Information, Codes and Ciphers

By Jeremy Le for MATH3411 24T3

1 Introduction

1.1 Mathematical Model

To give a mathematical framework for digital data transmission, define

- a **source alphabet** $S = \{s_1, s_2, \dots, s_q\}$ of q symbols
- a **code alphabet** A of r symbols probabilities $p_i = P(s_i)$
- a **code** that encodes each symbol s_i by a codeword which is a **string** of code symbols.

1.2 Assumed Knowledge

- Modular Arithmetic and the Division Algorithm
- Probability (Binomial Distribution and Bayes' Rule)
- Linear Algebra (Linear combination, independence, etc...)

1.3 Morse Code

Morse code is a **ternary** code (radix 3). Its alphabet is

1. • called **dot**
2. — called **dash**
3. `p` a **pause**

The codewords are strings of • and — **terminated** by `p`.

1.4 ASCII

<u>**A**</u>merican National <u>**S**</u>tandard <u>**C**</u>ode for <u>**I**</u>nformation <u>**I**</u>nterchange.

Binary code of fixed codeword length, namely 7, with $2^7 = 128$ encoded symbols.

The extended ASCII is a code like the 7-bit ASCII but with an extra bit in the front used as a check bit, requiring the number of 1's to be even.

1.5 ISBN

<u>**I**</u>nternational <u>**S**</u>tandard <u>**B**</u>ook <u>**N**</u>umber.

They have 10 bits, with it's last bit being a check bit, requiring

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

2 Error Detection and Correction Codes

We say that **x corrupted** to **y** is denoted by $\mathbf{x} \rightsquigarrow \mathbf{y}$.

2.1 ISBN-10 Error Capability

ISBN-10 numbers are capable of detecting the two types of errors:

1. getting a digit wrong,
2. interchanging two (unequal) digits.

2.2 Types of Codes

- **variable length code**: codewords have different lengths
- **block code**: codewords have the same lengths
- ***t*-error correcting code**: code can always correct up to t errors
- **systematic code**: code with **information digits** and **check digits** distinct

2.3 Binary Repetition Codes

A **binary *r*-repetition code** encodes $0 \to \overbrace{0\cdots0}^{r}$ and $1 \to \overbrace{1\cdots1}^{r}$.

The binary $(2t+1)$-repetition code is t-error correcting.
The binary $2t$-repetition code is $(t-1)$-error correcting and t-error detecting.

2.4 Information Rate and Redundancy

The **information rate** R is given by,

- For a code C of radix r and length n, $R = \dfrac{\log_r |C|}{n}$
- For a **systematic code**, $R = \dfrac{\text{\# information digits}}{\text{length of code}}$

We then define **redundancy** $= \frac{1}{R}$.

2.5 Binary Hamming Error-Correcting Codes

A Binary Hamming (n, k) code is a code of length n with k information bits, such that it is a single error correcting and has a parity check matrix, H, of size $n-k$ by n.

2.6 Hamming Distance, Weights

The **weight** of an n-bit word $\mathbf{x}$ is defined to be

$$w(\mathbf{x}) = \#\{i : 1 \le i \le n, x_i \ne 0\}.$$

Given two n-bit words, the **Hamming distance** between them is

$$d(\mathbf{x}, \mathbf{y}) = \#\{i : 1 \le i \le n, x_i \ne y_i\}.$$

Given some code with set of codewords C, we define **(minimum) weight** of C to

$$w = w(C) = \min\{w(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \ne \mathbf{0}\}.$$

Similarly, the **(minimum) distance of** C is defined by

$$d = d(C) = \min\{d(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C, \mathbf{x} \ne \mathbf{y}\}.$$

If $\mathbf{x} \rightsquigarrow \mathbf{y}$, then $d(\mathbf{x}, \mathbf{y})$ is the number of errors in $\mathbf{y}$.

2.7 Decoding Strategies

Minimum Distance Decoding Strategy Given a received word y, decode to *closest* codeword x.

Standard Strategy If received word y is distance at most t from a codeword x, then decode y to x; otherwise flag an error.

Pure Error Detection If received word y is not a codeword x, then flag an error.

