Web Security

This isn’t a security class

- This isn’t intended to be a lecture on all crypto

- I want you to appreciate the important principles, understand what’s important for TLS (and other protocols like it)

The Internet is insecure

- Designed for simplicity in a naïve era
- Lots of insecure systems that can be compromised
- Attacks look like normal traffic
- Internet’s federated operation obstructs cooperation for diagnosis/mitigation

Basic Requirements for Secure Communication

- **The big three (CIA Triad)**
  - **Confidentiality**: Can adversary read the data?
    - Sniffing, man-in-the-middle
  - **Integrity**: Do messages arrive in original form?
  - **Availability**: Will the network deliver data?
    - Infrastructure compromise, DDoS

Other desirable security properties

- **Authentication**: Who is this actor?
    - Spoofing, phishing
- **Provenance**: Who is responsible for this data?
    - Forging responses, denying responsibility
    - Not who sent the data, but who created it
- **Authorization**: is actor allowed to do this action?
    - Access controls
- **Accountability/Attribution**: who did this activity?
- **Audit/Forensics**: what occurred in the past?
    - A broader notion of accountability/attribution
- **Appropriate use**: is action consistent with policy?
    - E.g., no spam; no games during business hours; etc.
- **Freedom from traffic analysis**: can someone tell when I am sending and to whom?
- **Anonymity**: can someone tell I sent this packet?

What is TLS?

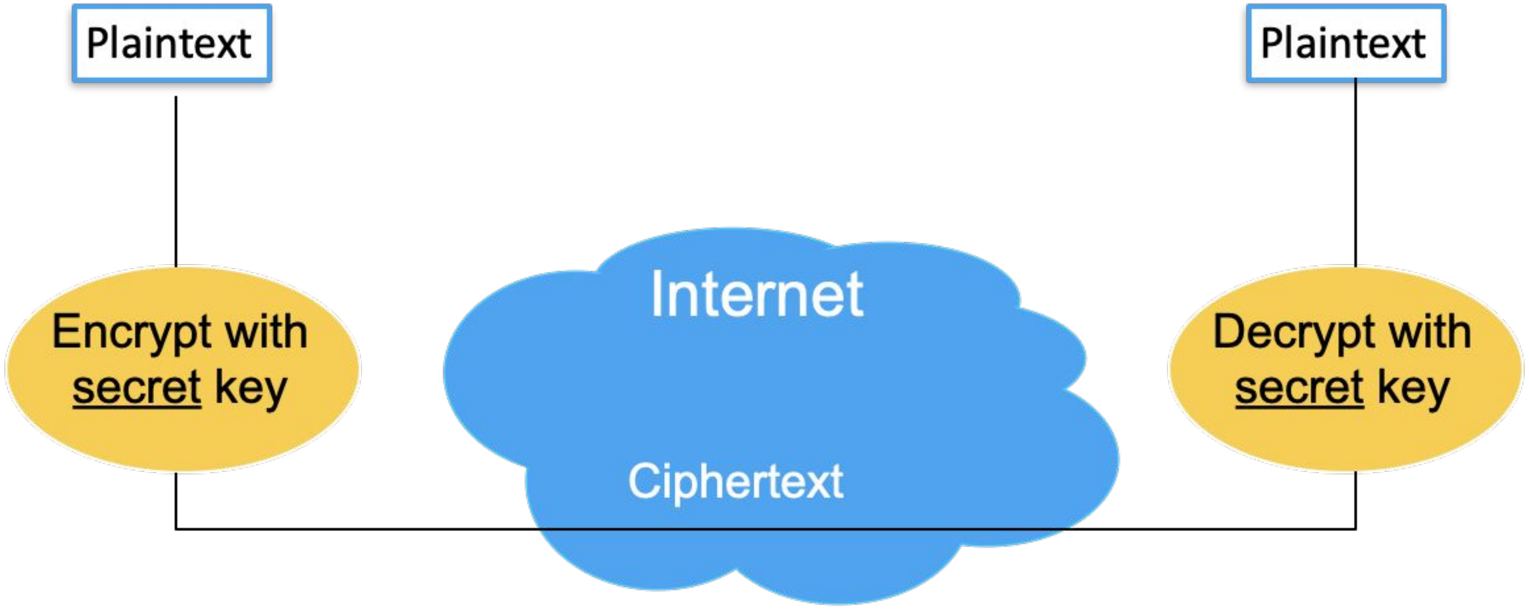
- Security for the transport layer

- Bidirectional pipe between two parties (client and server), but can enable:
    - Confidentiality
    - Integrity
    - Authentication

