Chapter 4

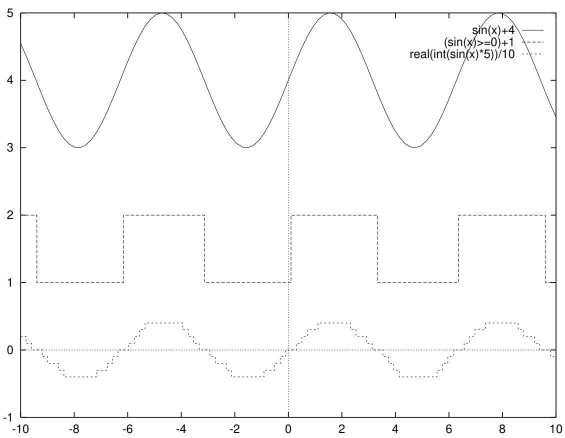
# Preliminaries - physical

When studying the transfer and storage of data, there are some underlying physical laws, representations and constraints to consider.

- Is the data analog or digital?
- What limits are placed on it?
- How is it to be transmitted?
- How can you be sure that it is correct?

## 4.1 Analog and digital

An analog signal is a continuous valued signal. A digital signal is considered to only exist at discrete levels.

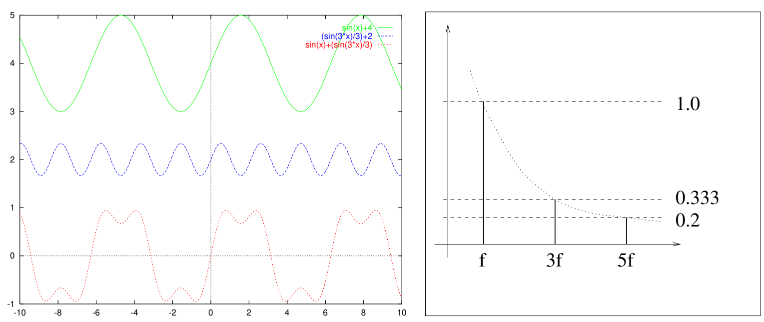


The (time domain) diagrams are commonly used when considering signals. If you use an oscilloscope, the display normally shows something like that shown on the previous page. The plot is **amplitude versus tim**e. With any analog signal, the repetition rate (if it repeats) is called the *frequency*, and is measured in Hertz (pronounced *hurts*, and written Hz). The peak to peak signal level is called the *amplitude*.

The simplest analog signal is called the sine wave. If we mix these simple waveforms together, we may create any desired periodic waveform. In figure 4.1, we see the sum of two sine waves - one at a frequency of 1,000Hz, and the other at three times the frequency (3,000Hz). The amplitudes of the two signals are 1 and $\frac{1}{3}$ respectively, and the sum of the two waveforms shown, approximates a *square* wave. If we were to continue summing these waves, in the same progression, the resultant waveform would be a square wave

$$\sum_{n=1}^{\infty} \frac{1}{n} \sin(2\pi n f) \text{ (for odd n)} \Rightarrow \text{ a square wave of frequency } f$$

We may also represent these signals by frequency domain diagrams, which plot the amplitude against *frequency*. This alternative representation is also shown in figure 4.1.




(a) Time (b) Frequency

Figure 4.1: Sum of sine waveforms.

## 4.2 Fourier analysis

One way of representing any simple periodic function is as a sum of simple sine (and cosine) waveforms. This representation method is known as *Fourier Analysis* after Jean-Baptiste Fourier, who first showed the technique.

The Fourier method can be viewed as a transformation between equivalent time domain and frequency domain representations. A piecewise continuously differentiable periodic function in the time domain may be transformed to a discrete aperiodic function in the frequency domain.

If our time domain function is $f(t)$ then we normally write the corresponding frequency domain function as $F(\omega)$, and we use the symbol $\leftrightarrow$ to represent the transformation:

$$f(t) \leftrightarrow F(\omega)$$

There are various *flavours* of Fourier analysis depending on the types of functions in each domain. The table below summarizes the methods used.

| **Time domain** | | **Frequency domain** | **Description** |
|---|---|---|---|
| Continuous, periodic | $\rightleftarrows$ | Discrete, aperiodic | **Fourier series** |
| Continuous, aperiodic | $\rightleftarrows$ | Continuous, aperiodic | **Fourier transform** |
| Discrete, periodic | $\rightleftarrows$ | Discrete, periodic | **Discrete Fourier series** |
| Discrete, aperiodic | $\rightleftarrows$ | Continuous, periodic | **Discrete Fourier transform** |

We can see an example of this deconstruction or construction of waveforms by examining a bipolar square wave which can be created by summing the terms:

$$\frac{4}{\pi}(sin(2\pi ft) + \frac{1}{3}sin(6\pi ft) + \frac{1}{5}sin(10\pi ft) + \frac{1}{7}sin(14\pi ft) + ...)$$

