Name: Index: Reg. No.: Class: Date:

Chapter 2

Data Transmission

2.1 Types and methods of data transmission

Data transmission is the transfer of data from one digital device to another. This transfer occurs via point-to-point data streams or channels.

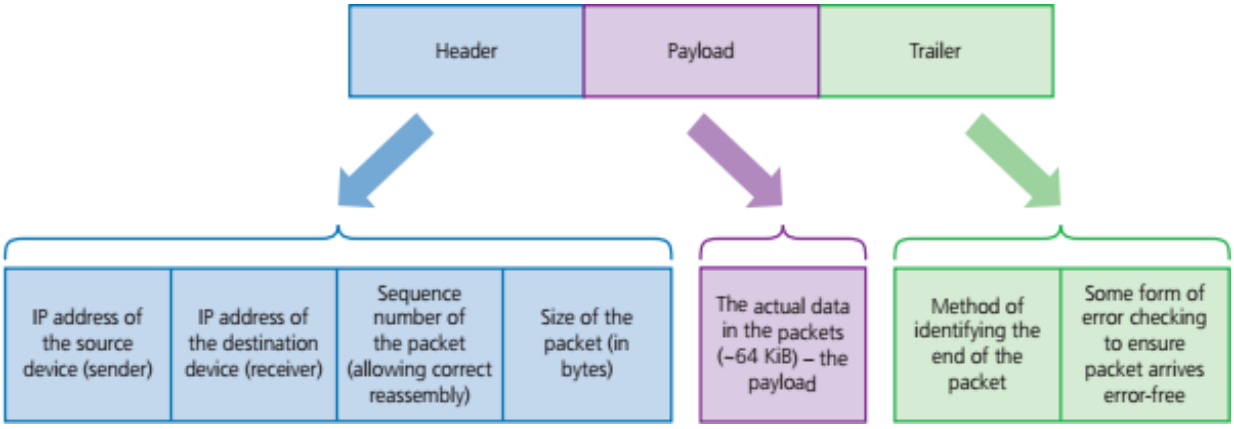
Packet structure

A typical packet is split up into:

» a packet header

» the payload

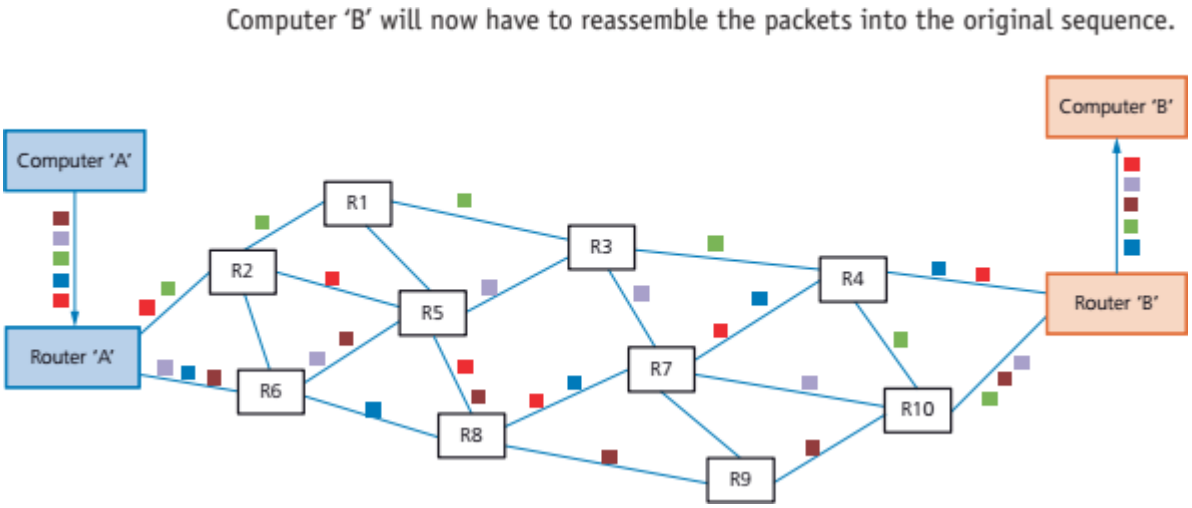
» a trailer.



