beta$, published data.

**Elgamoal Cipher**

i) $A$ wishes to send encrypted messages to $B$.

ii) $B$ chooses random private key $b \in (\mathbb{Z}_{p-1})^\times$, and computes $\beta = g^b (p)$, publishing $(p, g, \beta)$ as public key.

iii) $A$ wants to send $0 < m < p$, so chooses random $a \in (\mathbb{Z}_{p-1})^\times$ and sends the pair $(c_0, c_1) = (g^a, \beta^a m) \in \mathbb{Z}_p^\times \oplus \frac{\mathbb{Z}}{p\mathbb{Z}}$

iv) $B$ deciphers by computing $c_0^{-b} (p)$:

$$c_0^{-b} = (g^a)^{-b} = g^{ab} = \beta^a (p), \text{ so } G_0^{-b} = m (p)$$
v) If $E$ knows one plaintext $m_i$ and the corresponding ciphertext $(c_0, c_i)$ then the public keys $\beta = g^b(p)$, $\alpha = g^a(p)$ used in Diffie–Hellman. Breaking the Elgamal cipher is tantamount to breaking the Diffie–Hellman Key Exchange, which we hope is as hard as computing discrete logarithms.

18 Trapdoors and Signatures

Guarantees

A sends a message to B. Both are concerned about the following issues:

i) **Secrecy**: no third-party can read their message

ii) **Integrity**: no third-party has altered their message

18.1 Example

A is a customer at a bank owned by B. A sends an instruction to pay £100 to C. B must be sure that the amount and the recipient have not been altered. Suppose that the bank creates messages of the form $(a, m)$, where $a$ specifies the recipient, $m$ the amount to pay them.

Suppose that the messages are encrypted using RSA:

$$(c_0, c_i) = (a^e, m^e) \pmod{N}$$

$E$ then enters into a transaction with the bank, which adds £100 to her account. She observes the resulting encrypted message $(c_0', c_i')$, then transmits $(c_0', c_i^3)$.

We say that RSA is vulnerable to a **homomorphism attack** (i.e. uses knowledge of the form of the cipher) (cf. copy attack, same message represented twice, i.e. using timestamp).
iii) **Authenticity**: B wants to be sure that A sent the message (we would like some kind of digital signature)

iv) **Non-Repudiation**: B can prove to C that A sent the message.

**Trapdoor**

18.2 **Terminology**

Suppose \( h: \mathbb{A}^* \to \mathbb{B} \). There are infinitely many messages, but \( B \) is finite, so \( B \) many messages with the same \( h \) values. It should be hard to find 2 messages with the same \( h \) value: a collision. If it is easy to compute \( h(m) \) (the digest) for any message \( m \), but hard to find a collision then \( h \) is called a signature, trapdoor, or hash function.

**Example**

i) \( \text{md5} \) \( mm \), a 128-bit binary sequence computed for every file

ii) **RSA**, public key \((N, e)\), private key \( d \).

For any letter \( a \), compute \( a^e \mod N \), then apply a permutation to the binary bits of \( a^e \) to give \( \sigma(a^e) \). For \( m = a \ldots a \), compute \( \sigma_k(a^e) \) for a sequence of different permutations \( \sigma_k \), then add the result \( \mod N \). Changes to letters have an unpredictable effect on the digest, so it seems hard to find collisions.

18.3 **RSA Signature**

A wants to sign a message \( m \) to show that she sent it. She uses a public key cipher. Suppose \( e_A \), \( d_A \), are the public and private encryption and decryption functions, so that \( e_A d_A = d_{AES} = 1 \).
A takes her name $n$, computes $s = d_A(n)$, and appends $s$ to $m$.
B can calculate $e_A(s) = e_A(d_A(n)) = n$ to get A’s name.
E does not have access to A’s (private) $d_A$, so will find it hard to
sign $m$ in this way. But E could separate $s$ from $m$ and attach it
to other messages (existential forgery). How do we guarantee
integrity?

i) A wants to send $m$, and adds signature $s = d_A(h(m))$ where $h$ is
a (public) trapdoor, i.e. sends $(m, s)$.

ii) B receives $(m, s)$ and uses the public key to compute $e_A(s)$, then
checks $h(m) = e_A(s)$.
Only A has the decrypting function $d_A$, so
only she could have signed $m$ with signature such that
$e_A(s) = h(m)$.

