University of Portsmouth
BSc (Hons) Computer Science
Third Year

Advanced Networks (ADNET)

M21276

September 2025 - January 2026

20 Credits

Thomas Boxall
`thomas.boxall1@myport.ac.uk`

Contents

Page 1

Lecture - Signal Encoding Techniques

2025-09-29 11:00 Asim

There is a deck of slides on Moodle introducing this module's structure & assessments, etc.

1.1 Introduction to Concepts

1.1.1 Computer Networks

Definitions

Computer Network A system that connects two or more computing devices for transmitting and sharing information (data)

There are a number of different activities which can be done on a network:

- Watching Videos
- Playing Games
- Sending and Receiving Messages (including not just text)
- Paying Bills

The core function of the network is to exchange data between interconnected devices.

1.1.2 Data & Signals

Data is the information which is transmitted between devices on the network. The type of the data depends on the context and may include:

- Video
- Audio
- Text
- Images

For data to be able to travel on the network - it has to be converted into digital or analog format. Once in either of these formats, the resultant data is known as *signals*.

Definitions

Signal Electromagnetic Waves that carry data

Analog Signal are signals which vary smoothly over time and have no fixed point to change at and don't have fixed levels

Discreet Signal are signals which maintain constant level for some time then then change to another constant level

Digital Signal are signals which have only two levels - one high to indicate on, or 1, and one low to indicate off, or 0

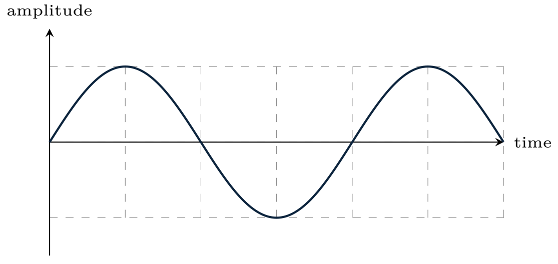

Figure 1.1: Analog Signal

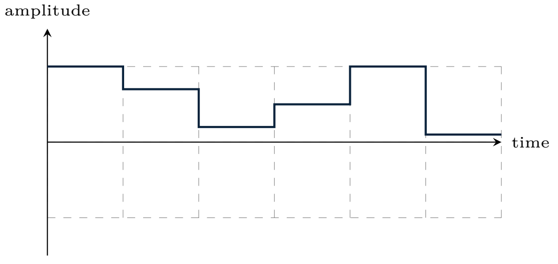

Figure 1.2: Discrete Signal

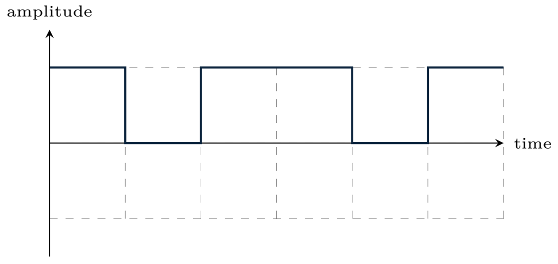

Figure 1.3: Digital Signal

1.2 Data Communications

Within a Data Communications Network - the data will convert between Digital and Analog data at various points. A *modem* may be used to complete this conversion.

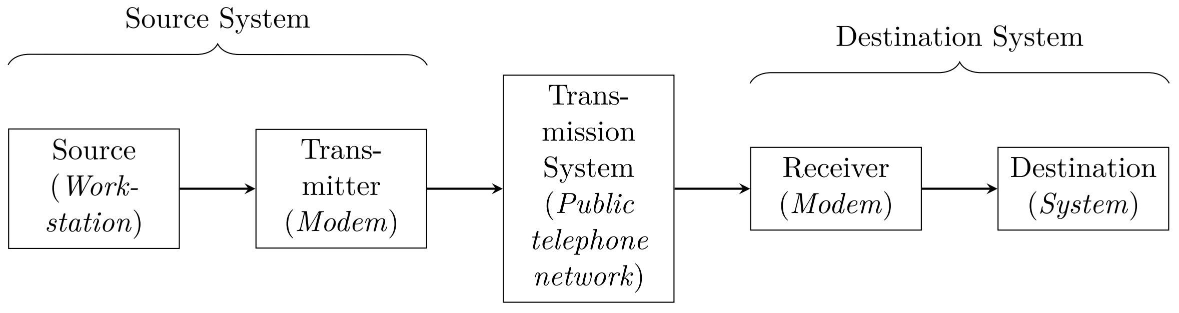

Figure 1.4: Block Diagram of Example Data Communications Signal Chain

In the above diagram, the *workstation* provides the *modem* with a digital signal. The modem then converts this digital signal into an analog signal which can be transmitted across the *Public Telephone Network* that uses analog signals. The receiving *modem* converts the signal to a digital format which the *server* receives.

In saying this, however, there are some devices which still work entirely on analog or digital signals.

For example, the analog telephone network including the switching and transmission is entirely analog. There is also the inverse whereby there are entirely digital signals are processed.

1.3 Digital-To-Digital Signal Encoding

Digital signals are transmitted such that an individual value is transmitted for a defined period of time called the *bit duration*. The *bit duration* is known to both the sender and receiver, allowing the receiver to correctly interpret the transmitted signal which the sender will always transmit at the beginning of the bit duration. We assume the sender and receiver are synchronised and therefore the clock or is not transmitted.

The move between two different defined voltages within the transmission is called the *transition*.

As digital signals are discrete (meaning there are defined, absolute data levels) encoding a digital signal as another digital signal is considerably simpler as we don't have to convert one data level to another.

1.3.1 Nonreturn to Zero Level (NRZ-L)

This method only has two levels of data transmitted. It works through mapping the high data level (1) and low data level (0) to a signal level:

- 0 - represented by a high level signal
- 1 - represented by a low level signal

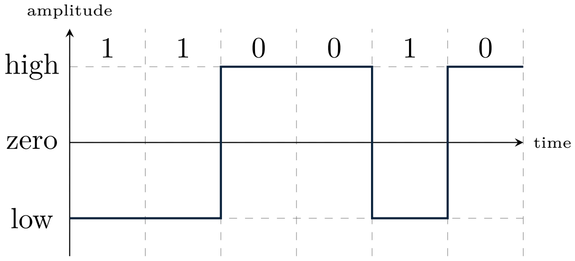

Figure 1.5: Example of NRZ-L Encoding using 110010

1.3.2 Nonreturn to Zero Inverted (NRZ-I)

This method is similar to NRZ-L in that it uses two levels of data transmitted: high and low. However it uses the transitions to define the data being transmitted. Looking at the bit following a transition point, even if there is no transition present:

