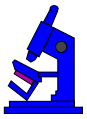
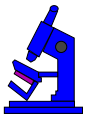



Chapter 5

Lecture 5 - Preliminaries


Last session

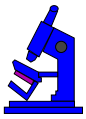
- Physical preliminaries
- Entropy



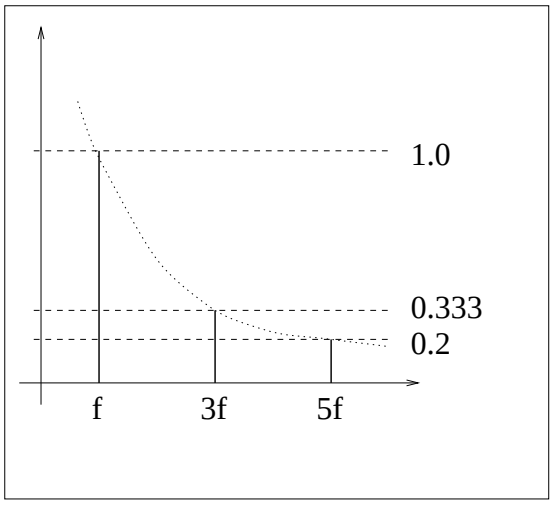
This session

- Channel properties
- Entropy
- Models

Hugh's bigger mistakes...

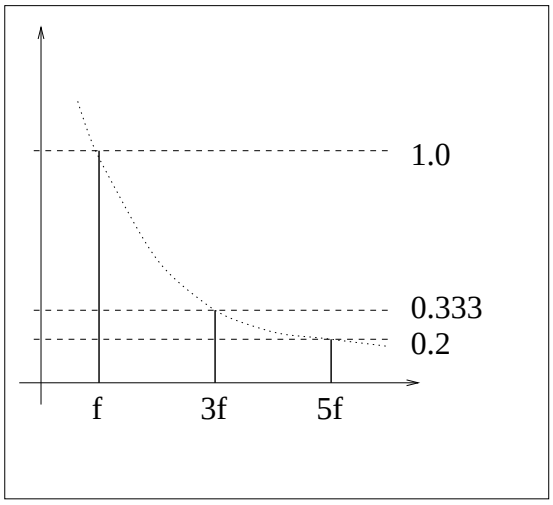


Incorrect

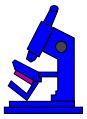


The plot is **frequency vs time**. (Frequency domain).

Correct



The plot is **amplitude vs frequency**. (Frequency domain).



Incorrect

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

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

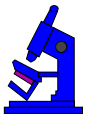
Correct

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

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

Note that

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



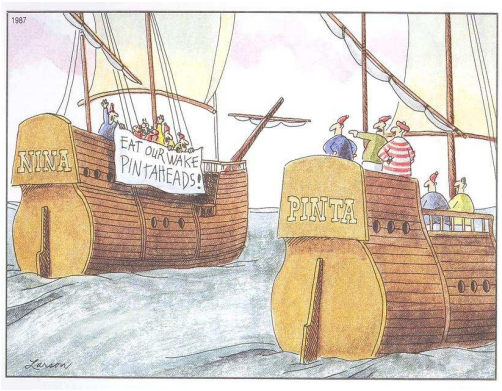
Page 46 of notes

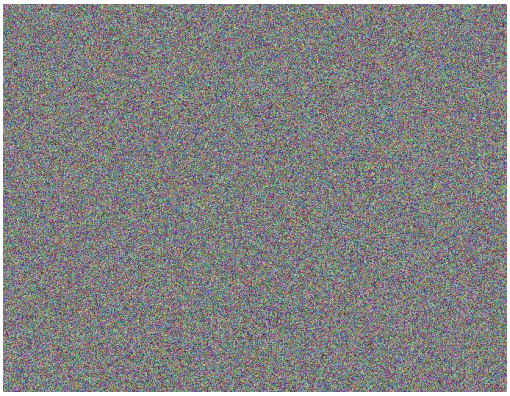
The first two equations that begin $H(X)$ should begin with $L(X)$.