Describe the structure of a data packet:

- it has a header that contains the sender address, destination address, packet number and size of the packet
- it has a payload that contains the actual data
- it has a trailer that contains the method of identifying the end of the packet and some form of error detection method.

Packet switching



Process of packet-switching

» Data is broken down into packets

» Each packet will follow its own **route**

» **Routers** will determine the route of each packet

» The **shortest possible path available** is always selected

» Packets may **arrive out of order** at destination

» Once the last packet has arrived, packets are **reordered**

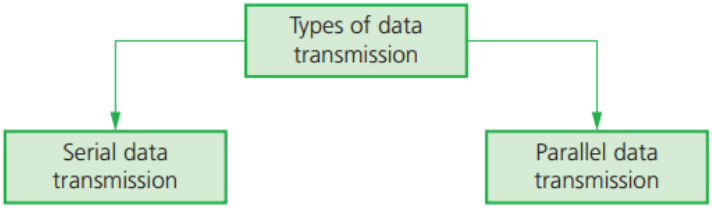
» If packet is missing/corrupted, it is requested

Advantages	Disadvantages
» there is no need to tie up a single communication line	» Packet may arrive in different order
» a high data transmission rate is possible	» packets can be lost and need to be re-sent
» it is relatively easy to expand package usage	» delay at the destination whilst the packets are being re-ordered.
» it is possible to overcome failed, busy or faulty lines by simply re-routing packets	

Purpose of a router in the packet-switching process.

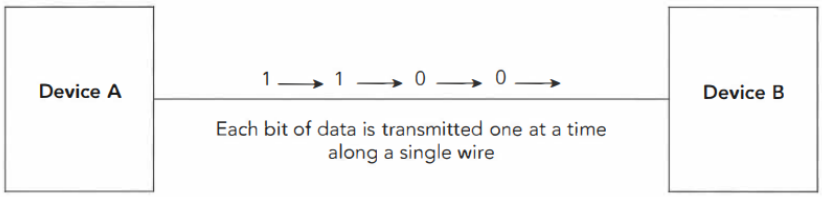
- Control the route the packet takes
- Send each packet towards its destination

2.1.2 Methods of Data Transmission



Serial Data Transmission

Serial data transmission occurs when data is sent **ONE BIT AT A TIME** over a **SINGLE WIRE/CHANNEL**. Bits are sent one after the other as a single stream.



Use of serial data transmission

- To send small amounts of data (due to slow rate) e.g. USB
- To send data over long distances e.g. telephone lines

Applications (examples) of serial data transmission

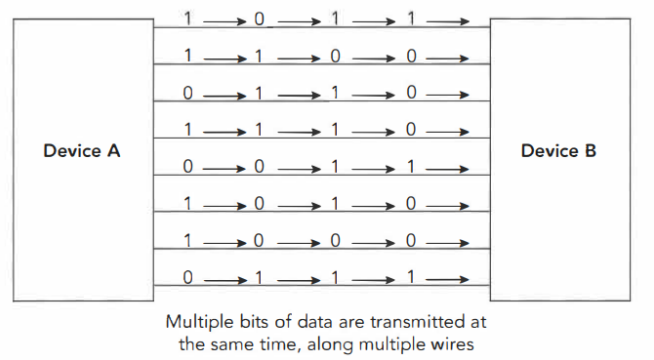
- Connecting computer to modem
- Connecting computer to printer via USB
- Connecting mouse to computer via USB
- Connecting keyboard to computer via USB

Advantages and disadvantages of serial data transmission.