- Where there is a transition (regardless of 1 to 0, or 0 to 1): the following signal bit is transmitted as is high
- Where there is not a transition (regardless of 1 to 1, or 0 to 0): the following signal bit is transmitted as low

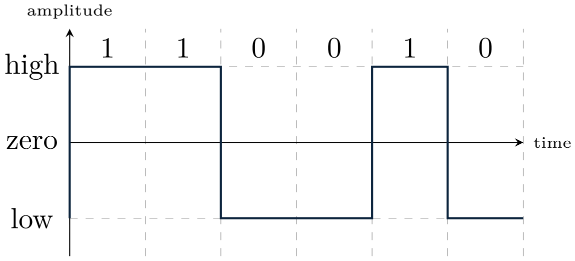

Figure 1.6: Example of NRZ-I Encoding using 110010

1.3.3 Bipolar-AMI

This has three signal levels: high, zero, and low. It works through setting the signal based on the bit to be transitioned to:

- 0 - represented by transmitting zero
- 1 - represented by alternating high and low values, regardless of it there is a 0 value in the middle

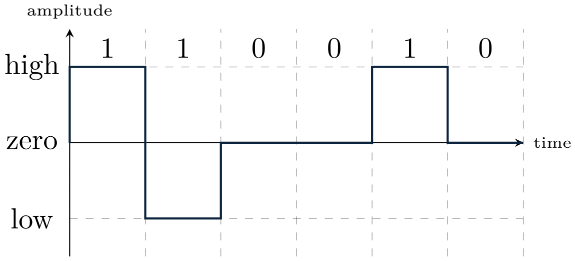

Figure 1.7: Example of Bipolar-AMI Encoding using 110010

1.3.4 Pseudoternary

This has three signal levels: high, zero, low. It works through setting the signal based on the bit to be transmitted:

- 0 - represented by alternating high and low values, regardless of if there is a 1 value in the middle of them
- 1 - represented by transmitting zero value

This is the inverse of Bipolar-AMI.

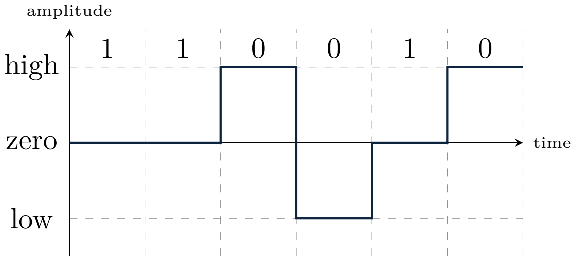

Figure 1.8: Example of Pseudoternary Encoding using 110010

1.3.5 Manchester

This has two signal levels: high and low. It works through having up-to two transitions per interval:

- 0 - represented by a transition from high to low in the middle of the interval (which for successive zeros will require a low-to-high transition at the start of the interval)
- 1 - represented by a transition from low to high in the middle of the interval (which for successive ones will require a high-to-low transition at the start of the interval)

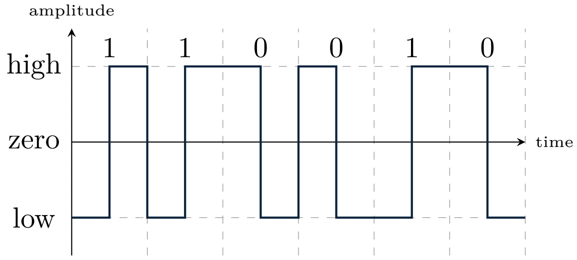

Figure 1.9: Example of Manchester Encoding using 110010

1.3.6 Differential Manchester

This has two signal levels: high and low. It works by always transitioning in the middle of the interval and then examining the transition at the beginning of the interval:

- 0 - represented by a transition at the beginning of the interval
- 1 - represented by no transition at the beginning of the interval

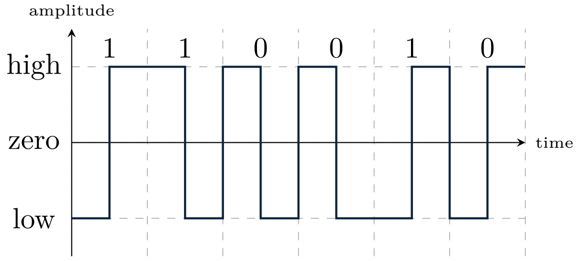

Figure 1.10: Example of Differential Manchester Encoding using 110010

1.3.7 Multi-Level Transmit 3 (MLT-3)

This uses three different signal levels: low, zero and high. It works by cycling through these states based on the bit to be transmitted:

- 0 - remain on the current signal level (regardless of signal level value)
- 1 - move to the next state in the cycle of signal levels (high - zero - low - zero repeat)

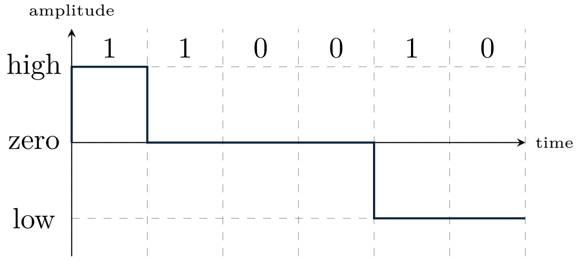

Figure 1.11: Example of MLT-3 Encoding using 110010

1.4 Electromagnetic Waves

Electromagnetic waves are the digital representation of an analog signal. They are considered to be smooth as they don't have fixed values. The key properties of any EM wave are as follows: Amplitude, Phase, Wavelength & Frequency.

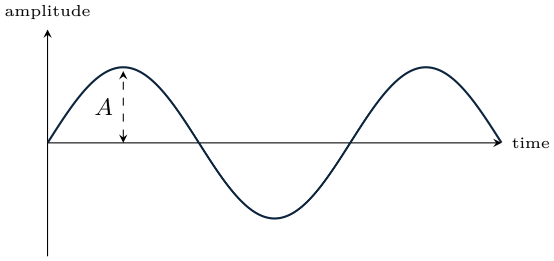

Figure 1.12: Electromagnetic Wave showing Amplitude (A)

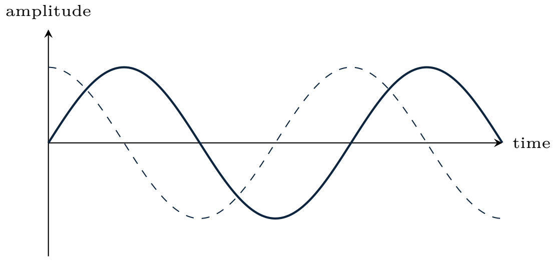

Figure 1.13: Electromagnetic Wave showing Phase (dashed)

