# Security for Web APIs

HTTPS

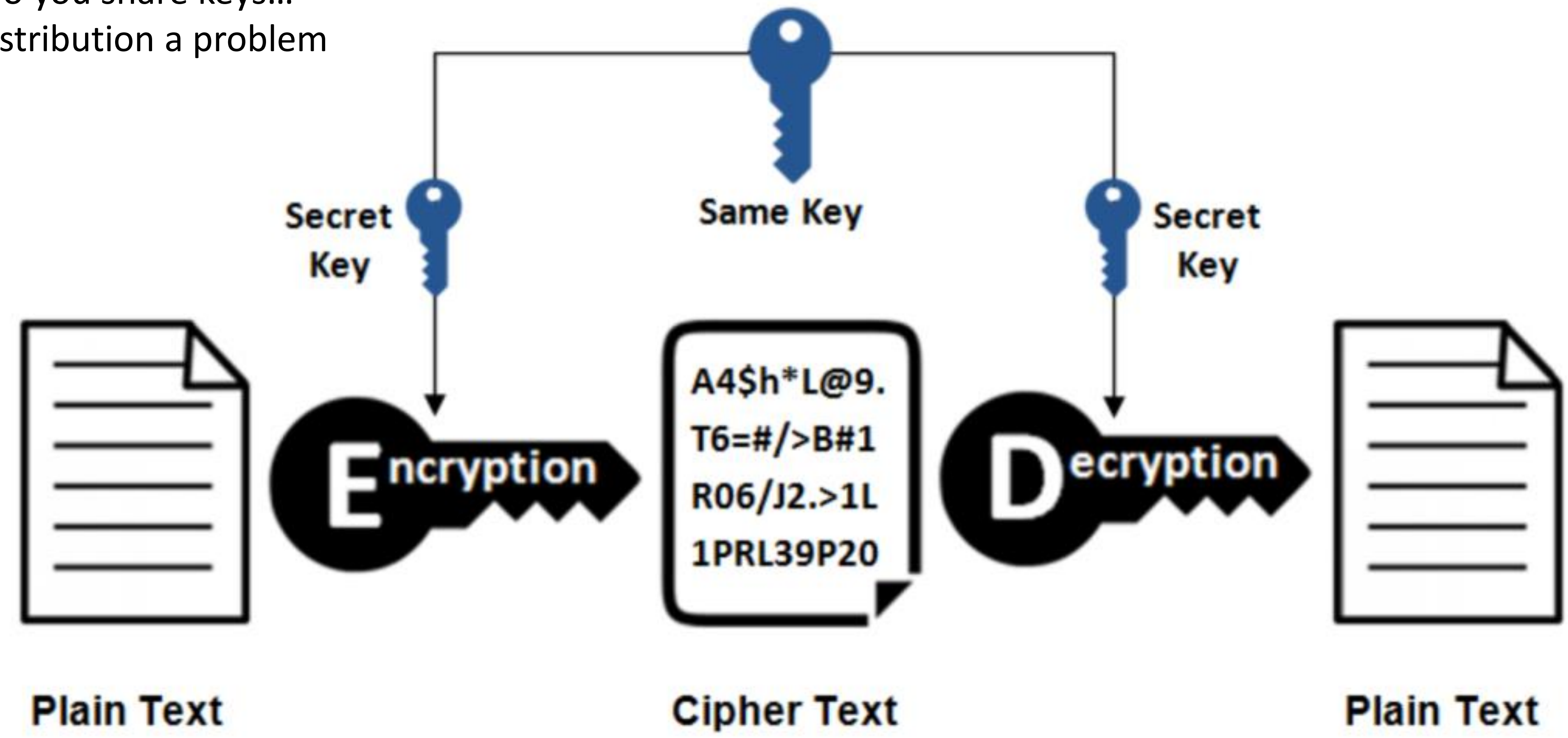
**HTTP over TLS**

# Web Security

- Anything you do on Web usually involves sending/receiving messages from a server, often with HTTP
- How do you know messages have not been changed/viewed in transit?
  - Man in the Middle attack
- Improvement:
  - Encrypt/Decrypt message using a shared Key
    - But how do you share the key is you've never met. That could be intercepted if sent (MITM attack again)
  - Encrypt/Decrypt using public/private key. But do you trust the person your communicating with is the person you think they are (Trust)
  - Public Certificate Authority. Trusted Authority can digitally "sign" certificates that contain public key.
- HTTPS explained with Pigeons: https://www.freecodecamp.org/news/https-explained-with-carrier-pigeons-7029d2193351/