- Is this all the security properties we might want? No

Fundamental crypto: symmetric keys

Both the sender and the receiver use the same secret keys

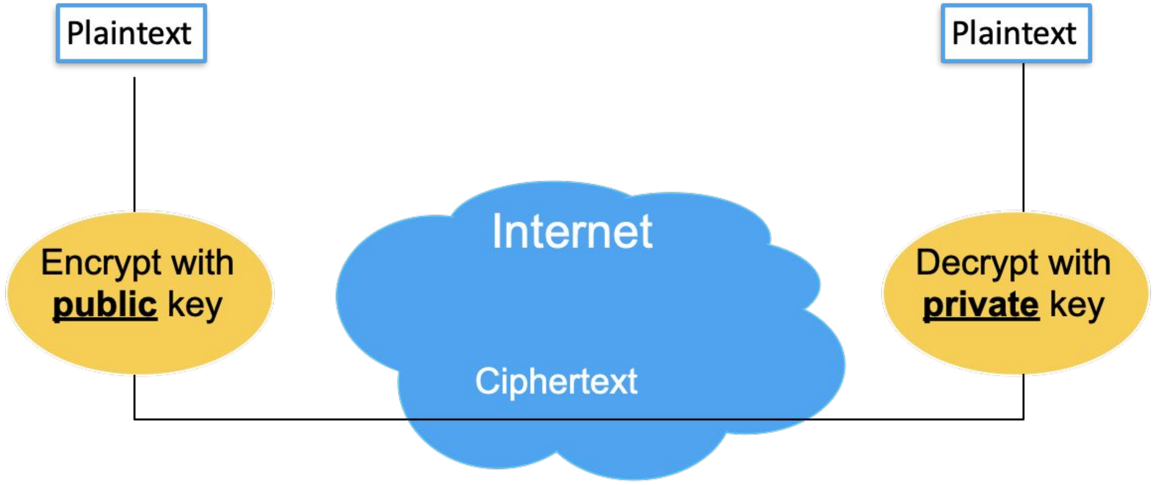


Fundamental crypto: asymmetric encryption (public key)