Advantages	Disadvantages
Works well over long distances	Slow rate of data transmission
Data arrives fully synchronised (in the correct order)	Harder to detect errors
Less expensive (fewer hardware requirements)	

Parallel Data Transmission

Parallel data transmission occurs when **SEVERAL BITS OF DATA** (usually one byte) are sent down **SEVERAL CHANNELS/WIRES** all at the same time. Each channel/wire transmits one bit:



Multiple bits of data are transmitted at the same time, along multiple wires

Use of parallel data transmission

- To send large amounts of data
- To send data over short distance

Applications (examples) of parallel data transmission

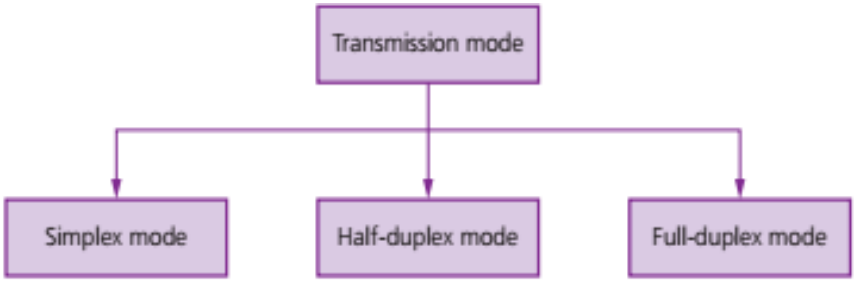
- Computer to printer using parallel ports
- Connection between internal circuits in computer

Advantages and disadvantages of parallel data transmission

Advantages	Disadvantages
Works well over short distances	More expensive to install
Faster data transmission rate	Can arrive out of sequence (skew)
Quick response time as data arrives fast	More interference, thus high chance of errors.

Simplex, Half-duplex and Full-duplex

Simplex, half-duplex and full-duplex transmission involve the **direction** in which the data is transmitted.



▲ **Figure 2.8** Transmission modes

Simplex data transmission

Simplex mode occurs when data can be sent in **ONE DIRECTION ONLY (uni-directional)**

Examples:

- Sending data from a computer to a printer (older printers).
- Sending data from a computer to a monitor
- Connecting keyboard to computer

Half-duplex data transmission

Half-duplex mode occurs when data is sent in **BOTH DIRECTIONS** but **NOT AT THE SAME TIME (bi-directional).**

Examples:

- Communication between walkie-talkies
- Old Hub-Based Ethernet Networks - Two computers cannot transmit at the same time (to avoid collision)

Full-duplex data transmission

Full-duplex mode occurs when data can be sent in **BOTH DIRECTIONS AT THE SAME TIME (bi-directional).**

Examples:

- Broadband internet connection.
- Video conferencing

KEEP IN MIND!

When a connection is created between two different devices it will either be a serial or parallel data connection. It will also be a simplex, half-duplex or full-duplex connection.

This means that a connection can be, for example, a serial simplex connection, or a parallel half-duplex connection, or possibly a serial full-duplex connection. You need to understand when each of the different methods of transmission are most suitable.

- **Serial-Simplex**

  Data is transmitted one bit at a time in a single direction on one wire

- **Serial-Half-duplex**

  Data can be transmitted in both directions on a single wire but only one bit at a time can be transmitted in one direction at a time

- **Serial-Full-duplex**

  Data can be transmitted in both directions at the same time on a single wire one bit at a time

- **Parallel-Simplex**

  Multiple wires transmit one bit at a time in one direction

- **Parallel-Half-duplex**

  Multiple wires send multiple bits of data in both directions but only one direction at a time

- **Parallel-Full-duplex**

  Multiple wires send multiple bits of data in both directions at the same time

Worked Example

A company has a website that is stored on a web server.
The company uses parallel half-duplex data transmission to transmit the data for the new videos to the web server.

Explain why parallel half-duplex data transmission is the most appropriate method. **[6]**

Answer

- Parallel would allow for the fastest transmission [1]
- as large amounts of data [1]
- can be uploaded and downloaded [1]
- but this does not have to be at the same time [1]
- Data is not required to travel a long distance [1]
- Therefore, skewing is not a problem [1]

2.2 Universal Serial Bus (USB)

- Universal Serial Bus (USB) is a form of data transmission.
- It uses serial data transmission means data are sent one at a time
- It is important to note that USB allows both half-duplex and full-duplex data transmission.

