Introduction to Cryptography:
Keys and Encryption

- - Core scenario: a sender and receiver communicating over an insecure channel, watched by an eavesdropper
- - Plaintext: readable data; Ciphertext: scrambled, unreadable output
- - Encryption: scrambling plaintext; Decryption: unscrambling ciphertext back to readable form
- - The key uniquely controls the transformation, acting as the essential secret ingredient
- - True security depends on key secrecy and handling, not just on the algorithm itself

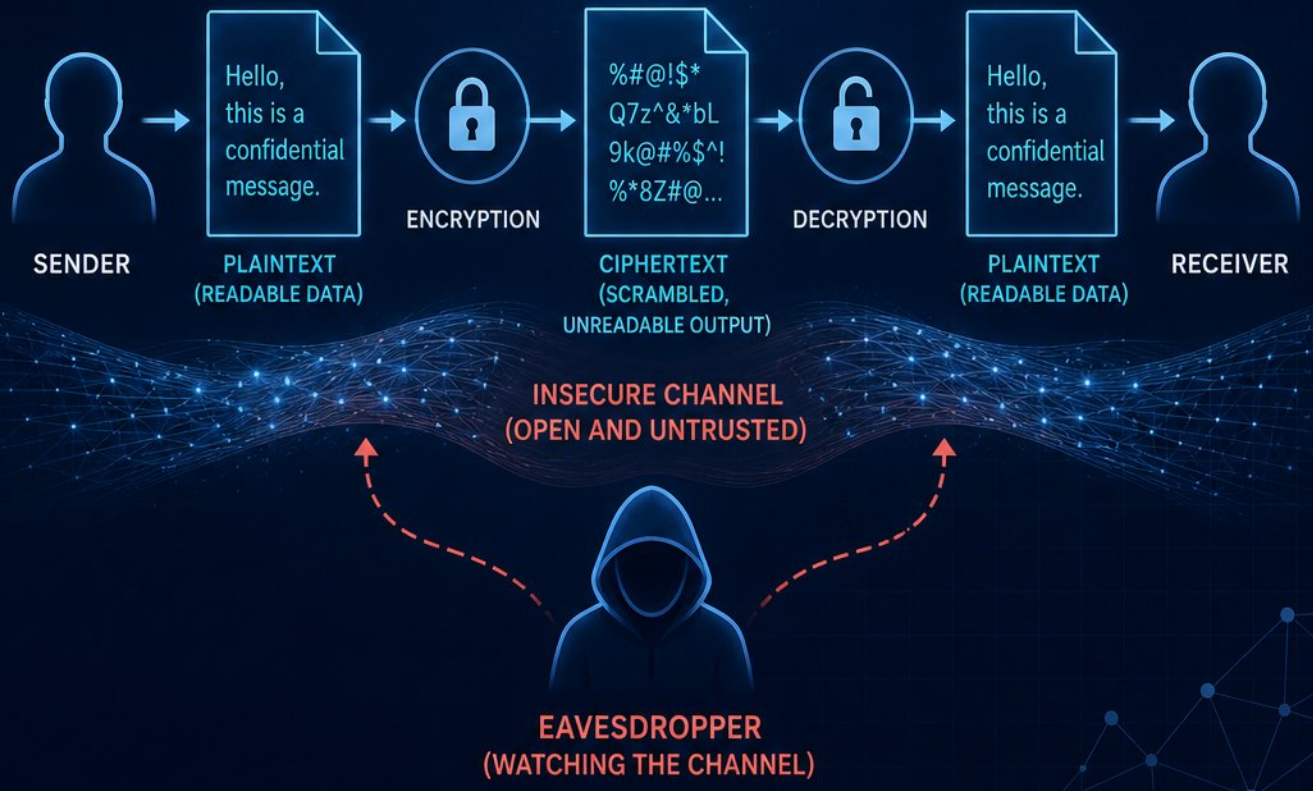
$$C = E_K(P)$$

$$P = D_K(C)$$

$$E_K : P \to C$$

$$D_K : C \to P$$

$$\sum_{i=1}^{n} x_i^2$$



A Simple Mental Model: The Lock and Key Analogy

- A lockable box represents encryption: only the correct key locks or unlocks it
- The lock is the algorithm, the physical key is the cryptographic key
- The item placed inside is plaintext; the locked box is ciphertext
- Physical keys can be stolen or copied—just like digital keys
- This simple model scales to secure all modern digital communication

$$E_k(m) = c$$
$$D_k(c) = m$$
$$c = \sum_{i=0}^{n-1} m_i \cdot 2^i \mod N$$
$$H(x) = h$$
$$p, q \text{ prime}$$
$$N = pq$$
$$\phi(N) = (p-1)(q-1)$$
$$c \in Z_N$$
$$k \in_R \{1, \ldots, \phi(N)-1\}$$

