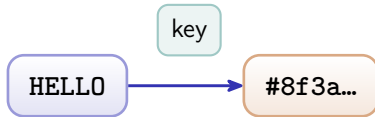


Security

A-Level Computer Science ■ Topic 17

A-Level Computer Science



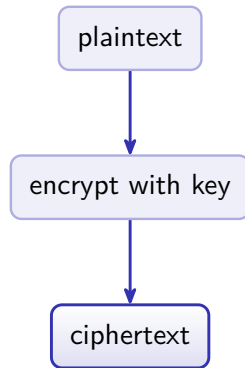
What we will cover

- 1 How encryption works
- 2 Symmetric, asymmetric, hybrid
- 3 Hashing
- 4 TLS, certificates, signatures
- 5 Recap



Plaintext in, ciphertext out

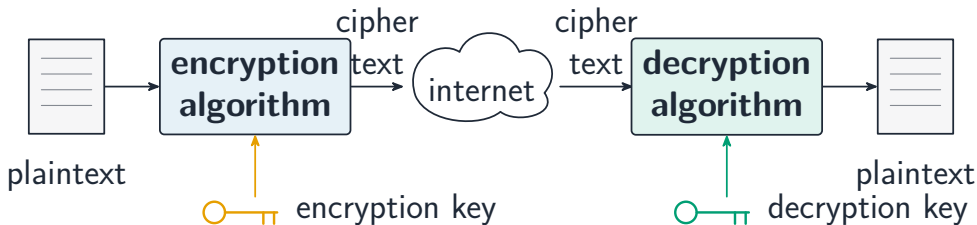
Encryption turns readable **plaintext** into unreadable **ciphertext** using a maths operation that depends on a **key**. **Decryption** reverses it – but only with the right key.



An interceptor without the key sees meaningless data. Trying every key would take far too long.

The encryption process

Plaintext and a key go in; ciphertext comes out. Security rests on the **key**, not on hiding the algorithm.



Symmetric vs asymmetric

symmetric

same **key** for encrypt and decrypt

fast – good for bulk
data (a disk, a video)

problem: **key distribution**

how do you share the key
safely in the first place?

asymmetric

a **pair**: **public key** pub-
lished, **private key** secret

encrypt with one, de-
crypt only with the other

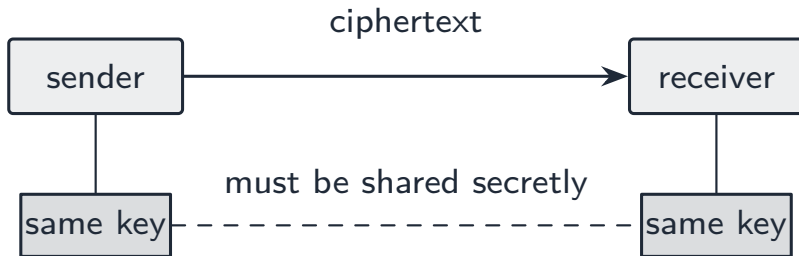
no prior key exchange

much slower – not for large data

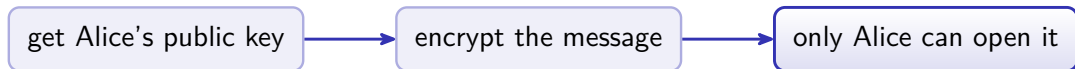
In practice both are used together: the slow asymmetric pair carries a fresh symmetric key, and the fast symmetric key then carries the data

One shared key

Symmetric encryption is fast, but both sides need the same secret – and getting it to them is the problem.



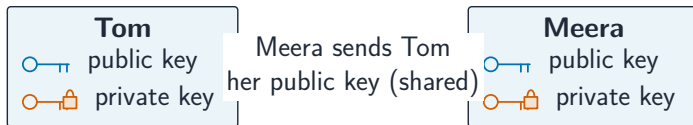
Sending a secret to Alice



One rule keeps the four keys straight: **encrypt** with the recipient's public key; **sign** with your own private key.

A public and a private key

Encrypt with the recipient's **public** key; only their private key undoes it, so nothing secret is ever sent.



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

The hybrid approach – how HTTPS actually works

Use asymmetric crypto **once** to share a **session key**, then fast symmetric crypto for the data.

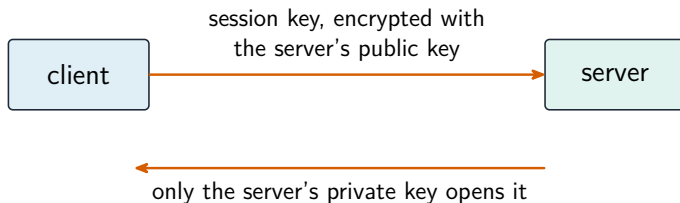
- 1 client makes a random session key
- 2 encrypts it with the server's public key
- 3 server opens it with its private key
- 4 both ends use fast symmetric encryption



The Enigma machine encrypted messages in the Second World War – an early, mechanical cipher devi

Hybrid: the best of both

Asymmetric carries a fresh symmetric key; the bulk of the traffic is then encrypted fast.



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

Where the private key lives, and what TLS gives you

asymmetric encryption

a public key encrypts, and only the matching private key decrypts

the private key

must stay secret – it is sometimes held on a hardware security key, so it never touches the disk

why hardware

a key that cannot be copied off the device cannot be stolen by malware

what TLS produces

an encrypted, authenticated, integrity-checked tunnel for higher-level protocols to run inside

each word matters

encrypted – unreadable; authenticated – you know who you reached; integrity-checked – unaltered in transit

quantum cryptography

threatens today's asymmetric algorithms, and offers key exchange that detects eavesdropping

HTTPS is HTTP inside that tunnel. Naming all three properties – not just “it is encrypted” – is what the question is asking for.

