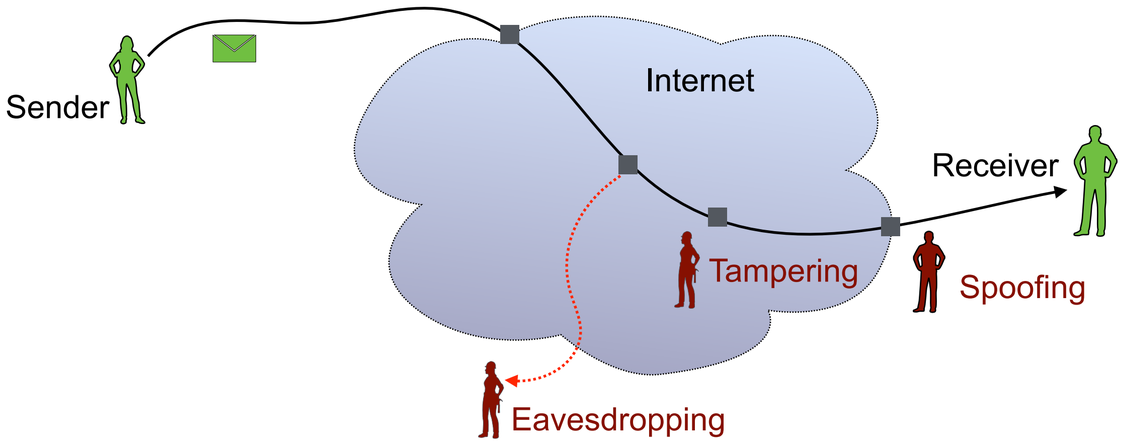
Principles of Secure Communication

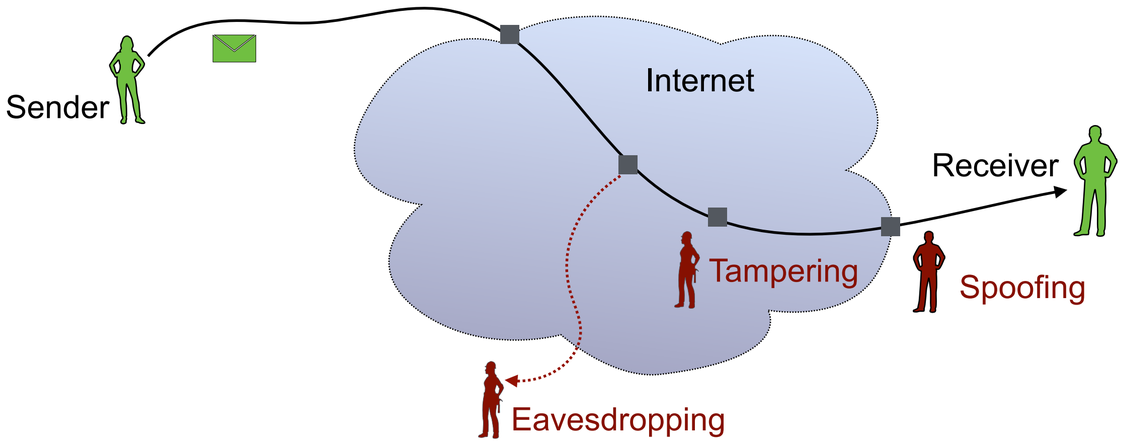
- Ensuring Confidentiality
- Authenticating Messages
- Validating Identity

Goals of Secure Communication



- Goal: deliver message from sender to receiver
  - Avoid eavesdropping → encrypt to provide confidentiality
  - Avoid tampering → authenticate to ensure the message is not modified in transit
  - Avoid spoofing → validate identity of sender

How to Provide Confidentiality? (1/2)