The Encryption Workflow: Scrambling and Unscrambling Data

- Plaintext + Key → Encryption Algorithm → Ciphertext
- Ciphertext + Key → Decryption Algorithm → Plaintext
- Without the correct key, decryption must be computationally infeasible
- Example: Caesar cipher with key=3 shifts A → D, B → E
- The algorithm is public; the key value must remain secret

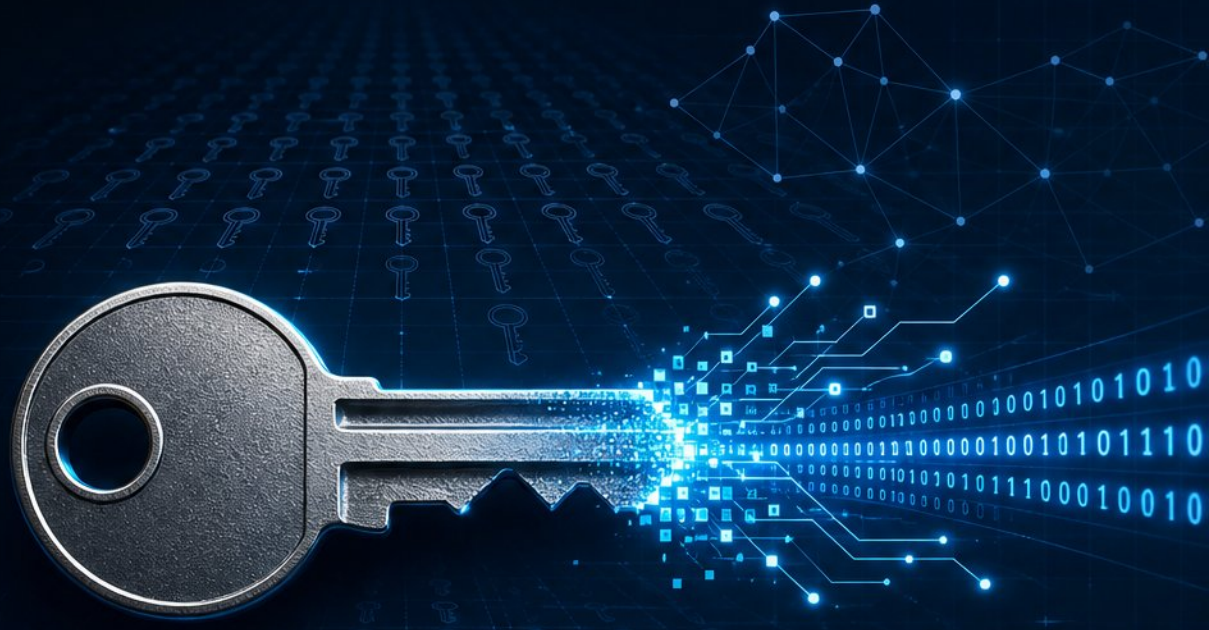
What Exactly Is a Key?
Secrecy, Uniqueness, and Strength

$$N = 2^n$$

$$|K| = 2^n$$

$$S = \{k_1, k_2, \ldots, k_N\}$$

- A key is a secret value giving encryption its unique 'personality'
- Keyspace: the complete set of all possible keys for an algorithm
- Longer keys create exponentially larger keyspaces, blocking brute-force attacks
- Example: 8-bit key = 256 possibilities; 128-bit key = 2^128 combinations
- The algorithm is the rulebook; the key is your secret choice from a vast library



$$T \propto 2^n$$

Symmetric Encryption:
One Key to Rule Them All

$$E_k(P) = C$$
$$D_k(C) = P$$

- Uses a single, shared secret key to encrypt and decrypt data
- Sender encrypts plaintext with the key; receiver decrypts ciphertext with same key
- AES is the global standard, encrypting 128-bit blocks with 128-, 192-, or 256-bit keys
- Fast and efficient, but burdened by the 'key distribution problem'
- Key challenge: securely sharing the secret key over insecure channels

Asymmetric Encryption: A Revolutionary Key Pair

- Uses a mathematically linked public key and private key pair
- Public key encrypts; only the matching private key can decrypt
- Public key can be freely shared with everyone
- Eliminates the risky need to share a secret key beforehand
- RSA and Elliptic Curve Cryptography (ECC) are common algorithms

