Message Authentication
Goal and Threat Model

• Goal is for Bob to verify the message is from Alice and unchanged
  • This is called integrity/authenticity
• Threat is Trudy will tamper with messages
  • Trudy is an active adversary (interferes)
Wait a Minute!

• We’re already encrypting messages to provide confidentiality

• Why isn’t this enough?
Encryption Issues

• What will happen if Trudy flips some of Alice’s message bits?
  • Bob will decrypt it, and ...
Encryption Issues (2)

• What will happen if Trudy flips some of Alice’s message bits?
  • Bob will receive an altered message
Encryption Issues (3)

• Typically encrypt blocks of data
• What if Trudy reorders message?
  • Bob will decrypt, and ...
Encryption Issues (4)

• What if Trudy reorders message?
  • Bob will receive altered message
MAC (Message Authentication Code)

- MAC is a small token to validate the integrity/authenticity of a message
  - Conceptually ECCs again
  - Send the MAC along with message
  - Validate MAC, process the message
  - Example: HMAC scheme
MAC (2)

• Sorta symmetric encryption operation – key shared
  • Lets Bob validate unaltered message came from Alice
  • Doesn’t let Bob convince Charlie that Alice sent the message
Digital Signature

• Signature validates the integrity/authenticity of message
  • Send it along with the message
  • Lets all parties validate
  • Example: RSA signatures
Digital Signature (2)

• Kind of public key operation – pub/priv key parts
  • Alice signs w/ private key, $K_A^{-1}$, Bob verifies w/ public key, $K_A$
  • Does let Bob convince Charlie that Alice sent the message
Speeding up Signatures

• Same tension as for confidentiality:
  • Public key has keying advantages
  • But it has slow performance!

• Use a technique to speed it up
  • Message digest stands for message
  • Sign the digest instead of full message
Message Digest or Cryptographic Hash

• Digest/Hash is a secure checksum
  • Deterministically mangles bits to pseudo-random output (like CRC)
  • Can’t find messages with same hash
  • Acts as a fixed-length descriptor of message – very useful!

\[
\text{I might be a tiny bit sick of networks...} \quad \text{Input}
\]
\[
e.g., \text{SHA1 (160 bits)} \quad \text{Hash function} \quad \text{Output}
\]
Speeding up Signatures (2)

• Conceptually similar except sign the hash of message
  • Hash is fast to compute, so it speeds up overall operation
  • Hash stands for msg as can’t find another w/ same hash
Preventing Replays

• We normally want more than confidentiality, integrity, and authenticity for secure messages!
  • Want to be sure message is fresh
• Need to distinguish message from replays
  • Repeat of older message
  • Acting on it again may cause trouble
Preventing Replays (2)

• Replay attack:
  • Trudy records Alice’s messages to Bob
  • Trudy later replays them (unread) to Bob
    • She pretends to be Alice

Hi Alice!

Bob

Password?

Trudy
Preventing Replays (3)

• To prevent replays, include a proof of freshness in the messages
  • Use a timestamp, or nonce

![Diagram showing the flow of a message from Alice to Bob with freshness, confidentiality, and authenticity/integrity labeled.]
Using Timestamps

A = Alice
B = Bob
$T_A = \text{Timestamp from Alice's clock}$

[Box]

= Digitally signed using Bob's private key
Takeaway

• Cryptographic designs can give us integrity, authenticity and freshness as well as confidentiality.

• Real protocol designs combine the properties in different ways
  • We’ll see some examples
  • Note many pitfalls in how to combine, as well as in the primitives themselves
Web Security
What should be the Threat Model for the Web?
Goal and Threat Model

- Much can go wrong on the web!
  - Clients encounter malicious content
  - Web servers are target of break-ins
  - Fake content/servers trick users
  - Data sent over network is stolen ...

Introduction to Computer Networks
Goal and Threat Model (2)

• Goal of HTTPS is to secure HTTP
• We focus on network threats:
  1. Eavesdropping client/server traffic
  2. Tampering with client/server traffic
  3. Impersonating web servers
HTTPS Context

- HTTPS (HTTP Secure) is an add-on
  - Means HTTP over SSL/TLS
  - SSL (Secure Sockets Layer) precedes TLS (Transport Layer Security)
HTTPS Context (2)

• SSL came out of Netscape
  • SSL2 (flawed) made public in ‘95
  • SSL3 fixed flaws in ‘96
• TLS is the open standard
  • TLS 1.0 in ‘99, 1.1 in ‘06, 1.2 in ‘08
• Motivated by secure web commerce
  • Slow adoption, now widespread use
  • Can be used by any app, not just HTTP
SSL/TLS Operation

• Protocol provides:
  1. Verification of identity of server (and optionally client)
  2. Message exchange between the two with confidentiality, integrity, authenticity and freshness

• Consists of authentication phase (that sets up encryption) followed by data transfer phase
SSL/TLS Authentication

• Must allow clients to securely connect to servers not used before
  • Client must authenticate server
  • Server typically doesn’t identify client

• Uses public key authentication
  • But how does client get server’s key?
  • With certificates »
Certificates

- A certificate binds pubkey to identity, e.g., domain
- Distributes public keys when signed by a party you trust
- Commonly in a format called X.509

---

I hereby certify that the public key
19836A8B03030CF83737E3837837FC3s87092827262643FFA82710382828282A
belongs to
Robert John Smith
12345 University Avenue
Berkeley, CA 94702
Birthday: July 4, 1958
Email: bob@superdupermet.com

Signed by CA
PKI (Public Key Infrastructure)

- Adds hierarchy to certificates to let parties issue
- Issuing parties are called CAs (Certificate Authorities)

I certified the ABC website!
PKI (2)

- Need public key of PKI root and trust in servers on path to verify a public key of website ABC
  - Browser has Root’s public key
  - {RA1’s key is X} signed Root
  - {CA1’s key is Y} signed RA1
  - {ABC’s key is Z} signed CA1

I certified the ABC website!
PKI (3)

- Browser/OS has public keys of the trusted roots of PKI
  - >100 root certificates!
- Inspect your web browser

Certificate for wikipedia.org issued by DigiCert
PKI (4)

• Real-world complication:
  • Public keys may be compromised
  • Certificates must then be revoked

• PKI includes a CRL (Certificate Revocation List)
  • Browsers use to weed out bad keys
TLS handshake
What can attacker (in the network) still learn from an HTTPS connection?

• “Metadata”
Takeaways

• SSL/TLS is a secure transport
  • For HTTPS and more, with the usual confidentiality, integrity / authenticity
  • Very widely used today

• Client authenticates web server
  • Done with a PKI and certificates
  • Major area of complexity and risk

• “Metadata” leaks
  • Use other tools (Tor or VPN) if you want to hide that