

Security

A-Level Computer Science 9618 · Topic 17 revision notes

Everything from the 3 lessons of this topic on a few pages: the key idea of each lesson, the diagram to remember, the points to learn and the traps examiners set. Read it, cover it, test yourself.

1. Symmetric and asymmetric encryption,
quantum cryptography.

2. SSL and TLS

3. Digital signatures and digital certificates

thelucidstem.com

1 Symmetric and asymmetric encryption, quantum cryptography

Syllabus 17.1

Encryption uses an algorithm and a **key** to turn **plaintext** into **ciphertext**; **decryption** reverses it. In **symmetric** cryptography the same secret key encrypts and decrypts, so the key must somehow be shared safely. In **asymmetric** cryptography each owner has a mathematically linked **public key** (given to anyone) and **private key** (kept secret): what one key encrypts, only the other can decrypt. To send a **private** message to someone, encrypt with **their public key**. To send a **verified** message to the public, encrypt with **your own private key**. **Quantum cryptography** uses photons to distribute keys so that any eavesdropping is detected.

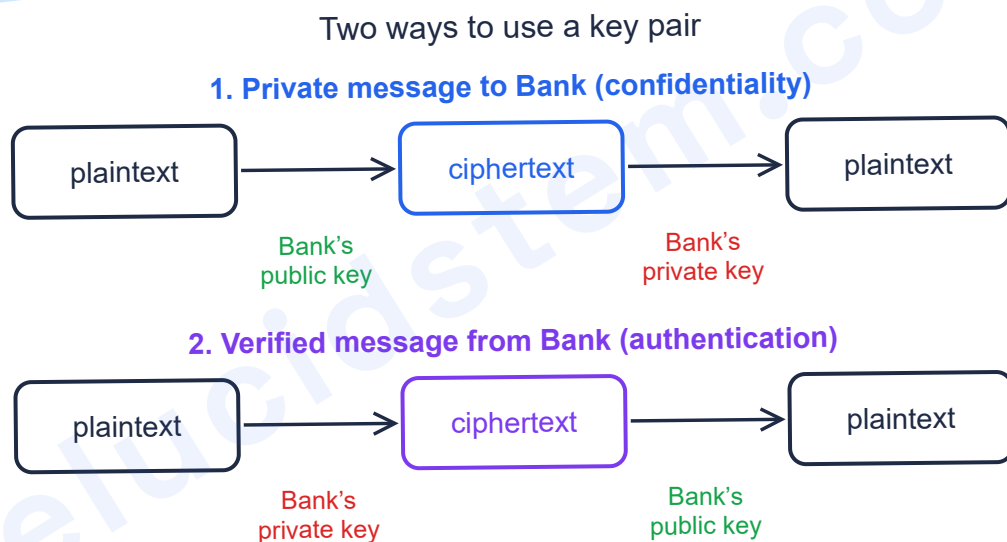


Fig. 1: the same key pair used two ways: the recipient's public key for privacy, the sender's private key for proof of origin

Worked example: A toy key pair (numbers far too small for real use)

A key pair uses $n = 55$, public key $e = 3$ and private key $d = 27$. Encrypt the value 8 to send privately, then decrypt it.

1. Encrypt with the recipient's public key: $\text{ciphertext} = 8^3 \text{ MOD } 55 = 512 \text{ MOD } 55$.
2. $512 = 9 \times 55 + 17$, so the ciphertext is 17.
3. Decrypt with the recipient's private key: $17^{27} \text{ MOD } 55 = 8$, the original plaintext.
4. An interceptor who knows $e = 3$ and $n = 55$ cannot decrypt 17 without finding d , which needs the factors of n (here 5 and 11; for a real key, impossible in practice).

Answer: Ciphertext 17; decrypting with the private key gives 8

Name the owner of the key.

- ✓ **Always say whose key:** "the receiver's public key", not just "the public key".
- ✓ **Privacy:** encrypt with the recipient's public key; decrypt with the recipient's private key.
- ✓ **Verification:** encrypt with the sender's private key; anyone decrypts with the sender's public key.
- ✓ **Quantum:** photons carry the key; eavesdropping changes their state, so it is detected.

Watch out!

"Encryption stops hackers getting the data" is wrong. The data can still be intercepted; encryption means it cannot be understood without the key.

