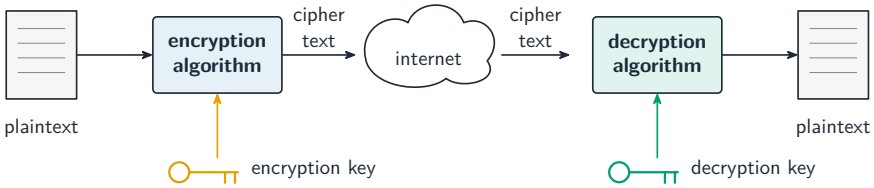
Security

A-Level Computer Science

How encryption works

Encryption 加密 turns readable **plaintext** 明文 (plain text) into unreadable **ciphertext** 密文 (cipher text) using a maths operation that depends on a **key**. Only someone with the right key can reverse it — **decryption** 解密 — to get the plaintext back. An attacker who intercepts the ciphertext without the key sees only meaningless data, because trying every possible key would take far too long. A newer approach, **quantum cryptography** 量子密码学, uses quantum physics to share a key in a way that reveals any eavesdropper.

The Enigma machine encrypted messages in the Second World War — an early, mechanical cipher device

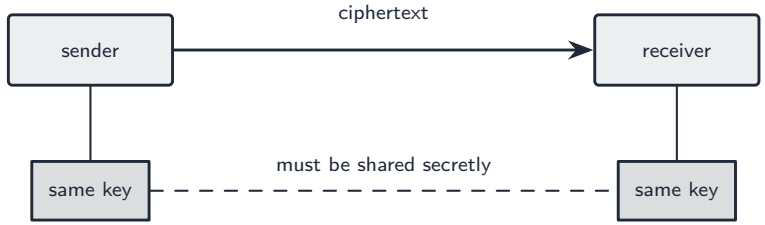


Encryption scrambles plaintext with a key; decryption reverses it

Symmetric encryption

Symmetric encryption 对称加密 (symmetric key cryptography) uses the **same key** for both encryption and decryption, so sender and receiver must both hold the secret key. It is **fast** and good for **bulk data** (a whole disk, a video stream). Its problem is **key distribution** 密钥分发: how do you share the key safely in the first place? Asymmetric encryption solves this.

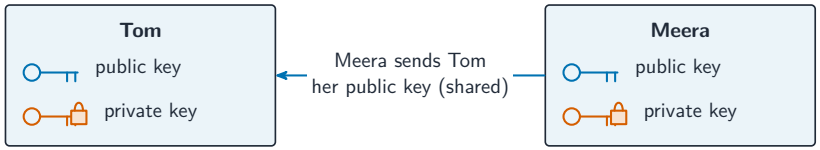
"Describe what is meant by symmetric key encryption" (two marks). *The same key is used to encrypt the plaintext and to decrypt the ciphertext*, so the key must be **shared** between sender and receiver and kept **secret** from everyone else. **Two drawbacks.** The key has to be **exchanged** before the message can be sent, and if it is intercepted in transit the interceptor can read every message; a separate key is needed for **every pair** of correspondents; and it gives no proof of **who** sent the message, because both ends hold the same key. **"Give two reasons for using key cryptography":** so that data is **unreadable** by anyone who intercepts it (confidentiality); so that the receiver can be sure the data came from the claimed sender and was **not altered** (authenticity and **integrity** 完整性). The two methods are symmetric and asymmetric key cryptography.



Symmetric encryption uses the same secret key at both ends

Asymmetric encryption (public-key)

Asymmetric encryption 非对称加密 (asymmetric key cryptography) gives each user a **pair** of related keys: a **public key** 公钥 they publish, and a **private key** 私钥 they keep secret. Data encrypted with the public key can be decrypted **only** with the matching private key, and vice versa.



encrypt with the recipient's **public key** — only their matching **private key** can decrypt

Each user has a public key to share and a private key to keep secret

To send a secret message to Alice: get her published public key, encrypt with it, and send. Only Alice — holding the matching private key — can decrypt. No prior key exchange is needed. The trade-off is that it is **much slower** than symmetric, so it is not used for large data.