When a device is connected to a USB port the computer:

- Automatically detects that the device has been connected
- Automatically recognises and the appropriate device driver is loaded so that the device can communicate with the computer
- If the device is new, the computer will look for a matching device driver
- If one cannot be found then the user must download and install an appropriate driver manually

Advantages	Disadvantages
Can only be inserted in one way	Standard USB cable only supports a maximum cable length of 5 meters.
The speed of a USB connection is relatively high	Early USB versions may not always be supported
Universal connection, industry standard	Cannot supply high power for large devices (e.g., printers, monitors may need extra power)
Automatically detected and device drivers are automatically loaded up	
No need for external power source since cable supplies +5V power	
Easy to add more USB ports using USB hubs	
Backward compatible (supports older versions)	

Worked Example

Julia uses a USB connection to transfer data onto her USB flash memory drive.

One benefit of using a USB connection is that it is a universal connection.

(i) State two other benefits of using a USB connection [2]

(ii) Identify the type of data transmission used in a USB connection. [1]

Answer

(i) Any two of:

- It cannot be inserted incorrectly [1]
- Supports different transmission speeds [1]
- High speed transmission [1]
- Automatically detected [1]
- Powers the device for data transfer [1]

(ii) Serial

2.3 Methods of Error detection

Errors can occur during data transmission due to:

» **Interference:** which can cause data to be corrupted or even lost

» **Problems during packet switching:** this can lead to data loss

» **Skewing of data:** can cause data corruption if the bits arrive out of synchronisation

Methods for Detecting Errors *After Transmission*

Parity Checks

- A parity can be set to odd or even
- Sender & receiver make an agreement on the type of parity used
- The left-most bit in the byte is the parity bit, which makes sure the number of 1-bits is even/odd
- If odd parity is used then there must be an odd number of 1's in the byte, including the parity bit
- If even parity is used then there must be an even number of 1's in the byte, including the parity bit
- Receiver re-calculates the parity of the byte sent
- If there is a change in parity, there is a data transmission error
- If there is no change in parity, the data is correct

For example, consider the byte:

In this example, if the byte is using **even parity**, then the parity bit needs to be set to 0, since there is already an even number of 1-bits in the byte (four 1-bits).

We thus get:

In this example, if the byte is using **odd parity**, then the parity bit needs to be set to 1, since we need to have an odd number of 1-bits in the byte.

We thus get:

Describe situations when parity checks fails:

When **two** bits or an even number of bits are inter changed, parity check fails. Therefore, parity block checks needed.

Parity Byte & Block Check

- Parity blocks and parity bytes can be used to check an error has occurred and where the error is located
- Parity checks on their own do not pinpoint where errors in data exist
- A parity block consists of a block of data with the number of 1's totalled horizontally and vertically

Example 1

In this example, nine bytes of data have been transmitted. Agreement has been made that **even parity** will be used. Another byte, known as the **parity byte**, has also been sent. This byte consists entirely of the parity bits produced by the vertical parity check. The parity byte also indicates the end of the block of data.

Table 2.3 shows how the data arrived at the receiving end. It is now necessary to check the parity of each byte horizontally (bytes 1 to 9) and vertically (columns 1 to 8). Each row and column where the parity has changed from even to odd should be flagged:

▼ **Table 2.3** Parity block showing nine bytes and parity byte

	Parity bit	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8
Byte 1	1	1	1	1	0	1	1	0
Byte 2	1	0	0	1	0	1	0	1
Byte 3	0	1	1	1	1	1	1	0
Byte 4	1	0	0	0	0	0	1	0
Byte 5	0	1	1	0	1	0	0	1
Byte 6	1	0	0	0	1	0	0	0
Byte 7	1	0	1	0	1	1	1	1
Byte 8	0	0	0	1	1	0	1	0
Byte 9	0	0	0	1	0	0	1	0
Parity byte	1	1	0	1	0	0	0	1

A careful study of Table 2.3 shows the following:
» byte 8 (row 8) now has incorrect parity (there are three 1-bits)
» bit 5 (column 5) also now has incorrect parity (there are five 1-bits).

