Encryption — Symmetric vs Asymmetric

One key or two keys? Learn how symmetric and asymmetric encryption work, and why the internet uses both together.

Introduction

In the last lesson you locked a message with one key — the same number locked it and unlocked it. That is **symmetric encryption**: one key, shared by both sides. It is fast and strong, but it has one hard problem: how do two strangers share the secret key without anyone else hearing it?

Asymmetric encryption solves that problem with two keys instead of one: a **public key** you can share with the whole world, and a **private key** you never share with anyone. A message locked with the public key can only be opened with the private key. Now a stranger can send you a secret without you ever meeting to share a key.

The real internet uses both: asymmetric encryption to share a secret safely, then symmetric encryption for the fast work that follows. That combination is called **hybrid encryption**, and it is what happens every time you open a website with a little padlock in the address bar.

In this lesson you will see both kinds, learn their strengths and weaknesses, and trace what happens when your browser opens a secure website.

(Roman Urdu: Symmetric mein aik hi chabi lock aur unlock dono karti hai — tez hai magar chabi share karna mushkil. Asymmetric mein do chabiyan: public (sab ko de do) aur private (kabhi kisi ko na do). Internet dono ko mila kar use karta hai.)

Beginners often ask which one is "better". Neither — they solve different problems. Symmetric is a fast lock with a key-sharing problem; asymmetric is a slow lock with no key-sharing problem. Together they cover each other's weakness.

Key points

- **Symmetric encryption uses one shared secret key.** The same key encrypts and decrypts. Examples: AES (the world standard today), ChaCha20.
- **Symmetric is fast.** It can encrypt a whole video call in real time. Its weakness is key distribution: getting the key to the other person safely.
- **Asymmetric encryption uses a key pair: public + private.** Anyone may lock a message with your public key; only your private key opens it. Examples: RSA, ECC (elliptic curve).
- **Asymmetric is slow.** It is used for small things — sharing a key, signing a message — not for encrypting a whole film.
- **A public key is not a secret; a private key is the crown jewel.** If your private key leaks, anyone can pretend to be you. Guard it like your house key.

- **Hybrid encryption is the real-world answer.** Asymmetric encryption safely delivers a fresh symmetric key; then symmetric encryption does the heavy work. This is what TLS (the padlock in your browser) does.
- **Digital signatures use the keys the other way round.** Locking with the private key proves the message came from you — anyone can open it with your public key and check it was really yours (and unchanged).
- **Key length matters.** AES-256 and RSA-2048 are today's safe sizes; short keys from the 1990s are broken now.

Examples

Example 1 — Symmetric: one key, both sides

```
Ali and Sara share one secret key: K

Ali:   lock("Meet at 5") with K  ->  "X9#mQ2z"
Sara:  unlock("X9#mQ2z") with K  ->  "Meet at 5"
```

Fast and simple. But how did Sara get K? If Ali emailed it, anyone reading the email has it too. That delivery problem is the whole weakness.

Example 2 — Asymmetric: two keys, no meeting needed

```
Sara's keys:  public key P (on her profile), private key S (only in her phone)

Ali:   lock("Meet at 5") with P  ->  "Z7!kP1q"
Sara:  unlock("Z7!kP1q") with S  ->  "Meet at 5"
```

Ali never needed to meet Sara or receive a secret. He used her public key — which is public. Only Sara's private key opens the message, and only Sara has it.

Example 3 — Hybrid: what your browser really does (TLS)

```
1. Browser asks the website for its public key (in a certificate).
2. Browser makes a fresh symmetric key K for this visit only.
3. Browser locks K with the website's public key and sends it.
4. Only the website's private key can open K.
5. Both sides now share K — and use fast symmetric encryption
   for the whole visit.
```

The slow asymmetric step happens once; the fast symmetric step carries the whole page, the photos, and the video. Best of both.

Example 4 — A digital signature (keys reversed)

```
Sara signs:  lock(hash of "I agree") with her PRIVATE key S
Anyone:      unlock with her PUBLIC key P, compare the hash
Result:      only Sara could have made it (only she has S),
             and the message was not changed (the hash matches).
```

This is how software updates prove they really came from the maker, and how the "integrity" part of the CIA Triad (CS.2) is delivered.

Example 5 — Symmetric encryption in Python (runnable)

```
# A tiny XOR demo — NOT real security, just to see one key do both jobs.
def xor_crypt(text, key):
    return "".join(chr(ord(c) ^ key) for c in text)

secret = xor_crypt("Meet at 5", 42)
print(repr(secret))
print(xor_crypt(secret, 42))
```

What you will get:

```
'... scrambled characters ...'
Meet at 5
```

The same key (42) locks and unlocks — that is symmetric encryption in miniature. Real apps use AES instead of XOR, because XOR with one small key is trivially broken.

Quick recap

- Symmetric: one shared key, fast, but the key must be delivered safely.
- Asymmetric: public key locks, private key opens; slow, but no key delivery problem.
- Hybrid: asymmetric delivers a fresh symmetric key, symmetric does the heavy work — this is TLS.
- A signature locks with the private key and proves who sent it.
- Public keys are public; private keys are never shared.

Common mistakes

1. **Sharing your private key** — even with a friend, even "just once". Fix: a private key is shared with nobody, ever.
2. **Using asymmetric encryption for everything** — it is hundreds of times slower than symmetric. Fix: use it to deliver a key, then switch to symmetric.

3. **Thinking a public key must be hidden** — it is designed to be public; hiding it adds nothing. Fix: publish it freely, protect the private one.
4. **Reusing one symmetric key forever** — if it leaks, every past message opens. Fix: fresh keys per session, the way TLS does it.
5. **Trusting a public key without checking whose it is** — an attacker can hand you their own public key and pretend to be the bank. Fix: certificates and trusted authorities check the identity behind the key.

Practice

1. In two lines, explain symmetric encryption to a friend using a house key: one key, two people, one problem (how does the second person get a copy?).
2. Explain asymmetric encryption with a mailbox: anyone can drop a letter in (public key), only the owner with the mailbox key can take letters out (private key).
3. Open any https website, click the padlock, and find the word "TLS" or "certificate". Write down what the certificate says about who the site is.
4. Run the XOR demo above with your own message and key 7. Then try to unlock it with key 8 — what do you get, and why?
5. Name one job for each: one thing symmetric encryption is used for, one thing asymmetric encryption is used for, and one thing a digital signature is used for.

Practice quiz

1. What is the main weakness of symmetric encryption?

- It is too slow
- The shared key must be delivered safely to the other person
- It needs two keys
- It cannot encrypt text

Answer: The shared key must be delivered safely to the other person

2. In asymmetric encryption, which key locks a message for you?

- Your private key
- Your public key — anyone may use it to lock a message only you can open
- A shared secret key
- The website's password

Answer: Your public key — anyone may use it to lock a message only you can open

3. What is hybrid encryption, as used by TLS?

- Using two passwords at once
- Asymmetric encryption delivers a fresh symmetric key, then symmetric encryption does the heavy work

- Encrypting everything twice with the same key
- Mixing hashing and encryption into one step

Answer: Asymmetric encryption delivers a fresh symmetric key, then symmetric encryption does the heavy work

---

Watch the Code Scope channel: https://www.youtube.com/@CodeScopePK — this lesson's own video joins the channel when it is recorded.

Read online: https://scopetreasure.com/notes/cs-encryption-symmetric-asymmetric · More free tools and notes at https://scopetreasure.com