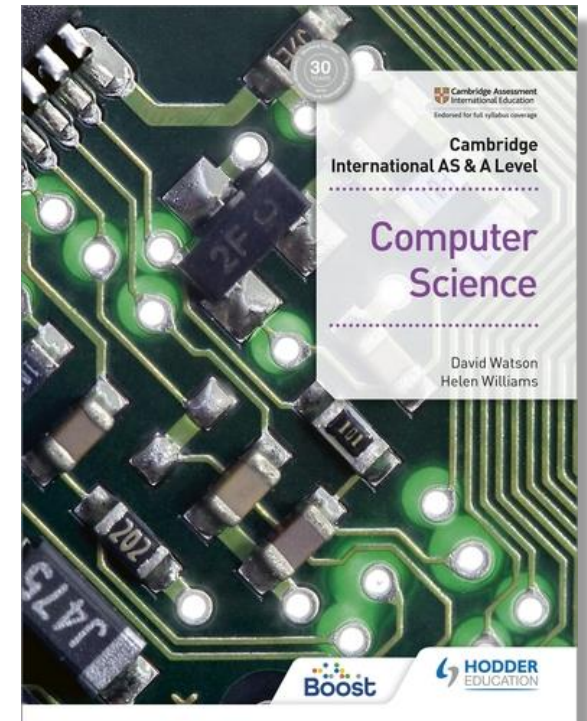


# Chapter 6

## Security, Privacy and Data Integrity

6.1 Data Security

6.2 Data Integrity



# 4. Security, Privacy and Data Integrity

## LEARNING OBJECTIVES:

1. The terms security, privacy and integrity of data
2. The need for security of data and security of computer systems
3. Security measures to protect computer systems such as user accounts, passwords, digital signatures, firewalls, antivirus and anti-spyware software and encryption
4. Security threats such as viruses and spyware, hacking, phishing and pharming
5. Methods used to reduce security risks such as encryption and access rights
6. The use of validation to protect data integrity
7. The use of verification during data entry and data transfer to reduce or eliminate errors.

# 6.1 Data Security

## KEY TERMS (1/2):

- **Data privacy** – the privacy of personal information, or other information stored on a computer, that should not be accessed by unauthorised parties.
- **Data protection laws** – laws which govern how data should be kept private and secure.
- **Data security** – methods taken to prevent unauthorised access to data and to recover data if lost or corrupted.
- **User account** – an agreement that allows an individual to use a computer or network server, often requiring a user name and password.
- **Authentication** – a way of proving somebody or something is who or what they claim to be.
- **Access rights (data security)** – use of access levels to ensure only authorised users can gain access to certain data.
- **Malware** – malicious software that seeks to damage or gain unauthorised access to a computer system.
- **Firewall** – software or hardware that sits between a computer and external network that monitors and filters all incoming and outgoing activities.
- **Anti-spyware software** – software that detects and removes spyware programs installed illegally on a user's computer system.

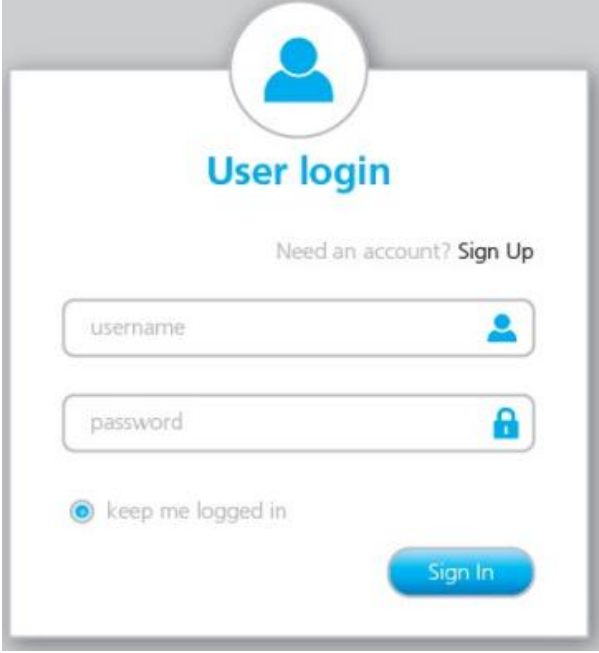
# 6.1.1 Data Privacy

- **Data Privacy Requirement:** Data stored about a person or an organization must remain **private**, and **unauthorized access** to the data must be **prevented**.
- **Data Protection Laws:** These laws vary **country to country**, but all follow the **same eight guiding principles**.
- **Coverage:** Data protection laws usually cover **organizations** rather than private individuals.
- **Legal Consequences:** Such laws are no guarantee of privacy, but the **legal threat** of **fines** or **jail sentences** deters most people.