- The error could be fixed automatically or a retransmission request could be sent to the sender

Checksum

The checksum process is as follows:

» Sender calculates checksum using an algorithm from the block of data

» The checksum is then transmitted with the block of data

» Receiver re-calculates the checksum from the block of data

» Receiver compares the re-calculated checksum to the one sent by the sender

» If it doesn't match, the data is incorrect and a request is made to re-send the block of data.

» If it matches, the data is correct

For example:

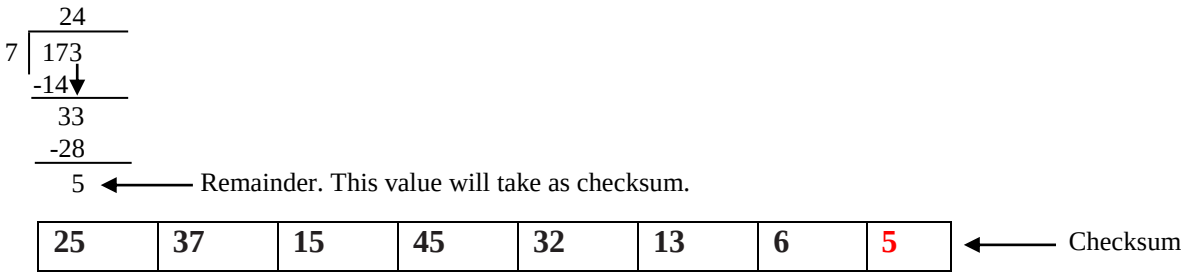
25	**37**	**15**	**45**	**32**	**13**	**6**	

Checksum

Algorithm:

1. Sum all the numbers
   25+37+15+45+32+13+6 = 173

2. Divide the sum by 7. Let the remainder of the division to checksum



What is the checksum value calculated from?
The actual data / payload that is being transmitted.

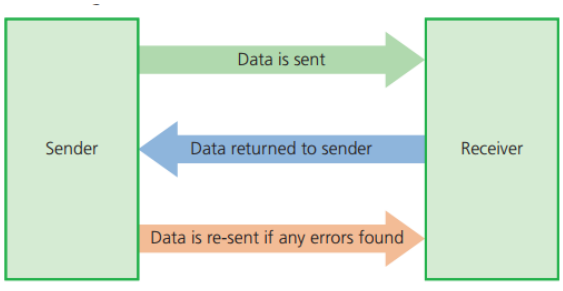
What happens to the checksum value when it has been calculated before transmission?
It is attached / appended to the data that is being transmitted, so that it is transmitted with the data.

Why would checksum values that do not match show that an error has occurred?
If the same calculation is performed on the data, it should get the same result. If the results are different, the data must have changed during transmission, meaning that errors must have been introduced.

Echo Check

The process of echo check:

- An echo checks involve transmitting the received data back to the sender
- Sender compares the received data to the original data
- If it doesn't match, there is a data transmission error, Sender resends data
- If it matches, the data is correct



▲ **Figure 2.14** Echo check diagram

Drawback of echo check

- Impossible to know whether the error occurred when sender sent the data or when receiver sent it back for checking
- Echo checks require a lot of extra data to be transmitted.

Methods for Detecting Errors *in Data Entry*

Check Digit

How check digit is used to detect errors?

- A calculation is performed using a standard algorithm
- A check digit is the last digit included in a code or sequence
- When the data is entered or scanned, the check digit is recalculated
- If the recalculated check digit matches the stored one, the data is correct.
- If the recalculated check digit does not match the stored one, the data is incorrect.