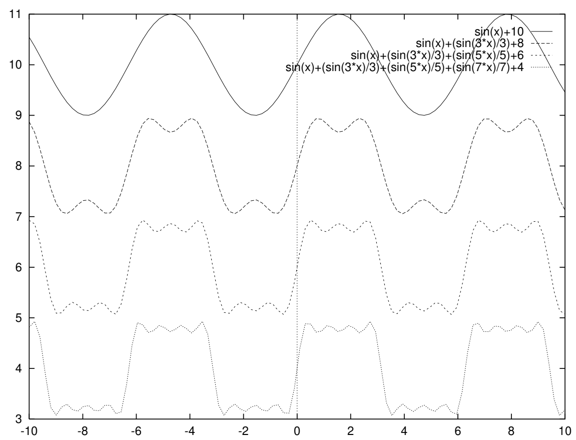
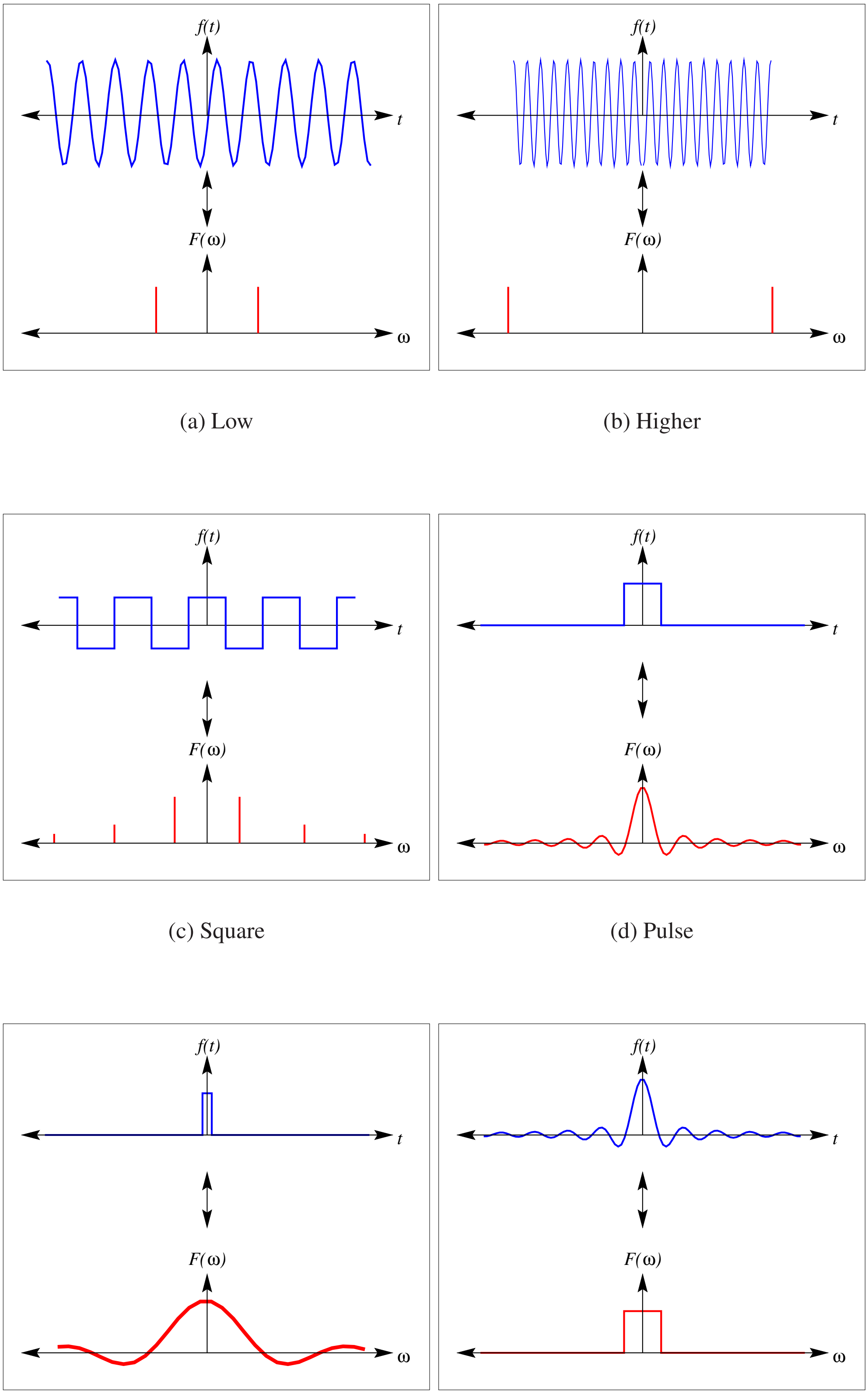



Figure 4.2: Successive approximations to a square wave.

In figure 4.2, we see four plots, showing the resultant waveforms if we sum the first few terms in the series. As we add more terms, the plot more closely approximates a square wave.

Note that there is a direct relationship between the bandwidth of a channel passing this signal, and *how accurate* it is. If the original (square) signal had a frequency of 1,000Hz, and we were attempting to transmit it over a channel which only passed frequencies from 0 to 1,000Hz, we would get a sine wave.

Another way of stating this is to point out that the higher frequency components are important - they are needed to re-create the original signal faithfully. If we had two 1,000Hz signals, one a triangle, one a square wave - if they were both passed through the 1,000Hz bandwidth limited channel above, they would look identical (a sine wave).




(a) Low

(b) Higher

(c) Square

(d) Pulse

(e) Shorter

(f) The

Figure 4.3: Sample plots showing functions and their transforms.

### 4.2.1 Fourier transform

With aperiodic waveforms, we consider the **Fourier Transform** of our function $f(t)$, which is the function $F(\omega)$ given by

$$F(\omega) = \int_{-\infty}^{\infty} f(t)e^{-j\omega t}dt$$

This transform may be inverted to give

$$f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{j\omega t}d\omega$$

In figure 4.3 we see various simple transforms. Note that if a function in one domain is *widened*, its transform *narrows*.