"State what is meant by a private key." *A key known only to its owner (never transmitted), used to decrypt data that was encrypted with the matching public key, and to create digital signatures.* **"Describe the process of asymmetric encryption" (four marks):** (1) the receiver generates a **pair** of keys, a public key and a private key, mathematically related; (2) the **public key** is made available to anyone who wants to send to them; (3) the sender **encrypts** the plaintext with the receiver's **public key**; (4) the ciphertext can only be **decrypted with the receiver's private key**, which never leaves the receiver, so nobody who intercepts the message can read it.

Worked example. Fred wants to send Sheila a confidential document. Explain how asymmetric encryption is used.

Sheila has a key pair; she sends Fred her **public** key (or he obtains it from her certificate). Fred encrypts the document with **Sheila's public key** and sends the ciphertext. Only Sheila's **private key** can decrypt it, and only Sheila holds that, so nobody else, including Fred once it is encrypted, can read the document. The keys are used the **receiver's** way round: her public key to lock, her private key to unlock. An organisation that holds a key pair "to receive secure transmissions" does exactly this: it publishes the public key, keeps the private key, and decrypts what arrives.

Two differences between symmetric and asymmetric encryption. Symmetric uses **one** key for both directions; asymmetric uses **two** related keys, one to encrypt and the other to decrypt. In symmetric encryption the key must be **kept secret by both** parties and exchanged securely; in asymmetric encryption the public key can be **published** and only the private key is secret. Symmetric encryption is much **faster** and suits large amounts of data; asymmetric is slower, so it is used for keys and signatures rather than bulk data.

A private key must stay secret, so it is sometimes kept on a small **hardware security key** 硬件安全密钥. You plug it in or tap it to prove who you are, and the secret key

never leaves the device.

A hardware security key stores a secret key to prove who you are

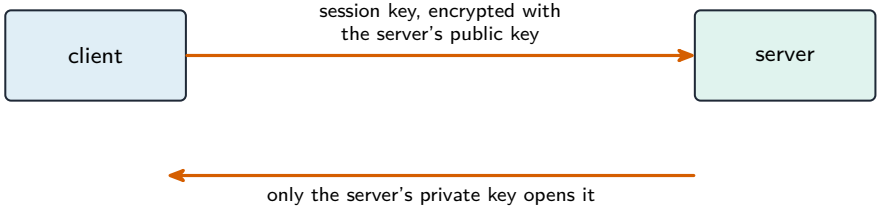
Hybrid approach (used by almost every real system)

Use asymmetric encryption to exchange a fresh **session key** 会话密钥, then use that symmetric key for the data:

1. the client makes a random session key.
2. it encrypts the session key with the server's public key.
3. the server decrypts it with its private key.
4. both ends now share the session key and use fast symmetric encryption for the rest.

This is how HTTPS and SSH work.

The exam's version of the key-exchange problem. "A symmetric key is to be exchanged before the message is sent. Explain how the key can be exchanged securely." The sender encrypts the **symmetric key** with the **receiver's public key** and sends it; the receiver decrypts it with their **private key**; both now hold the symmetric key, which was never exposed in transit, and use it for the messages. Asymmetric encryption solves the distribution problem; symmetric encryption then does the fast work.

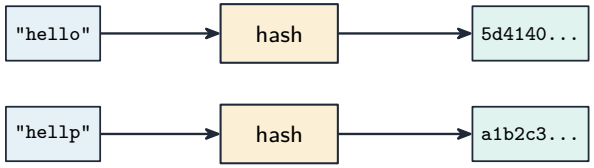


both ends now share the session key → fast **symmetric** encryption for all the data

The hybrid approach: asymmetric crypto shares a session key once, then fast symmetric encryption protects the data

Hashing (related, not encryption)

A **cryptographic hash** 密码散列 function takes any input and gives a fixed-size **digest** 摘要 such that the same input always gives the same digest, it is infeasible to find two inputs with the same digest, and a tiny change in input changes the digest completely. Hashing is **one-way** — you cannot get the input back. It is used for storing password checks, **integrity** checks, and digital signatures.



one letter changed → a completely different digest; one-way (cannot reverse)

A cryptographic hash gives a fixed digest; a tiny input change changes it completely, and it cannot be reversed

Quantum cryptography

Quantum cryptography uses the physics of light to distribute keys: the bits of a key are sent as **photons** whose quantum states encode the values. **"Describe its purpose":** to transmit an encryption key **securely**, in such a way that any attempt to intercept it can be **detected**, because measuring a photon **changes its state**; an **eavesdropper** 窃听者 therefore leaves evidence, and the corrupted key is thrown away and a new one sent. **Benefits:** interception is always detectable; the key cannot be copied without being altered; it is secure against future advances in computing power (a mathematical key can eventually be cracked, a quantum one cannot be read without disturbing it). **Drawbacks:** it needs specialised, **expensive** equipment; it works only over **limited distances** on dedicated optical fibre (or line of sight), not across the existing internet; it distributes the **key only**, so ordinary encryption still protects the message; and it is a **new** technology with few suppliers and little experience.

