Certificates

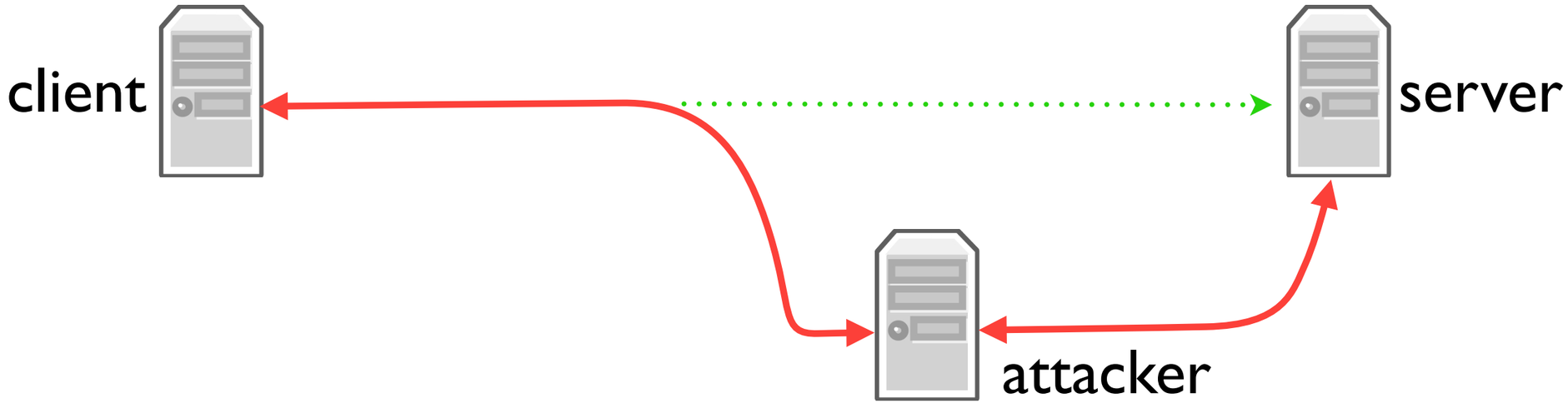
Establishing a chain of trust

Problem

- Want to communicate securely with a server, e.g. www.amazon.com
  - Using public key cryptography, given a public key K, we can verify that another program has the associated private key K^{-1}
    - Verify: $V(K, \{m\}_{K^{-1}}, m) \rightarrow$ {yes, no}
  - Use public key cryptography to exchange symmetric keys

- Problem: key management
  - How do we get the server's public key?
  - How can we be sure it's the server's public key?