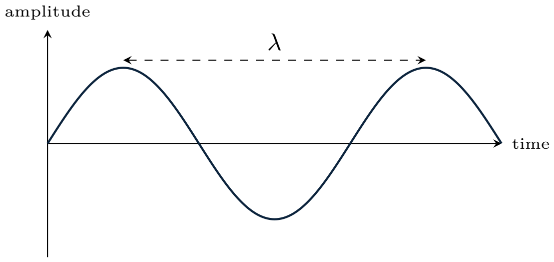

Figure 1.14: Electromagnetic Wave showing Wavelength (λ)

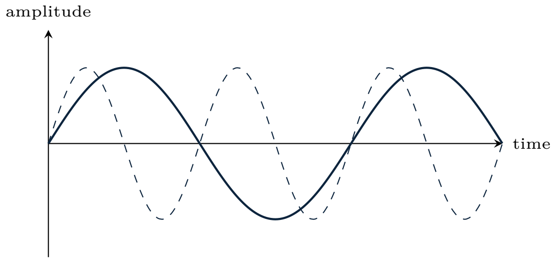

Figure 1.15: Electromagnetic Wave showing increased frequency (dashed)

1.4.1 Carrier Waves & Modulation

Definitions

Carrier Wave a continuous, periodic waveform that carries no information

A *carrier wave* is modified by an information-bearing signal to convey information. The modification can be by either changing: it's amplitude, frequency, phase, or some combination of the three. The process of modifying a carrier wave is called *modulation*.

1.5 Digital Data, Analog Signals

There are many examples of where digital data has to be transmitted through an analog medium. The most well-known of which being the public telephone network. This is designed to transmit voices, within the frequency of 300 to 3400 Hz; therefore a problem is presented when a digital device is connected to the network. This problem is overcome by connecting the digital device to a Modem (Modulator-Demodulator) which converts digital data to analog signals and the other way around.

There are a number of different methods which can be used to convert digital data onto a analog signal. These methods employ a carrier wave for the digital data to be modulated onto. For the subsequent examples - it's assumed the carrier wave is a sine wave.

1.5.1 Amplitude-Shift Keying

Amplitude-Shift Keying (ASK) modulation works by modulating the digital signal onto the amplitude of the carrier wave. Meaning that the amplitude of the carrier wave is increased for a digital 0 and decreased for a digital 1.

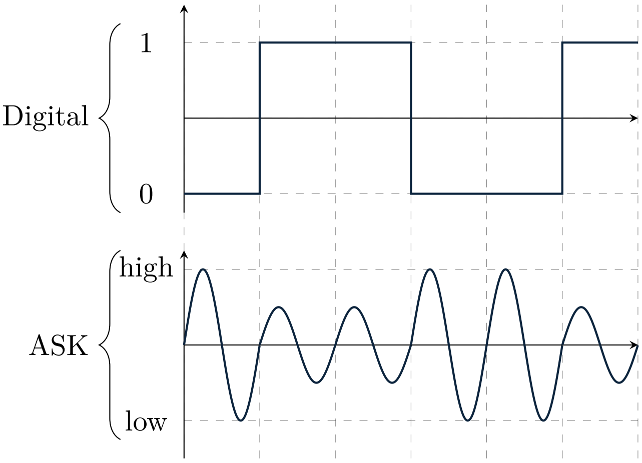

Figure 1.16: Example of ASK Modulation

1.5.2 Frequency-Shift Keying

Frequency-Shift Keying (FSK) modulation works by modulating the digital signal onto the frequency of the carrier wave. This means that the frequency of the carrier wave is increased for a digital 1 and decreased for a digital 0.

The sender and receiver may use different frequencies to allow full-duplex transmissions on the same channel. It is less susceptible to errors than ASK modulation is. FSK can be used for higher frequencies (3 - 30 MHz), radio and Local Area Network transmissions. It can also support Multiple Levels in MFSK.

The *bandwidth* of the transmission is the difference between the frequency of the high frequency (representing a 1) and the low frequency (representing a 0). Different frequency carrier waves will be used for sending and receiving on the same line.

The most common form of FSK is Binary FSK (BFSK) in which the two binary values are represented by two different frequencies near the carrier frequency:

- 0 - represented by $A\cos(2\pi f_1 t)$
- 1 - represented by $A\cos(2\pi f_2 t)$

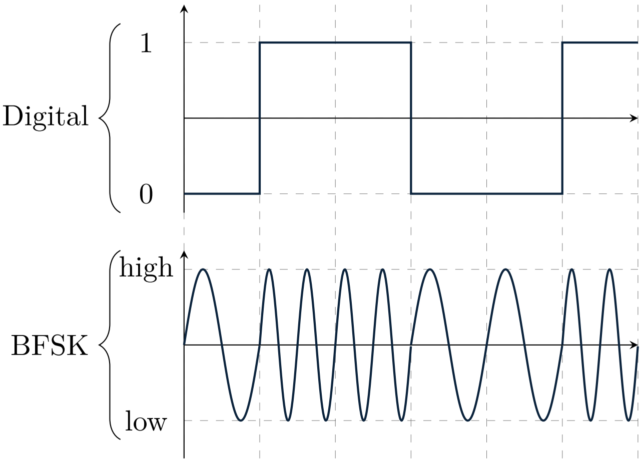

Figure 1.17: Example of BFSK Modulation

1.5.3 Phase Shift Keying

Phase Shift Keying (PSK) Modulation works by modulating the digital signal onto the phase of the carrier wave. This means that the phase of the carrier wave is adjusted to represent digital 0 and digital 1, respectively. There are two different types of PSK studied here.

1.5.3.1 Binary Phase Shift Keying

Binary PSK (BPSK) works by using two phases to represent two different binary digits (0 and 1) and shifting between them:

- 0 - represented by the sine wave $A\cos(2\pi ft + 180)$ which equals the sine wave $-A\cos(2\pi ft)$
- 1 - represented by the sine wave $A\cos(2\pi ft)$

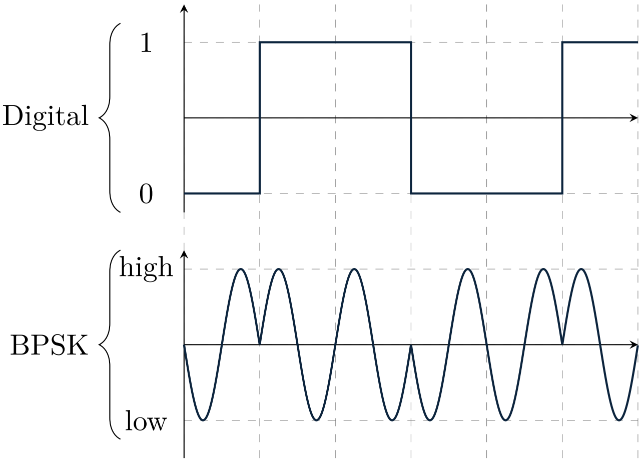

Figure 1.18: Example of BPSK Modulation

1.5.3.2 Differential Phase Shift Keying

Differential PSK (DPSK) works by referencing the previous bit in the current bit transmitted. DPSK removes the need for an accurate local oscillator phase at the receiver which is matched with the transmitter, because so long as the preceding phase is received correctly - the phase reference is accurate.

