

Security, Privacy & Integrity

AS Computer Science ■ Topic 6

A-Level Computer Science



What we will cover

- 1 Data security: threats and measures 6.1
- 2 Data integrity: validation and verification 6.2
- 3 Recap



1

Data security

Sheet 6.1 – what you need to know

You must be able to:

- tell security, privacy and integrity apart
- describe threats and the measure that stops each
- explain encryption, digital signatures and parity checks

What the exam asks

“Identify one security measure that protects the data when it is being transmitted...” [3]

“Identify and describe two threats to the data. Identify one security measure for each.” [6]

“Complete the description of parity check when Computer A is transmitting data to Computer B.” [5]

Practise on exercise sheet 6.1, then try the real exam questions on past-paper sheet 6.1.

Three different ideas

security 安全

protect data from
unauthorised
access or change

privacy 隐私

who is allowed to
see personal data

integrity 完整性

data is accurate
& complete,
not corrupted

Security, privacy and integrity are three different things

security

keeping data **safe from unauthorised access**, damage or loss

privacy

controlling **who is allowed** to see data about a person – a legal and ethical question, not a technical one

integrity

keeping data **accurate and consistent** – uncorrupted in storage and unchanged in transit

they can conflict

a full audit log improves integrity and reduces privacy

DoS/DDoS 拒绝服务

floods a server so **real users cannot reach it** – an attack on availability, not on secrecy

An encrypted file that has been silently altered is secure and *not* intact. Naming which of the three a scenario threatens is the first mark.

Threats come in three groups

malware

software written to harm: viruses and worms that spread on their own; a **Trojan horse** 木马, spyware, ransomware or adware that hides

tricking people

phishing 网络钓鱼 emails and **pharming** 域名欺骗 redirects that send you to a convincing fake

attacking the network

hacking 黑客入侵 through weak passwords or software flaws;
denial of service 拒绝服务 flooding a server;
eavesdropping 窃听 on open Wi-Fi

Two things need protecting: the **data** (confidential, intact, available) and the **system** itself – a compromised machine goes on to attack others.

Malware

two ways **malware** 恶意软件 behaves

self-spreading 自我传播

- **virus** 病毒
attaches to a host file
- **worm** 蠕虫
needs no host file

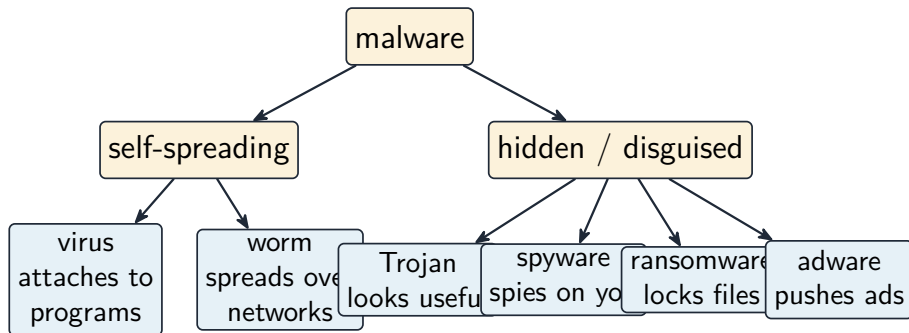
spreads copies on its own

hidden / disguised 伪装

- **Trojan** 木马
- **spyware** 间谍软件
logs keystrokes and passwords
- **ransomware** 勒索软件
encrypts files, demands payment
- **adware** 广告软件
pushes unwanted adverts

Malware sorted by behaviour

Malware by behaviour: self-spreading (virus, worm) versus hidden/disguised (Trojan, spyware, ransomware, adware)



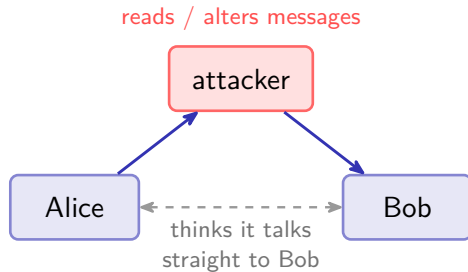
Social and network attacks

Tricking people:

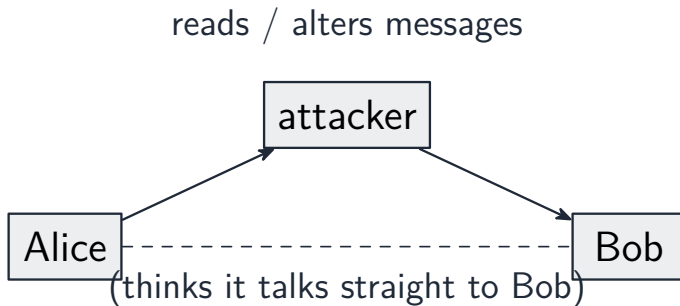
- **phishing** 网络钓鱼 – fake emails/sites
- **pharming** 域名欺骗 – redirect to a fake site

On the network:

- **DoS / DDoS** 拒绝服务 – flood a server
- **man-in-the-middle** 中间人攻击 – relay/alter messages



The man in the middle

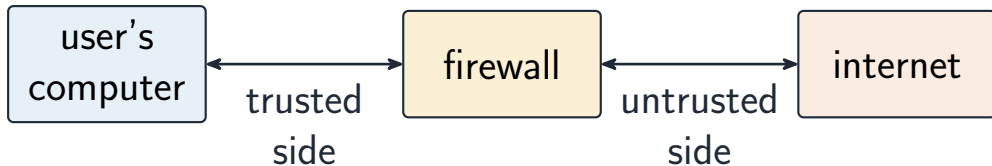


The attacker relays both sides, so each believes it is talking straight to the other – what HTTPS and **digital signatures** 数字签名 exist to stop.

A networked PC

A firewall sits between the user's computer and the internet

inspects & filters traffic
(allows or blocks each packet)



Security measures



- **standalone PC:** a strong password, **antivirus** 杀毒软件 kept up to date, prompt updates, **backup** 备份 to separate media, full-disk **encryption** 加密, a locked screen
- **networked PC:** all of that, plus a **firewall** 防火墙, per-user permissions (admin rights only for admins), central **user accounts** 用户账户 and **audit logs** 审计日志
- **across the internet:** a **VPN** 虚拟专用网 encrypting traffic to the gateway, HTTPS/TLS, **digital signatures** 数字签名 proving who sent a message and that it was not altered, and intrusion detection watching for known attack patterns

Match the measure to the threat

interception → **encrypt**

malware → **antivirus, anti-spyware** 反间谍软件, **patching**

insiders → **least-privilege**
最小权限 + auditing

unauthorised access
→ **authentication**

social engineering 社会工程
→ **training + filtering**

DDoS → **rate limiting**

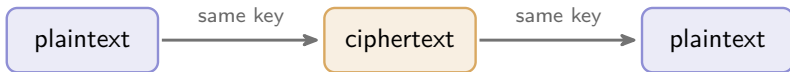
Threat, what it does, how to stop it: the six-mark table

threat	description	measure
hacking	unauthorised access to the system, to read, change or delete data	firewall; strong passwords; access rights
virus	malware that replicates itself and damages or deletes files	anti-virus software, kept updated
spyware	records what the user does and sends it to a third party	anti-spyware; do not install unknown software
keylogger	spyware that records key presses , to capture passwords	anti-spyware; on-screen keyboard; two-step login
phishing	an email links to a fake website that asks for details	do not click links; check the address
pharming	code redirects a correct address to a fake site	check for HTTPS and the certificate

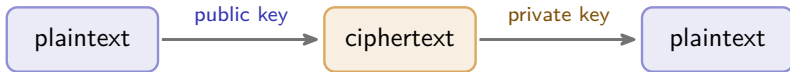
Data being **transmitted**: encryption, so intercepted data cannot be understood without the key.