Examples of errors that a check digit can help to identify are:

1. **Incorrect digits entered**
   Eg: 4093 instead of 4193
2. **Transposition errors:** Two numbers change position
   Eg: 4093 instead of 4903
3. **Phonetic errors**
   Eg: 30 (thirty) instead of 13 (thirteen)
4. **Omitted or extra digits**
   Eg: 409 instead of 4093 or 40931 instead of 4093

Examples of where check digits can be used include:

- ISBN book numbers
- Barcodes

ISBN Book Numbers

- Each book has a unique ISBN number that identifies the book
- A standard ISBN number may be thirteen digits, for example, 978-0-340-98382-9
- The check digit value is the final digit (9 in this example).
- This number is chosen specifically so that when the algorithm is completed the result is a whole number (an integer) with no remainder parts
- A check digit algorithm is performed on the ISBN number and if the result is a whole number, then the ISBN is valid

Barcodes

- Barcodes consist of black and white lines which can be scanned using barcode scanners
- Barcode scanners shine a laser on the black and white lines which reflect light into the scanner
- The scanner reads the distance between these lines as numbers and can identify the item
- Barcodes also use a set of digits to uniquely identify each item
- The final digit on a barcode is usually the check digit, this can be used to validate and authenticate an item

Two common methods will be considered here:

- ISBN 13
- Modulo-11

▲ **Figure 2.15** Sample barcode (ISBN 13 code with check digit)

Example 1: ISBN 13

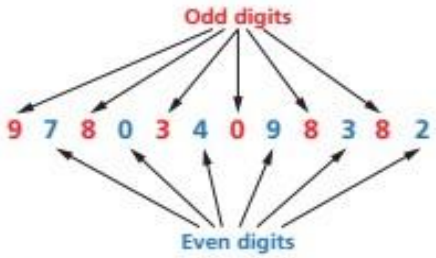
The check digit in ISBN 13 is the thirteenth digit in the number. We will now consider two different calculations. The first calculation is the generation of the check digit. The second calculation is a verification of the check digit (that is, a recalculation).

Calculation 1 – Generation of the check digit from the other 12 digits in a number

The following algorithm generates the check digit from the 12 other digits:

1. add all the odd numbered digits together
2. add all the even numbered digits together and multiply the result by 3
3. add the results from 1 and 2 together and divide by 10
4. take the remainder, if it is zero then use this value, otherwise subtract the remainder from 10 to find the check digit.

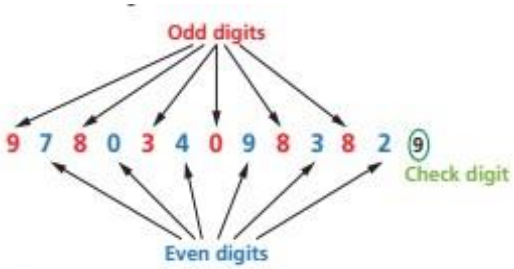
Using the ISBN **9 7 8 0 3 4 0 9 8 3 8 2** (note this is the same ISBN as in Figure 2.15):



▲ **Figure 2.16** ISBN (no check digit)

1. 9 + 8 + 3 + 0 + 8 + 8 = 36
2. 3 × (7 + 0 + 4 + 9 + 3 + 2) = 75
3. (36 + 75)/10 = 111/10 = 11 remainder 1
4. 10 – 1 = 9 the check digit

Verifying whether a check digit is correct using ISBN 13



▲ **Figure 2.17** ISBN (including the check digit)

Calculation 2 – Re-calculation of the check digit from the thirteen-digit number (which now includes the check digit)

To check that an ISBN 13-digit code is correct, including its check digit, a similar process is followed:

1 add all the odd numbered digits together, **including** the check digit
2 add all the even number of digits together and multiply the result by 3
3 add the results from 1 and 2 together and divide by 10
4 the number is correct if the remainder is zero.