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

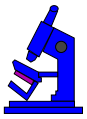
1/2 of data through correctly...

Received data is corrupted 50% of the time:





Before After



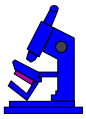
Convolution

Applet to do convolution:

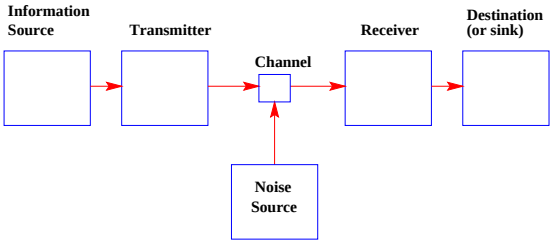
http://www.cs.brown.edu/exploratories/freeSoftware/repository/edu/brown/cs/exploratories/applets/convolution/convolution_java_browser.html

This session

- Channel properties
- Entropy
- Security models



Shannon and Nyquist



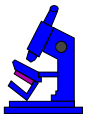
Maximum BPS = $W \log_2(1 + \frac{S}{N})$ bits/sec

Shannon and Nyquist 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

The maximum data rate over a limited bandwidth (W) channel with V discrete levels is:

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

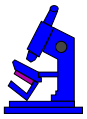
For example, two-Level data cannot be transmitted over the telephone network faster than 6,000 BPS, because the *bandwidth* of the telephone channel is only about 3,000Hz.

Nyquist 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).



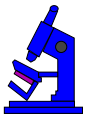
This session

- Channel properties
- Entropy
- Security models

Maximum entropy

In practical systems, signals rarely have maximum entropy, and we can do better - there may be methods to compress the data[2].

[2] **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.



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?*".

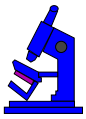
The Huffman encoding minimizes the bit length given the frequency of occurence of each symbol[3]. The resultant bit string in the best case will be the length predicted from the calculation of the source entropy.

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

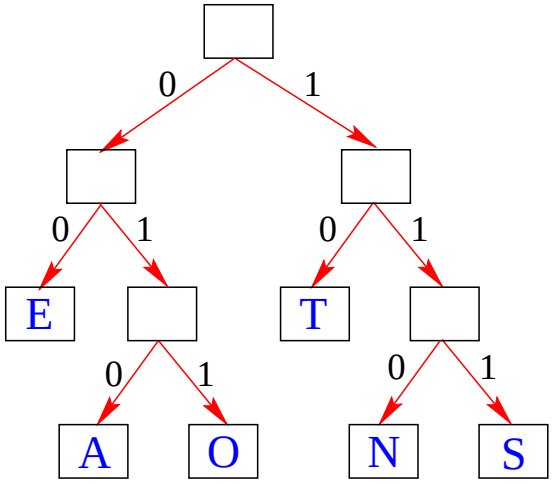
Huffman encoding

✔ How can we get knowledge about the frequency of (say) the letters in the English language?

(answer) - we read snapple bottle tops...



Huffman encoding



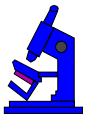
Less common characters use longer bit strings.

Huffman encoding

Our algorithm for encoding is simple - we calculate the tree encoding knowing the frequency of each letter:

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

To decode, traverse the tree taking a left or right path according to the bit. The leaf has our symbol.



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).

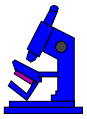
V42.bis does not have this property - it uses a large dictionary, and will not try to compress an already compressed stream.

MNP5

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.



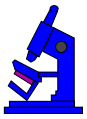
Adaptive frequency encoding

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

$\frac{3}{4}$ of our codewords are *larger* than they would be if we did not use this encoding scheme

Further study

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



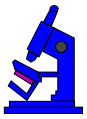
This session

- Channel properties
- Entropy
- Security models

Preliminaries - security models

Definition: a range of formal policies for specifying the security of a system in terms of a (mathematical) model.