# Symmetric Encryption

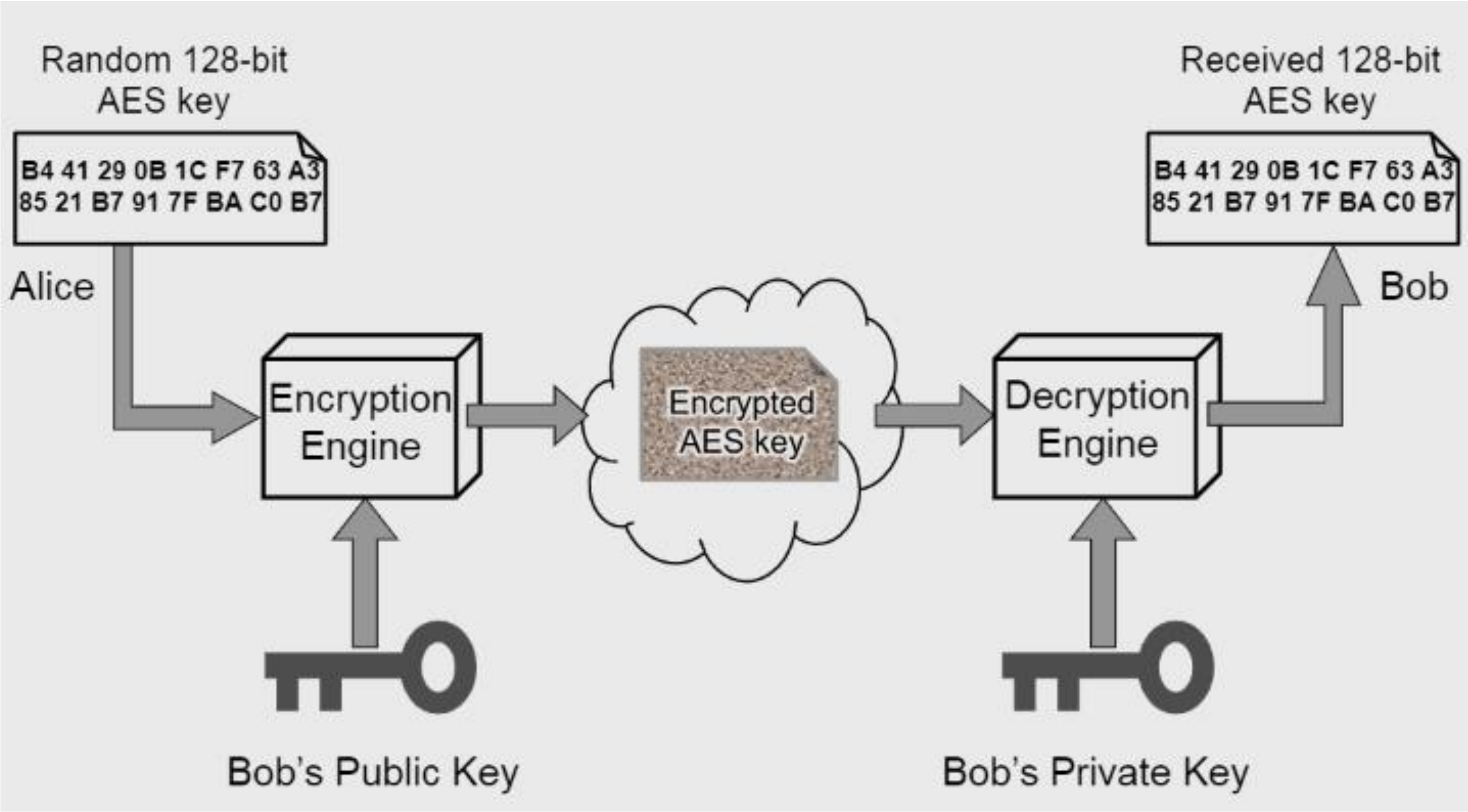
How do you share keys…
Key Distribution a problem