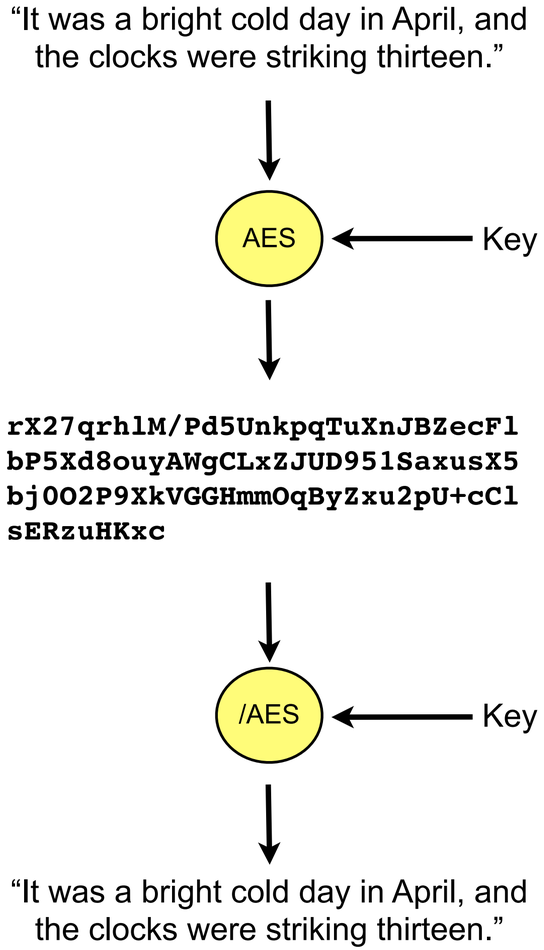
- Data traversing the network can be read by any device on the path
  - Can eavesdrop on packets as they traverse a link
  - Configure a switch or router to snoop on data as it's forwarded between links
- The network operator can *always* do this; if their network has been compromised, maybe so can others

How to Provide Confidentiality? (2/2)

- Data can always be read → use **encryption** to make it useless if intercepted
- Two basic approaches
  - Symmetric cryptography
    - Advanced Encryption Standard (AES)
  - Public key cryptography
    - The Diffie-Hellman algorithm
    - The Rivest-Shamir-Adleman (RSA) algorithm
    - Elliptic curve-based algorithms
  - Complex mathematics – will not attempt to describe

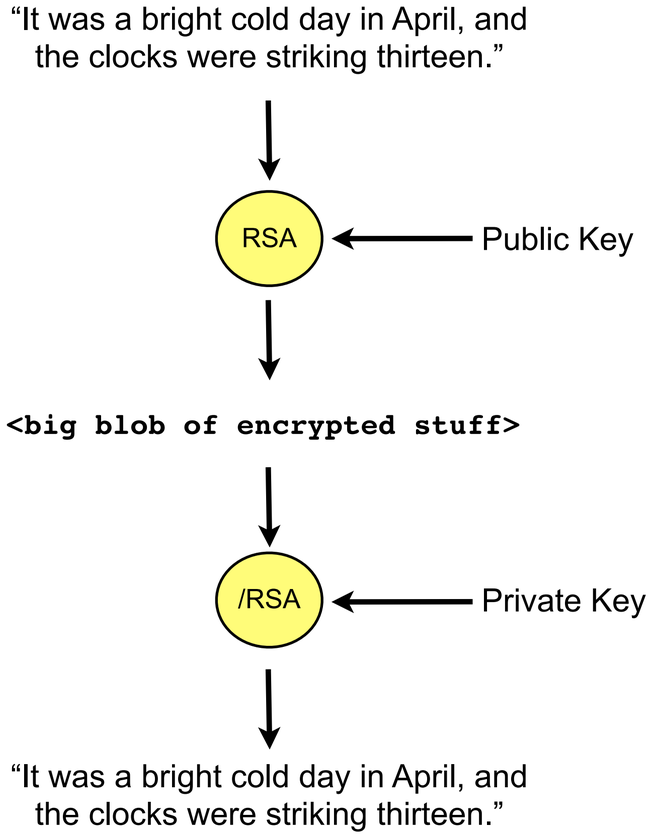
Symmetric Cryptography

- Symmetric encryption converts plain text into cipher-text
  - A secret **key** control the encryption and decryption process
    - Same key used to encrypt as is used to decrypt
    - Provided the key is secret and only known to sender and receiver, the conversation is secure – problem: **how to securely distribute the key?**
  - Very fast – suitable for bulk encryption
  - Encryption and decryption algorithms are public
    - US Advanced Encryption Standard (AES) is widely used – based on Rijndael algorithm, developed by Vincent Rijmen and Joan Daemen https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf



Public Key Cryptography

- Public key encryption also converts plain text into cipher-text
  - Again, the algorithms are public:
    - Diffie–Hellman
    - Rivest–Shamir–Adleman (RSA)
    - Ephemeral Elliptic Curve Diffie-Hellman
  - Public key algorithms use two related keys:
    - The **public key** for a user is widely distributed
    - **The corresponding private key must be kept secret**
    - If one key is used to encrypt, the other key is needed to decrypt the message
  - Solves key distribution problem
    - Look-up the public key of the receiver in a directory
    - Sender uses the public key to encrypt the message → can only be decrypted by the private key
    - If receiver is trusted to keep private key secret, only it can decrypt the message
  - Problem: **very slow** to encrypt and decrypt



Hybrid Cryptography

- Modern communications use a combination of **both** public-key and symmetric cryptography for security and speed
  - Sender chooses a random value, K_s, that can be used as key for the symmetric encryption algorithm
  - Sender looks up the receiver's public key, K_{pub}, uses it to encrypt K_s, and sends the result to the receiver; receiver uses the corresponding private key, K_{priv}, to decrypt the message and retrieve K_s
    - Securely transfers K_s from sender to receiver
    - Public key encryption is very slow, but the key K_s is small, so this doesn't matter
  - Sender encrypts future messages using symmetric cryptography with key K_s, receiver also has k_s, which it uses to decrypt the messages
    - Symmetric cryptography is fast, but requires the key to be exchanged securely
    - The public key algorithm has been used to securely exchange the key
- Ensures confidentiality of communication with good performance

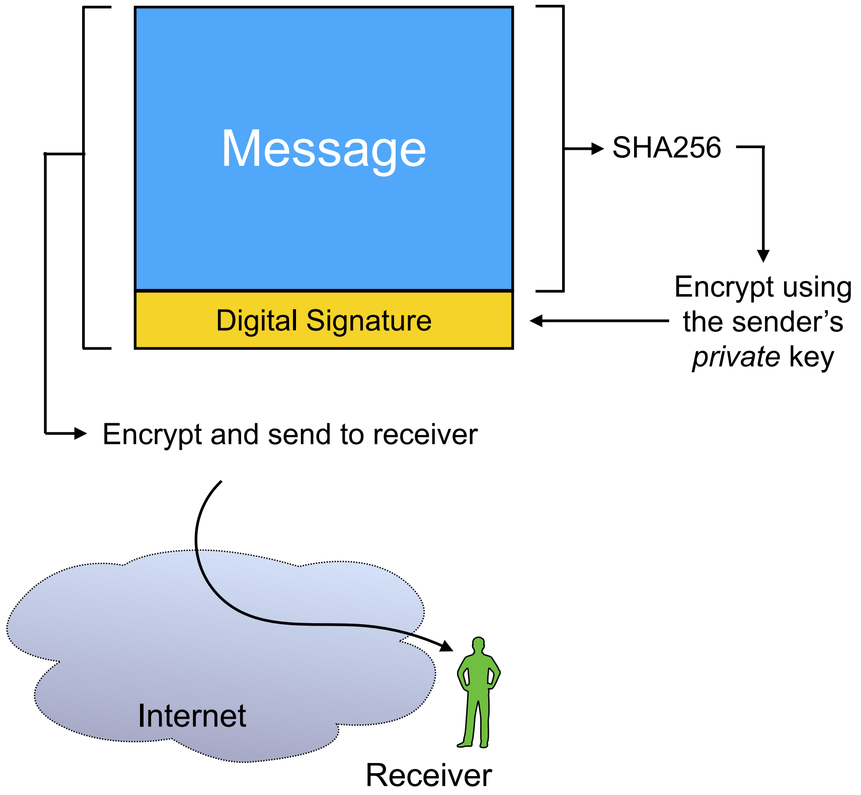
Authentication

- Encryption can ensure confidentiality – but also need to verify identify of sender and ensure messages have not been modified in transit
  - Generate a **digital signature** to authenticate the message
  - Relies on public key cryptographic and a cryptographic hash