SSL / TLS

TLS 传输层安全 (Transport Layer Security, the successor to the **Secure Socket Layer**, SSL) is a **protocol** that gives encryption and authentication for data sent over a network. It **encrypts** the data in transit, **authenticates** the server with a certificate, and provides **integrity** (detecting tampering).

Outline of a TLS handshake:

1. the client connects and proposes cipher options.
2. the server picks one and sends its **digital certificate** (with its public key) — issuing and validating these certificates is **digital certification**.
3. the client checks the certificate.
4. the two ends exchange a fresh **session key** using asymmetric crypto.
5. all later traffic uses fast symmetric encryption with the session key.

The result is an encrypted, authenticated, integrity-checked tunnel for higher-level protocols (HTTP, SMTP). It is appropriate wherever **sensitive information** is sent: HTTPS web browsing, online banking and payments, secure email, and VPNs.

"Describe the purpose of SSL/TLS" and "state two functions." The purpose is to provide **secure communication** between a client and a server over a network. Its functions: it **encrypts** the data sent, so that it cannot be read if intercepted; it **authenticates** 认证 the server (and optionally the client) by means of a digital certificate, so the client knows it is talking to the genuine site; and it **checks the integrity** of the data, so that changes in transit are detected. Two examples of where it is appropriate: **online banking** and **online shopping** (card payments); also logins, private email, file transfer, VoIP and instant messaging: any transaction in which private data crosses the internet.

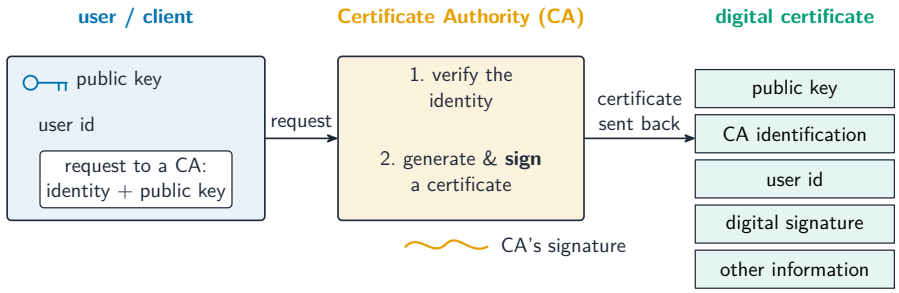
The two protocols that make up TLS. The **handshake** 握手 protocol sets up the session: it agrees the encryption algorithms (cipher suite), authenticates the server with its certificate, and exchanges the session key. The **record** protocol then carries the data: it encrypts each message with the session key, adds an integrity check, and passes it to the transport layer.

How a secure session starts: the certificate proves who the server is, the server's public key protects the session key on its way over, and the session key protects everything after that

"Explain how SSL/TLS is used when client–server communication is initiated" (six marks). (1) The client (browser) sends a request to the server for a **secure connection**, saying which encryption methods it supports. (2) The server sends back its **digital certificate**, which contains its **public key**. (3) The client **checks the certificate** is valid (issued by a trusted Certificate Authority, not expired, for the right domain). (4) The client generates a **session key**, encrypts it with the server's **public key** and sends it. (5) The server **decrypts** the session key with its **private key**. (6) Both sides now hold the session key and all further data is sent using **symmetric encryption** with it. Give the steps in this order; the marks are for the certificate, the public key, the session key and the switch to symmetric encryption.

Digital certificates

A **digital certificate** 数字证书 binds an identity (a domain, an organisation) to a public key, and is signed by a trusted **Certificate Authority** 证书颁发机构 (CA). It contains the subject (who it identifies), the subject's public key, the issuer (the CA), a validity period, and the CA's signature over all of it.



