Public Key Infrastructures

Chapter 5 – Public Key Infrastructures

Content by Ralph Holz

Network Architectures and Services
Technische Universität München

Version: November 2014, updated January 7, 2016

Public Key Infrastructures (PKIs)

You already know why PKIs are needed. Next:

- How can PKIs be organised?
- Where are PKIs used in practice?
- How are they deployed?
- Practical problems in deployment

Certificates: the essence of PKI

Definition of a certificate

A certificate is a cryptographic binding between an identifier and a public key that is to be associated to that identifier.

Semantics of the binding

- The identifier often refers to a person, business, etc. While much less common, the identifier may also indicate some attribute with which the key is associated (e.g., access right).
- **Always necessary:** Verification that identifier and corresponding key belong together.
- **If the identifier is a name:** verify that the entity behind the name is the entity it claims to be.

Certificate creation

PKIs are created by issuing certificates between entities

- Entity responsible for creating a certificate: the issuer I.
- I has a public key, $K_{I\text{-}pub}$, and private key, $K_{I\text{-}priv}$.
- X is an identifier to be bound to a public key, $K_{X\text{-}pub}$.
- Let I create a signature: $\mathrm{Sig}_{K_{I\text{-}priv}}(X|K_{X\text{-}pub})$
- The tuple $(X, K_{X\text{-}pub}, \mathrm{Sig}_{K_{I\text{-}priv}}(X|K_{X\text{-}pub}))$ is then a certificate.
- In practice, we add (much) more information.

Chains can be established:

I_1 may certify I_2, who certifies X: $I_1 \rightarrow I_2 \rightarrow X$. Each arrow means a certificate is issued from left side to right side.

Common forms of PKI

We can now classify PKIs by looking at:

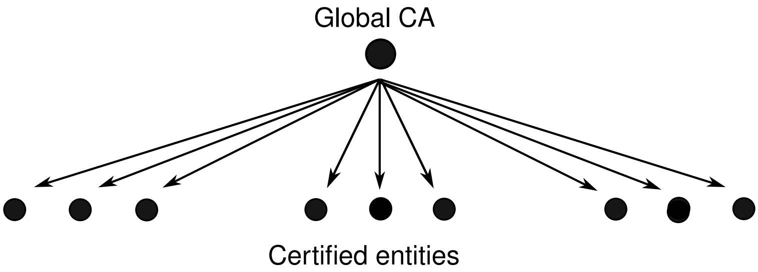
- Who are the issuers?
- Which issuers must be trusted = which TTPs exist?
- How do issuers verify that X and $K_{X\text{-}pub}$ belong together, or that X is really X?

Some terminology

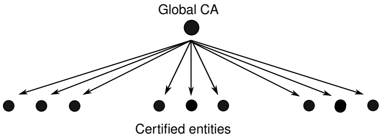
- Depending on the PKI, different words for issuer
- Often in hierarchical PKIs: “Certification Authority” (CA)
- In non-hierarchical PKIs sometimes: “endorser”
- These words often hint at the role (power) of the issuers

Hierarchical PKIs

Naive form



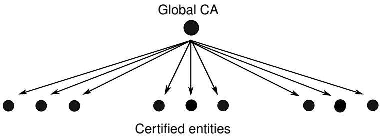
Hierarchical PKIs



This is a very impractical form.

- Why?

Hierarchical PKIs

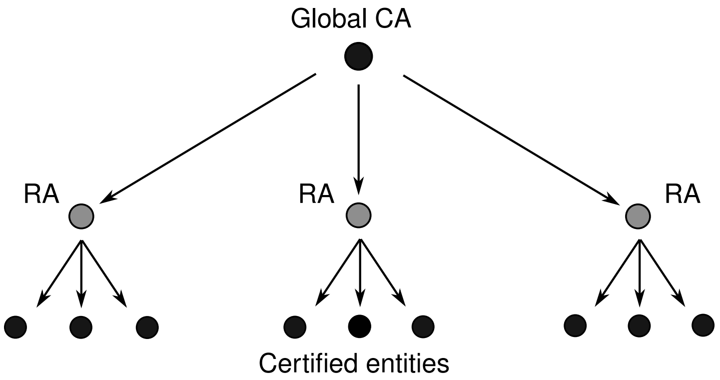


This is an infeasible form.