1. Data must be fairly and lawfully processed.
2. Data can only be processed for the stated purpose.
3. Data must be adequate, relevant and not excessive.
4. Data must be accurate.
5. Data must not be kept longer than necessary.
6. Data must be processed in accordance with the data subject's rights.
7. Data must be kept secure.
8. Data must not be transferred to another country unless that country also has adequate protection.

# 6.1.2 Preventing data loss and restricting data access

- **Data Security:** Refers to the methods used to prevent **unauthorized access** to data and data recovery methods if it is **lost**.
- **User Accounts:**
  - Used to **authenticate** a user, proving their identity.
  - Employed on both **standalone** and **networked** computers for multi-user access.
  - **Authentication Process:** Involves a **username** and **password** prompt.
- **Access Control:**
  - User accounts dictate **access rights**.
  - Involves **access level hierarchy**.
  - **Example:** In a hospital, cleaners wouldn't access patient data, while consultants would.
  - **Hierarchy:** Access levels linked to a person's security level via **username** and **password**.



A screenshot of a user login interface. At the top, there is a blue circular icon of a person. Below it, the text "User login" is displayed in blue. Underneath, there is a link that says "Need an account? Sign Up". The form contains two input fields: "username" with a blue person icon on the right, and "password" with a blue padlock icon on the right. Below the password field is a radio button labeled "keep me logged in". At the bottom right, there is a blue "Sign In" button.

# 6.1.2 Preventing data loss and restricting data access

- **Use of Passwords:**
  - **Restriction:** Passwords restrict access to data or systems.
  - **Security:** Passwords should be **hard to crack** and **changed frequently**.
  - **Biometrics:** Passwords can take the form of **biometrics**, like on a mobile phone.
  - **Applications:**
    - Accessing **email accounts**
    - Carrying out **online banking** or **shopping**
    - Accessing **social networking sites**
- **Password Protection:**
  - Ways to protect passwords:
    - Run **anti-spyware software** to prevent password relay by spyware.
    - **Regularly change** passwords to prevent unauthorized access.
    - Create passwords that are **difficult to crack or guess**.
      - Avoid using **personal information** like date of birth or pet's name.
- **Password Strength:**
  - Passwords are grouped as **strong** (hard to crack) or **weak** (easy to crack).
  - Strong passwords should contain:
    - **At least one capital letter**
    - **At least one numerical value**
    - **At least one other keyboard character** (e.g., @, \*, &)
  - Example of a **strong password**: Sy12@#TT90kj=0
  - Example of a **weak password**: GREEN



# 6.1.2 Preventing data loss and restricting data access

- **Firewalls:** Firewalls are like guards for computers.
  - They can be **software** or **hardware**.
  - They sit between the computer and the internet.
  - They decide what comes in and goes out.
  - They protect from bad things like hacking and viruses.
- **Firewall Tasks:**
  - **Checking traffic** between computer and internet.
  - Seeing if data is good.
  - **Blocking bad data** and warning about it.
  - Keeping a record of traffic.
  - Stopping access to bad sites.
  - Helping stop viruses and hackers.
  - Warning if software wants to connect outside.
- **Types of Firewalls:**
  - Hardware ones are like gateways.
  - Software ones are like computer helpers.
- **Firewalls Can't Always Help:**
  - Can't stop people using own modems to bypass it.
  - Can't control bad behavior or mistakes.
  - Can't stop some users from turning it off.



## 6.1.2 Preventing data loss and restricting data access

- **Managing Firewalls:** To make firewalls work well, people need to take charge.
- **Antivirus Software:**
  - Keeps computer safe from **viruses**.
  - Checks for viruses all the time.
  - Different types of **antivirus** work differently.
  - They:
    - Check software before using it.
    - Compare against **virus database**.
    - Look for virus **behavior**.
    - Keep suspicious things separate.
    - Can delete viruses automatically or ask you.
  - Keep antivirus updated to catch new viruses.
  - Check whole computer often to catch hidden viruses.