Hashing is one-way – it is not encryption

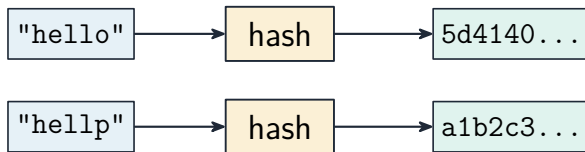
A **cryptographic hash** maps any input to a fixed-size **digest** .



- same input $\Rightarrow$ same digest; a tiny change $\Rightarrow$ a totally different digest
- infeasible to find two inputs with the same digest
- you **cannot** get the input back – used for passwords, **integrity** , signatures

A one-way function

Easy to compute, infeasible to reverse, and any change to the input changes the digest completely.



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



TLS – an encrypted, authenticated tunnel

TLS (successor to SSL) encrypts data in transit, authenticates the server, and detects tampering.

- 1 client connects and proposes ciphers
- 2 server picks one and sends its **digital certificate**
- 3 client checks the certificate
- 4 they exchange a fresh session key (asymmetric)
- 5 all later traffic uses fast symmetric encryption

What a digital certificate contains

A certificate binds an identity to a public key, and is signed by a trusted **Certificate Authority** (CA).

subject

public key

issuer (CA)

validity

CA signature

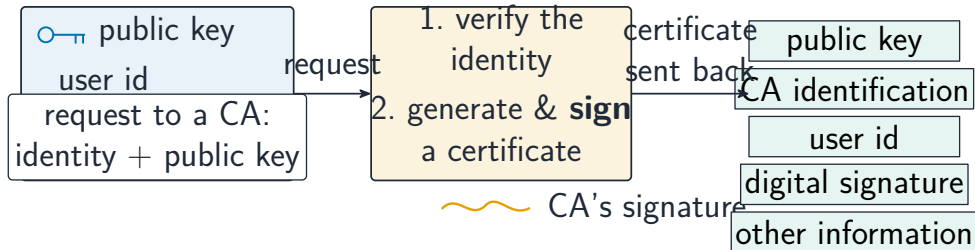
To verify:

check expiry, check the name matches the URL, check it is signed by a trusted CA, follow the chain to a trusted root. Fail any step $\Rightarrow$ "Your connection is not private".

What a certificate carries

A public key, an identity, and a CA's signature binding the two – which is what your browser checks.

user / client **Certificate Authority (CA)** **digital certificate**



Digital signatures – who sent it, and was it changed?

To sign

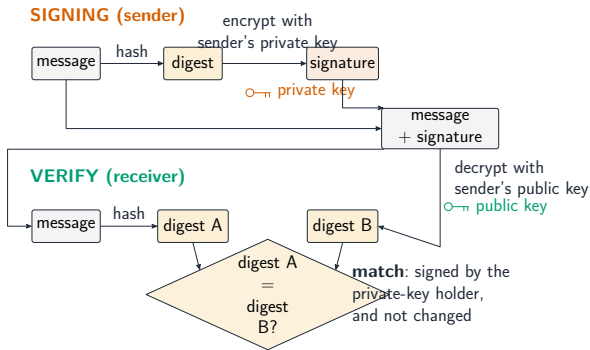
- 1 hash the message
- 2 encrypt the hash with the sender's private key
- 3 send the message and the signature

To verify

- 1 hash the received message
- 2 decrypt the signature with the sender's public key
- 3 compare the two digests

A match proves **authentication** (only the private-key holder could have signed) *and* integrity (the message was not changed). A signature does **not** hide the message.

Digital signatures



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

Putting it together

A secure request to `https://www.bank.com`: the server sends its certificate; the client verifies it against trusted CAs.

Encryption stops eavesdroppers, the certificate proves the server's identity, and integrity checks stop a **man-in-the-middle** altering the data.

Worked example – a contract for Bob

Alice sends Bob a contract. She wants him certain it came from her and was not altered, *and* she wants nobody else to read it. Which keys?

Signature (authentication + integrity): Alice hashes the contract and encrypts that hash with **her private key**. Bob checks with **Alice's public key**.

Confidentiality: Alice encrypts the contract with **Bob's public key**, so only **Bob's private key** can open it.

Sign with your own private key; encrypt with the recipient's public key.

Exam tips

- Distinguish **symmetric** encryption (one shared key, fast) from **asymmetric** (a public/private key pair).
- Explain the **TLS handshake** and why a **digital certificate** from a CA proves identity.
- Explain a **digital signature**: hash the message, then encrypt the hash with the private key – it proves integrity and origin.



Topic 17 – key takeaways

1 Keys

symmetric = one shared key;
asymmetric = a pair

2 Hybrid

asymmetric shares a session key;
then symmetric

3 TLS

certificate from a CA proves the
server's identity

4 Sign

hash, encrypt with private key;
does not hide

Check yourself

- 1 Why is symmetric encryption used for bulk data?
- 2 Is a hash reversible?
- 3 What does a CA's signature on a certificate prove?
- 4 Which key do you use to *sign*?

Answers 1. it is fast 2. no – it is one-way 3. the public key belongs to that identity
4. your own private key