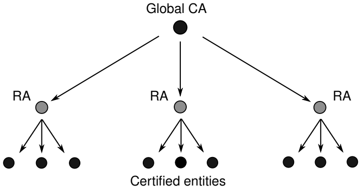
- Who decides which global authority is trustworthy for the job?
- What are the agreed verification steps?
- Namespace is global—unique global identifiers needed
- This, and the high load on the CA, may make it easier to trick the CA into misissuing a certificate to, e.g., wrong entity (X')
- Hard to imagine any government would rely on an authority outside its legal reach.

Improved (but still simple form)

Introduce intermediate entities helping the CA



Registration Authorities (RAs)



Role of RAs

- Do the verification step: identify X, verify it has $K_{X\text{-}priv}$
- Verification may be according to local law
- RAs do *not* issue certificates—they are mere proxies
- Problem of single trusted authority remains
- The namespace remains global

‘Practical’ solutions to the problem

Many global CAs

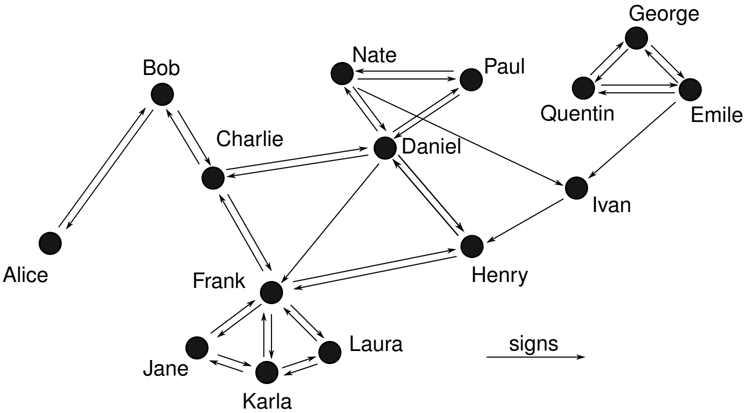
- One global CA is infeasible, even with RAs
- Use many CAs, in different legislations, accept them all equally
- There are serious weaknesses in this model
- Which ones?

Defining CAs as trusted

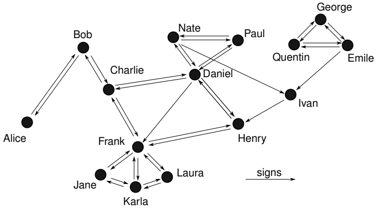
- A CA must be trusted by participants in order to be useful
- How should participants decide which CAs to trust?
- ‘Solution’: operating systems and software like browsers come preconfigured with a set of trusted CAs

Form without hierarchy: Webs of Trust

Every participant may issue certificates



Webs of Trust



Webs of Trust may also take many forms:

- Trust metrics to automatically reason about authenticity of bindings between entity and key
- E.g. introduce rules how many delegations are allowed, store explicit trust values, etc.
- Namespace may be global or local (→ PGP vs. SPKI, later)
- CAs may act as ‘special’ participants

