CS 5430

Certificate Authorities

Prof. Clarkson
Spring 2016

Review: Certificates

- Digital certificate is a signature binding together:
  - **identity** of principal
  - **public key** of that principal (might be encryption or verification key)
- **Notation:** Cert(S; I) is a certificate issued by principal I for principal S
  - let b = id_S, K_S
  - Cert(S; I) = b, Sign(b; k_I)
  - Issuer I is certifying that K_S belongs to subject id_S

Review: PKI

- System for managing distribution of certificates
- Two main philosophies:
  - **Decentralized:** anarchy, no leaders (PGP)
  - **Centralized:** oligarchy, leadership a few elite (CAs)

Recap of PGP

PGP offers authentication of humans through machines:

- Identity is that of a human
- Private key is part of human's identity
- Private key is stored on trusted machine
- Need the machine to handle storage and computation
- So line is blurred between which we're really authenticating

PKI Example 2: CAs

- Uses a centralized PKI philosophy (at least as evolved in marketplace)
- Invented (?) by Digital [Gasser et al. 1989], used in early Netscape browsers
- Certificate authority (CA): principal whose purpose is to issue certificates

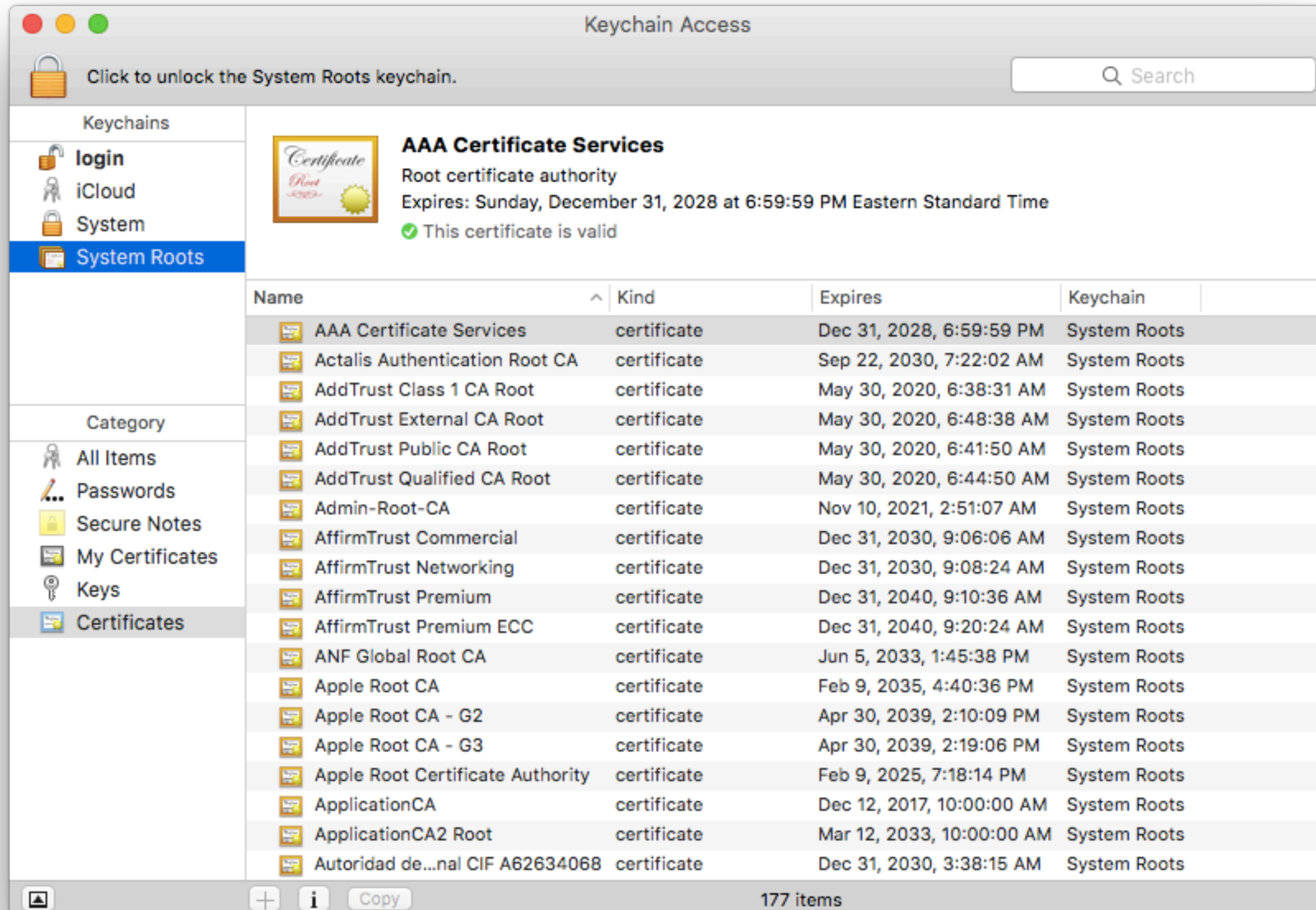
Using a CA

- Everyone enrolls with the CA to get a certificate
  - E.g., Alice enrolls and gets Cert(Alice; CA)
- Your system comes pre-installed with CA's self-signed certificate Cert(CA; CA)
- When you receive a message signed by Alice:
  - you contact CA to get Cert(Alice; CA)
  - or Alice just includes that certificate with her message

Many CAs

- There can't be **only one**
  - No single CA is going to be trusted by all the world's governments, militaries, businesses
  - Though within an organization such trust might be possible
- So there are **many**
  - Around 1500 observed on public internet
  - Your OS and/or browser comes with some pre-installed
- Organizations act as their own CA, e.g....
  - Company issues certificates to employees for VPN
  - Bank issues certificates to customers
  - Central bank issues certificates to other banks
  - Manufacturer issues certificates to sensing devices

Demo: OS X Keychain Access



Certificate chains with CAs

Conflicting goals:

- CA private signing key must be kept secret
  - the public verification key is pre-installed on user systems; hard to update
  - if ever leaked, signing key could be used to forge certificates
  - easy way to realize goal: keep it in *cold storage*
- CA private signing key must be available for use
  - to sign new certificates when users request them
  - easy way to realize goal: keep it in computer's memory

Certificate chains with CAs

Solution: use root and intermediate CAs

- **root CA:** the certificate at root of trust in a chain; pre-installed; key kept in highly secure storage
- **intermediate CA(s):** certified by root CA, themselves certify user keys; might be run by a different organization than root
- example: https://www.facebook.com

Enrollment with a CA