- ✔ access control matrix
- ✔ Bell-LaPadula
- ✔ Biba
- ✔ Clark-Wilson



Security model

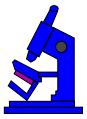
- ✔ Have a model
- ✔ Determine properties
- ✔ Verify implementations

Access control matrix

Rows of the matrix are subjects, columns are objects:

		Objects			
		f_1	f_2	f_3	f_4
Subjects	s_1	`read` `execute`	`execute`		
	s_2	`write`		`read`	`execute`
	s_3	`read`	`write`		`execute`
	s_4		`read`	`write`	`read`

s_4 cannot read f_1. But subjects may collude...



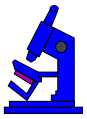
Bell-LaPadula, confidentiality

- ✔ Military style to assure *confidentiality* services.
- ✔ Security levels in a (total) ordering formalizing a policy which restricts information flow from a higher security level to a lower security level.
- ✔ Lower-level subjects from accessing higher-level objects.
- ✔ Section 5.2 in textbook

Bell-LaPadula, levels

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

where $T > S > C > U$. Access operations visualized using an access control matrix, and are drawn from {**`read`**, **`write`**}.



BLP security property

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.

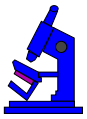
This single property is insufficient to ensure the restriction we need for the security policy.

BLP Trojan Horse property

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?



BLP extended

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.

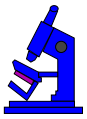
✔ Lattice - chapter 30 in textbook

BLP extended

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.



BLP extended

The new properties are:

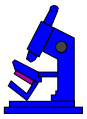
- **No read-up-2:** s can **`read`** o if and only if s **dom** o, and s has **`read`** access in the access control matrix.
- **No write-down-2:** s can **`write`** o if and only if o **dom** s, and s has **`write`** access in the access control matrix.

BLP security

✔ 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 example

From textbook, p128:

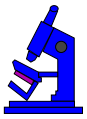
✔ DG UNIX uses access controls and BLP-like behaviour

BLP limits

BLP is a static model, not providing techniques for changing access rights or security levels[4], 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.

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

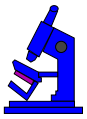


Biba model, integrity

- ✔ Trustworthiness of data and programs - assurance for *integrity* services.
- ✔ Levels like **`clean`** or **`dirty`** (in reference to database entries).
- ✔ Biba model (chapter 6.2) is a kind of *dual* for Bell-LaPadula. *integrity* vs *confidentiality*.

Biba levels

- ✔ The integrity levels $\mathcal{I}$ are ordered as for the security levels
- ✔ Function $i : \mathcal{O} \rightarrow \mathcal{I}$ ($i : \mathcal{S} \rightarrow \mathcal{I}$) which returns the integrity level of an object (subject).



Biba properties

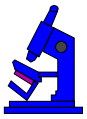
The properties/rules for the *main* (static) Biba model are:

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

Biba - dynamic

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



Biba - ring

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 $i(s_2) \leq i(s_1)$.

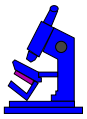
Each of these policies have an application in some area. - Example in textbook, p155 (LOCUS OS)

Clark-Wilson, integrity

Transactions 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)



Information flow (Chapter 16)

- ✔ 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[5].

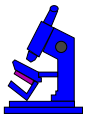
[5] For example, "*System X reveals no more than 25% of the input values*".

Information flow

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.



Information flow

Consider the example on page 409 of the textbook:

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

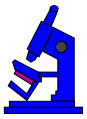
After this code segment, we can determine if $x = 1$ from y' even though we do not ever assign y' directly from some function of x. In other words we have an implicit flow of information from x to y'.

Information flow

Formal treatment 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'. Paper gives some background.



Confinement and covert channels

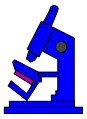
The confinement problem is one of preventing a system from leaking (possibly partial) information.

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.

Confinement and covert channels

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.



Confinement and covert channels

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.