## 6.1.2 Preventing data loss and restricting data access

- **Anti-spyware Software:**
  - Stops and removes sneaky **spyware** on computers.
  - Uses rules or known structures to find spyware.
- **Encryption:**
  - Keeps data safe from hackers.
  - Makes data unreadable without special **keys**.
  - Hackers can't use the data even if they get it.

# 6.1.2 Preventing data loss and restricting data access

- **Biometrics:** New way to keep computers safe.
  - Uses human body features.
  - Examples: fingerprints, eye scans, face, and voice.
- **Fingerprint Scans:**
  - Compares fingerprints with saved ones.
  - Looks at patterns of ridges and valleys.
  - Fairly unique (1 in 500 chance of mistake).
- **Retina Scans:**
  - Uses infrared to look at blood vessels in the eye.
  - Very secure, hard to copy.
  - Unique patterns (1 in 10 million chance of mistake).
- **Mobile Phones:**
  - Use biometrics to check if user is the owner.

# 6.1.3 Risks to the security of stored data

- **Hacking:**
  - Word used in this book a lot.
  - Two types: **malicious** and **ethical**.
- **Malicious Hacking:**
  - Bad hacking.
  - Gets into computers illegally.
  - Wants to do bad things.
  - Steal, break, or change things.
  - Guarded by strong passwords, firewalls, and special software.
- **Ethical Hacking:**
  - Good hacking.
  - Companies ask for it.
  - Checks if they're safe from bad hacking.
  - Legal and paid for.

# 6.1.3 Risks to the security of stored data

- **Malware:** Bad things that can hurt computer data.
  - Some antivirus apps can remove these.
- **Viruses:**
  - Copies itself to break things.
  - Needs other programs to work.
- **Worms:**
  - Like viruses, but travels to other computers.
  - Uses weak computers to spread.
- **Logic Bombs:**
  - Hidden in programs.
  - Does bad things when conditions are met.
- **Trojan Horses:**
  - Pretends to be good software.
  - Hurts computer when used.
- **Bots (Internet Robots):**
  - Can do helpful things.
  - Can also control computers and attack.
- **Spyware:**
  - Watches what you type.
  - Sends info to others.

# 6.1.3 Risks to the security of stored data

- **Phishing:**
  - Bad people send real-looking emails.
  - They want you to click links or give personal info.
  - They pretend to be trusted sources like banks.
  - You need to do something (click link) for it to work.
- **Preventing Phishing:**
  - **Know** about new scams.
  - Don't click links if not sure.
  - Use anti-phishing tools on web browsers.
  - Look for **https** and green padlock.
  - Check accounts and change passwords.
  - Use good browser with updates and firewall.
  - Watch out for pop-ups, close with **X**.

# 6.1.3 Risks to the security of stored data

- **Pharming:**
  - Bad code on computer or server.
  - Sends user to fake site without them knowing.
  - Gets personal info.
  - Seems real but is fake.
- **Why It's Bad:**
  - Takes users to fake sites.
  - Uses tricks like DNS poisoning.
  - Changes real address to fake.
  - Makes computer go to bad site.
- **Protection from Pharming:**
  - Use **antivirus software**.
  - Use good web browsers.
  - Check spelling of websites.
  - Look for **https** and green padlock.
- **Harder to Stop:**
  - If DNS server is bad, it's tough to fix.

# 6.1.3 Risks to the security of stored data

Data Recovery	
Accidental loss of data (for example, accidental deletion of a file)	<ul style="list-style-type: none"><li>• use back-ups in case the data is lost or corrupted through an accidental operation</li><li>• save data on a regular basis</li><li>• use passwords and user IDs to restrict access to authorised users only</li></ul>
Hardware fault (such as head crash on the HDD)	<ul style="list-style-type: none"><li>• use back-ups in case data is lost or corrupted through the hardware fault</li><li>• use uninterruptable power supply (USP) in case power loss causes hardware malfunction</li><li>• save data on a regular basis</li><li>• use parallel systems as back-up hardware</li></ul>
Software fault (for example, incompatible software installed on the system)	<ul style="list-style-type: none"><li>• use back-ups in case the data is lost or corrupted through the software fault</li><li>• save data on a regular basis in case the software suddenly 'freezes' or 'crashes' while the user is working on it</li></ul>
Incorrect computer operation (for example, incorrect shutdown or procedure for removing memory stick)	<ul style="list-style-type: none"><li>• use back-ups in case data is lost or corrupted through wrong operation</li><li>• correct training procedures so users are aware of the correct operation of hardware</li></ul>



## 6.1.3 Risks to the security of stored data

- **Data Backup:**
  - Save data often on another thing like cloud or HDD.
  - Do it every day, even automatically.
  - Keep it safe somewhere else.
- **Important:**
  - Have someone do backups always.
  - Backup helps if something bad happens.
- **Not Always Good:**
  - Backups can have viruses, so not always helpful.