- You create a key pair: **you** do this so that CA doesn't learn your private key
- You generate a certificate signing request (CSR); it contains the identity you are claiming
- You send the CSR to a CA, perhaps along with payment
- The CA verifies your identity (maybe)
- The CA signs your key, creating a certificate, and sends certificate to you

EV certificates

- Extended validation (EV):
  - CA does extra checking of your identity
  - Certificate marked as having received EV
  - Web browser reflects EV mark in UI
- Examples of extra checking:
  - Verify legal existence of organization including some sort of registration number
  - Verify physical operation of organization by a site visit
  - Verify phone number as listed by a public phone company
- CA record all those data in the certificate as part of subject's identity
- Example: https://www.paypal.com

Recap of CAs (as used in web)

- Browser authenticates web server
  - Server possesses private key associated with organization
  - Certificate states website's domain name as part of identity
  - Browser verifies that matches
- Machines are authenticating machines

Authentication

	Humans	Machines
Humans authenticating…	Faces, passwords	Secure attention key, visual secrets
Machines authenticating…	Passwords, biometrics	Tokens, CAs as used in web

Success!

We've solved the phonebook problem!

To publish public key, user can:

- distribute it as part of web of trust
- or engage CA to provide certificate

...or, have we???

PROBLEMS WITH PKI

Problem 1: Revocation

- Keys (subject's, issuer's) get compromised
- Or subject leaves an organization

  ...certificates therefore need to be revoked

- **There's no perfect solution**
  - Certificate revocation lists (CRLs)
  - Online certificate validation
  - Fast expiration

Revocation

Fast expiration

- **Idea:** Validity internal is short, e.g. 10 min to 24 hr; any compromise is bounded
- **Problem:**
  - CAs have to issues new certificates frequently, including checking identities
  - Machines have to update certificates frequently

Revocation

Certificate revocation lists (CRLs)

- **Idea:** CA posts list of revoked certificates; clients download and check every time they need to validate certificate
- **Problems:**
  - Clients don't (because usability)
  - Or they cache leading to TOCTOU
  - CRL must always be available (so an attractive target)
- Chromium currently does this, with a CRL limited to 250kb

Revocation

Online certificate validation

- **Idea:** CA runs *validation server*; clients contact it each time to validate certificate
- **Problems:**
  - Clients don't
  - Server must always be available (so an attractive target)
  - Reveals to CA which websites you want to access
- **Follow-on solution:** stapling, in which certificates must be accompanied by fresh assertion from CA that certificate is still valid; whoever presents certificate to client is responsible for acquiring assertion
- Firefox currently does this but doesn't *hard fail* because "[validation servers] aren't yet reliable enough"