Any alteration to $m$ would require a corresponding change
to $d_A(h(m))$. 
18.4 ElGamal Signature Scheme

i) Take a large prime $p$, and let $g$ be a primitive root mod $p$.

ii) $A$ chooses $a \in \mathbb{Z}_p^*$ and publishes $\alpha = g^a \in \mathbb{Z}_p^*$

iii) Let $h$ be a hash function. $A$ sends $1 \leq m \leq p-1$ by choosing random $k$ coprime to $p-1$ and puts $r = g^k(p)$, $s = \frac{h(m) - ar}{k}(p-1)$

iv) $A$'s signature is $(r, s)$. $(m, r, s)$ is sent to $B$.

v) $B$ receives $(m, r, s)$ and checks that $g^{h(m)} = \alpha^r s^s (p)$

(because $g^{h(m)} = g^{ar + sk} = (g^a)^r (g^k)^s = \alpha^r s^s (p)$)

It is believed that the only way to forge this signature is to solve the discrete logarithm problem.

Lemma 18.5

In 18.4, one must choose different random $k$ for each message.

Proof

Suppose that $m_1, m_2$ are signed using the same $k$ giving $(m_1, r, s_1)$ and $(m_2, r, s_2)$. Then

$h(m_1) - h(m_2) \equiv k(s_1 - s_2) (p-1)$. 

(Recall that if $xs \equiv h(m)$ has either no solutions for $x$ or else has $(m, s)$ solutions mod $m$.)

Hence there are $(p-1, s_1 - s_2)$ solutions for $k \mod p-1$.

Choose the one that gives the correct value for $r = g^k(p)$.

Then $s_1 \equiv \frac{h(m_1) - ar}{k}(p-1) \Rightarrow ar = h(m_1) - ks_1 (p-1)$

This gives $(p-1, r)$ solutions for $a$. Choose the one giving $\alpha = g^a (p)$. We have found $A$'s private key $a$ and the
exponent \( k \) that is used for signatures.

19. Bit-Commitment

A wants to send a message to B such that

i) B cannot read the message until Alice sends further information.

ii) A cannot change the message.

For example, B tosses a coin, and A calls, both in different rooms, so that A cannot see B tossing the coin. How do we ensure that A does not change the call on discovering the result, and that B cannot change the reported result, when A's guess is announced?

Commit and Reveal Process

A writes down a guess, and places it in an envelope, and gives it to B. B tosses a coin and reports the result. Together, they open the envelope to see if A's guess was correct.

19.1 Bit Commitment using a Public Key Cipher

Let \( e_A, d_A \) be encrypting and decrypting functions for the public key cipher used by A.

i) \( e_A \) is published, \( d_A \) is secret, known only to A.

ii) A chooses \( m \in \mathbb{F}_2 \) and commits to B the encrypted message \( c = e_A(m) \) which B cannot decipher.

iii) To reveal the choice, A sends to B the private key so that \( d_A \) can be found and \( d_A(c) = m \) computed, revealing A's choice. B can check that \( d_A, e_A \) are inverse functions, hence can be confident that A has not sent the wrong private key.
19.2 Using Noisy Coding

A \xrightarrow{\delta} B \quad C_1, \text{ Channel 1, a BSC, error probability } \delta < 1/2
C_2, \text{ Channel 2, a clear (error-free) channel.}

i) B chooses some linear (binary) code } C \subseteq \mathbb{F}_2^n, \text{ min. distance } \delta.

ii) A chooses some random, non-trivial linear map } \Phi : C \rightarrow \mathbb{F}_2^n
\text{ These are both made public.}

iii) To send } m \in \mathbb{F}_2^n, A \text{ chooses a (random) } c \in C \text{ such that } \Phi(c) = m,
\text{ and sends } c \text{ to } B \text{ using } C_1.

iv) B receives } r = c + \varepsilon \text{ (certain component of } c \text{ may be altered).}
\text{ The expected value for } d(r, c) = \text{wt}(\varepsilon) \approx np
\text{ (provided that the variance of the BSC can be made sufficiently small).}

v) Later on, A sends } C \text{ using } C_2, \text{ and B checks that } d(r, c) \approx np
\text{ If so, then } \Phi(c) \text{ is accepted as A's choice.}

1. Why can B not read the original message?
We can arrange that } \delta << \eta. \text{ This means that B cannot tell what the original codeword } C \text{ was, hence can't find } \Phi(c) = m
\text{ and determine A's choice.}

2. Why can't A change choice?
A knows that } C \text{ was sent, but does not know } r. \text{ If later on, }
A \text{ sends } c' \neq c, \text{ it will only be accepted if } d(c', r) \approx np
\text{ A's only safe option is to choose } c' \text{ very close to } C, \text{ but if }
\delta \text{ is sufficiently large, } \delta >> np \text{, this forces } c = c'.