Using the ISBN **9 7 8 0 3 4 0 9 8 3 8 2 9** (including its check digit) from Figure 2.17:

1 **9 + 8 + 3 + 0 + 8 + 8 + 9 = 45**
2 **3 × (7 + 0 + 4 + 9 + 3 + 2) = 75**
3 (45 + 75)/10 = 120/10 = 12 remainder 0
4 remainder is 0, therefore number is correct.

Example 2: Modulo-11

The modulo-11 method can have varying lengths of number which makes it suitable for many applications, such as product codes or VINs. The first calculation is the generation of the check digit. The second calculation is a verification of the check digit (that is, a recalculation).

Calculation 1 – Generation of the check digit from the other digits in a number

(In this example, we will assume the original number contained only 7 digits.)

The following algorithm generates the check digit from the other 7 digits:

1 each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3 or 2 starting from the left (weightings start from 8 since the number will become eight-digit when the check digit is added)
2 the digit is multiplied by its weighting and then each value is added to make a total
3 the total is divided by 11
4 the remainder is then subtracted from 11 to find the check digit (note if the remainder is 10 then the check digit 'X' is used).

The example to be used has the following seven-digit number:

1 7-digit number: 4 1 5 6 7 1 0

weighting values: 8 7 6 5 4 3 2

2 sum: (8 × 4) + (7 × 1) + (6 × 5) + (5 × 6) + (4 × 7) + (3 × 1) + (2 × 0)
= 32 + 7 + 30 + 30 + 28 + 3 + 0
total = 130
3 divide total by 11: 130/11 = 11 remainder 9
4 subtract remainder from 11: 11 – 9 = 2 (check digit)

So we end up with the following eight-digit: 4 1 5 6 7 1 0 2

Verifying whether a check digit is correct using Modulo-11

Calculation 2 – Re-calculation of the check digit from the eight-digit number (which now includes the check digit)

To check that the eight-digit number is correct, including its check digit, a similar process is followed:

1. each digit in the number is given a weighting of 8, 7, 6, 5, 4, 3, 2 or 1 starting from the left
2. the digit is multiplied by its weighting and then each value is added to make a total
3. the total is divided by 11
4. the number is correct if the remainder is zero

Using the 8-digit number: 4 1 5 6 7 1 0 2

1. weighting values: 8 7 6 5 4 3 2 1
2. sum: (8 × 4) + (7 × 1) + (6 × 5) + (5 × 6) + (4 × 7) + (3 × 1) + (2 × 0) + (1 × 2)
   = 32 + 7 + 30 + 30 + 28 + 3 + 0 + 2
   total = 132
3. divide total by 11: 132/11 = 12 remainder 0
4. remainder is 0, therefore number is correct

Automatic Repeat Requests (ARQs)

Process of ARQ

- ARQ uses **acknowledgement** and **timeout**
- If receiver does not detect errors in data transmission, it sends a **positive acknowledgement** to sender
- If an error is detected, it sends a **negative acknowledgement** to sender and requests to re-send
- Sender will re-send data if:
  a) it receives a request to re-send or
  b) **timeout** occurs without **acknowledgement** from receiver
- This automatically repeats until there is a **positive acknowledgement** or limit has been reached

Acknowledgement is a message that is sent from one device to another to indicate whether data is received correctly.

Timeout is a period of time that is set and used to wait for an acknowledgement to be received.

Worked Example

Explain how Automatic Repeat reQuests (ARQ) are used in data transmission and storage

Answer

- Set of rules for controlling error checking/detection // it's an error detection method // used to detect errors
- Uses acknowledgement and timeout
- Request is sent (with data) requiring acknowledgement
- Positive acknowledgment will be sent to sender if data received error free
- If no response/acknowledgment within certain time frame data package is resent
- When data received contains an error a negative acknowledgement is sent (automatically) to resend the data
- The resend request is repeatedly sent until packet is received error free/limit is reached/acknowledgement received

