# CAIE Computer Science IGCSE

# 2 - Data Transmission

Advanced Notes

# 2.1 Types and methods of data transmission

Data transmission involves sending data from one device to another. To do this efficiently, the data is often broken down into smaller, manageable chunks called packets.

## Packet structure

Each packet of data contains a packet header, payload and trailer.

| Packet component | Description |
|---|---|
| Header | This contains control information to route the packet correctly. It includes the destination address (typically an IP address, specifying where the packet is going), the packet number (A sequential identifier assigned to each packet, used to reorder them at the destination), and the originator's address (typically an IP address, specifying where the packet came from). |
| Payload | This is the actual data being sent. |
| Trailer | This includes error-checking information to ensure the data arrived intact. |

## Packet switching

Packet switching is the process of sending data over a network by splitting it into packets. It involves the following steps:

1. The data to be transmitted is broken down into several packets.
2. Each packet is sent individually across the network.
3. Routers control the route each packet takes, directing the packets along the most efficient path. Because of this, packets may take different routes to the same destination due to changes in the network.
4. Packets may arrive out of order. Once the last packet arrives, the packets are reordered, using their packet numbers.

If any packets are missing, they can be resent (depending on the protocols being used).

## Methods of data transmission

Data can be transmitted using different methods, each with its own advantages and disadvantages.

### Serial transmission

Bits are sent one after another along a single channel or wire.

| Advantages | Disadvantages |
|---|---|
| As the data is only sent over a single wire, it is less susceptible to skew (data arriving at slightly different times). | It can be slow to send bits over just one channel/wire. |
| Only one wire is needed, making this method less expensive. | |

### Parallel transmission

Sends multiple bits at once using multiple channels or wires.

| Advantages | Disadvantages |
|---|---|
| Faster data transmission, as several channels/wires are used. | More expensive, as several wires are needed. |
| | Susceptible to skew (data arriving at slightly different times), especially over long distances. |

### Simplex transmission

Data flows in one direction only (e.g. from a keyboard to a computer).

| Advantages | Disadvantages |
|---|---|
| No chance of data collision. | No way for the receiver to give feedback to the sender, such as to request data again if it wasn't received properly. |
| Simple to implement. | |

**Half-duplex transmission**

Data flows in both directions, but only one direction at a time (e.g. walkie-talkies).

| Advantages | Disadvantages |
|---|---|
| Allows two-way communication using fewer resources than full-duplex. | Slower than full-duplex due to switching time. |
| Simpler to design than full-duplex systems. | Cannot send and receive data simultaneously. |

**Full-duplex transmission**

Data flows in both directions at the same time (e.g. phone calls).

| Advantages | Disadvantages |
|---|---|
| Facilitates fast, efficient communication. | More complex and expensive to implement. |
| No need to wait for the line to be free. | Requires more bandwidth. |

## Universal Serial Bus (USB)

USB (Universal Serial Bus) is a standard interface used to connect peripheral devices (e.g. keyboards, mice, and flash drives) to a computer. It allows serial data transmission (one bit at a time).

### How USB transmits data

1. The data to be transmitted is broken down into several packets.
2. The host (computer) initiates communication and controls the flow of data. Each packet is sent serially (one bit at a time) along the USB cable to the connected device.
3. The connected device receives the packets and checks for errors using the error-checking information in the packets' trailer. If an error is found, the packet is re-sent.
4. The connected device reassembles the packets to complete the data transmission.

| Benefits | Drawbacks |
|---|---|
| USB is widely used and supported by many modern devices. | USB devices are prone to physical damage, such as bent connectors or broken housings, especially if dropped or mishandled. |
| USB allows plug-and-play functionality, making it easy to use. | USB cables have a limited cable length of roughly 5 metres, meaning that they aren't suitable for long distance use cases. |
| USB can transmit data and power simultaneously, which is particularly useful for charging devices like smartphones and powering peripherals like keyboards and mice. | |

# 2.2 Methods of error detection

## Why is error detection needed?

When data is transmitted from one device to another, errors can occur due to interference or issues on the transmission medium. These errors may result in:

- Data loss (where some data isn't received)
- Data gain (unintended increases in data volume, which may include data corruption)
- Data change (e.g. flipping of bits)

## Error detection methods

To ensure data is received correctly, error detection methods are used to identify these errors.