### 4.2.2 Convolution

One of the important theorems in Fourier analysis is the convolution theorem, which states that:

> If $f(t)$ and $g(t)$ are two functions with Fourier transforms $F(\omega)$ and $G(\omega)$, then the Fourier transform of the convolution $f(t) \star g(t)$ is the product of the Fourier transforms of the functions $F(\omega)$ and $G(\omega)$, *and vice versa*.

$$\begin{array}{rcl} f(t) \star g(t) & \leftrightarrow & F(\omega) \times G(\omega) \\ f(t) \times g(t) & \leftrightarrow & F(\omega) \star G(\omega \end{array}$$

The convolution $k(t)$ of $f(t)$ and $g(t)$ may be expressed as

$$k(t) = f(t) \star g(t) = (f \star g)(t) = \frac{1}{T}\int_{-\frac{T}{2}}^{+\frac{T}{2}} f(t-\tau)g(\tau)\,d\tau$$

but it also has a *graphical* interpretation. We can use convolution to easily predict the functions that result from complex signal filtering or sampling[1].

In figure 4.4, we see a sine wave and a sampling window, each with their own Fourier transform. By multiplying the two waveforms, we end up with a single cycle of the sine wave, and we can deduce its frequency domain representation by convolving the two Fourier transforms.

[1] In class, we will use this technique to demonstrate the impossibility of a *perfect* filter.

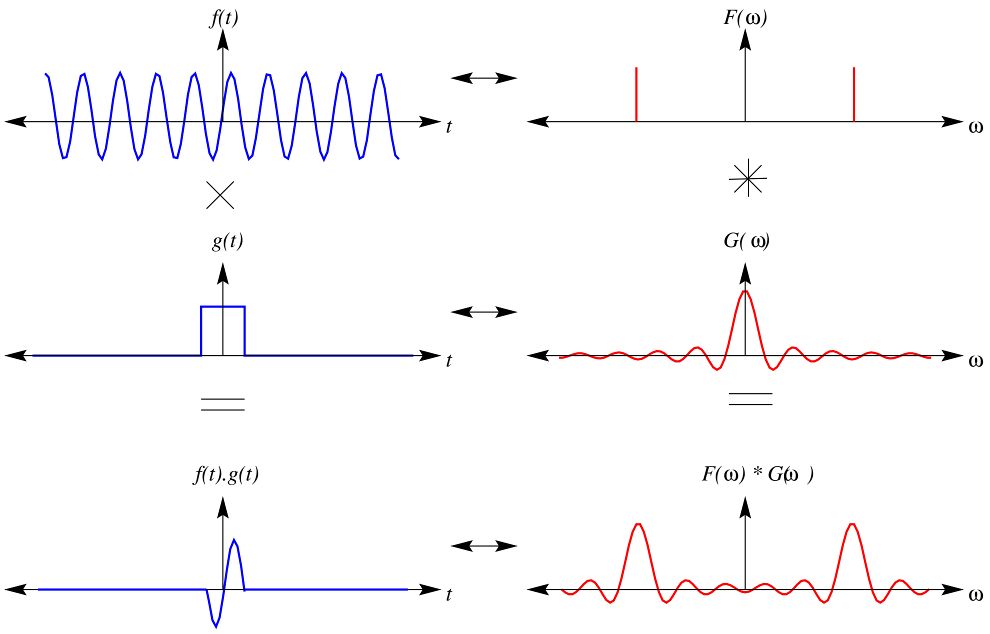



Figure 4.4: Window sampling.

## 4.3 Modulation

A baseband signal is one in which the data component is directly converted to a signal and transmitted. When the signal is imposed on another signal, the process is called modulation.

We may *modulate* for several reasons:

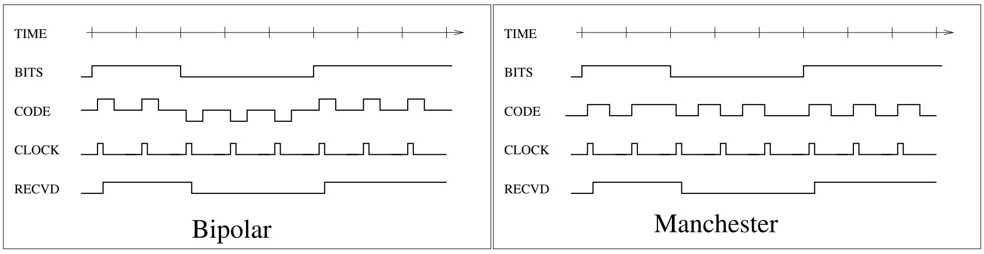
- The media may not support the baseband signal
- We may wish to use a single transmission medium to transport many signals

We use a range of modulation methods, often in combination:

- Frequency modulation - frequency shift keying (FSK)
- Amplitude modulation
- Phase modulation - phase shift keying (PSK)
- Combinations of the above (QAM)

### 4.3.1 Baseband digital encoding

The simplest encoding scheme is just to use a *low* level for a *zero* bit, and a *high* level for a *one* bit. As long as both ends of a channel are synchronized in some manner, we can transfer data. On the other hand, if the ends of the channel are not synchronized we might use a simple encoding scheme, such as *Bipolar* or *Manchester* encoding, to transfer synchronizing (clock) information on the same channel.