This Photo by Unknown Author is licensed under CC BY

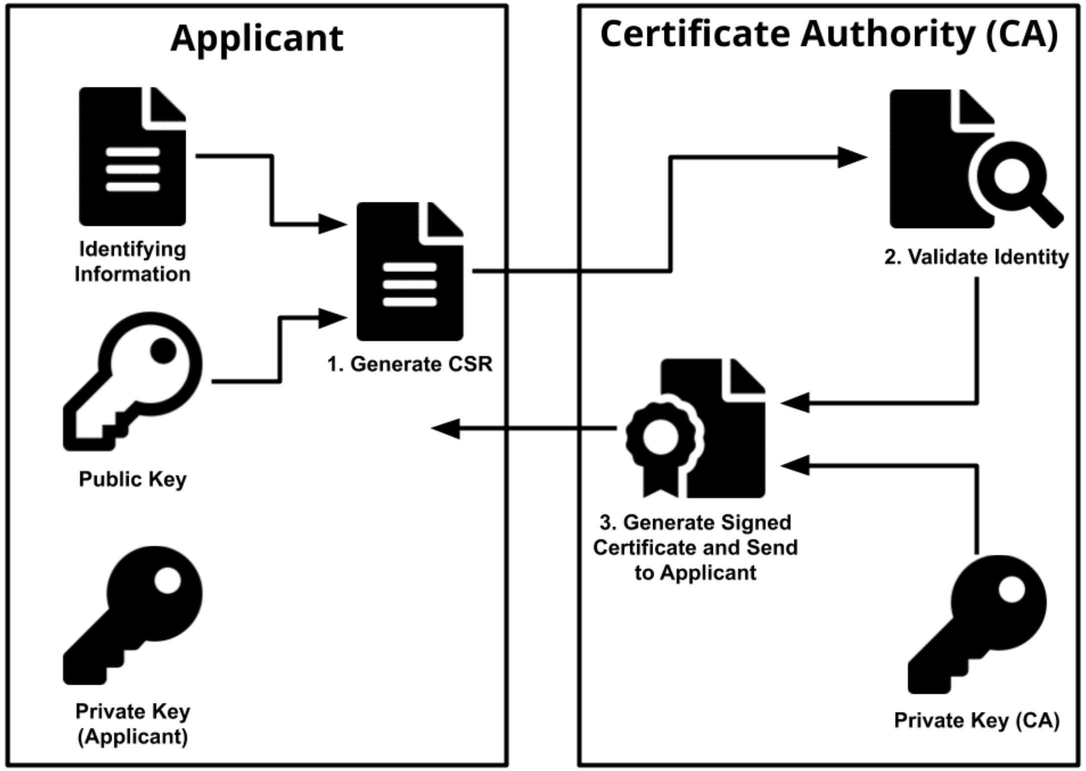
# Public Key Crypto

- Sender and Receiver have different keys
- Susceptible to Man in the Middle Attacks.
- How do you trust the public key is the correct one. Is it issued to you by the real owner???