- 0 - send a signal similar to the previous one
- 1 - send a signal with phase shift as compared to the previous one

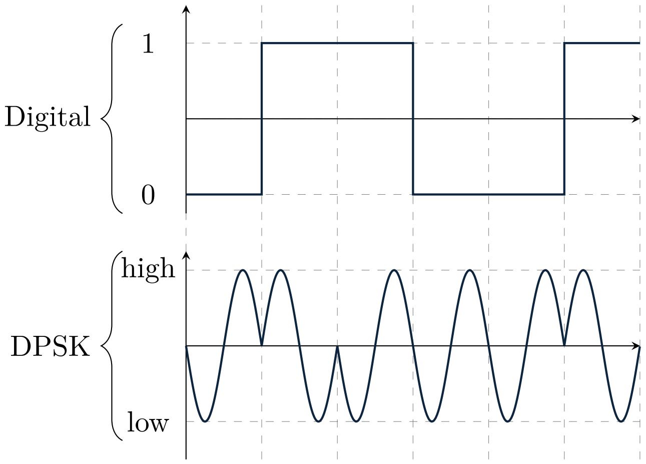

Figure 1.19: Example of DPSK Modulation

Page 2

Lecture - Spread Spectrum and Walsh Codes

📅 2025-10-06 🕚 11:00 Asim

2.1 Communication Channels

There are two different types of Communications Channels (also sometimes referred to as *Communication Media* or *Transmission Media*).

Definitions

Guided Media Wired (Bounded) Media (i.e Twisted Pair, Coaxial, Fibre Optic)

Unguided Media Wireless Media (i.e. Microwave, Radio Wave. Cellular, Infrared, Satellite)

Within Guided Media, the electromagnetic waves are guided along a physical path. The medium can be considered to be *point-to-point* if it provides a direct link between two devices and those two devices are the only devices sharing the medium.

Unguided media is the opposite - where the electromagnetic waves are not guided through any physical containment, rather they transmit through air, vacuum or seawater.

The term *Direct Link* is used to refer to a transmission path where the signal is transmitted directly from sender to receiver without any intermediate devices. This can apply to both guided and unguided media. A *multipoint guided configuration* is a configuration such that more than two devices share the same medium.

A transmission may be simplex, half duplex or full duplex.

Definitions

Simplex a transmission in which signals are only transmitted in one direction; one station is the transmitter and the other is the receiver.

Half-Duplex a transmission in which both stations can transmit and receive but only one can transmit at one time therefore

Full-Duplex a transmission in which both stations can transmit and receive at the same time; which requires a medium is required for signals to be transmitted in both directions at the same time

In a full-duplex transmission system there may sometimes be an overlap in the frequency ranges used for each direction of transmission. This is sometimes acceptable and sometimes not - depending on the application. In any case, the overlap would be at the very edges of the frequency range; however this would still cause some interference.

There is a diagram detailing the frequencies used for different unguided transmissions available both in the slides on Moodle & in the textbook on page 109.

2.2 Interference and Noise

Often with transmissions - our signal may be interrupted in some way. This will alter the signal being transmitted which could change the data it represents - therefore garbling the resultant wave. Often this interference will come from *noise*.

Definitions

Interference The combination of two or more electromagnetic waveforms to form a resultant wave in which the displacement is either reinforced or cancelled

Noise An unwanted signal which is combined with desired signal

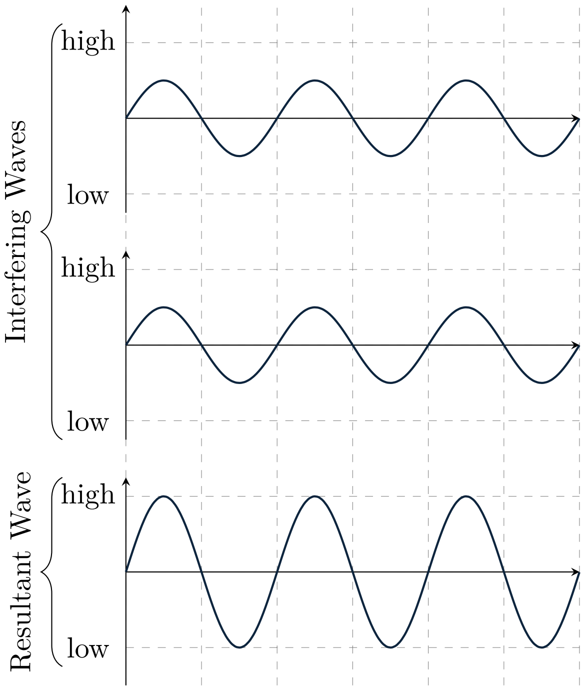

Figure 2.1: Constructive Interference

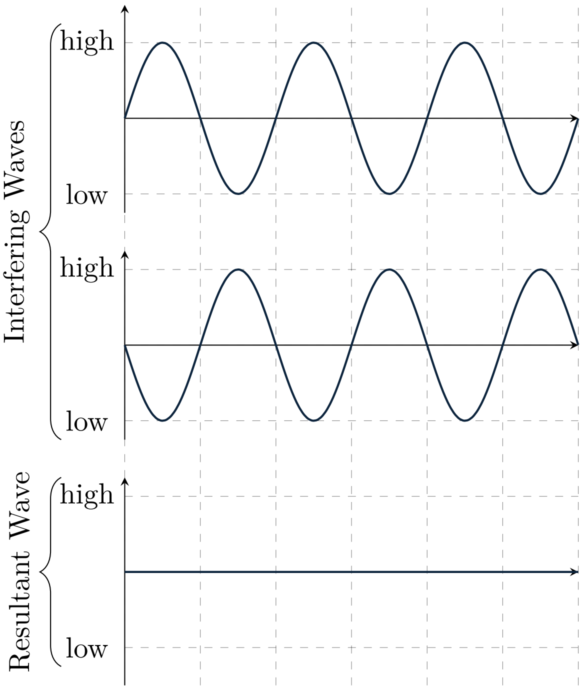

Figure 2.2: Destructive Interference

As we can see in the above figure, where we have two waves which are of the same phase interfering with each other - they will *interfere constructively* to increase the amplitude of the resultant wave. However where two waves in opposite phase interfere with each other - they will *interfere destructively* to effectively cancel each other our and the resultant wave has no amplitude.