Encryption

symmetric 对称加密

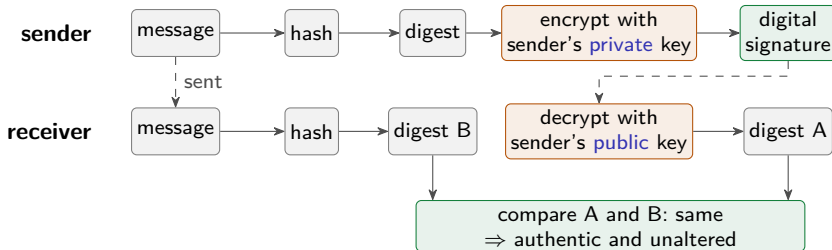


asymmetric 非对称加密



- **symmetric** 对称加密: one shared key locks & unlocks (AES)
- **asymmetric** 非对称加密: **public key** 公钥 locks, **private key** 私钥 unlocks (RSA)

Digital signatures, and whose key



Signing uses the **sender's private** key; anyone can check with the **sender's public** key. Matching digests: it came from the sender, and was not changed.

Sending a secret to one customer: encrypt with the **customer's public** key, so only the customer's **private** key can decrypt it.

Authentication

Three factors, strongest combined – **two-factor authentication** 双因素认证.

Authentication proves *who* you are; **authorisation** 授权 decides what you may then do.

- **know** – a password
- **have** – a token or phone code
- **are** – **biometrics** 生物识别 (fingerprint, face)

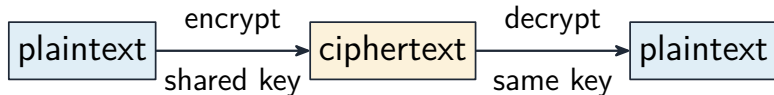


“something you have”

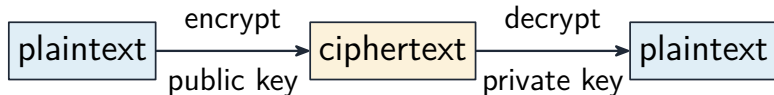
Protecting the data itself

Access control 访问控制 – file permissions enforced by the OS; **backups**, some off-site; **physical security** – locked server rooms and cable locks; and encryption: symmetric shares one key, asymmetric encrypts with a public key and decrypts with a private one.

symmetric



asymmetric



Parity blocks, and recognising a face

Completing the description of a parity check

A and B agree on **odd** or **even** parity. Data is sent in groups of **7 bits**, with a parity bit making the 8th. After a block of bytes, a **parity byte** is sent, with a parity bit for each column. B recounts: a wrong row and a wrong column locate the **flipped bit**, which can be corrected.

How does AI recognise a face?

The camera captures the face; features (distance between the eyes, shape of the nose and jaw) are **measured**; they are compared with stored faces; a model **trained** on many images decides the closest match, and improves with more data.

Security keeps data **safe** from loss or misuse; privacy controls **who may see** it.

2

Data integrity

Sheet 6.2 – what you need to know

You must be able to:

- tell validation from verification, with examples
- calculate a check digit, and say what it misses
- write pseudocode that extracts digits with DIV and MOD

What the exam asks

“Identify an example of the incorrect attempt to enter 41623 and explain why this number would not be rejected.” [2]

“Write pseudocode for the module Generate()” [7]

“Give two similarities and one difference between spyware and a virus.” [3]

Practise on exercise sheet 6.2, then try the real exam questions on past-paper sheet 6.2.

Validation vs verification

validation 验证

"is the data sensible?"

- range / type / format
- check digit

catches nonsense data

verification 核对

"was it copied correctly?"