Currently deployed PKIs

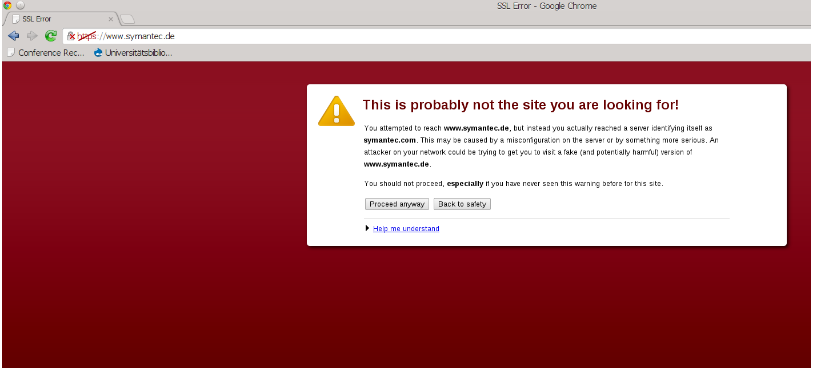
Hierarchical PKI(s) with *many* CAs

- Most widely deployed PKI type at the moment, based on the X.509 standard
  - Very common: X.509 for the Web (SSL/TLS + HTTP), regulated
  - Common, but less regulated: X.509 + SSL/TLS to secure IMAP, SMTP
  - X.509 also used with IPSec, etc.
- Common: X.509 for email (S/MIME)
- Much less common: X.509 for code signing

Webs of Trust

- OpenPGP for email
- OpenPGP for code-signing

A typical X.509 experience



WWW: SSL/TLS + HTTP = HTTPS

SSL/TLS

- Backbone protocols for securing many other protocols.
- SSL/TLS works as a layer between TCP/IP and the application layer.
- Goals: authentication, confidentiality, integrity
- SSL/TLS employ X.509

Origins of X.509

- Part of the X.500 family of standards (ITU)
- X.500 vision: global directory to store and retrieve entity information
- All information stored in a tree—strict naming discipline
- X.509 is the certificate standard in X.500
- CAs and subCAs responsible for controlling access to subtrees
- X.500 never saw much deployment
- But the X.509 certificate standard was reused by the IETF to create a certification standard, in particular to link domain names to public keys
- The concept of a tree was given up—any CA can issue certificates for any domain

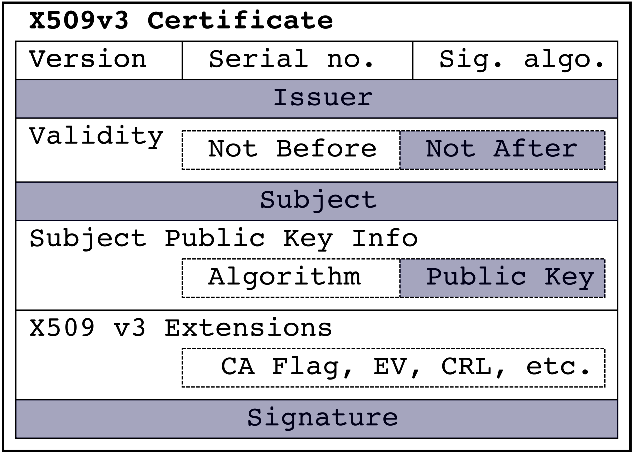
X.509 and SSL/TLS

SSL/TLS include certificate-based authentication

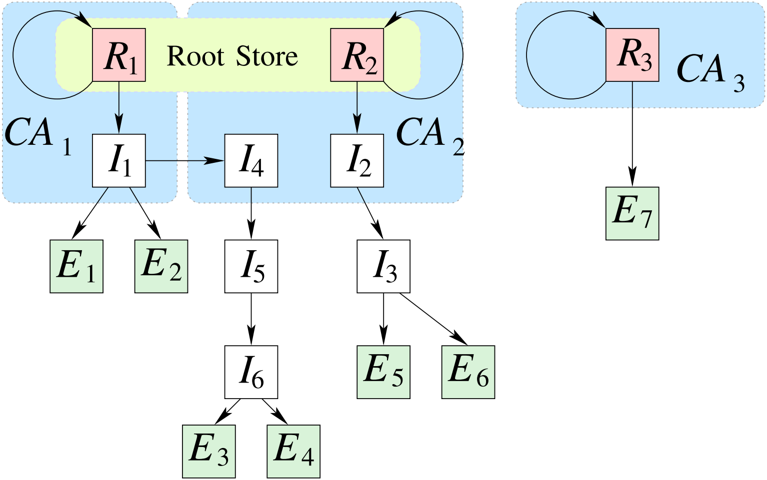
- Original design of SSL by Netscape (Mozilla!)
- Goal: protect sensitive information like cookies, user input (e.g., credit cards)
- The attack model in mind was more a criminal attacker, less a state-level attacker