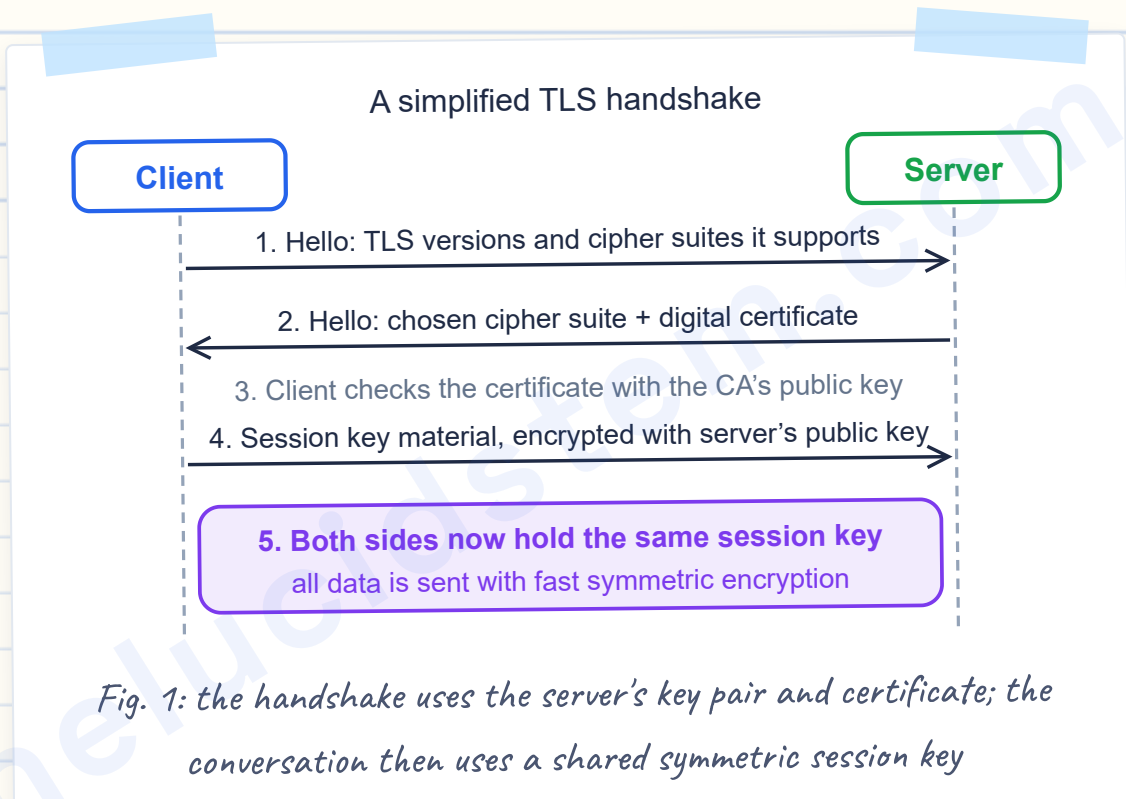
Never say the sender encrypts a private message with their own private key. Anyone could decrypt that with the sender's public key, so it gives authentication, not privacy.

Quantum cryptography does not stop someone intercepting the photons. It makes interception detectable, so the compromised key is thrown away before any data is sent with it.

2 SSL and TLS

Syllabus 17.1

SSL (Secure Socket Layer) and its successor **TLS** (Transport Layer Security) are **protocols** that provide a secure connection between two computers, normally a **client** and a **server**. Their purpose is to give **encryption** (data cannot be understood if intercepted), **authentication** (the server proves who it is with a **digital certificate**) and **data integrity** (any change in transit is detected). A **handshake** uses asymmetric encryption to agree a **session key**; the data is then encrypted with that **symmetric** key. HTTPS is HTTP running over SSL/TLS.



Worked example: Why two kinds of encryption?

Explain why TLS uses asymmetric encryption in the handshake but symmetric encryption for the web page data.

1. Symmetric encryption is fast, which matters for the large amount of data in a session, but both sides need the same secret key.
2. Sending that key in the clear would let an interceptor read everything.
3. Asymmetric encryption solves this: the key material is encrypted with the server's public key, so only the server's private key can recover it.
4. Asymmetric encryption is too slow for bulk data, so it is used only once, to set up the session key.

Answer: Asymmetric for safe key exchange, symmetric for speed

Certificate, key, session.

- ✓ **Purpose:** encryption, authentication of the server, integrity of data in transit.
- ✓ **Handshake:** client hello, server sends certificate (public key), client checks it.
- ✓ **Session key:** exchanged using the server's public key; only its private key can recover it.
- ✓ **Data:** then encrypted symmetrically with the session key, because symmetric is faster.