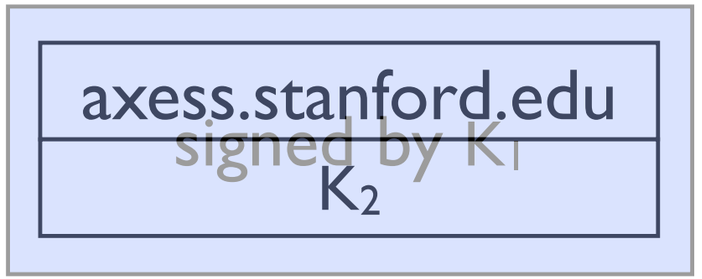
Attack: Man in the Middle



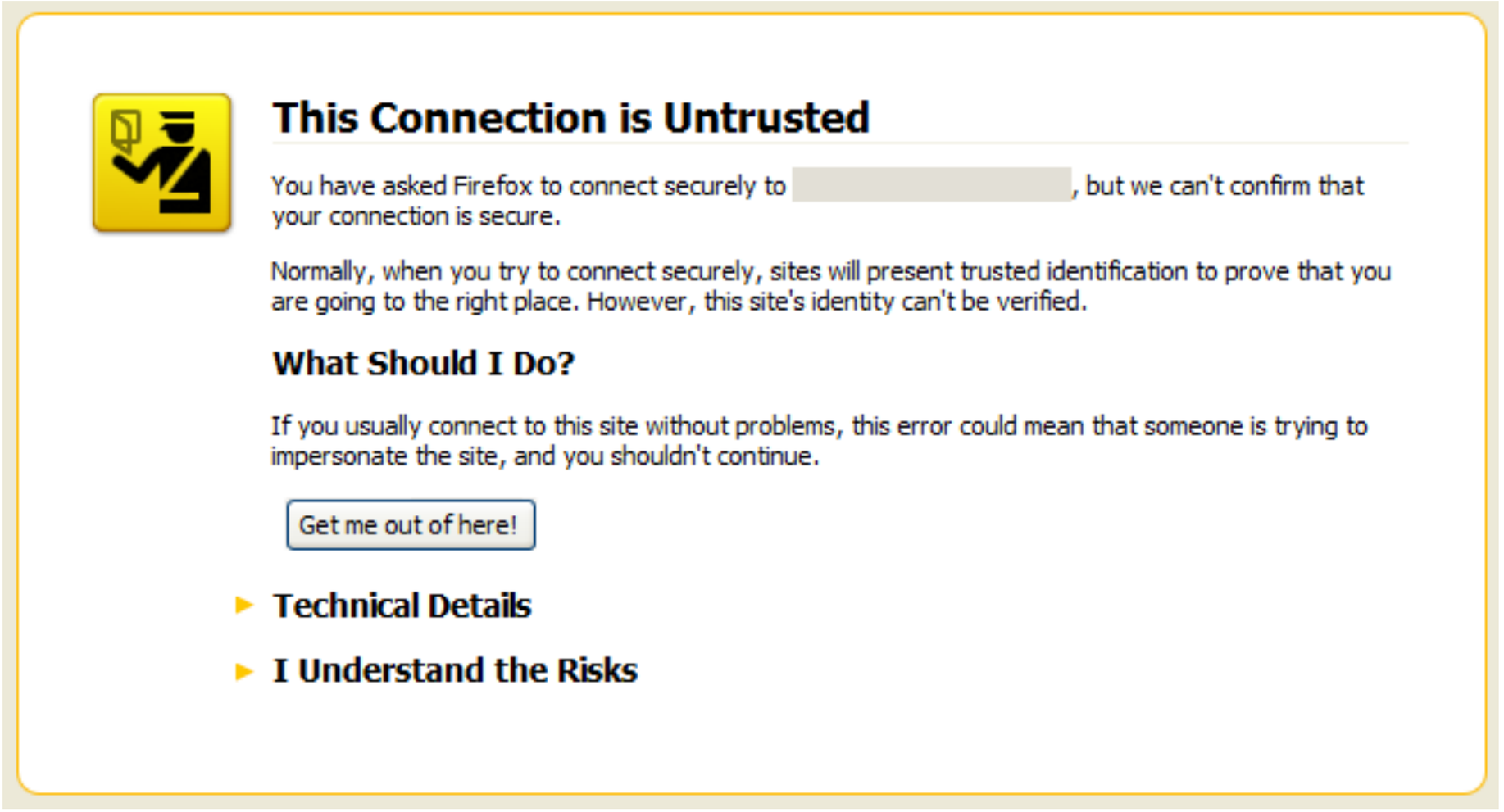
- Attacker pretends to be server, gives its own public key
- Mounts Man in the Middle attack
  - Looks just like server to client (except different public key)
  - Attacker sees, re-encrypts sensitive traffic
  - Attacker can insert new traffic

Certificate

- Certificate: document signed by a private key K_1^{-1} that binds a public key K_2 to an identity/name N
  - "The public key of www.ebay.com is ...."
  - "The public key of axess.stanford.edu is..."
- If we trust the signing party and know their public key, we can use K_2 when communicating with N



Bad Certificate!



Bootstrapping Trust

- "Everyone" trusts a few signing authorities and knows their public keys
  - Hard-baked into your browser or OS
  - Examples: Verisign, Microsoft, Google
- Root authorities sign certificates (e.g., for Stanford)
- Can then use those certificates to sign further certificates
  - "Chain of trust" - GeoTrust to Google to www.google.com
- Certificate only says that someone testifies that a host has this key! Have to trust every step along chain
- This is how TLS/HTTPS works today (the padlock in your browser bar)