In Bipolar encoding, a 1 is transmitted with a positive pulse, a 0 with a negative pulse. Since each bit contains an initial transition away from zero volts, a simple circuit can extract this clock signal. This is sometimes called *return to zero* encoding.

In Manchester (phase) encoding, there is a transition in the center of each bit cell. A binary 0 causes a high to low transition, a binary 1 is a low to high transition. The clock retrieval circuitry is slightly more complex than before.

## 4.4 Information theory

The term ***information*** is commonly understood. Consider the following two sentences:

1. The sun will rise tomorrow.
2. The Fiji rugby team will demolish the All Blacks (New Zealand rugby team) the next time they play.

**Question:** Which sentence contains the most information[2]?

Two early researchers - Nyquist (1924) and Hartley (1928) laid the foundation for a formal treatment of information.

Hartley showed that the information content of a message is proportional to the *logarithm* of the number of possible messages. He used morse encodings, but the same can be applied to binary encodings - if we wish to encode integers between 1 and $n$, we need $\log_2 n$ bits.

[2]Most of us would have no hesitation in stating that the *second* statement contains more information than the *first* statement. In general, the less predictable the message, the more information in it.

Shannon developed a more complete mathematical treatment of communication and information in a important paper [Sha48] at http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html. The paper develops a general theory of communication over a noisy channel. We model the communication system in schematic form in figure 4.5.

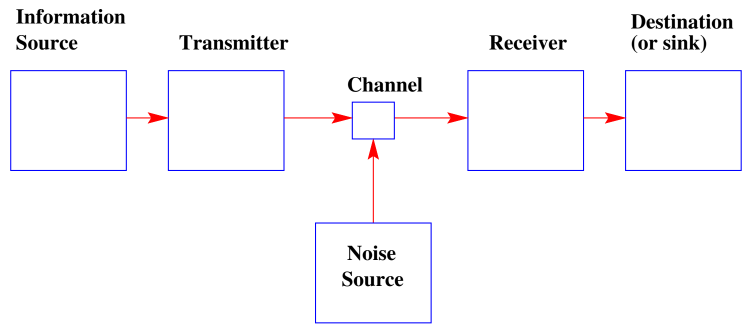



Figure 4.5: Model of communication system.

In the following presentation, we assume that the unit of information is the binary digit (or bit), as most computer systems use this representation. There is a strong parallel found in other scientific areas - for example the study of statistical mechanics has a similar concept of entropy.

The relevance of Shannon's theory of communication to the study of secrecy systems is explored in another important paper [Sha49] at

http://www.cs.ucla.edu/˜jkong/research/security/shannon.html

### 4.4.1 Entropy

In our communication model, the units of transmission are called *messages*, constructed from an alphabet of (say) $n$ symbols $x \in \{x_1, \ldots, x_n\}$ each with a probability of transmission $P_x$. We associate with each symbol $x$ a quantity $H_x$ which is a measure of the *information* associated with that symbol.

$$H_x = P_x \log_2 \frac{1}{P_x}$$

If the probability of occurence of each symbol is the same, by adding these for all symbols $x$ we can derive Hartley's result, that the average amount of information transmitted in a single symbol (the source *entropy*) is

$$H(X) = \log_2 n$$

where $X$ is a label referring to each of the source symbols $x_1, \ldots, x_n$. However, if the probability of occurence of each symbol is not the same, we derive the following result, that the source *entropy* is

$$H(X) = \sum_{i=1}^{n} P_{x_i} \log_2 \frac{1}{P_{x_i}}$$

Shannon's paper shows that $H$ determines the channel capacity required to transmit the desired information with the *most* efficient coding scheme. Our units for entropy can be *bits/second* or *bits/symbol*, and we also sometimes use unit-less *relative* entropy measures (relative to the entropy of the system if all symbols were equally likely). We can also define the entropy of a continuous (rather than the discrete) distribution over $x$ with density $p(x)$ as

$$H(x) = \int_{-\infty}^{\infty} p(x) \log_2 \frac{1}{p(x)} dx$$

**Example:** If we had a source emitting two symbols, $0$ and $1$, with equal probabilities of occuring, then the entropy of the source is

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P_{x_i} \log_2 \frac{1}{P_{x_i}} \\ &= 0.5 \log_2 2 + 0.5 \log_2 2 \\ &= 1 \text{ bits/symbol} \end{aligned}$$

**Example:** If we had a source emitting two symbols, $0$ and $1$, with probabilities of $1$ and $0$, then the entropy of the source is

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P_{x_i} \log_2 \frac{1}{P_{x_i}} \\ &= \log_2 1 + 0 * \log_2 0 \\ &= 0 \text{ bits/symbol} \end{aligned}$$

Note that:

$$\lim_{y \to 0} y \log_2 \frac{1}{y} = 0$$

The information rate for a source providing $r$ symbols/sec is $R = rH(X)$ bits/sec, and the channel capacity is

$$C = \lim_{T \to \infty} \frac{\log_2 N(T)}{T}$$

**Example:** If we were transmitting a sequence of letters *A,B,C,D,E* and *F* with probabilities $\frac{1}{2}, \frac{1}{4}, \frac{1}{16}, \frac{1}{16}, \frac{1}{16}$ and $\frac{1}{16}$, the entropy for the system is

$$\begin{aligned} H(X) &= \frac{1}{2} \log_2 2 + \frac{1}{4} \log_2 4 + \frac{4}{16} \log_2 16 \\ &= 0.5 + 0.5 + 1.0 \\ &= 2 \text{ bits/symbol} \end{aligned}$$