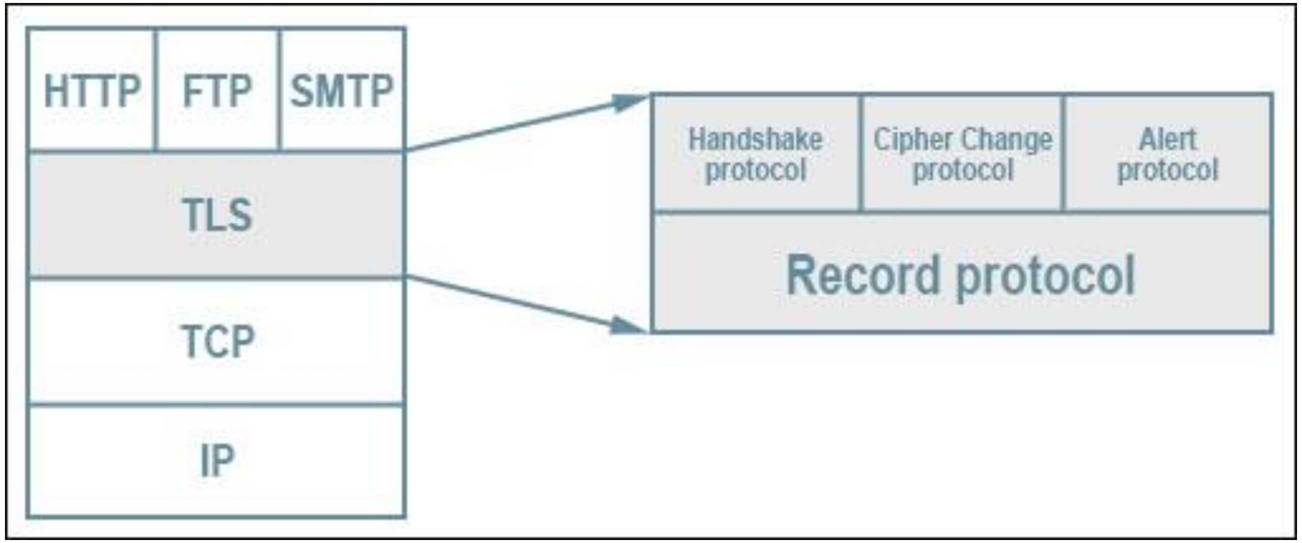
# Certificates

- How do you know if a Public Key is genuine?
- A key could be signed by someone who’s public key you have AND you know it’s correct AND you trust them!
- A digital cert is an electronic document
  - Signed by a third party Certificate Authority
  - Cert user trust the CA to issue valid certs
- Digital Cert contains:
  - The owner
  - The Public Key
  - Issuer (Cert Authority)
  - Issuers digital signature
  - Valid Period

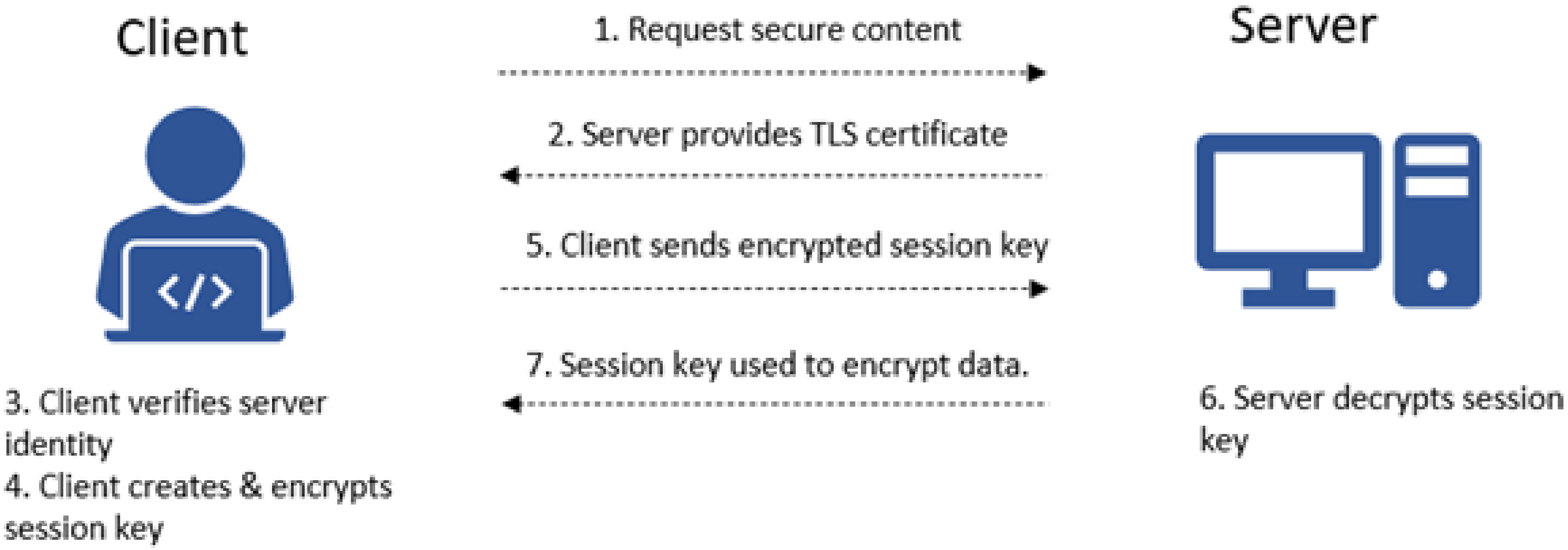


# Securing Web APIs

- HTTP provides no security
- The accepted approach is to add security on top of Transport Layer
- This is Transport Layer Security (TLS)



# TLS Process Overview



Client

1. Request secure content

Server

2. Server provides TLS certificate

5. Client sends encrypted session key

7. Session key used to encrypt data.

3. Client verifies server identity

4. Client creates & encrypts session key

6. Server decrypts session key