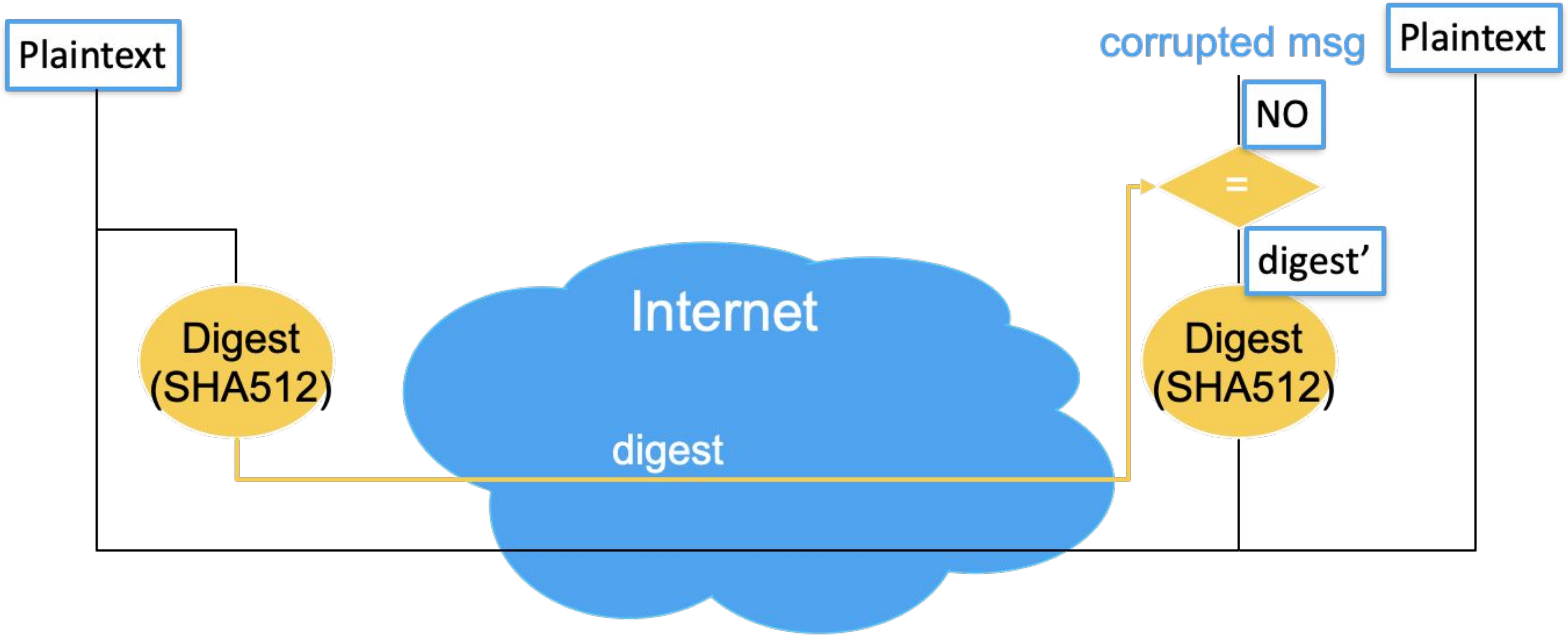
- Idea: use two different keys, one to encrypt (**e**) and one to decrypt (**d**)
  - A key pair
- Crucial property: knowing **e** does not give away **d**
- **e** can be public: everyone knows it
- If Alice wants to send to Bob, she fetches Bob’s public key (say from Bob’s home page) and encrypts with it
  - Alice can’t decrypt what she’s sending to Bob …
  - … but then, neither can anyone else (except Bob)

Public Key / Asymmetric Encryption

- Sender uses receiver's public key
  - Advertised to everyone
- Receiver uses complementary private key
  - Must be kept secret