For any data transmission, the received signal will consist of the transmitted signal, modified by the various distortions used by the transmission system, plus additional unwanted signals that are inserted somewhere between transmission and reception (which is referred to as *noise*).

Noise can be divided into four categories:

- Thermal Noise
- Intermodulation Noise

- Crosstalk
- Impulse Noise

2.3 Basic Definitions

Definitions

Data Rate The rate, in bits per second (bps), at which data can be communicated

Error The reception of a 1 where 0 was transmitted, or the reception of a 0 when a 1 was transmitted

Error Rate The rate at which errors occur

Frequency Bandwdith The difference between the upper and lower frequencies in a continuous band of frequencies

Channel Capacity The maximum rate at which information can be transmitted through a communication channel

Signal to Noise Ration (SNR) The ratio of the signal power to the noise power, measured in Decibels $10 \log_{10} \frac{\text{signal power}}{\text{noise power}}$

The *Channel Capacity* can be calculated using:

$$C = 2B \log_2 M$$

Where C is the channel capacity; B is the bandwidth; M is the signal or voltage levels.

The Nyquist Bandwidth Theory stipulates that if the rate of signal transmission is $2B$ then a signal with frequencies no greater than B is sufficient to carry the signal rate. The converse is also true. This limitation is due to the effect of intersymbol interference, which is produced by delay distortion. This is in essence based on the Nyquist Sampling Theorem (flashback to A-Level Electronics).

From this we can see that permitting all other things being equal, when we double the bandwidth - we double the error rate. The error rate then only gets worse as we increase the data rate because a higher data rate will mean the bits are shorter so more bits are affected by a given pattern of noise. Mathematician *Claude Shannon* tied these into a formula:

$$C = B \log_2(1 + SNR)$$

Where C is the capacity of the channel in bps and B is the bandwidth of the channel in Hertz.

2.4 Multiplexing

Definitions

Multiplexing A technique that allows the simultaneous transmission of multiple signals through the same channel or link; several signals are combined into a single composite signal

2.4.1 Frequency Division Multiplexing

In Frequency Division Multiplexing (FDM), the different message signals are modulated onto different carrier frequencies. This then means the signals being transmitted are separate from each other in the frequency domain. These modulated signals are then combined together to form the composite signal and this signal is sent over the shared medium or channel. To avoid the interference between the different message signals, a guard band is also kept between the message signals.

On the transmitter end, the signals to be transmitted are modulated onto the different carrier frequencies. Then on the receiver end - the frequencies first pass through a Band Pass Filter (definition not required for this module), then through the demodulator, then finally through a low pass filter (again, definition not needed in this module). The band pass filter is used to separate the specific signal from the composite signal, and the low pass filter is used to filter out the higher frequencies introduced as part of the FDM process.

2.4.2 Time Division Multiplexing

In Time Division Multiplexing (TDM), the channel is divided into several time slots, and each signal allocated during it's time slot. As a result - several signals share the channel without interfering with each other.

2.5 Spread Spectrum

When transmitting our analog signals (whether these originated as digital or analog), we can spread the signal over a wider bandwidth to avoid jamming and frequency interception.

Spread Spectrum is a technique used by military and intelligence applications which is also used in Wireless and Cordless networks. There are a number of different techniques which can be used, we will explore 3 of them.

Definitions

Pseudorandom Noise (PN) is a deterministic sequence of bits which satisfies one or more of the standard tests for statistical randomness while being repeatable after a period

2.5.1 Frequency Hopping Spread Spectrum

In Frequency Hopping Spread Spectrum (FHSS), the signal is broadcast over a number of different radio frequencies, and the frequency used is changed at a fixed intervals which are generally extremely short (i.e. $1ms$). The receiver will hop between the different frequencies used in sync with the transmitter.

If the transmission is compromised, then the attacker would only hear unintelligible blips of the transmission. It would also thwart attempts to jam the signal as the attacker would only be able to block a few bits of the signal.

FHSS transmission systems tend to work with the binary data being fed into a modulator using a digital-to-analog encoding scheme (for example FSK or BPSK). The resultant signal is centred in a frequency. A PN source serves as an index into a table of frequencies; this is the spreading code. Each k bits of the PN source specifies one of the carrier frequencies. At each pre-agreed interval, a new carrier frequency is selected. This frequency is then modulated by the signal produced from the initial modulator to produce a new signal with the same shape, but now centred on the selected carrier frequency.

On reception - the spread spectrum signal is demodulated using the same sequence of frequencies derived from the PN source, and then demodulated to produce the output data.

2.5.2 Direct Sequence Spread Spectrum

Direct Sequence Spread Spectrum (DSSS) works by encoding a single bit to be transmitted (i.e. 1) as a multi-but sequence (i.e. 0110) using a spreading code. The *spreading code* spreads the signal across a wider frequency band in direct proportion to the number of bits used. Which means that a 4-bit spreading code spreads 1-bit of signal across a frequency band which is 4 times greater than a 1-bit spreading code. Of course, it doesn't have to be a 4-bit spreading code; it could be 10-bit or 20-bit.

A common method for encoding DSS is to combine the digital data input signal with a *Pseudorandom Noise (PN)* sequence (the individual bits within are called *chips*). The combined output signal is then referred to as a *chip sequence*.

In reality, this encoding process works by combining the digital data input signal with the Chip Sequence using an Exclusive Or (XOR) operation.

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 1 \oplus 0 &= 1 \\ 1 \oplus 1 &= 0 \end{aligned}$$

This produces a *combination bit stream* which has the data rate of the spreading code sequence, so therefore has a higher bandwidth than the information stream.

Example: DSSS

If we take the first bit we want to transmit, 0, and the first for bits of PN sequence 0110; we then perform the XOR operation for each of them:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 0 \oplus 1 &= 1 \\ 0 \oplus 1 &= 1 \\ 0 \oplus 0 &= 0 \end{aligned}$$

This gives us the transmitted spread-sequence representing our first bit: 0110. This process is then continued for all bits to be transmitted.

On the receiving end, the receiver has the same PN sequence so they can perform an XOR function on the received signal and PN sequence:

$$\begin{aligned} 0 \oplus 0 &= 0 \\ 1 \oplus 1 &= 0 \\ 1 \oplus 1 &= 0 \\ 0 \oplus 0 &= 0 \end{aligned}$$

The un-spread single bit output is then the output of the XOR. In this case, that would be 0. Obviously, this decoding process is repeated for as many bits as needed.