Cryptographic Hash Functions

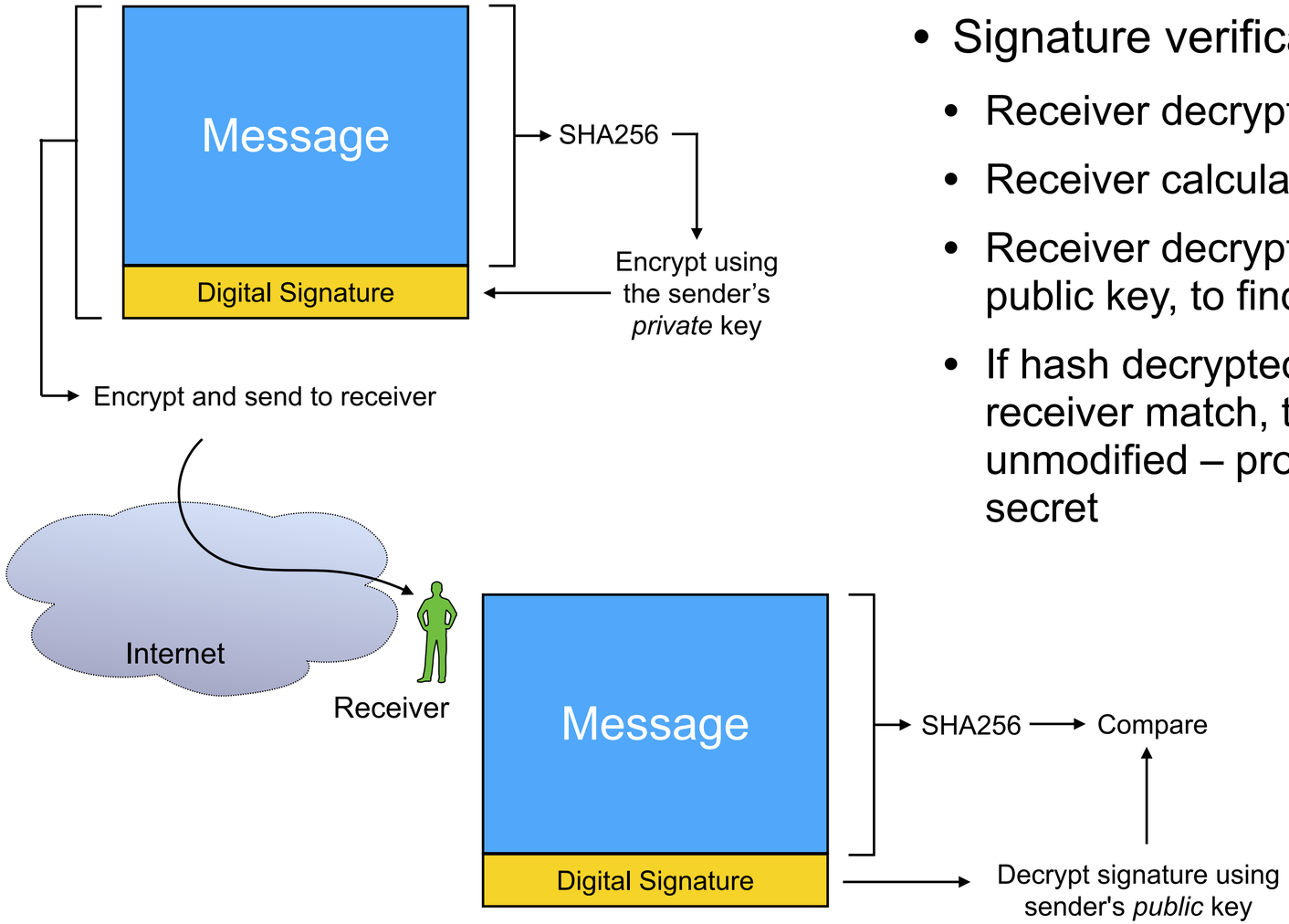
- **Cryptographic hash** takes arbitrary input and produces a fixed length output hash
  - Any change to input generates different output
  - Infeasible to find two inputs that give the same output
  - Calculating a cryptographic hash is fast
  - Reversing a hash, to find the input given only the output, is infeasible
- Many cryptographic hash algorithms exist:
  - Recommendation: SHA256 algorithm
    - https://tools.ietf.org/html/rfc6234
  - Older algorithms, e.g., MD5 and SHA1, have known security flaws