2.8 Sphere Packing

The **sphere** of radius r around $\mathbf{c}$:

$$S_r(\mathbf{c}) = \{\mathbf{x} \in \mathbb{Z}_2^n : d(\mathbf{x}, \mathbf{c}) \le r\}.$$

The volume of this sphere is its size $|S_r(\mathbf{c})|$.

Sphere-Packing Condition Theorem A t-error correcting binary code C of length n has minimum distance $d = 2t+1$ or $2t+2$, and

$$|C| \sum_{i=0}^{t} \binom{n}{i} \le 2^n.$$

If we have equality in the bound, then we say that the code is perfect. This means that codewords are evenly spread around in $\mathbb{Z}_2^n$ space.

More generally for radiux r:

$$|C| \le \frac{r^n}{\sum_{i=0}^{t} \binom{n}{i} (r-1)^i}.$$

2.9 Binary Linear Codes

A linear code C is a vector space over some field $\mathbb{F}$. Equivalently it is the null-space of

$$C = \{\mathbf{x} \in \mathbb{F}^n : H\mathbf{x}^T = \mathbf{0}\}$$

of an $m \times n$ parity check matrix H with $m = \text{rank}(H)$.

- $\dim C = k = n - m$ by the Rank-Nullity Theorem.
- If C is binary, then $|C| = 2^k$.
- C is systematic.
- If H is **reduced echelon form**, then we can choose the non-leading columns of H to be **information bits** and the leading columns of H to be **check bits**.

Minimum Distance for Linear Codes If C is a linear code with parity check matrix H, then

- $w(C) = d(C)$,
- $d(C) = \min\{r : H \text{ has } r \text{ linearly dependent columns}\}$.

For a linear code C, the **row space** (or **row span**) of a $k \times n$ **generator matrix** G over $\mathbb{F}$ generates C, in the sense that C is a set of linear combinations of G.

2.10 Standard Form Matrices

For a linear code C of dimension k and length $n = k + m$,

- $H = (I_m \mid B)$ is a parity check matrix for C *if and only if*
- $G = (-B^T \mid I_k)$ is a generator matrix C.

Linear codes $C$and C' are **equivalent** if C' is obtained by permuting the codeword entries of C by a fixed permutation:

$$C' = CP = \{\mathbf{x}P : \mathbf{x} \in C\} \text{ for some permutation matrix } P$$

Note that $G' = GP$ and $H' = HP$.

2.11 Extending Linear Codes

The **extension** of a linear code C:

$$\hat{C} = \{x_0 x_1 \cdots x_n : x_1 \cdots x_n \in C, x_0 = -(x_1 + \cdots + x_n)\}.$$

The **extension** $\hat{C}$ is a linear code with minimum distance $d(C)$ or $d(C)+1$.

2.12 Radix r Hamming Codes

- Let r be a prime number and $m \ge 1$ some integer.
- Write the numbers $1, \ldots, r^m - 1$ in base r, as length m column vectors.
- Of each set of $r-1$ parallel columns, delete all whose *first nonzero entry* is not 1.
- This gives the **radix r Hamming code** of length $n = \frac{r^m-1}{r-1}$.

3 Compression Coding

Definitions

source S	with	**symbols**	$s_1, \dots, s_q$
	with	**probabilities**	$p_1, \dots, p_q$
code C	with	**codewords**	$\mathbf{c}_1, \dots, \mathbf{c}_q$
		of **lengths**	$\ell_1, \dots, \ell_q$
		and **radix**	r

3.1 Instantaneous and UD Codes

A code C is

- **uniquely decodable (UD)** if it can always be decoded **unambiguously**
- **instantaneous** if no codeword is a **prefix** of another. Such a code is an **I-code**.

Decision trees can represent I-codes.

- Branches are numbered from the top down.
- Any radix r is allowed.
- Two codes are equivalent if their decision trees are isomorphic.
- By shuffling source symbols, we may assume that $\ell_1 \leq \ell_2 \leq \dots \leq \ell_q$.