There is an alternative representation of DSSS using a graph to represent the different signals in the Slides available on Moodle.

2.5.3 Code Division Multiple Access Spread Spectrum

Code Division Multiple Access (CDMA) is a multiplexing technique used with spread spectrum.

To understand how CDMA works, we first have to understand how *Walsh Code* works. Walsh Code is a matrix from which codes can be taken by reading the rows of the matrix. The Walsh matrix starts with the general structure:

$$W_1 = (-1)$$

We can then double n to provide larger matrices, for the events in which we need longer sequences of chip codes.

$$W_{2n} = W_2 = \begin{pmatrix} W_1 & W_1 \\ W_1 & \overline{W_1} \end{pmatrix} = \begin{pmatrix} -1 & -1 \\ -1 & +1 \end{pmatrix}$$

Obviously, this can be taken further to see a 4 × 4 matrix which is a common size we will interact with in this module.

$$W_{2n} = W_4 = \begin{pmatrix} W_2 & W_2 \\ W_2 & \overline{W_2} \end{pmatrix} = \begin{pmatrix} -1 & -1 & -1 & -1 \\ -1 & +1 & -1 & +1 \\ -1 & -1 & +1 & +1 \\ -1 & +1 & +1 & -1 \end{pmatrix}$$

CDMA works by taking the data signal, with a bit-rate R, and selecting a unique code of n chips according to the Walsh Matrix; for example *row 1 for a; row 2 for b; etc.* Then if a the user, k, sends a 1 - the transmitter sends the chip code c_k; alternatively if the user k sends a 0 - the transmitter sends the chip code $\overline{c_k}$ (which may also be represented in this module as c'_k).

The signal to be transmitted is the built by summing the "columns" of chip codes (i.e. all first indexes summed for A, B, C & D; all second indexes summed for A, B, C & D; etc; in a situation where 4 signals are being transmitted together). The receiver then decodes this signal by performing inner-product-multiplication for the user k. This works by them taking the original Chip Code (not the inverted chip where a -1 has been transmitted) and multiplying each element with the corresponding element in the received signal (i.e. first element of chip code multiplied with first received bit, etc). The output from these multiplications are then summed together, and where the result is the number of bits in the chip code, n: a 1 was transmitted; and where the result is $-n$: a 0 was transmitted.

Example: CDMA

If we take the following users to have the following Chip Codes:

- User A Chip Codes: $(-1, -1, -1, -1)$
- User B Chip Codes: $(-1, +1, -1, +1)$
- User C Chip Codes: $(-1, -1, +1, +1)$
- User D Chip Codes: $(-1, +1, +1, -1)$

Case 1

User A sends 1, user B sends 1, user C sends 1, and user D sends 0 (represented as -1).

We sum their chip codes, for A, B, and C - these are as specified above; however for D we need to invert the chip codes because D is transmitting a 0 (-1).

$$\begin{aligned} A + B + C + D' &= (-1, -1, -1, -1) + (-1, +1, -1, +1) + (-1, -1, +1, +1) + (+1, -1, -1, +1) \\ &= (-2, -2, -2, 2) \end{aligned}$$

The receiver will receive this and perform an inner product multiplication using the Chip Code of A.

$$\begin{aligned} f &= (-2,-2,-2,2) \times (-1,-1,-1,-1) \\ &= (-2 \times -1) + (-2 \times -1) + (-2 \times -1) + (2 \times -1) \\ &= 2+2+2-2 \\ &= 4 \end{aligned}$$

Which therefore confirms that A is sending a bit 1.

Case 2

User A sends 0 (-1), user B sends 1, user C sends 1, and user D sends 0 (-1).

We sum their chip codes, this time inverting A and D because they're transmitting 0s.

$$\begin{aligned} A+B+C+D' &= (+1,+1,+1,+1) + (-1,+1,-1,+1) + (-1,-1,+1,+1) + (+1,-1,-1,+1) \\ &= (0,0,0,4) \end{aligned}$$

Then we can find what the receiver decodes using A's chip code.:

$$\begin{aligned} f &= (0,0,0,4) \times (-1,-1,-1,-1,) \\ &= (0 \times -1) + (0 \times -1) + (0 \times -1) + (4 \times -1) \\ &= 0+0+0+-4 \\ &= -4 \end{aligned}$$

As this results to $-n$ (remembering n is the length of the users chip codes) - we know that A transforms to bit 0.

Page 3

Lecture - Security

📅 2025-10-13 🕚 11:00 Asim

3.1 Security Requirements

Within *Network Security* there are three important requirements. The three requirements work together to ensure that network transmissions are of a good security standard.

Definitions

Confidentiality Ensuring that only authorised parties can read the message; therefore restricting unauthorised third parties from accessing it

Integrity Ensuring that the transmitted message is exactly the same as the received message and that only authorised parties can modify, delete, or reply to the message

Availability Ensuring the message is available on demand to authorised parties only

3.2 Attack Types

There are two main types of attack used.

Definitions

Passive Attacks Adversaries analyse the traffic or read message content

Active Attacks Adversaries modify the message content, deny the service to a request, replay to a message and masquerade (pretend to be a different entity)

We can attempt to overcome the threats using cryptography to encrypt the message on the transmitter's side, and then ensure that only the designated recipient can decode the message.

The message we want to encrypt, known as the *plaintext*, is transformed by a function that is parametrised by a *key*. The output of the encryption process, known as the *ciphertext*, is then transmitted. If this ciphertext is intercepted by the adversary, they are unable to directly interpret it as they do not know what the key used in the encryption process was. However, it may be possible for the adversary to crack the cipher depending on the encryption method used.

There is a standard notation for the plaintext, ciphertext and keys. $C = E_K(P)$ is used to represent the encryption, E of the plaintext, P using the key, K which results in the ciphertext, C. The decryption is defined as $P = D_K(C)$ where the plaintext, P, is the result of applying the decryption, D, with the key, K, on the ciphertext, C.

3.3 Encryption Techniques

There are a number of Encryption Techniques used commonly.

3.3.1 Substitution cipher

In a *substitution cipher*, each letter or group of letters is directly replaced by another letter or group of letters which disguises it.

A well-known example of this is the Ceaser Cipher.

Example: Substitution Cipher

If we take the following substitution:

```
Plaintext:   a b c d e f h g i j k l m n o p q r s t u v w x y z
Ciphertext:  Q W E R T Y U I O P A S D F G H J K L Z X C V B N M
```

Figure 3.1: Substitution Matrix

We can see how we convert from plaintext to ciphertext.