20 Quantum Cryptography  (Cane, Chapter 22)

Problems with public-key cryptosystems:

i) Based on the belief that some maths problems are "hard", we could be wrong.

ii) Computers get faster. Yesterday's securely encrypted message is easily read tomorrow.

We aim to construct a key exchange that is secure conditional on the laws of physics. Consider a simple quantum system which describes the state of a single polarized photon. The photon is described by a wave function $\psi$.

Definition 20.1

A quantum bit or qubit is a linear combination

$$\Psi = \alpha |0\rangle + \beta |1\rangle$$

vertical polarization  horizontal polarization

$\alpha, \beta \in \mathbb{C}$

$|\alpha|^2 + |\beta|^2 = 1$
Coding and Cryptography

Two states $|0\rangle$, $|1\rangle$ are a basis for state space. Denote the basis by $\{\pm\}$. Measuring $\Psi$ gives $|0\rangle$ with probability $|\alpha|^2$ and $|1\rangle$ with probability $|\beta|^2$. After measurement the qubit collapses to the state observed, i.e. either $|0\rangle$ or $|1\rangle$.

There are many other possible bases, for example:

\[ + = \frac{1}{2} |0\rangle + \frac{1}{2} |1\rangle \]
\[ - = \frac{1}{2} |0\rangle - \frac{1}{2} |1\rangle \]

\{ denote by $\times$.

Basic idea: A generates a sequence of qubits and sends them to B. By comparing notes afterwards, they can detect E's presence.

![Light interference diagram]

Each photon passing through the first filter then passes through the second with probability $\cos^2 \theta$. Endow $|\psi\rangle = (|0\rangle + |1\rangle, \alpha, \beta, \overline{\alpha}, \overline{\beta})$ with an inner-product:

\[ \langle \alpha, \beta \rangle \cdot \langle \alpha_2, \beta_2 \rangle = \alpha \overline{\alpha}_2 + \beta \overline{\beta}_2 \]

If $\Psi = \frac{1}{2} |+\rangle + \frac{1}{2} |-\rangle$ then observation gives $|+\rangle$ with probability $\frac{1}{4}$ and $|-\rangle$ with probability $\frac{1}{8}$.

20.2 BB84 Protocol for Quantum Key Exchange (Bennet, Brassard, 1984)

A and B want to agree a secret key of $n$ binary bits.

Step 1

A sends B a stream of $4n$ qubits, with randomly chosen polarisations $|0\rangle, |1\rangle, |+\rangle, |-\rangle$ with probability $\frac{1}{4}$. 


Step 2
B measures the qubits using either the + basis or the x-basis, deciding which at random.
 Afterwards, A and B announce which bases they have used and B announces which bits were measured with the right bases (we expect roughly 2n of these).
A, B share 2n bits. They compare n of these bits, and if they agree, they use the other n bits as their key.
If E can predict which basis A is using too send, or B is using to measure, she will remain undetected. Otherwise, E will change about 25% of the 2n bits shared.
(If E intercepts a photon A → B, and knows that it is either vertically or horizontally polarized, she can use a vertical filter. If the photon passes through, then it was vertically polarized when sent by A, and E can pass on a vertically polarized photon to B. If not, then it was polarized the other way, and a horizontally polarized photon is sent to B. If A’s photon was actually diagonally polarized, then the procedure results in E sending B a photon “wrongly polarized.

Problems and Remarks
i) There is high noise in the system (20% bits can be corrupted even when E is not present). One can use error-correcting codes to deal with this. So it is possible to exchange keys with
arbitrarily small probability of error and arbitrarily small probability that an enemy has intercepted it undetected. It is secure because of the laws of quantum mechanics rather than because a maths problem is "hard".

i) There is no device (yet) for producing single photons (low power lasers are used so that each qubit consists of many photons each with the same polarization.

ii) "Man in the middle" attacks: If E can intercept all the data sent between A and B, E can impersonate each of them to the other, so that each of A, B will exchange a key with E.

iii) We have sketched a protocol for exchanging keys using the polarisation of photons. Other methods exist such as Ekert's Quantum Entanglement Protocol.