The Kraft-Mcmillan Theorem The following are equivalent:

1. There is a radix r **UD-code** with codeword lengths $\ell_1 \leq \ell_2 \leq \dots \leq \ell_q$
2. There is a radix r **I-code** with codeword lengths $\ell_1 \leq \ell_2 \leq \dots \leq \ell_q$
3. $K = \sum_{i=1}^{q} (\frac{1}{r})^{\ell_i} \leq i$

3.2 Minimal UD-Codes

The (expected or) **average length** and **variance** of codewords in C are

$$L = \sum_{i=1}^{q} p_i \ell_i \qquad V = (\sum_{i=1}^{q} p_i \ell_i^2) - L^2$$

A UD-code is **minimal** with respect to $p_1, \dots, p_q$ if it has minimal length.

Minimal UD-Codes If a UD-code has minimal average length L with respect to $p_1, \dots, p_q$, then, possibly after permuting codewords of equally likely symbols,

- $\ell_1 \leq \ell_2 \dots \leq \ell_q$
- $\ell_{q-1} = \ell_q$
- If C is instantaneous, then $\mathbf{c}_{q-1}$ and $\mathbf{c}_q$ differ only in their last place.
- If C is binary, then

$$K = \sum_{i=1}^{q} 2^{-\ell i} = 1$$

3.3 Huffman's Algorithm

Binary Case

1. Write the symbols in a column, with highest probability at the top and lowest probability at the bottom.
2. Merge the bottom two (least frequent) symbols s_q and s_{q-1} into one big symbol of probability $p_q + p_{q-1}$.
3. Write the resulting $q - 1$ symbols in a new column to the right in same order as before. Make sure to place the newly created symbol as high as possible in this column.
4. Draw branches from the newly created symbol to its two constituent symbols, and label them 0 and 1.
5. Repeat the above, until there is only one symbol left.

Huffman Code Theorem For any given source S and corresponding probabilities, the Huffman Algorithm yields an instantaneous minimum UD-code.

Knuth The average codeword length L of each Huffman code is the sum of all child node probabilities.

3.4 Extensions

For a source $S = \{s_1, \dots, s_q\}$ with probabilities $p_1, \dots, p_q$, the n**-th extension** of S is the Cartesian product S^n, containing all strings of n symbols in S.

The probability of each symbol in S^n is the product of the probabilities of constituent symbols. We also order the new symbols in non-increasing probability.

3.5 Markov Sources

A k-**memory source** S is one whose symbols each depend on the previous k.

- If $k = 0$, then no symbol depends on any other, and S is **memoryless**.
- If $k = 1$, then S is a **Markov source**.
- $p_{ij} = P(s_i \mid s_j)$ is the probability of s_i occurring right after a given s_j.
- The matrix $M = (p_{ij})$ is the **transition matrix**.
- Entry p_{ij} is the probability of getting from state s_j to state s_i.

A Markov process M is in **equilibrium p** if $\mathbf{p} = M\mathbf{p}$.

We will assume that

- M is **ergodic**: we can get from any state j to any state i.
- M is **aperiodic**: the gcd of cycle lengths is 1.

Under the above assumptions, M has a non-zero equilibrium state.

3.6 Arithmetic Coding

Consider a source $\{s_1, \ldots, s_q\}$ where $s_q = \bullet$ is called a stop symbol, with probabilities $p_1, \ldots, p_q$. In this context, a message will always end with a stop symbol. Encoding a message $s_{i_1} \ldots s_{i_n}$ involves the following steps:

- Split up the interval $[0, 1)$ into sub-intervals of size $p_1, \ldots, p_q$.
- Choose the i_1-th sub-interval.
- Split up this sub-interval again, in proportion to $p_1, \ldots, p_q$.
- Choose the i_2-th sub-interval.
- Repeat this for the rest of the symbols, and output any number inside the final sub-interval found.

3.7 Dictionary Methods

Encoding Consider a message $m = m_1 m_2 \ldots m_n$. To encode m:

- Begin with an empty table D, and set the 0-th entry to $\emptyset$, representing an empty string.
- Find the longest prefix s of m in D (possibly the empty string $\emptyset$), and say s is in entry k.
- Find the symbol c just after s.
- Add a new entry sc to D, remove sc from m, and output (k, c).
- Repeat until m is fully encoded.