For example, taking the plaintext *attack* - we get the ciphertext *QZZQEA*; or taking the plaintext *london*, we get the Ciphertext *SGFRGF*.

Using a standard alphabet with 26 characters, assuming we only use lowercase letters, then there are 4×10^{26} possible combinations. Which on a computer with one million CPU cores - can take up to 10,000 years to crack.

3.3.2 Transposition Cipher

In a *transposition cipher*, we take a key and use this key to design a grid which we populate with our plaintext, working row-by-row. We then read the ciphertext from the grid reading column-by-column in alphabetical order of the key's letter.

Example: Transposition Cipher

If we take the plaintext *pleasetransferonemilliondollarstomyswissbankaccountsixtwotwo* and the key *MEGABUCK*.

We start by writing out the key at the top of the grid, and assigning a number to each column. In our case, the numbers are assigned in ascending order of the alphabet.

```
M E G A B U C K
7 4 5 1 2 8 3 6
```

Figure 3.2: Transposition Cipher Matrix with Key and Number shown

We can then take our plaintext and write that into the matrix working down the rows. We pad the end of the string to fill out the last row entirely, in our case we're using the start characters of the alphabet.

```
M E G A B U C K
7 4 5 1 2 8 3 6
p l e a s e t r
a n s f e r o n
e m i l l i o n
d o l l a r s t
o m y s w i s s
b a n k a c c o
u n t s i x t w
o t w o a b c d
```

Figure 3.3: Transposition Cipher Matrix with plaintext added

To get the ciphertext out of the matrix, we read from the matrix vertically in order of the column numbers: *afllsksoselawaiatoossctclnmomantesilyntwrnntsowdpaedobuoeriricxb*.

To decode the message, a similar process is followed: setup the matrix by writing the key, and the column numbers; then write in the ciphertext in columns based on the order of the numbers; then read rows to obtain the plaintext.

3.4 Symmetric Encryption

Symmetric Encryption is an encryption method in which the sender and receiver both use the same key to encode and decode the message. This is where it's alternative name, *shared key encryption* comes from.

Symmetric Encryption works through taking the plaintext input which is then encrypted using the key, K. The encryption algorithm will perform various substitutions or transformations on the plaintext, which will be key-dependent. This means that the same plaintext, with two different keys and the same encryption algorithm will produce two different ciphertexts. The ciphertext is then transmitted to the receiver. The receiver then decrypts the message using that pre-shared-key, K. The decryption algorithm is effectively the encryption algorithm run in reverse. This gives them the plaintext.

To denote a shared public key between A and B, we may see the key notated as K_{AB+}.

We can see an example of a Symmetric Encryption in the DES encryption method.

3.4.1 DES

The *Data Encryption Standard*, or DES, is a symmetric-key encryption algorithm which was developed in the 1970s and has been hugely influential in modern-day cryptography.

The plaintext input for DES is always 64-bits in length, and the key is always 56-bits. If the total plaintext needing to be encrypted using DES is longer than 64-bits then it gets chunked into 64-bit chunks so then the DES algorithm can process its 64-bit blocks.

The DES algorithm works by taking the 56-bit key and generating 16 sub-keys from it. These 16 sub-keys are each used for their own round of encryption, therefore the DES algorithm encrypts the plaintext 16 times through a sequence iterating on the output of the previous as the input to the next. The keys are used in reverse order - meaning K_{16} is used on the first iteration, K_{15} on the second, and so on until K_1 is used on the final iteration.

DES works through the 16 rounds of encryption following a modified *Feistel Network* structure. This works by taking the plaintext and splitting it into two substrings of equal length. One half of the key is passed to the F-Function (Feistel Function) which performs Expansion, Key-Mixing, Substitution and Permutation operations on the input and that round's key. The output from the F-Function then gets XOR'd with the other half of the input string. The next round is then prepared by feeding the output from the XOR operation into the next F-Function, and the input to the current round's F-Function to the next rounds XOR. There are 16 rounds of this in total. The final ciphertext is created by concatenating the output of the final XOR with the output of the penultimate XOR operation (which fed into the F-Function, the output from which fed into the input to the final XOR operation).

> There is a diagram explaining the Feistel Network in the slides on Moodle.

3.4.2 3DES

The *Triple Data Encryption Standard*, or 3DES, is a symmetric-key encryption algorithm which apples the DES algorithm three times to each block.

3DES works with three keys, K_1, K_2, and K_3. It will use K_1 on the first round of DES encrypting the data. Then the second round of DES will decrypt the data using K_2. The third and final round of DES will encrypt the data using K_3. The middle decryption doesn't return the data to the plaintext state, as $K_1 \neq K_2$.

Decryption follows the same encryption process but in reverse: decrypting with K_3, encrypting with K_2 before decrypting with K_1.

3.4.3 AES

The *Advanced Encryption Standard*, or AES, is a symmetric-key encryption algorithm which was designed to replace DES.

AES supports three different key lengths: 128-bit, 192-bit and 256-bit. Similarly to how DES works on 64-bits of data at a time, AES works on 128-bit blocks of data and will perform n rounds depending on the different key-length used. 128-bit keys get 10 rounds, 192-bit keys get 12 rounds and 256-bit keys get 14 rounds.

3.5 Asymmetric Encryption

Asymmetric Encryption is an encryption method where the sender and receiver both use different keys to encode and decode the message. This is where it's alternative name, *Public Key Encryption* comes form.

The sender encrypts the plaintext with the receivers public key to produce the ciphertext. The ciphertext is then transmitted over the network to the receiver. The receiver uses their private key to decrypt the message thus producing the plaintext.

The public key of the receiver is made available to anyone who wants to send the receiver a message. The public key of anyone is restricted to them and them alone - leaking of a public key would mean that an adversary could decode the message. Due to the receivers public keys being required to be able to send the receiver a message - senders will have as many public keys as the number of people they send messages to.

A major flaw with this model is that an adversary could intercept the public-key encoded message and stop it from reaching the receiver. Whilst they couldn't read that message, they could spoof that message by sending their own payload encoded with the receivers public key. The receiver would receive this and decode it.

They keys may be notated as follows:

- Senders public key: K_{A+}

- Senders private key: K_{A-}
- Receivers public key: K_{B+}
- Receivers private Key: K_{B-}

We can see an example of the Asymmetric Encryption in the RSA algorithm.

3.5.1 RSA

The *Rivest-Shamir-Adlerman* cryptosystem, or RSA, is a public-key cryptosystem which was developed in 1973 at GHCQ.

RSA works through taking two large prime numbers where their product is used to form the public key (along with other values used). The original prime numbers are kept secret. Then anyone can use the public key to encrypt the message. The message is transformed using mathematical operations which involve the public key. On the receiving end - only the corresponding private key can decrypt the message. The decryption process involves the original prime factors to reverse the encryption.