Lets compare two encodings, first a fixed size 3-bit code, and then a more complex code:

| Symbol | 3-bit code | Complex code |
|---|---|---|
| **A** | 000 | 0 |
| **B** | 001 | 10 |
| **C** | 010 | 1100 |
| **D** | 011 | 1101 |
| **E** | 100 | 1110 |
| **F** | 101 | 1111 |

The average length of the binary digits needed to encode a typical sequence of symbols using the 3-bit code is

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P_{x_i} \bullet \text{sizeof}(x_i) \\ &= \frac{1}{2} * 3 + \frac{1}{4} * 3 + \frac{4}{16} * 3 \\ &= 1.5 + 0.75 + 0.75 \\ &= 3 \text{ bits/symbol} \end{aligned}$$

But we can do much better if we encode using the other encoding. The average length of the binary digits needed to encode a typical sequence of symbols using the complex encoding is

$$\begin{aligned} H(X) &= \sum_{i=1}^{n} P_{x_i} \bullet \text{sizeof}(x_i) \\ &= \frac{1}{2} * 1 + \frac{1}{4} * 2 + \frac{4}{16} * 4 \\ &= 0.5 + 0.5 + 1.0 \\ &= 2 \text{ bits/symbol} \end{aligned}$$

**Example:** If our source was transmitting $0$ and $1$ bits with equal probability, but the received data was corrupted 50% of the time, we might reason that our rate $r(X)$ of information transmission was $0.5$, because half of our data is getting through correctly.
However, a better argument is to consider the difference between the entropy of the source and the conditional entropy of the received data:

$$r(X) = H(X) - H(X \mid y)$$

where $H(X \mid y)$ is the conditional entropy of the received data.

$$\begin{aligned} H(X \mid y) &= 0.5 * \log_2 2 + 0.5 * \log_2 2 \\ &= 1 \\ \text{and} \quad H(X) &= 1 \qquad \text{(shown before)} \\ \text{so} \quad r(X) &= H(X) - H(X \mid y) \\ &= 0 \text{ bits/symbol} \end{aligned}$$

This is a much better measure of the amount of information transmitted when you consider that you could model the system just as effectively by using a random bit generator connected to the receiver.

### 4.4.2 Redundancy

The ratio of the entropy of a source $H(X)$ to what it would be if the symbols had equal probabilities $H'(X)$, is called the *relative* entropy. We use the notation $H_r(X)$, and

$$H_r(X) = \frac{H(X)}{H'(X)}$$

The *redundancy* of the source is $1 - H_r(X)$.

$$R(X) = 1 - H_r(X)$$

If we look at English text a symbol at a time[3], the redundancy is about $0.5$. This indicates that it should be simple to compress English text by about 50%.

### 4.4.3 Shannon and Nyquist

In white noise, the distribution of the power densities for a signal with noise power $N$ is a gaussian function:

$$p(x) = \frac{1}{(2\pi N)} e^{-\frac{1}{2N}x^2}$$

If $p(x_1, ..., x_n)$ is a gaussian density distribution function for $n$ samples $x_1, \ldots, x_n$ (i.e. sampled white noise), then Shannon derives the *power* entropy $H(X)$ of the function:

$$H(X) = W \log_2 2\pi e N$$

the maximum possible entropy for a given average power $N$.

Assume we have a composite signal with entropy $H(Y)$, consisting of an information source $H(S)$ and a noise source with entropy $H(N)$. If the noise is independent of the signal, our channel capacity is

$$C = H(Y) - H(N)$$

If all these sources are essentially random (i.e. they have maximum entropy), then

$$\begin{aligned} H(Y) &= W \log_2 2\pi e(S + N) \\ H(N) &= W \log_2 2\pi e N, \quad \text{and so} \\ C &= W \log_2(1 + \frac{S}{N}) \end{aligned}$$

This result is commonly used for noisy (thermal noise) channels, expressed in the following way:

**Maximum BPS =** $W \ \log_2(1 + \frac{S}{N})$ bits/sec

[3] That is, without considering letter *sequences*.

**Example:** If we had a telephone system with a bandwidth of 3,000 Hz, and a S/N of 30db (about 1024:1)

$$\begin{aligned} D &= 3000 * \log_2 1025 \\ &\approx 3000 * 10 \\ &\approx 30000 \quad \text{bps} \end{aligned}$$

This is a typical maximum bit rate achievable over the telephone network.

Nyquist shows us that the maximum data rate over a limited bandwidth (W) channel with V discrete levels is:

**Maximum data rate** = $2W \log_2 V$ bits/sec

For example, two-Level data cannot be transmitted over the telephone network faster than 6,000 BPS, because the *bandwidth* of the telephone channel is only about 3,000Hz.

**Example:** If we had a telephone system with a bandwidth of 3,000 Hz, and using 256 levels:

$$\begin{aligned} D &= 2 * 3000 * \log_2 256 \\ &= 6000 * 8 \\ &= 48000 \quad \text{bps} \end{aligned}$$

In these equations, the assumption is that the relative entropies of the signal and noise are a maximum (that they are random). In practical systems, signals rarely have maximum entropy, and we can do better - there may be methods to compress the data[4].