# 6.2 Data Integrity

## KEY TERMS:

- **Data integrity** – the accuracy, completeness and consistency of data.
- **Validation** – method used to ensure entered data is reasonable and meets certain input criteria.
- **Verification** – method used to ensure data is correct by using double entry or visual checks.
- **Check digit** – additional digit appended to a number to check if entered data is error free.
- **Modulo-11** – method used to calculate a check digit based on modulus division by 11.
- **Checksum** – verification method used to check if data transferred has been altered or corrupted, calculated from the block of data to be sent.
- **Parity check** – method used to check if data has been transferred correctly that uses even or odd parity. Parity bit – an extra bit found at the end of a byte that is set to 1 if the parity of the byte needs to change to agree with sender/receiver parity protocol.
- **Odd parity** – binary number with an odd number of 1-bits.
- **Even parity** – binary number with an even number of 1-bits.
- **Parity block** – horizontal and vertical parity check on a block of data being transferred.

# 6.2 Data Integrity

- **Data Integrity:**
  - Data on a computer must be **accurate**, **consistent**, and **up to date**.
  - **Validation** and **verification** methods are used to ensure data integrity.
- **Data Accuracy Compromised:**
  - During **data entry** and **data transmission**.
  - Through **malicious attacks** (like malware and hacking).
  - Due to **accidental data loss** from hardware problems.

# 6.2.1 Validation

- **Validation** is a method of **checking** if entered data is **reasonable** (and within a given criteria), but it **cannot check** if data is **correct** or **accurate**.
- For example, if somebody **accidentally** enters their age as **62** instead of **26**, it is **reasonable** but **not accurate** or **correct**.
- **Validation** is carried out by **computer software**.

## KEY TERMS:

- **Parity byte** – additional byte sent with transmitted data to enable vertical parity checking (as well as horizontal parity checking) to be carried out.
- **Automatic repeat request (ARQ)** – a type of verification check.
- **Acknowledgement** – message sent to a receiver to indicate that data has been received without error.
- **Timeout** – time allowed to elapse before an acknowledgement is received.

## 6.2.1 Validation

Validation test	Description	Example of data failing validation test	Example of data passing validation test
type	checks whether non-numeric data has been input into a numeric-only field	typing sk.34 in a field which should contain the price of an item	typing 34.50 in a field which should contain the price of an item
range	checks whether data entered is between a lower and an upper limit	typing in somebody's age as -120	typing in somebody's age as 48
format	checks whether data has been entered in the agreed format	typing in the date as 12-12-20 where the format is dd/mm/yyyy	typing in the date as 12/12/2020 where the format is dd/mm/yyyy
length	checks whether data has the required number of characters or numbers	typing in a telephone number as 012 345 678 when it should contain 11 digits	typing in a telephone number as 012 345 678 90 when it should contain 11 digits
presence	checks to make sure a field is not left empty when it should contain data	please enter passport number:.....	please enter passport number: AB 1234567 CD

## 6.2.1 Validation

existence	checks if data in a file or a file name actually exists	data look up for car registration plate A123 BCD which does not exist	data look up for a file called books_in_stock which exists in a database
limit check	Checks only one of the limits (such as the upper limit OR the lower limit)	typing in age as -25 where the data entered should not be negative	typing in somebody's age as 72 where the upper limit is 140
consistency check	checks whether data in two or more fields match up correctly	typing in Mr in the title field and then choosing female in the sex field	typing in Ms in the title field and then choosing female in the sex field
uniqueness check	checks that each entered value is unique	choosing the user name MAXIMUS222 in a social networking site but the user name already exists	choosing the website name Aristooo.com which is not already used

# 6.2.2 Verification

- **Verification** is a way of **preventing errors** when **data** is **entered manually** (using a keyboard, for example) or when **data** is **transferred** from **one computer** to **another**.
- **Verification during Data Entry:**
  - **Manual data entry** needs verification to prevent errors.
  - **Three ways** to verify: **double entry**, **visual check**, and **check digits**.
- **Double Entry:**
  - Data is entered twice by different people.
  - Entries are compared later or during entry.
- **Visual Check:**
  - Entered data is compared with **original document**.
  - Screen data is matched with data on paper.