The RSA process can be formalised below:

1. Chose two large primes, p and q.
2. $n = p \times q$ and $z = (q - 1) \times (q - 1)$
3. Choose a number relatively prime to z and call it d
4. Find e such that $e \times d = 1 \mod z$
5. To encrypt a message, P, compute $C = P^e (\mod n)$
6. To decrypt the ciphertext, C, compute $P = C^d (\mod n)$
7. The public key is (e, n) and the private key is (d, n).

Example: RSA

We can see the above steps with real numbers substituted in below:

1. Let $p = 3$ and $q = 11$
2. Therefore $n = 33$ and $z = 20$
3. Let $d = 7$
4. We can calculate e using $e \times 7 = \mod 20$ which sets $e = 3$
5. We can encrypt our message, for example 19 by performing $C = 19^3 (\mod 33) = 28$
6. We can decrypt the ciphertext, in this example 28 by performing $P = 28^7 (\mod 33) = 19$
7. This means we know that the public key was $(3, 33)$ and the private key was $(7, 33)$.

There is a more fleshed-out example of the RSA process for the string “SUZANNE” in the slides on Moodle.

3.6 Ensuring Confidentiality

As we have already seen - we need to ensure confidentiality within the transmission to ensure that adversaries cannot intercept and access the message.

3.6.1 Within a Symmetric Encryption Network

As both the sender and the receiver share the same private key (K_{AB}), the adversary cannot decode the message as they do not have the key.

The sender encodes the plain text message into a cipher text $C = D_{K_{AB}}(M)$. Only the receiver, who has the key, is able to decrypt it using $P = D_{K_{AB}}(E_{K_{AB}}(M))$

3.6.2 Within an Asymmetric Encryption Network

The adversary is not able to read the transmitted message.

Both the sender (A) and receiver (B) compute their own private keys (K_{A-}, K_{B-}) and public keys (K_{A+}, K_{B+}). The public keys are advertised on the network.

When a transmission occurs - the sender will use the receivers public key to encrypt the message $C = E_{K_{B-}}(P)$. Therefore only the receiver can decode the message using their private key $P = D_{K_{B+}}(C)$, alternatively shown as $P = D_{K_{B+}}(E_{K_{B-}}(P))$.

3.7 Ensuring Integrity

We have to ensure integrity of our transmitted messages to ensure that the user who says they transmitted the message are actually the person who transmitted the message. This is for in the event that an adversary intercepts and replaces a message transmitted from Sender to Receiver with a message sent from adversary to Receiver pretending it was from the original sender.

3.7.1 Digital Signatures

The use of a *Digital Signature*, or *Reverse Public Key* is a method of ensuring integrity which removes confidentiality from the encryption process.

Digital Signatures work by the sender encrypting the message using their private key (K_{A-}). The receiver then uses the public key of the sender (K_{A+}) to decrypt the received message. This means that then the sender and only the sender can encrypt messages, however anyone on the network who has the senders public key is also able to decrypt the message. This therefore means that any adversaries on the network cannot change the contents of the message.

3.7.2 Message Digests

The *Message Digests* method works through producing a short fingerprint of the message, which is a hash of the message, represented as $H(M)$. The hash function works by encrypting a small block fo the message which is the function of the document. The hash of two different messages will always be different and it is impossible to find M from $H(M)$.

The has is encrypted with the senders private key, while the message itself is encrypted with the receivers public key. The hash therefore serves as a signature verifying the origin of the document, content and sequencing. This works because the adversary does not have the senders private key, so they cannot spoof a hash value.

3.8 Authentication

Authentication verifies that the messages came from an authentic source. Authentication can be achieved by conventional techniques:

- Private Key for authentication source
- Error detection and sequencing for message alteration

- Timestamp for messages delayed

Authentication is important because it shows that the message has come from a trusted, authentic source. Adversaries may record the messages and then re-play messages at a later time which they have captured. This would look like it's coming form the right person as they are correctly encrypted using the correct keys - however in reality it's a retransmission. This type of attack is called a *Play-Back Attack*

3.8.1 Symmetric Key Authentication

Authentication can be introduced to the symmetric-key encryption model through the following steps:

1. Sender sends their ID (i.e. a password) to receiver
2. Receiver responds with a one-time random number (called a *nonce*). The adversary as well as the sender could pick up the nonce R_B.
3. The sender encrypts using the private shared key, K_{AB} and sends this back to the receiver. The adversary may also pick this up, however they cannot decrypt because they don't have the key. The receiver decrypts the message and now knows they are talking to the sender.
4. The sender sends their own Nonce number, R_A to the receiver. The receiver encrypts this and sends it back to the sender. Only the sender can decrypt this, again because shared keys, and they now know they are talking to the sender.

This prevents playback because of the session established between the sender and the receiver. If the adversary tried to re-send a message later in time - the session would most likely have expired and so the receiver of that message would know to discard because it's not valid any more.

3.8.2 Public Key Authentication

In what can only be described as *the pinnacle of this lecture*, we will now see how Confidentiality, Integrity and Availability are maintained while transmitting secure messages across a communications network.

Public Key Authentication involves three parties: the sender, the receiver and the PKI (*Public Key Infrastructure*). The PKI is a centralised repository for public keys for devices on a given network. The Sender and Receiver's public keys are submitted to the PKI through an authentication-heavy trust-laden process (which we don't need to know about in this module).

The Public Key Authentication process is numbered below (now featuring A and B rather than sender and receiver):

1. A sends a request for B's public key to the PKI
2. PKI returns B's public key to the A (K_{B+})
3. Encrypted with the B's public key, A sends B their identification and a Nonce: $K_{B+}(A, R_A)$
4. B then requests A's public key from the PKI
5. The PKI sends B A's public key (K_{A+})
6. B now sends A their Nonce, A's Nonce, and a shared key to use for this session - all of which gets encrypted with A's public key: $K_{A+}(R_B, R_A, K_S)$
7. A receives this and decodes it using their private key, and then returns just B's nonce encrypted with the shared session key to B: $K_S(R_B)$

A authenticates B when they receive R_A in step 6. This proves it's a fresh message, not playback. B authenticates A when they receive R_B in step 7.

The adversary can fabricate a message at step 3, but as soon as A receives the wrong R_A back in step 6 - the session will be terminated.

The session key will be used as a private key for the duration of the communication; therefore achieving what we know to be the most secure form of communication. The short-lived session keys prevent playback because if a adversary attempts to play back a message, it's session will have expired so the message is discarded.

⚠ Add diagrams for generic model of secure network as this will be needed in the exam.