### Parity check

A parity bit is a single bit added to a transmission that can be used to check for errors in the transmitted data. Its value is calculated based on the transmitted data itself.

There are two types of parity bit, even parity and odd parity.

In even parity, the value of the parity bit is chosen so as to make the total number of 1s in the transmitted data even. For example, if the data 01101110 (which contains 5 1s) were to be transmitted, the parity bit would be set to 1, so that the total number of 1s is even.

Odd parity works in a similar way to even parity, but adds a parity bit so that the total number of 1s in the transmitted data is odd.

When data is received, a parity check is carried out. If the value of the received parity bit conforms to the type of parity (odd or even) in use, then the received data is treated as correct. Otherwise, the computer will request that the sender re-transmits the data.

| Data to transmit | Even parity applied | | Data received | Parity check |
|---|---|---|---|---|
| 1101 | 11010 | → | 11010 | No error detected |
| 0000 | 00000 | → | 00100 | Error detected |
| 0100 | 01001 | → | 11001 | Error detected |
| 1001 | 10010 | → | 11110 | No error detected |

In the first example, there is no error in transmission. When the parity check is applied, no error is found and so the transmitted data is treated as correct.

In the second and third examples, an error has resulted in the value of 1 bit being changed (highlighted in red). After a parity check is applied, the error is detected and the computer would request that the data is retransmitted.

In the fourth example, an error has resulted in the values of two bits changing. However, when a parity check is applied, no error is detected as the total number of 1s in the data is still even.

This highlights the major issue with parity bits. Whether using odd or even parity, if an even number of bits are changed during transmission, the error is not detected.

| Advantages | Disadvantages |
|---|---|
| Minimal additional data transfer required, as only a small number of extra bits are appended to the data per transmission. | If an even number of bits are changed during transmission, the error is not detected. |
| Effective at detecting single-bit errors (such as a bit being flipped). | |

**Parity Byte & Block Check**

Like a parity bit, a parity byte is added to a block of data to help detect errors during data transmission or storage. The byte checks the parity (even or odd) of an entire group of bytes, creating a new byte that helps detect errors across the whole group.

Each bit position (0-7) across all bytes in a block is examined, with the number of 1s in each position being counted. The parity byte is then constructed so that each bit ensures an even or odd number of 1s for its corresponding bit position across all bytes.

A parity byte is a type of block check, checking entire data blocks at once.

**Example:**

Suppose you have three bytes and want to add an even parity byte to detect errors.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Byte 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 |
| Byte 2 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| Byte 3 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| Parity | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |

The parity byte: 01000110 ensures that each bit position's total number of 1s is even.

**Checksum**

As with parity bits, checksums involve adding a value, determined by the data itself, to the transmitted data.

An algorithm is used to determine the value of a checksum based on the data being transmitted. There is no agreed algorithm for this and different systems will use their own solutions. A simple algorithm that could be applied is the modulo function, which returns the remainder after a division.

Data to send

46 (denary) = 101110 (binary)

Calculate value of checksum

46 MOD 8 = 6 = 110

Data transmitted

101110110

In the example above, the value of the checksum is calculated using the function MOD 8 which returns the remainder when the value to send is divided by 8. This value is then appended to the original data in binary before being transmitted.

Once received, the recipient can remove the checksum and apply the same algorithm as was used when sending the data to ensure that the checksum matches the transmitted data. If the two do not match, the recipient cannot correct the error itself so must request that the sender re-transmits the data.

| Advantages | Disadvantages |
|---|---|
| Can detect a wider range of errors, including some multiple-bit errors that parity checks would miss. | Some patterns of errors can result in the same checksum, meaning the error goes undetected. |
| Useful for verifying larger blocks of data. | Adds extra processing time on both the sending and receiving ends, which may not be suitable for time-sensitive systems. |

### Echo check

To conduct an echo check, the sender sends data to the receiver, and the receiver sends back the exact same data to the sender (an echo). The sender compares the echoed data to the original. If they match, the transmission is assumed to be error-free.

| Advantages | Disadvantages |
|---|---|
| Verifies that data was received exactly as it was sent. | Inefficient, as it doubles the data traffic by requiring the receiver to send all data back to the sender. |
| Can detect a wide range of transmission errors, including both single and multiple-bit errors. | |

## Check digits

A check digit is a type of checksum in which only a single digit is added to the transmitted data. This reduces the number of different algorithms that could be used to calculate the value of the check digit and so reduces the variety of errors that the method can detect.