Hashing for Integrity

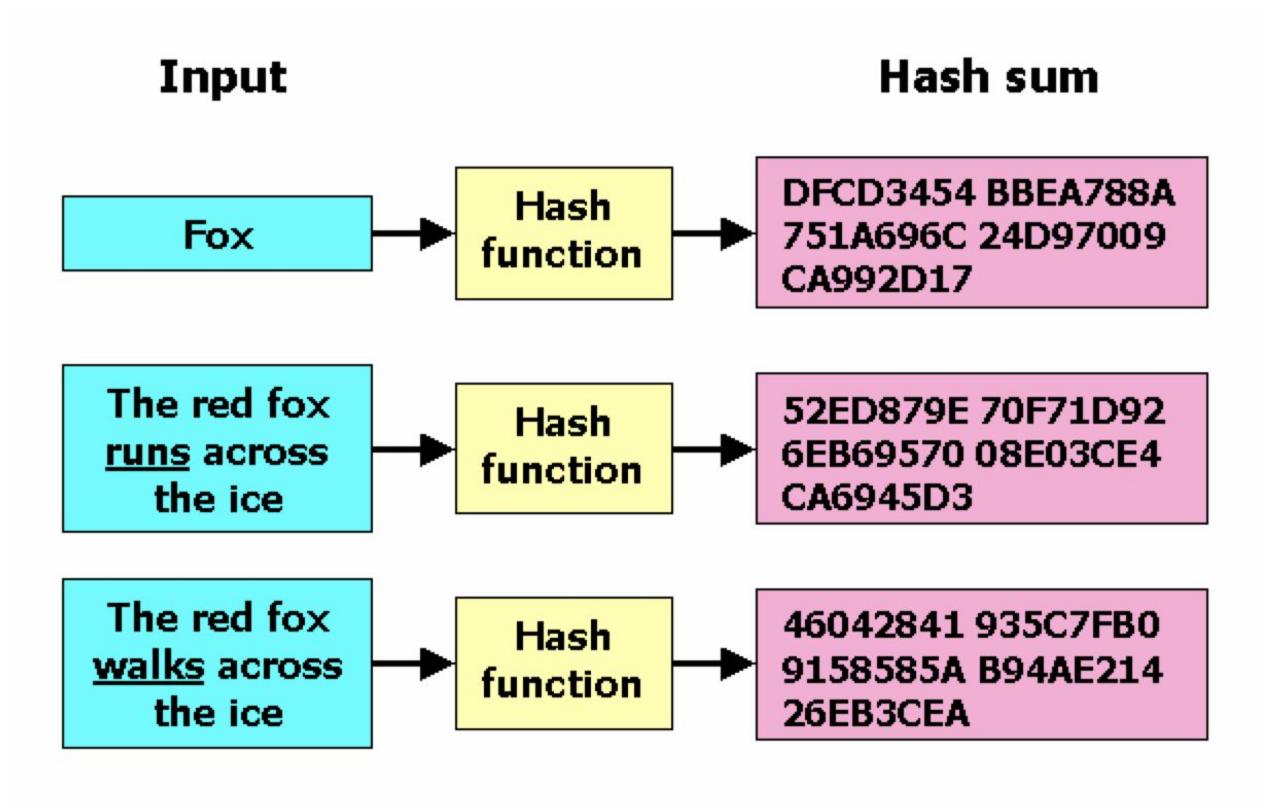


Cryptographically Strong Hashes

- Hard to find collisions
  - Adversary can’t find two inputs that produce same hash
  - Someone cannot alter message without modifying digest
  - Can succinctly refer to large objects

- Hard to invert
  - Given hash, adversary can’t find input that produces it
  - Can refer obliquely to private objects (e.g., passwords)
    - Send hash of object rather than object itself

Effects of cryptographic hashes


Input
Hash sum
Fox
Hash function
DFCD3454 BBEA788A 751A696C 24D97009 CA992D17
The red fox runs across the ice
Hash function
52ED879E 70F71D92 6EB69570 08E03CE4 CA6945D3
The red fox walks across the ice
Hash function
46042841 935C7FB0 9158585A B94AE214 26EB3CEA

Public key authentication

Each side need only to know the other side’s public key

No secret key need be shared

A encrypts a nonce (random number) x using B’s public key

B proves it can recover x

A can authenticate itself to B in the same way

Basic crypto toolkit

- If we can securely distribute a key, then
  - Symmetric ciphers (e.g., AES) offer fast, presumably strong confidentiality
- Public key cryptography can make this easier (can share public keys anywhere)
  - But not as computationally efficient
  - Use public key crypto to exchange **session key**, which is used for symmetric encryption

Public Key Infrastructure (PKI)

- In reality, hierarchy of trust
- Root CAs sign certificates for Intermediate CAs
- Intermediate CAs sign certificates for general users/sites

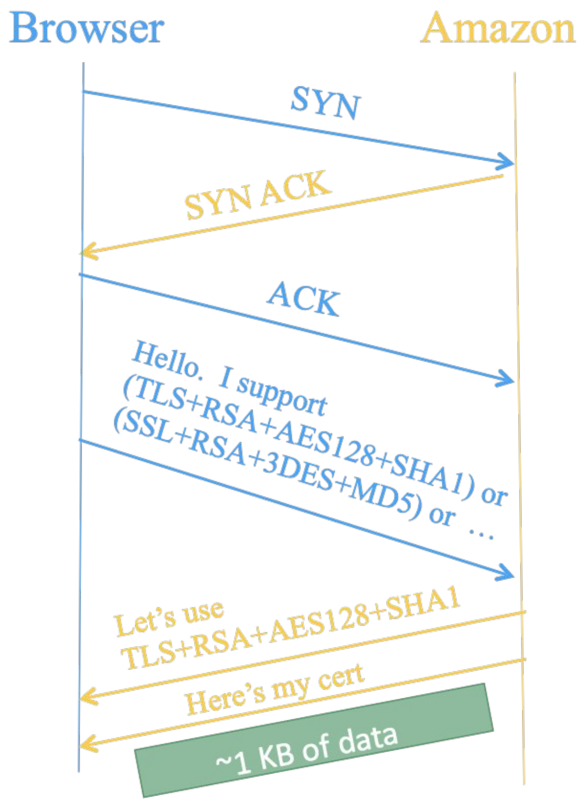
- The further up the hierarchy, the more protections it needs
  - CA’s often use Hardware Security Modules (HSMs), other physical protections…
  - What happens if a CA is compromised?