2.4 Symmetric and Asymmetric Encryption

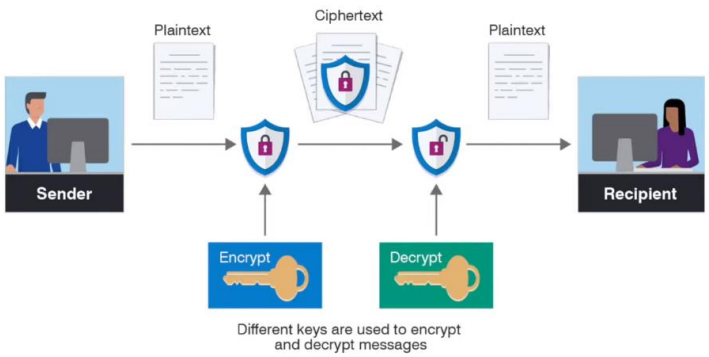
Encryption

Encryption is the process of encoding data or a message so that it cannot be understood by anyone other than its intended recipient.

The purpose of encryption

- If the data is intercepted it cannot be understood by hackers
- To help keep the data secure
- To make the data meaningless

Plain text and Cipher text



Encryption key: a type of algorithm that is used to encrypt data.

PLAIN TEXT: Data before encryption takes place it is called **plain text.**

CIPHER TEXT: An encryption algorithm, called an encryption key, is used to scramble the data and make it meaningless. This meaningless data or data after the encryption is called **cipher text**.

One important thing to note about encryption is that is does not **stop a hacker stealing the data** that is transmitted. It just means that the data that is stolen will be **meaningless**, as it will be encrypted and therefore scrambled. They will not understand any of the data in the file.

How the data are encrypted?

- An encryption algorithm is used to scramble data
- The original data is called the plain text
- A key is used to encrypt the data
- The key is applied to the plain text
- Plain text is encrypted into cypher text

Symmetric Encryption

A type of encryption that uses the same key to encrypt and decrypt data.

The process for symmetric encryption is:

- Data is encrypted using an **algorithm** that uses a key
- The same **encryption key (algorithm)** is used to **encrypt** and **decrypt** a message
- Data before encryption is **plain text**, data after encryption is **cypher text**
- The **cipher text** and the **encryption key** are sent separately to the receiving device.

Advantages of symmetric encryption:

- Fast and efficient than asymmetric encryption (it uses a simple mathematics)
- Can encrypt large file quickly
- Low resource like memory is using

Disadvantages of symmetric encryption:

- Difficult key sharing method
- Less secure for communication
- No identity check – Can't confirm who sent the data.

Asymmetric Encryption

A type of encryption that uses two different keys (public and private keys) to encrypt and decrypt data. Public key is used to encrypt data and a private key used to decrypt data.

The process for asymmetric encryption is:

- Data is encrypted using an **algorithm** that uses a key
- A matching pair of **public** and **private keys** is generated from receiver's computer
- A **public key** is made available to everybody, a **private key** is kept at a secure place by the owner of the public key
- Sender uses receiver's **public key** to **encrypt** the plain text
- The cipher text is transmitted to the receiver
- Receiver uses the matching **private key** to **decrypt** the cipher text

Advantages of asymmetric encryption:

- No need to share secret keys (private key)
- Can verify who sent the data.
- Less risk of leaking the data as it uses two keys.
- Can send same public key to many users

Disadvantages of asymmetric encryption:

- Slower than symmetric as it takes more time to encrypt and decrypt.
- Uses more resources as needs more processing power and memory.
- Not ideal for large data as its too slow for encrypting big files.

Difference between Symmetric and Asymmetric Key Encryption

Symmetric Key Encryption	Asymmetric Key Encryption
It only requires a single key for both encryption and decryption.	It requires two keys, a public key to encrypt and private key to decrypt.
Symmetric sends key with the data	Asymmetric does not send key with data
The encryption process is very fast.	The encryption process is slow.
It is used when a large amount of data needs to be transferred.	It is used to transfer small amount of data.
Security is lower as only one key is used for both encryption and decryption purposes.	Security is higher as two keys are used, one for encryption and the other for decryption.