Check digits are used in International Standard Book Numbers (ISBN) and bar codes.

## Automatic Repeat Query (ARQ)

ARQ is a method that ensures data is received correctly using acknowledgements and timeouts.

1. The sender transmits data and waits for a response.
2. To determine whether the data contains errors, a method such as a checksum, parity bit or echo check is used. If the receiver gets the data without errors, it sends a positive acknowledgement. If there are errors, then the receiver sends a negative acknowledgement.
3. If the data is incorrect or no response is received within a set timeout period, the sender automatically resends the data.
4. This continues until the correct data is confirmed.

# 2.3 Encryption

## The need for and purpose of encryption

Encryption is essential for protecting data during storage and transmission. Its primary purpose is to convert readable data (plaintext) into an unreadable format (ciphertext) so that only authorised users with the correct decryption key can access the original information. This prevents unauthorised access, even if the data is intercepted or stolen.

Encryption is particularly important when sending sensitive data over networks, such as passwords, personal details, or financial information. It helps maintain confidentiality, ensures data integrity, and supports secure communication between individuals, organisations, and systems.

## Symmetric encryption

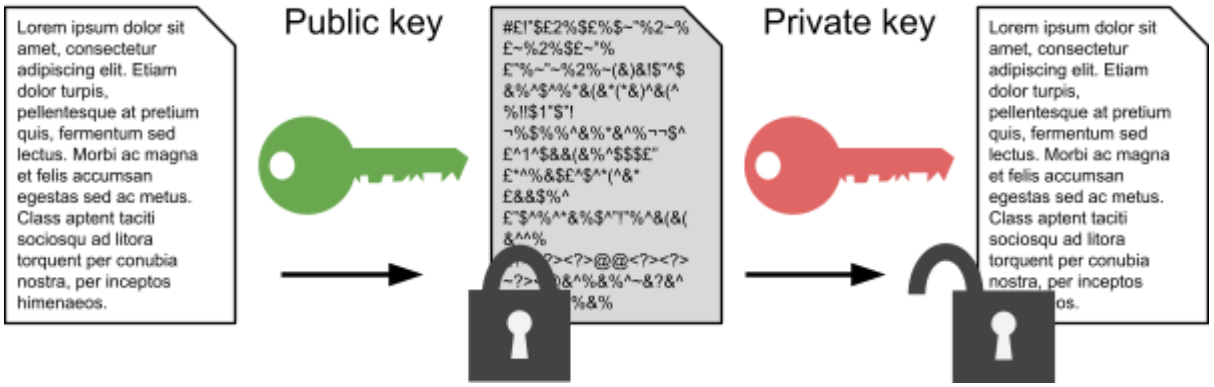
In symmetric encryption, both the sender and receiver share the same private key. This key is used to both encrypt and decrypt data sent between the two parties.

Before sending any information, the sender and receiver must participate in a key exchange to ensure that they both have a copy of their shared key. If the key is exchanged over a network, it is vulnerable to interception. This is a major flaw in symmetric encryption that asymmetric encryption overcomes.

## Asymmetric encryption

When two devices communicate using asymmetric encryption, four different keys are used. Each device has a pair of mathematically related keys, one of which is kept secret (the private key) and the other shared on the Internet (the public key).

When a message is encrypted with a public key, only the corresponding private key (typically only held by one user) can decrypt it and vice versa.




Before a message is sent, it is encrypted by the sender using the recipient's public key. This means that the message can only be decrypted by the corresponding private key (as explained earlier), the recipient's private key, which only the recipient has access to. This means that the recipient is the only person who can decrypt the message.