Decoding Consider an encoded message $(k_1, c_1) \ldots (k_n, c_n)$. To decode this message, take the following steps:

- Begin with a table D with $\emptyset$ in the 0-th entry.
- Let s_1 be the k_1-th entry in the table. Append $s_1 c_1$ to the table, and output $s_1 c_1$.
- Let s_2 be the k_2-th entry in the table. Append $s_2 c_2$ to the table, and output $s_2 c_2$.
- Keep doing this until the message is fully decoded.

4 Information Theory

Define $I(s_i) = I(p_i) = -\log_2 p_i$.
Define the (Shannon) **entropy** of S:

$$H_r(S) = \sum_{i=1}^{q} p_i I_r(p_i) = -\sum_{i=1}^{q} p_i \log_r p_i$$

This expresses the average information per source symbol.

Gibb's Inequality If $p_1, \dots, p_q$ and $p'_1, \dots, p'_q$ are probability distributions, then

$$-\sum_{i=1}^{q} p_i \log_r p_r \leq -\sum_{i=1}^{q} p_i \log_r p'_i.$$

Equivalently,

$$\sum_{i=1}^{q} p_i \log_r \frac{p'_i}{p_i} \leq 0.$$

Furthermore, there is equality if and only if $p_i = p'_i$ for all i.

Maximum Entropy Theorem For any source S with q symbols, the base r entropy satisfies

$$H_r(S) \leq \log_r q$$

with equality if and only if all symbols are equally likely.

First Source-Coding Theorem For each radix r UD-code C for source S,

$$H_r(S) \leq L_r$$

with equality iff $p_i = r^{-\ell_i}$ for all i and $K_r = \sum_{i=1}^{q} r^{-\ell_i} = 1$.

4.1 Entropy of Extensions for Memoryless Sources

Entropy of Extensions

$$H_r(S^n) = nH_r(S).$$

Sannon's Source Coding Theorem Encoding S^n by an SF-code or a Huffman code allows the average codeword lengths to be arbitrarily close to the entropy:

$$\frac{L_r^{(n)}}{n} \to H_r(S) \quad \text{for } n \to \infty.$$

4.2 Entropy for Markov Sources

Consider a Markov source $S = \{s_1, \dots, s_q\}$ with probabilities $p_1, \dots, p_q$, transition matrix $M = (p_{ij}) = (P(s_i \mid s_j))$ and equilibrium $\mathbf{p} = (p_j)$.
The **conditional information** of s_i given s_j is

$$I(s_i \mid s_j) = -\log P(s_i \mid s_j) = -\log p_{ij}.$$

The **conditional entropy given** s_j is

$$H(S \mid s_j) = \sum_{i=1}^{q} p_{ij} I(s_i \mid s_j) = -\sum_{i=1}^{q} P(s_i \mid s_j) \log P(s_i \mid s_j)$$

The **Markov entropy** of S is

$$\begin{aligned} H_M(S) &= \sum_{j=1}^{q} p_j H(S \mid s_j) \\ &= -\sum_{i=1}^{q}\sum_{j=1}^{q} p_j p_{ij} \log p_{ij} \\ &= -\sum_{i=1}^{q}\sum_{j=1}^{q} P(s_j s_i) \log P(s_i \mid s_j). \end{aligned}$$

The **equilibrium entropy** of S is

$$H_E(S) = -\sum_{j=1}^{q} p_j \log p_j.$$

Theorem on Markov Entropy For a Markov source S,

$$H_M(S) \leq H_E(S).$$

There is equality if and only if the symbols in S are independent.

4.3 Noisy Channels

Source entropy: $$H(A) = -\sum_{j=1}^{u} P(a_j) \log P(a_j)$$

Output entropy: $$H(B) = -\sum_{i=1}^{v} P(b_i) \log P(b_i)$$

Conditional entropies: $$H(B \mid a_j) = -\sum_{i=1}^{v} P(b_i \mid a_j) \log P(b_i \mid a_j)$$

$$H(A \mid b_i) = -\sum_{j=1}^{u} P(a_j \mid b_i) \log P(a_j \mid b_i)$$

Joint entropy: $$H(A, B) = -\sum_{i=1}^{v}\sum_{j=1}^{u} P(a_j \cap b_i) \log P(a_j \cap$$

4.4 Channel Capacity

The **channel capacity** is

$$C = C(A, B) = \max I(A, B)$$

where the maximum is taken over all possible probabilities for A's symbols.