Putting It All Together: HTTPS

- Steps after clicking on https://www.amazon.com
- https = “Use HTTP over TLS”
  - SSL = Secure Socket Layer (older version)
  - TLS = Transport Layer Security
    - Successor to SSL, and compatible with it
  - RFC 4346, and many others
- Provides security layer (authentication, encryption) on top of transport layer
  - Fairly transparent to the app (once set up)

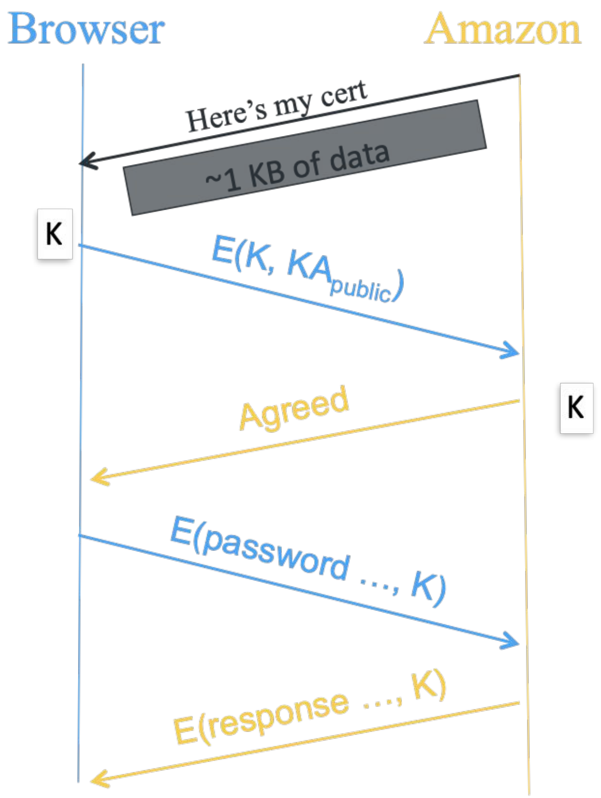
HTTPS Connection (SSL/TLS), con't

- Browser (client) connects via TCP to Amazon's HTTPS server
- Client sends over list of crypto protocols it supports
- Server picks protocols to use for this session
- Server sends over its certificate
- (all of this is in the clear)


Browser
Amazon
SYN
SYN ACK
ACK
Hello. I support (TLS+RSA+AES128+SHA1) or (SSL+RSA+3DES+MD5) or …
Let's use TLS+RSA+AES128+SHA1
Here's my cert
~1 KB of data

HTTPS Connection (SSL/TLS), con't

- Browser constructs a random session key K
- Browser encrypts K using Amazon's public key
- Browser sends E(K, KA_{public}) to server
- Browser displays 🔒
- All subsequent communication encrypted w/ symmetric cipher using key K
  - E.g., client can authenticate using a password