- double entry
- parity / checksum

catches copying errors

The validation checks, by what each catches

range check	the value lies between two limits – a month between 1 and 12
type check	the data is of the right type – a number where a number is required
length check	the right number of characters – a 16-digit card number
format check	matches a pattern – an email must contain @
presence check	required fields are not empty
existence check	the referenced item exists – a product code is in the table
a check digit 校验位	an extra digit computed from the others (ISBN, card numbers) that spots transcription errors

Validation asks *is this sensible*; verification asks *is this what was sent*. A validated wrong value passes every check on this list.

Working out a check digit

Rule 1: the sum of the digits, MOD 10

41623: $4 + 1 + 6 + 2 + 3 = 16$, check digit **6**, so 41623**6**.

Typed as **14623**: the sum is still 16, so the check digit still matches and the error is **not** rejected.

A plain sum cannot see the **order** of the digits, so it misses a transposition.

Rule 2: weighted, MOD 11

Weights 6, 5, 4, 3, 2:

$$4 \cdot 6 + 1 \cdot 5 + 6 \cdot 4 + 2 \cdot 3 + 3 \cdot 2 = 65$$

$$65 \text{ MOD } 11 = 10; \text{ check } = 11 - 10 = 1$$

$$14623: 6 + 20 + 24 + 6 + 6 = 62,$$

$$62 \text{ MOD } 11 = 7, \text{ check } 4 \neq 1: \text{ **caught**.}$$

A check digit is **validation**: it tests a value is possible, not that it is the right one.

Writing Generate(): extract the digits, append the check

```
FUNCTION Generate(Number : INTEGER) RETURNS INTEGER
  DECLARE Total, Temp : INTEGER
  Total ← 0
  Temp ← Number
  WHILE Temp > 0
    Total ← Total + MOD(Temp, 10)
    Temp ← DIV(Temp, 10)
  ENDWHILE
  RETURN Number * 10 + MOD(Total, 10)
ENDFUNCTION
```

MOD(Temp, 10) gives the **last digit**; DIV(Temp, 10) **removes** it. Multiply by 10 and add, and the check digit becomes the new last digit: $\text{Generate}(41623) = 416236$.

Verification – was the data entered or transferred correctly?

Validation checks the data makes sense; verification checks it was copied without change

validation
"is this data sensible?"

verification
"was it copied correctly?"

checks rules *before* storing:

- range / type / format
- length / presence
- check digit

catches nonsense data

checks the data did not change:

- double entry
- parity check
- checksum / CRC

catches copying errors

Verification on transfer – parity

even parity:

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

 number of 1s = 4 (even)

parity bit 7 data bits

odd parity:

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

 number of 1s = 5 (odd)

parity check 奇偶校验: count the 1s – a flipped bit changes the count, so the error shows up.

A **parity block check** 奇偶块校验 goes further: give each byte a row parity bit and compute a column parity byte too, and the crossing row and column **locate** the flipped bit.

Worked example: locate the flipped bit

parity bits

	1	2	3	4	5	6	7	8	
byte 1	1	0	1	1	0	1	0	0	
byte 2	0	1	1	0	0	1	1	1	odd!
byte 3	1	1	0	1	1	0	0	0	
byte 4	0	0	1	1	0	1	1	0	
parity byte	0	0	1	1	0	1	0	1	

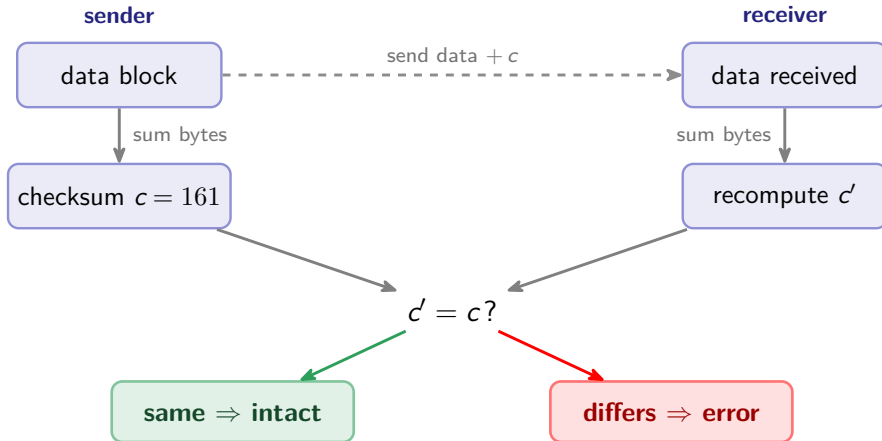
odd!