Theorem The channel capacity of a binary symmetric channel with crossover probability p is $1 - H(p)$.

5 Number Theory and Algebra

5.1 Revision of Discrete Mathematics

- Division Algorithm
- (Extended) Euclidean Algorithm
- Bezout's Identity
- Inverses
- Chinese Remainder Theorem

5.2 Number Theory Results

Given $m \in \mathbb{Z}^+$, the set of invertible elements in $\mathbb{Z}_m$ is denoted by

$$\mathbb{U}_m = \{a \in \mathbb{Z}_m : \gcd(a, m) = 1\}$$

and its elements are the **units** in $\mathbb{Z}_m$.
Euler's **phi-function** is defined by

$$\phi(m) = |\mathbb{U}_m|.$$

Formula for $\phi(m)$

1. If $\gcd(m, n) = 1$, then $\phi(mn) = \phi(m)\phi(n)$.
2. For a prime p and $\alpha \in \mathbb{Z}^+$, we have $\phi(p^\alpha) = p^\alpha - p^{\alpha-1}$.
3. Hence, if $m = p_1^{\alpha_1} p_2^{\alpha_2} \cdots p_r^{\alpha_r}$ is the prime factorisation of m, then

$$\phi(m) = (p_1^{\alpha_1} - p_1^{\alpha_1-1})(p_2^{\alpha_2} - p_2^{\alpha_2-1}) \cdots (p_r^{\alpha_r} - p_r^{\alpha_r-1}).$$

Primitive Element Theorem Given a prime p, there exists $g \in \mathbb{U}_p$ such that

$$\mathbb{U}_p = \{g^0 = 1, g, g^2, \ldots, g^{p-2} \quad \text{and} \quad g^{p-1} = 1.$$

Primitve Powers If g is primitive in $\mathbb{Z}_p$, then g^k is primitive if and only if $\gcd(k, p-1) = 1$ and hence there are $\phi(p-1)$ primitive elements in $\mathbb{Z}_p$.

Euler's Theorem If $\gcd(a, m) = 1$, then $a^{\phi(m)} \equiv 1 \pmod{m}$.

Corollary If $\gcd(a, m) = 1$, then $\text{ord}_m(a) \mid \phi(m)$.

Fermat's Little Theorem For prime p and any $a \in \mathbb{Z}$, $a^p \equiv a \pmod{p}$

5.3 Finite Fields

Finite Field Theorem If p is prime, $m(x)$ a monic irreducible in $\mathbb{Z}_p[x]$ of degree n, and α denotes a root of $m(x) = 0$, then

1. $\mathbb{F} = \mathbb{Z}_p[x]/\langle m(x) \rangle$ is a field,
2. $\mathbb{F}$ is a vector space of dimension n over $\mathbb{Z}_p$,
3. $\mathbb{F}$ has p^n elements,
4. $\{\alpha^{n-1}, \alpha^{n-2}, \ldots, \alpha, 1\}$ is a basis for $\mathbb{F}$,
5. $\mathbb{F} = \mathbb{Z}_p(\alpha)$ i.e. the smallest field containing $\mathbb{Z}_p$ and α,
6. there exists a primitive element γ of order $p^n - 1$ for which $\mathbb{F} = \{0, 1, \gamma, \gamma^2, \ldots, \gamma^{p^n-2}\}$,

if a field $\mathbb{F}$ has a finite number of elements, then $|\mathbb{F}| = p^n$ where p is prime, and $\mathbb{F}$is isomorphic to $\mathbb{Z}_p[x]/\langle, (x)\rangle$. Hence ALL fields with p^n elements are isomorphic to one another.

5.4 Primality Testing

Pseudo-Prime Test No if we fine n is composite.