Beyond Secrecy: How Keys Prove Your Identity

$$e(m,n)^d \equiv m \pmod{n}$$

- Asymmetric keys serve a dual purpose: confidentiality and authentication
- Digital signatures: signing with a private key creates verifiable proof of origin
- Workflow: signer uses private key; verifier uses the signer's public key
- This is not separate technology—just a fundamental property of the same key pair

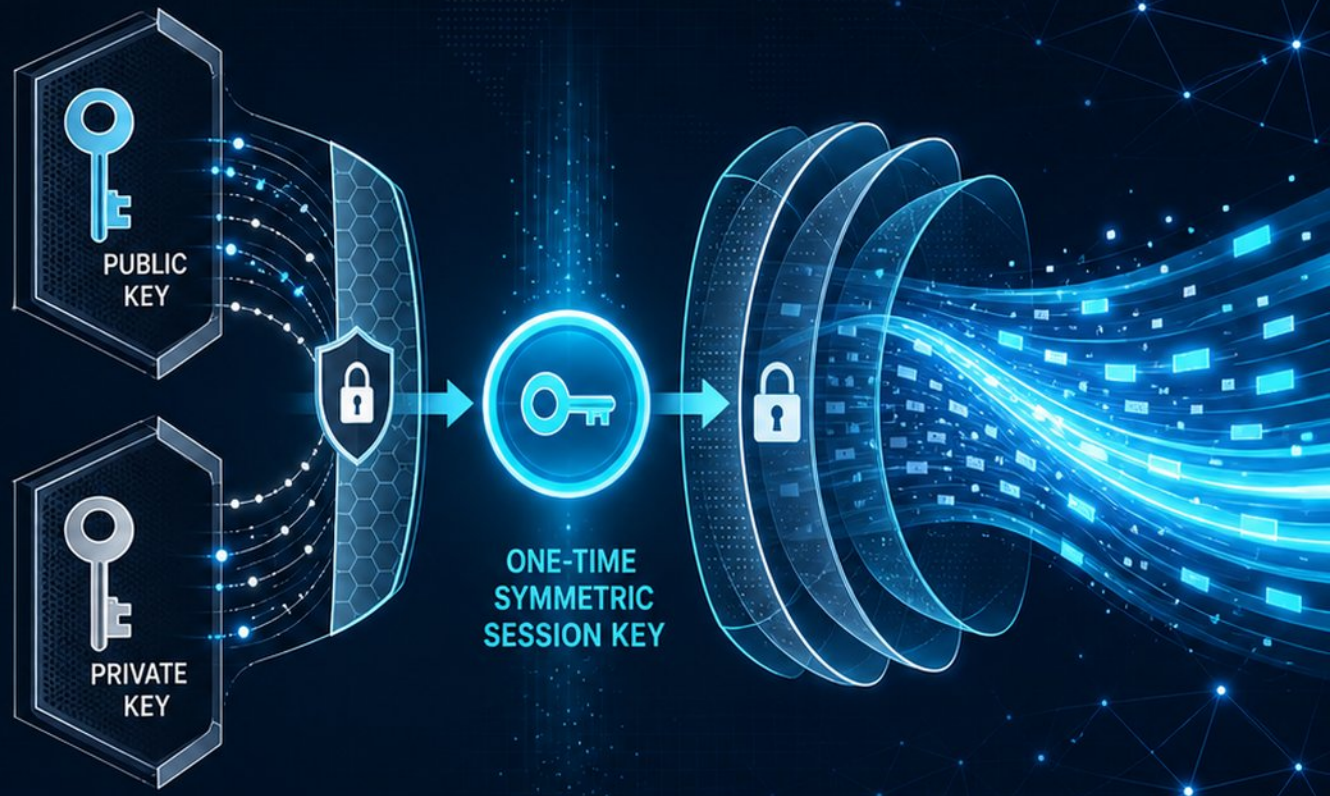
$$D_{k_{pub}}(E_{k_{priv}}(m)) = m$$

$$H(m) \rightarrow s = \text{Sign}_{k_{priv}}(H(m))$$

$$\text{Verify}_{k_{pub}}(m, s) = true$$

Practical Cryptography: The Hybrid Approach

- Asymmetric encryption is slow for bulk data; symmetric is fast but hard to share
- To solve this, use asymmetric encryption to securely exchange a one-time symmetric session key
- Then use that fast symmetric session key to encrypt the actual bulk data transfer
- Example: HTTPS/TLS—the browser padlock icon. An asymmetric handshake sets up a safe session, then fast symmetric encryption protects all subsequent data