## 6.2.2 Verification

- **Check Digits**: Extra digit added to a number, usually on the right.
- Used in **barcodes**, **ISBNs** (book codes), and **VINs** (car ID numbers).
- Helps ensure the **number** is **right**.
- **Catches Errors** like wrong digit, swapped numbers, missing or extra digits, and phonetic mistakes.



a barcode with an ISBN-13 code  
with check digit

## 6.2.2 Verification

- An example of a **check digit calculation** is **modulo-11**.
- The following **algorithm** is used to **generate** the **check digit** for a **number** with **seven digits**:
  1. **Each digit** in the **number** is given a weighting of **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** and the **remainder subtracted** from **11**.
  4. The **check digit** is the **value** generated; note if the **check digit** is **10** then **X** is used.

Example:

Seven digit number:

4 1 5 6 7 1 0

Weighting values:

7 6 5 4 3 2 1

Sum:

$$(7 \times 4) + (6 \times 1) + (5 \times 5) + (4 \times 6) + (3 \times 7) \\ + (2 \times 1) + (1 \times 0)$$

$$= 28 + 6 + 25 + 24 + 21 + 2 + 0$$

Total:

$$= 106$$

Divide total by 11:

9 remainder 7

subtract remainder from 11:

$$11 - 7 = 4 \text{ (check digit)}$$

final number:

4 1 5 6 7 1 0 4

When this number is entered, the check digit is recalculated and, if the same value is not generated, an error has occurred.

# 6.2.2 Verification

- **Verification during Data Transfer:**
  - When data is sent from one device to another, there's risk of **corruption** or **loss**.
  - Ways to reduce risk exist.
- **Checksums:**
  - Check if data changes during transfer.
  - Data sent in blocks with an **extra value** (checksum) at the end.
- **How Checksums Work:**
  - Assume checksum is **1 byte long** (max value 255).
  - If sum of data bytes  $\leq 255$ , checksum is sum .
  - If sum  $> 255$ , use a **simple algorithm** for checksum calculation.

## 6.2.2 Verification

- In the example we will assume the value of  $X = 1185$ .

①  $(X = 1185): 1185/256 = 4.629$

② Rounding down to nearest whole number gives  $Y = 4$

③ Multiplying by 256 gives  
 $Z = Y * 256 = 1024$

④ The difference  $(X - Z)$  gives the checksum:  $(1185 - 1024) = 161$

⑤ This gives the checksum: 161

divide the sum,  $X$ , of the bytes by  
256



round the answer down to the  
nearest whole number,  $Y$



$$Z = Y * 256$$



calculate the difference  $(X - Z)$



the value is the checksum

# 6.2.2 Verification

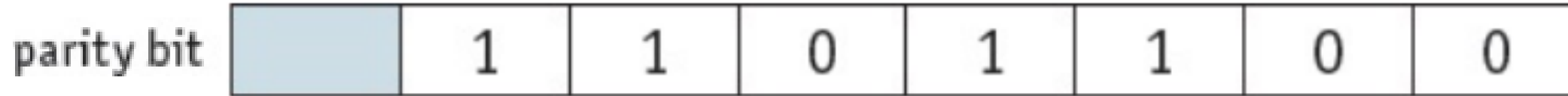
- **Checksum Calculation:**
  - Before data transmission, **calculate checksum** for bytes.
  - **Send checksum** with **data block**.
- **Receiving End:**
  - **Recalculate checksum** from received data.
  - **Compare** calculated and transmitted **checksums**.
  - If **same**, data is **error-free**.
  - If **different**, ask for **re-transmission**.

## 6.2.2 Verification

- **Parity Checks**: Another way to check if data changed during transmission.
- Each data byte gets a **parity bit**.
- Parity bit assigned before sending.
- **Even parity** has even 1-bits.
- **Odd parity** has odd 1-bits.