Digital Signatures (1/2)



- Sender generates a digital signature
  - Sender calculates the cryptographic hash of the message
  - Sender encrypts the hash with their own *private* key
    - Anyone can use the sender's public key to decrypt this, but only the sender can have encrypted it (if trusted to keep their private key secret)
  - Attaches encrypted hash to the message
- Message and its digital signature are encrypted and sent to receiver using hybrid encryption

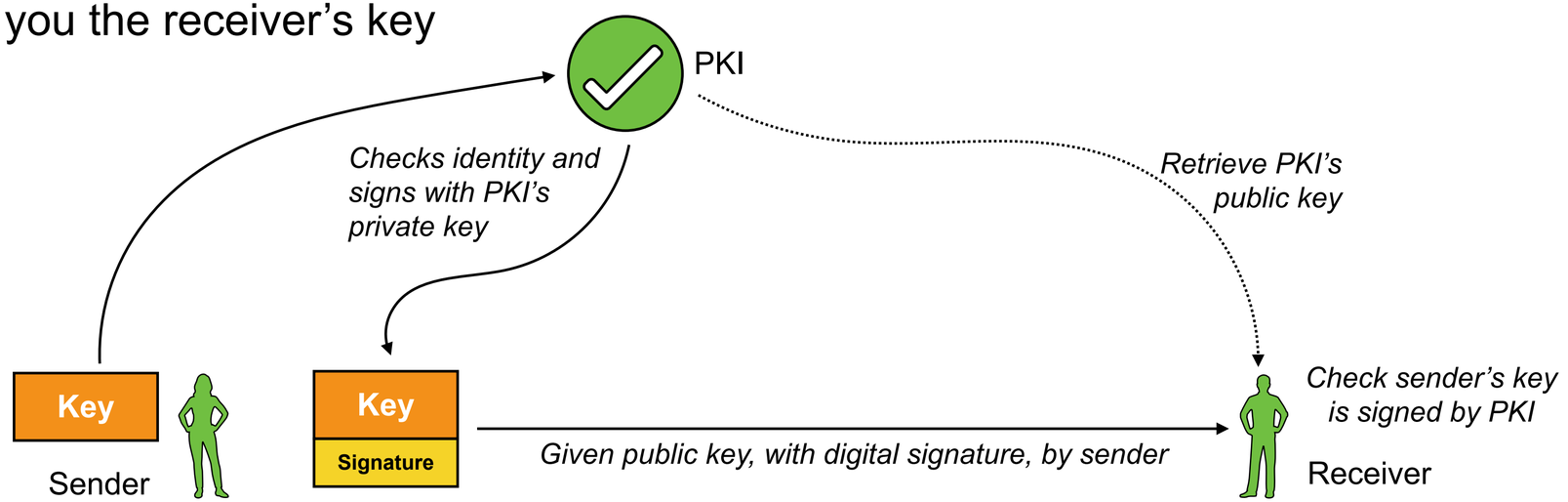
Digital Signatures (2/2)



- Signature verification process:
  - Receiver decrypts the message
  - Receiver calculate cryptographic hash of the message
  - Receiver decrypts digital signature using sender's public key, to find the hash that the sender calculates
  - If hash decrypted hash and the hash calculated by the receiver match, then the message is authentic and unmodified – provided the sender kept its private key secret

Trust and Public Key Infrastructure

- How to know what public key corresponds to a particular receiver?
  - The receiver gave you their key in person
  - The receiver sent you their key, authenticated by someone you trust
  - Someone you trust gave you the receiver's key



- A **public key infrastructure** can authenticate keys
  - PKI verifies sender's identity, then adds their digital signature to sender's public key
  - If receiver trusts PKI, can verify the digital signature to confirm identity of sender

Principles of Secure Communication

- Ensuring Confidentiality
- Authenticating Messages
- Validating Identity