Watch out!

HTTPS is not a different protocol from HTTP that "is encrypted". It is ordinary HTTP carried inside an SSL/TLS connection; say SSL/TLS provides the encryption and authentication.

The session key is not the server's private key. The private key never leaves the server; a new session key is created for each session.

Name a situation and the reason: "online banking, because account details sent across the internet must not be readable if intercepted" earns more than "banking" on its own.

3 Digital signatures and digital certificates

Syllabus 17.1

A **digital certificate** is an electronic document, issued by a trusted **certificate authority (CA)**, that links an owner's identity to their **public key**. The CA signs it with the **CA's private key**. To make a **digital signature**, the sender **hashes** the message to produce a **message digest** and encrypts the digest with the **sender's private key**. The receiver decrypts the signature with the sender's public key (taken from the certificate), hashes the received message with the same algorithm and **compares** the two digests. A match proves the message is **authentic** and **unaltered**.

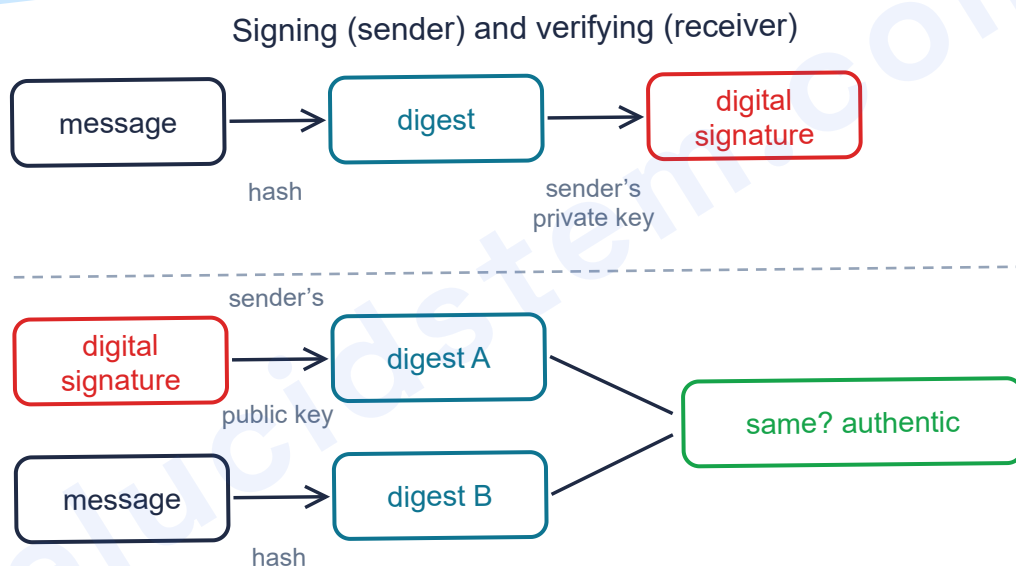


Fig. 1: the sender signs a digest with their private key; the receiver recomputes the digest and compares

Worked example: Signing with the toy key pair

A sender's key pair uses $n = 55$, public key $e = 3$, private key $d = 27$. Suppose the digest of a message is 8. Produce the signature and show how the receiver verifies it.

1. Sign: encrypt the digest with the sender's private key: $8^{27} \text{ MOD } 55 = 2$. The signature is 2.
2. The receiver decrypts the signature with the sender's public key: $2^3 \text{ MOD } 55 = 8$ (digest A).
3. The receiver hashes the message it received and gets 8 (digest B).
4. Digest A = digest B, so the message is authentic and unaltered. If the message had been changed, digest B would differ.

Answer: Signature 2; it verifies because $2^3 \text{ MOD } 55 = 8$ matches the recomputed digest

Hash, sign, compare.

- ✓ **Certificate:** applicant sends public key and identity to a CA; CA verifies identity and signs with the CA's private key.
- ✓ **Sign:** hash the message to a digest; encrypt the digest with the sender's private key.
- ✓ **Verify:** decrypt the signature with the sender's public key (from the certificate); re-hash the message; compare.
- ✓ **Provides:** authentication, integrity, non-repudiation; not confidentiality.

Watch out!

A certificate never contains the owner's private key. It carries the public key; the private key must stay secret with its owner.

The receiver does not "decrypt the message to check the hash". They decrypt the signature with the sender's public key and re-hash the message, then compare the two digests.

A digital signature is not a scanned image of a handwritten signature. It is an encrypted message digest that depends on both the message and the private key.