- sent with **even** parity in every row *and* column
- receiver counts: row 2 odd, column 5 odd
- the error sits **where they cross** – flip it back to 1

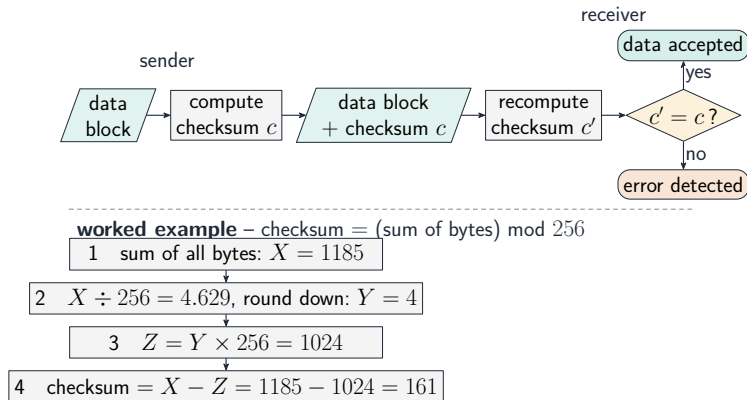
One parity bit only **detects** an error; a parity **block** locates it – so it can be corrected.

Verification on transfer – checksum

A **cyclic redundancy check** 循环冗余校验 (CRC) is a stronger checksum using polynomial division.



Working out a checksum for a block of data



The sender computes the checksum and sends it with the data; the receiver computes it again and compares. A difference proves the data changed – but a match is not proof it did not.

Worked example – validation or verification?

A user types their date of birth as 31/02/2009, and types their email address twice. Which check catches which error?

- **Validation** asks “is this data *sensible*?” – a format or range check rejects 31/02/2009, because February never has 31 days.
- **Verification** asks “was it *entered correctly*?” – typing the email twice is double entry, and comparing the copies catches a typing slip.
- The limit: validation can never say the data is *right*, only that it is *possible*.

01/02/2009 passes every validation rule even when the user was born in March. That sentence is the answer to most questions on this topic.

Spyware and a virus: alike and different

“Give two similarities and one difference between spyware and a virus.”
[3]

Similar: both are malware; both are installed without the user's knowledge; both are stopped by security software.

Different: a virus **replicates** and damages files; spyware **gathers** information and sends it on, and does not replicate.

Validation or verification?

Validation: is the data **sensible**? (range, format, length, presence, type, check digit)

Verification: was it **copied correctly**? (double entry, visual check, parity, checksum)

3

Recap

Exam tips

- Keep the three ideas separate: **security** (keeping data safe), **privacy** (who may see it), **integrity** (keeping it correct).
- Match each **threat** (malware, hacking, phishing, interception) to a **measure** (firewall, encryption, authentication, access rights).
- Encryption protects **confidentiality**, **not integrity** – use a checksum, parity or check digit for integrity.
- Distinguish a **virus**, **worm** and **Trojan** and how each spreads.

Topic 6 – key takeaways

1

Three ideas

security, privacy, integrity differ

2

Threats

malware, phishing, DoS, man-in-the-middle

3

Encryption

symmetric (one key) vs
asymmetric (key pair)

4

Integrity

validation = sensible; verification
= copied right

Check yourself

- 1 Which malware spreads over a network with no user action?
- 2 Which key decrypts in asymmetric encryption?
- 3 Is a range check validation or verification?
- 4 What does a parity bit detect?



Answers 1. a worm 2. the private key 3. validation 4. a single-bit error