Problem 2: Authority

- CAs go rogue, get hacked, issue certificates that **they** should never have issued
  - e.g., Dutch CA DigiNotar, which was included in many root sets: 500 bogus certificates issued, including for Google, Yahoo, Tor
- Missing a means for authorization of who may issue certificates for which principals

Authority

There's no perfect solution

- Key pinning: upon first connection to a server, client learns a set of public keys for server; in future connections, certificate must contain one of those keys
- Certificate transparency: maintain a public log of issued certificates; require any presented certificate to provably be in that log; monitor log to notice misbehavior
- Certificate Authority Authorization (CAA): piggyback on DNS system; DNS record for entity specified allowed CAs

USING CAs IN SSL

SSL

Secure Sockets Layer (SSL)

- aka Transport Layer Security (TLS)
- SSL 3.1 = TLS 1.0 (1999)
  - Broken by attack in 2011 based on improper choice of IVs for CBC mode
- SSL 3.2 = TLS 1.1 (2006)
  - Fixes IVs
- SSL 3.3 = TLS 1.2 (2008)
  - Upgrades crypto primitives (AES, SHA-256, etc.)

Network stack

Layer	e.g.	Connects
Application	HTTP	processes
Transport	TCP	hosts
Internet	IP	networks
Link	WiFi	devices

Network stack

Layer	e.g.	Connects
Application	HTTP	processes
	SSL	
Transport	TCP	hosts
Internet	IP	networks
Link	WiFi	devices

- SSL provides secure channel atop underlying guarantees of transport layer
- HTTPS = HTTP + SSL

SSL terminology

- Record: message sent during session
- Session:
  - communication channel
  - between client and server
  - logical
  - bi-directional (and direction matters)
  - **optionally** secured for confidentiality and/or integrity against Dolev-Yao attacker

SSL protocols

- Handshake protocol: initial channel setup
- Record protocol: exchange of messages

Caveats:

- *what follows is common way of configuring those protocols, not the only way*
- *no official rationale for the protocol*

Record protocol

Connection state:

- **cmk:** client HMAC key
- **smk:** server HMAC key
- **cek:** client symmetric encryption key
- **sek**: server symmetric encryption key
- **civ:** client IV
- **siv:** server IV
- **cseq:** client sequence number
- **sseq:** server sequence number

Record protocol

Directional communication:

- both client and server are meant to know the entire state, but...
- from client to server uses cXX state
- from server to client uses sXX state

... defends against reflection attacks

Record protocol

For client to send record to server:

```
1. C: t = MAC(r, cseq; cmk);
      c = Enc(r, t; civ; cek);
      cseq++;  // if overflow, re-key
      civ = rand()
2. C -> S: c
```

MAC-then-Enc

Server to client is the same with sXX part of connection state

Handshake protocol

- Purpose:
  - Establish ciphersuite
  - Then establish connection state
- **Ciphersuite:** triple of cryptographic choices...
  1. Protocol for key establishment
  2. Block cipher and mode
  3. PRF (typically a hash function for HMAC)
- Example ciphersuites:
  - **`RSA, AES128/CBC, SHA-256`**
  - **`DH_anon, 3DES/CBC, SHA-1`** **(beware DH_anon!)**
  - **`null, null, null`**
- Henceforth assume RSA key establishment...

Handshake protocol

Warning:

- attacks on SSL sometimes involve rollback to deprecated algorithms that your crypto library still supports
- YOUR responsibility to make sure only current algorithms are enabled

Handshake protocol

Could be a chain

```
1. C->S: Suites_C, N_C
2. S->C: Suite_S, Cert(S; CA), N_S
3. C: PS = rand();  // premaster secret
      ePS = Enc(PS; K_S)
4. C->S: ePS
5. S: PS = Dec(ePS; k_S)
6. C and S:
     MS = PRF(PS, "master secret"; N_C+N_S);
     derive connection state from MS
       by splitting into bits
```

Handshake protocol

See online notes for some omitted details:

- Verify that client and server have agreed on same keys
- Unilateral vs. mutual authentication:
  - **unilateral:** server authenticates to client
  - **mutual:** server authenticates to client and client authenticates to server

Upcoming events

- [today] Clarkson office hours cancelled
- [today or tomorrow] A5 out
- [next week] Schneider subs for Clarkson

Do not believe anything just because you heard it from a seeming authority. – The Buddha