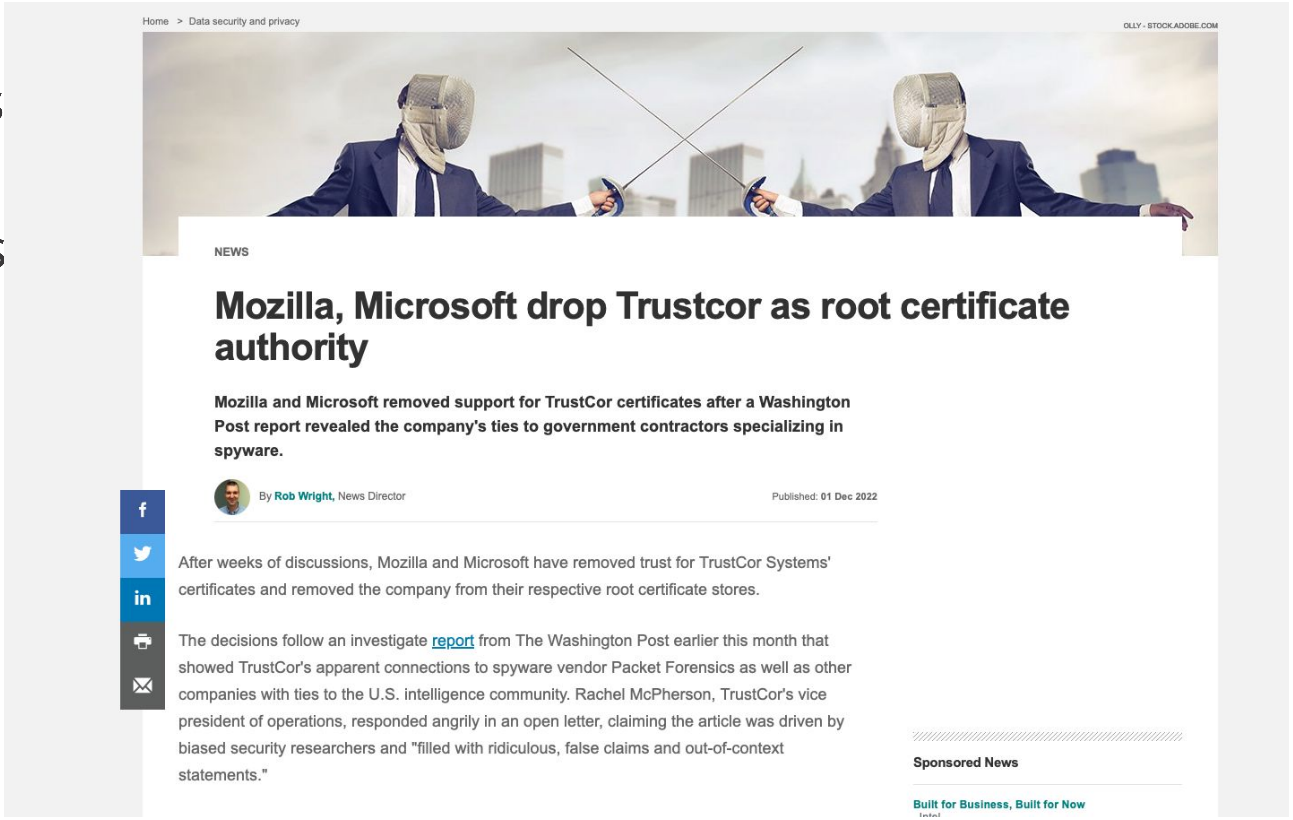


When does this break down?

- TLS is hard to implement
- Need to trust the CAs
- Users need to understand warnings

When does this break down?

- TLS is
- Need
- Users



NEWS

Mozilla, Microsoft drop Trustcor as root certificate authority

Mozilla and Microsoft removed support for TrustCor certificates after a Washington Post report revealed the company's ties to government contractors specializing in spyware.

By Rob Wright, News Director

Published: 01 Dec 2022

After weeks of discussions, Mozilla and Microsoft have removed trust for TrustCor Systems' certificates and removed the company from their respective root certificate stores.

The decisions follow an investigate report from The Washington Post earlier this month that showed TrustCor's apparent connections to spyware vendor Packet Forensics as well as other companies with ties to the U.S. intelligence community. Rachel McPherson, TrustCor's vice president of operations, responded angrily in an open letter, claiming the article was driven by biased security researchers and "filled with ridiculous, false claims and out-of-context statements."

Sponsored News

Built for Business, Built for Now