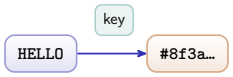
Security

A-Level Computer Science ▪ Topic 17

A-Level Computer Science



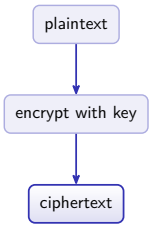
What we will cover

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

1 Encryption

Plaintext in, ciphertext out

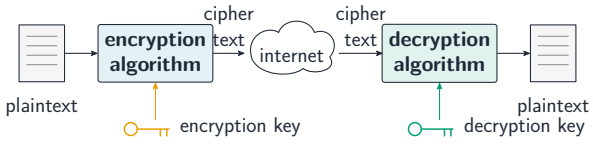
Encryption 加密 turns readable **plaintext** 明文 into unreadable **ciphertext** 密文 using a maths operation that depends on a key. **Decryption** 解密 reverses it – but only with the right key.



> 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
对称加密

same key for encrypt and decrypt

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

problem: **key distribution** 密钥分发

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

asymmetric
非对称加密

a pair: **public key** 公钥 published, **private key** 私钥 secret

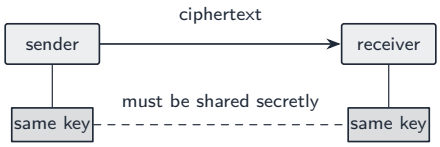
encrypt with one, decrypt only with the other

no prior key exchange

much slower – not for large data

One shared key

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



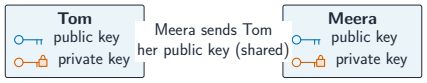
Sending a secret to Alice

get Alice's public key → encrypt the message → only Alice can open it

> 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

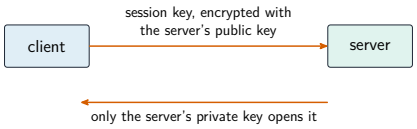
The hybrid approach – how HTTPS actually works

Use asymmetric crypto once to share a **session key** 会话密钥, 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

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

Hashing is one-way – it is not encryption

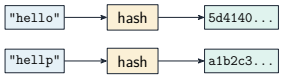
A **cryptographic hash** 密码散列 maps any input to a fixed-size **digest** 摘要.

`hello`	`hellp`
`a94a8fe5…`	`7d793037…`

- same input ⇒ same digest; a tiny change ⇒ a totally different digest
- infeasible to find two inputs with the same digest
- you cannot get the input back – used for passwords, **integrity** 完整性, 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)

2 TLS and certificates

TLS – an encrypted, authenticated tunnel

TLS 传输层安全 (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** 数字证书
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** 证书颁发机构 (CA).

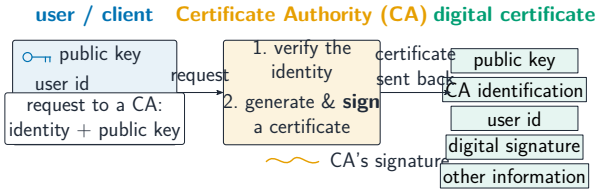
subject | **public key** | **issuer (CA)** | **validity** | **CA signature**

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 ⇒ "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.



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** 身份验证 (only the private-key holder could have signed) *and* integrity (the message was not changed). A signature does not hide the message.

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.

3 Recap

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