A Certificate Authority issues a digital certificate binding an identity to a public key

To verify one, the client (which holds a list of trusted root CAs):

1. checks the expiry dates.
2. checks the subject name matches the URL.
3. checks it is **signed by a trusted CA**, using the CA's public key to verify the signature.
4. follows the certificate chain up to a trusted root.

If anything fails, the browser shows the "Your connection is not private" warning. When it verifies cleanly, the client knows the identity was vetted by a trusted CA, the public key really belongs to that identity, and the certificate is current.

"Describe what is meant by a digital certificate" (two marks). *An electronic document, issued by a Certificate Authority, that verifies the identity of its owner (a person, organisation or website) and contains the owner's public key.* **Items found in one:** the **serial number**; the **name of the owner** (subject) and, for a website, its domain; the owner's **public key**; the name of the **issuing CA**; the **validity period** (dates); the signature **algorithm** used; and the **CA's digital signature** of the whole certificate.

"Explain how an organisation acquires a digital certificate" (four marks). (1) The organisation generates its own **key pair**, a public key and a private key. (2) It sends a **request** containing its public key and its **identity details** to a Certificate Authority. (3) The CA **verifies the identity** (checks that the applicant really is the organisation or owns the domain). (4) The CA creates the certificate containing the public key and the identity, **signs it with the CA's own private key**, and returns it. (5) The organisation installs the certificate on its server so that it can be sent to clients. The private key never leaves the organisation.

"Explain why a digital certificate is required to validate a digital signature." To check a signature the receiver needs the sender's **public key**, and needs to be sure that the key really **belongs to the claimed sender**; the certificate supplies the public

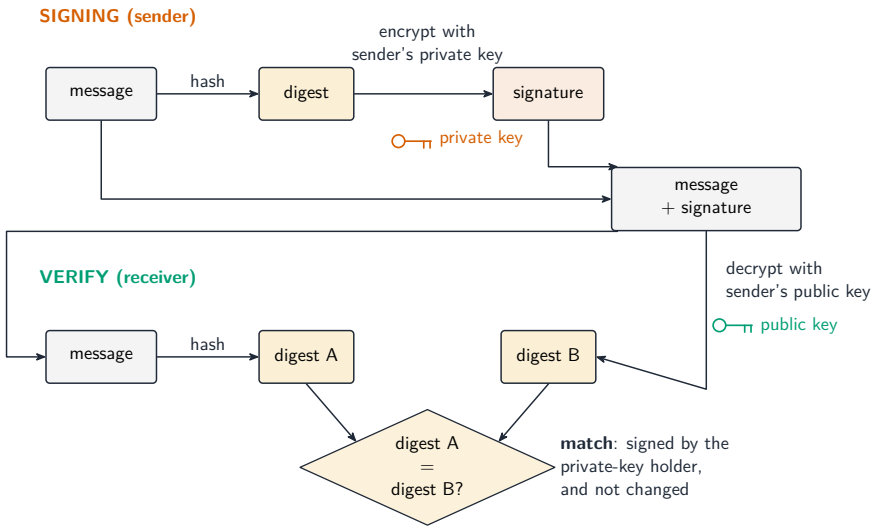
key together with the identity, and because the certificate is **signed by a trusted CA** the receiver can trust that binding. Without it an impostor could publish a public key in someone else's name and sign messages as them. The same reasoning answers "what should be included with a program downloaded from the internet to prove it is genuine": a **digital signature**, checked against the publisher's certificate.

Digital signatures

A **digital signature** 数字签名 proves who signed a message **and** that it was not changed. To sign:

1. compute a cryptographic hash of the message.
2. **encrypt the hash with the sender's private key** — that is the signature.
3. send the message and the signature.

To verify: compute the hash of the received message; **decrypt the signature with the sender's public key** to get the sender's hash; compare. If they match, the message was signed by the holder of the private key (**authentication** 身份验证) and was not changed (integrity). A signature does **not** hide the message — for confidentiality as well, encrypt **and** sign.



Signing hashes the message and encrypts the digest with the private key; the receiver checks it with the public key

"Explain the role of a digital certificate in creating a digital signature"

(three marks). The sender's certificate was issued by a CA and contains the sender's **public key** together with the sender's identity; the sender produces the signature by **hashing** the message and **encrypting the hash with their private key**, the partner of the key in the certificate; the receiver uses the **public key from the certificate** to decrypt the hash and, because the certificate binds that key to the sender, the signature proves **who** signed.