## 4.5 Huffman encoding

An immediate question of interest is "*What is the minimum length bit string that may be used to compress a string of symbols?*".

The Huffman encoding minimizes the bit length given the frequency of occurence of each symbol[5]. The resultant bit string in the best case will be the length predicted from the calculation of the source entropy.

A Huffman encoder uses a binary tree with symbols arranged at the leafs such that each leaf has a unique prefix. In the example in figure 4.6, the letter E is encoded by following the path from

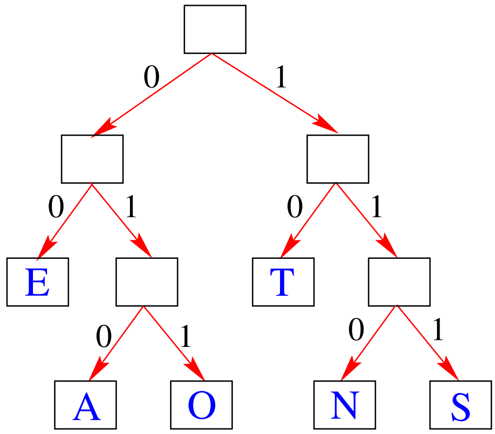



Figure 4.6: Tree encoding for Huffman codes.

the tree root “00”. This is the shortest path and shortest encoding, since E is the most commonly used letter in English text.

We can see that less common characters such as A, O, N and S, use longer bit strings. Our algorithm for encoding is simple - we calculate the tree encoding knowing the frequency of each letter, and just construct a table for each symbol:

| Symbol | Coding |
|---|---|
| E | 00 |
| T | 10 |
| A | 010 |
| O | 011 |
| N | 110 |
| S | 111 |

To decode a Huffman encoded string, we traverse the tree as each bit is received, taking a left path or a right path according to the bit being a 0 or a 1. When we reach the leaf, we have our symbol.

[4]**Note:** we must also differentiate between lossy and lossless compression schemes. A signal with an entropy of 0.5 may not be compressed more than 2:1 unless you use a lossy compression scheme. JPEG and Wavelet compression schemes can achieve huge data size reductions without visible impairment of images, but the restored images are not the same as the original ones - they just look the same. The lossless compression schemes used in PkZip, gzip or GIF files (LZW) cannot achieve compression ratios as high as that found in JPEG.

[5]Note that it presupposes knowledge about these frequencies.

## 4.6 Case study - MNP5 and V.42bis

MNP5 and V42.bis are compression schemes commonly used on modems. MNP5 suffers from the unfortunate property that it will *expand* data with maximum or near-maximum entropy (instead of compression). V42.bis does not have this property - it uses a large dictionary, and will not try to compress an already compressed stream.

MNP5 uses two different compression methods, switching between them as appropriate. The methods are:

- Adaptive frequency encoding
- Run-length encoding

Run length encoding sends the bytes with a byte count value, and doubles the size of a data stream with maximum entropy. Adaptive frequency encoding uses a similar scheme as that shown in our *complex-code* in section 4.4.1:

| 3-bit header | Body size | Total code size | Number of codewords |
|---|---|---|---|
| 000 | 1 bit | 4 bits | 2 |
| 001 | 1 bit | 4 bits | 2 |
| 010 | 2 bits | 5 bits | 4 |
| 011 | 3 bits | 6 bits | 8 |
| 100 | 4 bits | 7 bits | 16 |
| 101 | 5 bits | 8 bits | 32 |
| 110 | 6 bits | 9 bits | 64 |
| 111 | 7 bits | 10 bits | 128 |

We can see from this that $\frac{3}{4}$ of our codewords are *larger* than they would be if we did not use this encoding scheme, and with an input stream with an even spread of data (i.e. maximum entropy), our encoding will increase the size of data.

## 4.7 Summary of topics

In this section, we introduced the following topics:

- Physical preliminaries, Fourier analysis and convolution
- Entropy
- Encoding

---

## Supplemental questions for chapter 4

1. Assuming that the data transferred has maximum entropy, what is the maximum bit transfer rate using 16 level data over a cable with a bandwidth of 1MHz?
2. Assuming that the data transferred has an entropy of 0.2, what is the maximum bit transfer rate using 16 level data over a cable with a bandwidth of 1MHz?
3. Assuming that the signal-to-noise ratio of a communication system is 16:1, what is the maximum bit transfer rate over a cable with a bandwidth of 1MHz?
4. Calculate the entropy of a source transmitting 64 different characters, with the probabilities of E, T, A, O, N, S, H, R being $\frac{1}{4}$, $\frac{1}{8}$, $\frac{1}{16}$, $\frac{1}{16}$, $\frac{1}{16}$, $\frac{1}{16}$, $\frac{1}{16}$ and $\frac{1}{16}$ respectively and the other 56 characters being evenly distributed.
5. Devise a Huffman encoding for the above data.
6. Classify each of the diagrams in Figure 4.3 according to their periodicity and discreteness in each domain.

---

## Further study

- Textbook Chapter 32
- Shannon's paper on secrecy systems [Sha49] at http://www.cs.ucla.edu/~jkong/research/security/shannon.html.

---

Chapter 5

# Preliminaries - security models

The term *security model* refers to a range of formal policies for specifying the security of a system in terms of a (mathematical) model. There are various ways of specifying such a model, each with their own advantages and disadvantages. We will look at several models, beginning with the simple access control matrix model, and continuing with the Bell-LaPadula, Biba and Clark-Wilson models. Each of these models views security as a problem in *access*[1].