- Let $a \in \mathbb{N}$ with $a < n$.
  - If $\gcd(a, n) \neq 1$, then n must be composite so return no
  - Otherwise, if $a^{n-1} \not\equiv 1 \pmod{n}$, then n is composite so return no

Lucas' Test Possible answer as to whether n is prime

- Let $a \in \mathbb{N}$ with $a < n$.
  - If $\gcd(a, n) \neq 1$, then n must be composite so no
  - If $a^{n-1} \not\equiv 1 \pmod{n}$, then n must be composite so no
  - If $a^{(n-1)/p} \not\equiv 1 \pmod{n}$ for all primes $p \mid n-1$, then yes

Miller-Rabin Test If n is composite, then return no

- Write $n = 2^s t + 1$ where t is odd
- Choose some $a \in \{1, \ldots, n-1\}$ at random
- If $a^t \equiv 1 \pmod{n}$, then it is probably prime
- For $r = 0, \ldots, s-1$,
  - If $a^{2^r t} \equiv -1 \pmod{n}$, then n is probably prime.
- Otherwise, it is not prime.

Fermat Factorisation a two-factorisation of n

- For $t = \lceil \sqrt{n} \rceil, \ldots, n$:
  - If $s^2 = t^2 - n$ is square, then return $n = ab = (t-s)(t+s)$

6 Algebraic Coding

6.1 Single Error Correcting BCH Codes

Let $f(x) \in \mathbb{Z}_2[x]$ be a polynomial of degree m with a primitive root α. Let $n = 2^m - 1$ and $k = n - m$.

The matrix $H = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \end{pmatrix}$ is the check matrix of a binary Hamming (n, k) code C. Every binary Hamming (n, k) code can be obtained in this way.

Let $\mathbf{c} = (c_0, c_1, \dots, c_{n-1}) \in C$ be a codeword.

- $c_0, \dots, c_{m-1}$ are the check bits
- $c_m, \dots, c_{n-1}$ are the information bits

The first m bits of the codeword are check bits, since the first m columns of H are the leading columns. Therefore the information bits and the check bits are neatly divided.

The syndrome of the codeword $\mathbf{c}$ is

$$S(\mathbf{c}) = H\mathbf{c}^T = c_0 + c_1\alpha + c_2\alpha^2 + \dots + c_{n-1}\alpha^{n-1} = C(\alpha)$$

where $C(x) = c_0 + c_1x + c_2x^2 + \dots + c_{n-1}x^{n-1}$ is the codeword polynomial corresponding to $\mathbf{c}$.

- Since $\mathbf{c}$ is the codeword, its syndrome is 0. Therefore, $S(\mathbf{c}) = 0 = C(\alpha)$, so α is a root of $C(x)$.
- Since α is a root of $C(x)$, the minimal polynomial, $M_1(x)$ of α must divide $C(x)$ without a remainder.
- $M_1(x)$ is the primitive polynomial $f(x)$.

BCH Encoding

1. From our message $c_m, \dots, c_{n-1}$, we form the **information polynomial**
$$I(x) = c_m x^m + c_{m+1}x^{m+1} + \dots + c_{n-1}x^{n-1}.$$
2. Using polynomial long division, we find the **check polynomial** of degree at most $m - 1$
$$R(x) = I(x)(\mod M_1(x)) = c_0 + c_1x + \dots + c_{m-1}x^{m-1}.$$
3. Calculate the **codeword polynomial**
$$C(x) = I(x) + R(x).$$

The codeword is $(c_0, c_1, \dots, c_{n-1})$ where the first m bits are check bits and the last k bits are information bits.

BCH Error Correcting and Decoding Suppose that we receive $\mathbf{d} = \mathbf{c} + e_j$, where e_j is a unit vector with 1 in the a^j position and zero entries elsewhere.

1. Let us represent $\mathbf{c}$ and $\mathbf{d}$ as codeword polynomials $C(x)$ and $D(x)$
2. Calculating the syndrome of d gives us
$$S(\mathbf{d}) = D(\alpha) = C(\alpha) + a^j = a^j$$
The error is therefore in the α^j position, which is the $j + 1^{th}$ letter of the code. If the syndrome of the codeword we receive is 0, i.e $D(\alpha) = 0$, then there is no error.
3. To decode a codeword, look at the last k bits, which are the information bits $(c_m, \dots, c_{n-1})$.