"Explain how a digital signature is used to verify a message" (four marks). (1) The receiver **decrypts the signature** with the sender's **public key** (taken from the sender's certificate), which yields the hash that the sender computed. (2) The receiver **hashes the received message** with the same hash algorithm. (3) The two hashes are **compared**. (4) If they **match**, the message came from the holder of the private key (**authentic**) and has **not been altered** since it was signed (**integrity**); if they differ, the message is rejected. A banker receiving confidential data with a signature does exactly this before trusting it; the data itself may separately be encrypted with the banker's public key for confidentiality.

Putting it together

A secure request to `https://www.bank.com`: the server sends its certificate; the client verifies it against trusted CAs; the client uses the server's public key to exchange a session key; then data flows encrypted with that key. Encryption stops eavesdroppers, the certificate proves the server's identity, and integrity checks stop a **man-in-the-middle** 中间人攻击 altering the data.

Worked example. Alice sends Bob a contract. She wants Bob to be certain it came from her and was not altered, **and** she wants nobody else to be able to read it. Which keys does she use, and in which direction? These are two different jobs needing two different key pairs. For the **signature** (authentication and integrity): Alice hashes the contract and encrypts that hash with **her own private key**; Bob decrypts it with **Alice's public key** and compares it against his own hash of the message. Only Alice holds her private key, so only she could have produced it. For **confidentiality**: Alice encrypts the contract itself with **Bob's public key**, so only **Bob's private key** can open it. One rule keeps all four straight: you **sign with your own private key** and **encrypt with the recipient's public key**. A signature on its own does **not** hide the message.

Definitions the examiner accepts

A definition question is marked against fixed wording. Learn these exactly, and give one answer only.

Term	Definition
encryption	converting plaintext into ciphertext using an algorithm and a key so that it cannot be understood if intercepted
plaintext / ciphertext	the original readable data / the encrypted, unreadable form of it
symmetric key cryptography	the same secret key is used to encrypt and to decrypt, so it must be shared securely by both parties
asymmetric key cryptography	a pair of related keys is used: the public key encrypts and only the matching private key decrypts
public key	a key made available to anyone, used to encrypt messages to its owner and to verify the owner's signatures
private key	a key known only to its owner, used to decrypt messages encrypted with the public key and to sign
SSL/TLS	protocols that provide secure (encrypted, authenticated, integrity-checked) communication between a client and a server
digital certificate	an electronic document issued by a Certificate Authority that verifies the owner's identity and contains their public key
digital signature	a hash of a message encrypted with the sender's private key, proving who sent it and that it is unaltered
Certificate Authority	a trusted organisation that verifies identities and issues and signs digital certificates
quantum cryptography	the use of quantum states of photons to distribute keys so that any interception is detected

Exam tips

- Symmetric: one shared secret key, fast, key exchange is the weakness. Asymmetric: public key to encrypt, private key to decrypt, slow, no exchange problem. Two differences, two drawbacks, two reasons: the exam asks for them in pairs.
- Confidentiality uses the **receiver's** keys (public to lock, private to unlock); a signature uses the **sender's** keys (private to sign, public to check). Say whose key every time.
- The TLS start-up is six steps: request, certificate with public key, check, session key encrypted with the public key, decrypted with the private key, symmetric encryption from then on.

- A certificate is identity plus public key, signed by a CA; acquisition is key pair, request, verification, signing, installation. It is needed to validate a signature because it proves whose public key it is.
- A signature is a hash encrypted with the private key; verification is decrypt, re-hash, compare. Integrity and authenticity are the two things it proves.
- Quantum cryptography distributes keys and detects eavesdropping; its limits are cost, distance and novelty.

Common mistakes

- Saying a message is encrypted with the sender's public key; the receiver's public key encrypts, the receiver's private key decrypts.
- Describing a signature as "encrypting the message with the private key" instead of encrypting its hash.
- Claiming a certificate contains the private key; it holds the public key and the identity, signed by the CA.
- Listing "the server sends its private key" in the TLS handshake; only the public key travels, inside the certificate.
- Giving "SSL/TLS makes the connection faster" as a function; its functions are encryption, authentication and integrity.
- Confusing hashing with encryption: a hash cannot be reversed and has no key; encryption is reversible with the key.
- Answering "why is a certificate needed for a signature" with "to encrypt it"; it is needed to trust the public key.