Having a model of course is not the end of the story. We need to be able to determine properties of the model, and to verify that our implementations of the security model are valid. However the above models have formed the basis of various trusted operating systems.

## 5.1 Access control matrix

The access control matrix as described towards the end of [Den71] allows us to specify and formalize a set of rules that might be intended to implement a security policy. The rows of the matrix correspond to subjects, and the columns correspond to objects:

**Objects**

| **Subjects** | $f_1$ | $f_2$ | $f_3$ | $f_4$ |
|---|---|---|---|---|
| $s_1$ | `read` `execute` | `execute` | | |
| $s_2$ | `write` | | `read` | `execute` |
| $s_3$ | `read` | `write` | | `execute` |
| $s_4$ | | `read` | `write` | `read` |

[1]The terms *principal* and *subject* are sometimes used interchangeably in much of the security model literature, and both refer to the active entity in the model - say a *process*. However the term principal sometimes refers to other things (for example a public key is sometimes termed a principal), so we will avoid using it. The term *object* refers to the resource (a file, or another process).

The matrix can be considered a control element for access to each object. In an OS, if the object was a file, then our access permissions (**`read`**, **`write`** and **`execute`**) are managed by the file system. If the objects were processes, then we may have permissions like **`sleep`**, **`wakeup`** and so on. In this case, the permissions are managed by the scheduler.

By examining this matrix, we can see that $s_4$ cannot read $f_1$. However, if you examine it more closely, you may see a way that subjects may collude to allow $s_4$ to read $f_1$.

## 5.2 Bell-LaPadula for confidentiality

The Bell-LaPadula [BL75] model (no read-up, no write-down) provides a military viewpoint to assure *confidentiality* services. There is a brief introduction to this which is worth reading in [MP97]. In Bell-LaPadula, we have a model with security levels in a (total) ordering formalizing a policy which restricts information flow from a higher security level to a lower security level. That is, we want to stop lower-level subjects from accessing higher-level objects.

In [MP97], we have four levels $l \in \mathcal{L}$ of security classification:

1. Top secret $(T)$
2. Secret $(S)$
3. Confidential $(C)$
4. Unclassified $(U)$

where $T > S > C > U$. Access operations associated with a set of objects $\mathcal{O}$, subjects $\mathcal{S}$ may be specified or visualized using an access control matrix, and are drawn from {**`read`**, **`write`**}.

The clearance classification for a subject $s \in \mathcal{S}$ or object $o \in \mathcal{O}$ is denoted $L(s) = l_s$ or $L(o) = l_o$. We might then assume we can use this to construct a first simple security property:

- **No read-up-1:** $s$ can **`read`** $o$ if and only if $l_o \leq l_s$, and $s$ has **`read`** access in the access control matrix.

This single property is insufficient to ensure the restriction we need for the security policy. Consider the case when a low security subject creates a high security object (say a program) which then reads a high security file, copying it to a low security one. This behaviour is commonly called a Trojan Horse. A second property is needed:

- **No write-down-1:** $s$ can **`write`** $o$ if and only if $l_s \leq l_o$, and $s$ has **`write`** access in the access control matrix.

These two properties can be used to enforce our security policy, but with a severe restriction. For example, how does any subject write *down* without invalidating a security policy?

The BLP model is a little more developed than this, and includes a concept of security categories. A security category $c \in \mathcal{C}$ is used to classify objects in the model, with any object belonging to a set of categories. Each pair $(l \times c)$ is termed a *security level*, and forms a lattice. We define a relation between security levels:

- The security level $(l, c)$ dominates $(l', c')$ (written $(l, c)$ **dom** $(l', c')$) iff $l' \leq l$, and $c' \subseteq c$.

A subject $s$ and object $o$ then belong to one of these security levels. The new properties are:

- **No read-up-2:** $s$ can **`read`** $o$ if and only if $s$ **dom** $o$, and $s$ has **`read`** access in the access control matrix.
- **No write-down-2:** $s$ can **`write`** $o$ if and only if $o$ **dom** $s$, and $s$ has **`write`** access in the access control matrix.

A system is considered secure in the current state if all the current accesses are permitted by the two properties. A transition from one state to the next is considered secure if it goes from one secure state to another secure state. The basic security theorem stated in Theorem 5-2 in the textbook states that if the initial state of a system is secure, and if all state transitions are secure, then the system will always be secure.

BLP is a static model, not providing techniques for changing access rights or security levels[2], and there is an exploration and discussion into the limitations of this sort of security modelling in section 5.4 of the textbook. However the model does demonstrate initial ideas into how to model, and how to build security systems that are provably secure.

## 5.3 Biba model for integrity

The Biba [Bib75] models attempt to model the trustworthiness of data and programs, providing assurance for *integrity* services. An integrity level might be something like **`clean`** or **`dirty`** (in reference to database entries). We can consider the main Biba model as a kind of *dual* for the Bell-LaPadula model, concerned with *integrity* rather than confidentiality.

The integrity levels $\mathcal{I}$ are ordered as for the security levels, and we have a function $i : \mathcal{O} \rightarrow \mathcal{I}$ $(i : \mathcal{S} \rightarrow \mathcal{I})$ which return the integrity level of an object (subject).

[2] You might want to explore the Harrison-Ruzo-Ullman model for this capability.