6.2 Two Error Correcting BCH Codes

Theorem If p is prime and β is a root of $f(x) \in \mathbb{Z}_2[x]$ then so is β^{p^i} for all i.

Construction

1. Find a primitive root of minimal polynomial $M_1(x)$, with cyclotomic coset K_1
2. Select an index not belonging to $K_1 (i \in \{1, \dots, p^m - 1\} \setminus K_1)$
3. Find the minimal polynomial $M_i(x)$ for α^i
4. Define $M(x) = M_1(x)M_i(x)$
5. Define the check matrix
$$H = \begin{pmatrix} 1 & \alpha & \dots & \alpha^{n-1} \\ 1 & \alpha^i & \dots & (\alpha^i)^{n-1} \end{pmatrix}$$
6. Define the syndrome of a codeword $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$ as
$$S(\mathbf{c}) = H\mathbf{c}^T = \begin{pmatrix} c_0 + c_1\alpha + \dots + c_{n-1}\alpha^{n-1} \\ c_0 + c_1\alpha^i + \dots + c_{n-1}(\alpha^i)^{n-1} \end{pmatrix} = \begin{pmatrix} C(\alpha) \\ C(\alpha^i) \end{pmatrix}$$

Encoding and Decoding Encoding a double-error correcting BCH code is the same as encoding a single-error correcting BCH code, with the difference being

$$R(x) = I(x)(\mod M(x))$$

where $M(x)$ is used instead of $M_1(x)$.

Error Correcting and Decoding Suppose that we received $\mathbf{d} = \mathbf{c} + e_j + e_l$.

1. Let us represent $\mathbf{c}$ and $\mathbf{d}$ as codeword polynomials $C(x)$ and $D(x)$
2. Calculate the syndrome
$$S(\mathbf{c}) = \begin{pmatrix} D(\alpha) \\ D(\alpha^i) \end{pmatrix} = \begin{pmatrix} C(\alpha) + a^j + a^l \\ C(\alpha^i) + (\alpha^i)^j + (\alpha^i)^l \end{pmatrix} = \begin{pmatrix} \alpha^j + \alpha^l \\ \alpha^{ij} + \alpha^{il} \end{pmatrix} = ($$

The syndrome allows us to determine when there is 0, 1 or 2 errors.

- 0 errors when $D(\alpha) = D(\alpha^i) = 0$
- 1 error when $D(\alpha) \neq 0$ and $D(\alpha)^i = D(\alpha^i)$
- 2 errors when $D(\alpha) \neq 0$ and $D(\alpha)^i \neq D(\alpha^i)$

7 Cryptography (Ciphers)

7.1 Some Classical Cryptosystems

- Caesar Ciphers
- Simple (monoalphabetic) substitution Cipher
- Transposition Cipher
- Combined Systems
- Polyalphabetic Substitution Ciphers
- Non-Periodic Polyalphabetic Substitutions

Kasiski's Method For a message m of length n, the index of coincidence is given by

$$I_c = \frac{\sum(f_i^2) - n}{n^2 - n}$$

where f_i is the frequency of each letter in the message. Solving for r we get

$$r \approx \frac{0.0273n}{(n-1)I_c - 0.0385n + 0.0658}$$

7.2 Unicity Distance

The unicity distance is

$$n_0 = \lceil \frac{H_2(K)}{\log_2 q - R} \rceil$$

where K is the total number of keys, q is the number of letters in the source alphabet, and R is the rate of the language in bits per character.

For English text, $q = 26$ and $R \approx 1.5$. So

$$n_0 = \lceil \frac{H_2(K)}{\log_2 q - R} \rceil \approx \lceil \frac{H_2(K)}{4.7 - 1.5} \rceil = \lceil \frac{H_2(K)}{3.2} \rceil.$$

If the keys are equally likely, then

$$n_0 \approx \lceil \frac{log_2|K|}{3.2} \rceil$$