$E_k(M) = C$

$c = E_k(m)$

$D_k(C) = M$

$H(x) = -\sum p(x) \log p(x)$

$C = E_{k_{sess}}(M)$

The Achilles' Heel: Why Key Management Is Everything

- Key management governs the entire key lifecycle: generation, storage, distribution, use, rotation, backup, and destruction
- Even the strongest algorithm is useless if keys are mishandled, stolen, or lost
- Key management is the human, process, and policy side of cryptography—often the hardest part
- Most real-world security failures trace back to key mismanagement, not broken algorithms

$E_K(m) = C$

$H(K) = -\sum_i p_i \log p_i$

$\forall x \in M, D_K(E_K(x)) = x$

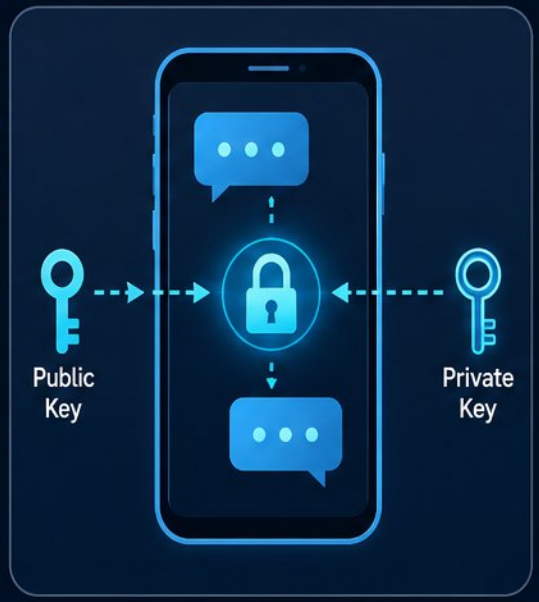
Managing Keys: Common Pitfalls and Disasters

- Hardcoded keys in source code or config files become public during a breach.
- No inventory leads to unmanaged, forgotten keys that protect critical data.
- Keys that never rotate extend the damage window of a silent compromise.
- Manual handling, like emailing keys, creates untraceable human-error risks.
- Orphaned keys without clear ownership break processes and block audits.

Key Management Best Practices:
Security by Design

- Use centralized KMS and HSMs to prevent direct exposure of key material in software
- Automate the entire lifecycle: generation, distribution, rotation, and secure destruction
- Enforce split knowledge and dual control; never allow one person to control a critical key
- Maintain a complete, centralized inventory and immutable audit trail for every key event
- Apply least privilege and role-based access for all cryptographic key operations

Building Your Mental Model: Real-World Scenarios



- **Secure Messaging:** Asymmetric key pair verifies identity, then symmetric session key encrypts chat messages.
- **Online Banking:** Asymmetric digital certificate authenticates the bank, then symmetric key transfers data.
- **Software Updates:** Developer signs with private key; your device verifies with public key.
- **Same hybrid model powers all three:** public-key handshake, symmetric speed.

Facing the Future:
Cryptography in a Post-Quantum World

TODAY

Encrypted data created today

"HARVEST NOW, DECRYPT LATER" VAULT

Adversaries store encrypted data today, with the goal of decrypting it later.

FUTURE QUANTUM COMPUTER

Large-scale quantum computers could break classic asymmetric cryptography.

POST-QUANTUM PROTECTION

Post-Quantum Cryptography (PQC) resists quantum attacks and secures the future.

- Large-scale quantum computers could break asymmetric algorithms like RSA and ECC
- "Harvest Now, Decrypt Later" threat stores data today for future quantum decryption
- Symmetric AES-256 remains strong, retaining 128-bit quantum security
- Industry transition to Post-Quantum Cryptography (PQC) is underway now

Summary and Your Path Forward

- Journeyed from plaintext and ciphertext to the hybrid encryption powering HTTPS
- Core insight: system security rests entirely on key secrecy and proper management
- Next step: Coursera's 'Cryptography I' by Dan Boneh for structured depth
- Next step: 'Cryptopals' challenges to learn by breaking real crypto
- Next step: NIST SP 800-57 for foundational key management standards

Deepen Knowledge
Build strong theoretical foundations in cryptography.

Sharpen Skills
Practice and solve challenges to develop hands-on expertise.

Apply Standards
Adopt proven standards to manage keys securely and effectively.

PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/2cebcf3e/public