The properties/rules for the *main* (static) Biba model are:

- **No read-down:** $s$ can **`read`** $o$ if and only if $i(s) \leq i(o)$.
- **No write-up:** $s$ can **`write`** $o$ if and only if $i(o) \leq i(s)$.
- **No invoke-up:** $s_1$ can **`execute`** $s_2$ if and only if $i(s_2) \leq i(s_1)$.

Biba models can also handle dynamic integrity levels, where the level of a subject reduces if it accesses an object at a lower level (in other words it has *got dirty*). The *low-watermark* policies are:

- **No write-up:** $s$ can **`write`** $o$ if and only if $i(o) \leq i(s)$.
- **Subject lowers:** if $s$ **`reads`** $o$ then $i'(s) = \min(i(s), i(o))$.
- **No invoke-up:** $s_1$ can **`execute`** $s_2$ if and only if $i(s_2) \leq i(s_1)$.

Finally, we have a *ring* policy,

- **All read:** $s$ can **`read`** $o$ regardless.
- **No write-up:** $s$ can **`write`** $o$ if and only if $i(o) \leq i(s)$.
- **No invoke-up:** $s_1$ can **`execute`** $s_2$ if and only if $i(s_2) \leq i(s_1)$.

Each of these policies have an application in some area.

## 5.4 Clark-Wilson model for integrity

The Clark-Wilson [CW87] model attempts to model the trustworthiness of data and programs, providing assurance for *integrity* services. In this model, the principal concern is with well-formed transactions operating over the system. The transactions are defined through certification rules. The Clark-Wilson model has the following terminology:

| Term | Definition |
|---|---|
| **CDI** | **C**onstrained **D**ata **I**tem (data subject to control) |
| **UDI** | **U**nconstrained **D**ata **I**tem (data not subject to control) |
| **IVP** | **I**ntegrity **V**erification **P**rocedures (for testing correct CDIs) |
| **TP** | **T**ransformation **P**rocedures (for transforming the system) |

In the textbook, Section 6.4.1, various certification and enforcement rules are given. Together these provide a (perhaps) less formal model than Bell-LaPadula, Biba, but the model has wider application than just simple access control.

## 5.5 Information flow

We may also more abstractly model some security policies by considering the flow of information in a system. We can use *entropy* to formalize this. In this context, we can establish quantitative results about information flow in a system, rather than just making absolute assertions[3]. In the textbook we have a definition of information flow based on the conditional entropy $H(x \mid y)$ of some $x$ given $y$:

**Definition 16-1.** The command sequence $c$ causes a flow of information from $x$ to $y'$ if $H(x \mid y') < H(x \mid y)$. If $y$ does not exist in $s$ then $H(x \mid y) = H(x)$.

We can use this to detect *implicit* flows of information, not just explicit ones in which we directly modify an object. Consider the example on page 409 of the textbook:

```
if x=1 then
   y := 0
else
   y := 1;
```

After this code segment, we can determine if $x = 1$ from $y'$ even though we do not ever assign $y'$ directly from some function of $x$. In other words we have an implicit flow of information from $x$ to $y'$. We may do this in a formal manner by considering the entropy of $x$. If the likelihood of $x = 1$ is $0.5$, then $H(x) = 1$. We can also deduce that $H(x \mid y') = 0$, and so

$$H(x \mid y') < H(x \mid y) = H(x) = 1$$

and information is flowing from $x$ to $y'$. The paper [Den76] gives some background.

## 5.6 Confinement and covert channels

The confinement problem is one of preventing a system from leaking (possibly partial) information. Sometimes a system can have an unexpected path of transmission of data, termed a covert channel, and through the use of this covert channel information may be leaked either by a malicious program, or by accident.

Consider the set of permissions on a file. An unscrupulous program could modify these permissions cyclically to transmit a very-low data-rate message to another unscrupulous program. We categorize covert channels into two:

1. **Storage channels:** using the presence or absence of objects
2. **Timing channels:** the speed of events

We can attempt to identify covert channels by building a shared resource matrix, determining which processes can read and write which resources.

[3]For example, "*System X reveals no more than 25% of the input values*".

## 5.7 Summary of topics

In this section, we introduced the following topics:

- Mathematical modelling for security
- Information flow
- Indirect channels of information flow

---

## Supplemental questions for chapter 5

1. Use the Bell-LaPadula model to specify the controls for a security policy that allows a General working in a high security area to make public announcements, and allows lower security operatives to report secrets up into the same security area. Your model must be secure.
2. Textbook, Exercise 5.8.2.
3. Textbook, Exercise 5.8.7.
4. Textbook, Exercise 6.8.2.
5. Textbook, Exercise 6.8.8.

---

## Further study

- *Access control matrix model*, textbook sections 2.1, 2.2.
- *Bell-LaPadula model*, textbook sections 5.1, 5.2, also the paper [MP97] at http://80-ieeexplore.ieee.org.libproxy1.nus.edu.sg/xpl/tocresult.jsp?isNumber=13172.
- *Biba model*, textbook sections 6.1, 6.2.
- *Clark-Wilson model*, textbook section 6.4, also the paper [CW87] at http://www.isg.rhul.ac.uk/msc/teaching/ic4/clark_wilson.pdf.
- *Information flow*, textbook sections 16.1, 16.2, also the paper [Den76] at http://www.cosc.georgetown.edu/˜denning/infosec/lattice76.pdf.
- *Confinement*, textbook sections 17.1, 17.2, 17.3.

---