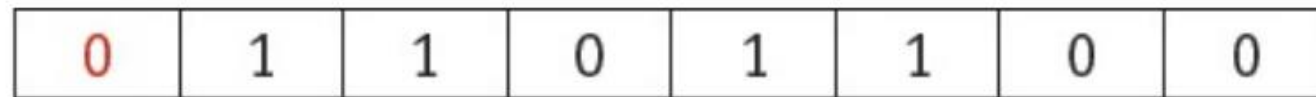
## 6.2.2 Verification

- Consider the following **byte**:



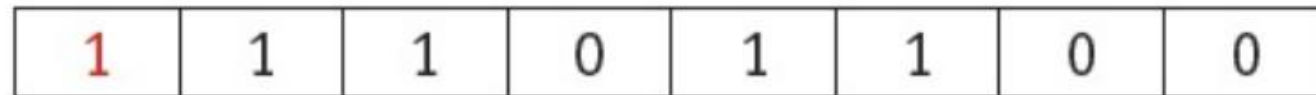
- If this byte is using **even parity**, then the **parity bit** needs to be **0** since there is **already** an **even number of 1-bits** (in this case, **four**).
- If **odd parity** is being used, then the **parity bit** needs to be **1** to **make** the **number** of **1-bits** **odd**. Therefore, the byte just before transmission would be:

either (even parity):



parity bit

or (odd parity):



parity bit

- Before data** is **transferred**, an **agreement** is made between **sender** and **receiver** regarding which of the **two types** of **parity** are **used**.
- This is an example of a **protocol**.



## 6.2.2 Verification

- If a **byte** has been **transmitted** from 'A' to 'B', and **even parity** is used, an **error** would be **flagged** if the **byte now** had an **odd number** of **1-bits** at the **receiver's** end.

- Example:

Sender's byte:

0	1	0	1	1	1	0	0
---	---	---	---	---	---	---	---

parity bit

Receiver's byte:

0	1	0	0	1	1	0	0
---	---	---	---	---	---	---	---

parity bit

- **Parity Mismatch:**
  - Receiver's byte has **odd parity** (3 1-bits).
  - Sender's byte had **even parity** (4 1-bits).
  - **Error in transmission** detected.
- **Error Detection:**
  - Computer **recalculates parity** of sent byte.
  - If agreed **even parity**, change to **odd parity** means **transmission error**.

# 6.2.2 Verification

- **Transmission Errors:**
  - Bits in example could change, causing errors.
  - Flagged error doesn't show which bit is wrong.
- **Parity Blocks:**
  - Use blocks of data to fix this.
  - Count 1-bits up and down.
  - **Find errors** and show where they are.
- **Example:**
  - Sent 9 bytes.
  - **Agreed even parity.**
  - Extra byte, **parity byte**, sent too.
  - Parity byte has **vertical parity bits**.
  - Also marks **end of data block**.

# 6.2.2 Verification

- Here, it shows **how** the **data arrived** at the **receiving end**:

	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

- byte 8** (row 8) has **incorrect parity** (there are three 1-bits)
- bit 5** (column 5) also has **incorrect parity** (there are five 1-bits).
- First, the table shows that an **error** has **occurred** following **data transmission**.
- Second, at the **intersection** of **row 8** and **column 5**, the **position** of the **incorrect bit** value (which caused the error) can be **found**. This means that **byte 8** should have been:

0	0	0	1	0	0	1	0
---	---	---	---	---	---	---	---

- Which would also correct **column 5** giving an **even vertical parity** (now has four 1-bits).

# 6.2.2 Verification

- Error Handling:
  - **Byte with errors** can be **fixed automatically**.
  - **Error message** can ask sender to **resend** data block.
- Important Note:
  - If **two bits change**, method may **not** find **error**.

• For example, using the above example again:

0	1	0	1	1	1	0	0
---	---	---	---	---	---	---	---

• This byte could reach the destination as:

0	1	1	1	1	1	0	1
---	---	---	---	---	---	---	---

- **All three** are clearly **incorrect**, but they have retained **even parity** so will **not trigger** an **error message** at the **receiving end**.
- Clearly, other methods to **complement parity** when it comes to **error checking** of transmitted data are required (such as **checksum**).

## 6.2.2 Verification

- **Automatic Repeat Request (ARQ):**
  - Checks data after transmission.
  - **Acknowledgement** (message saying data received right) and **timeout** used.
  - **Receiving device** asks for **data resend** if **error found**.
  - Sends **positive message** if **no error**.
  - Sender **re-sends** if asked or **timeout** happens.
  - Continues until **correct data** or **timeout**.
  - **Used in mobile networks** to ensure data is good.