X.509 for the WWW

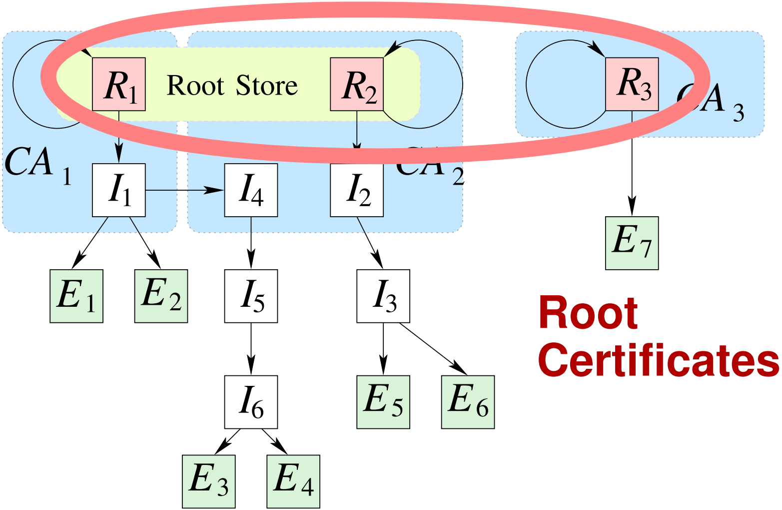
An X.509 Certificate



Many global CAs



Preconfigured as trusted in root stores



Root stores

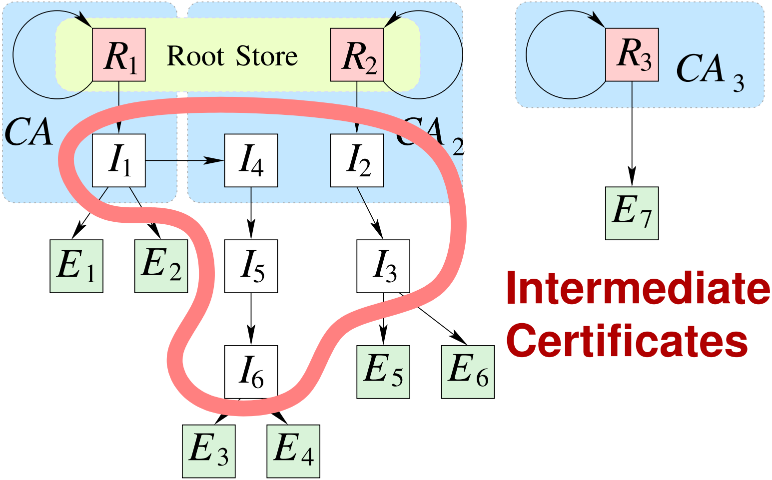
Root stores: certificates of trusted CAs

- ‘Trusted’ = trusted to issue certificates to the correct entities
- *Every* application that uses X.509 has to have a root store
- Operating Systems have root stores: Windows, Apple, Linux
- Browsers use root stores: Mozilla ships their own, IE uses Windows’ root store, etc.

Root store processes

- Every root store vendor has their own process to determine if a CA is added or not
- A CA’s *Certification Policy Statements (CPS)* are assessed
- Mozilla: open discussion forum (but very few participants)
- Commercial vendors (Microsoft, Apple): little to no openness

Intermediate Certificates



Intermediate Certificates

Intermediate certs: part of a certificate chain, but neither a root certificate nor an end-entity certificate.

There are two primary reasons to use intermediate certificates:

- To delegate signing authority to another organisation: sub-CA
- Protect your main root certificate:
  - Intermediate cert is operated by the same organisation
  - Allows to store root cert in the root store, but private key may remain offline in some secure location
  - Online day-to-day operations can be done using the private key of the intermediate cert
  - Also makes it very easy to replace the intermediate cert in case of compromise, or crypto breakthroughs (e.g. hash algorithms) etc.

Hazards of Intermediate Certificates

Intermediate certs have the same signing authority as root certs:

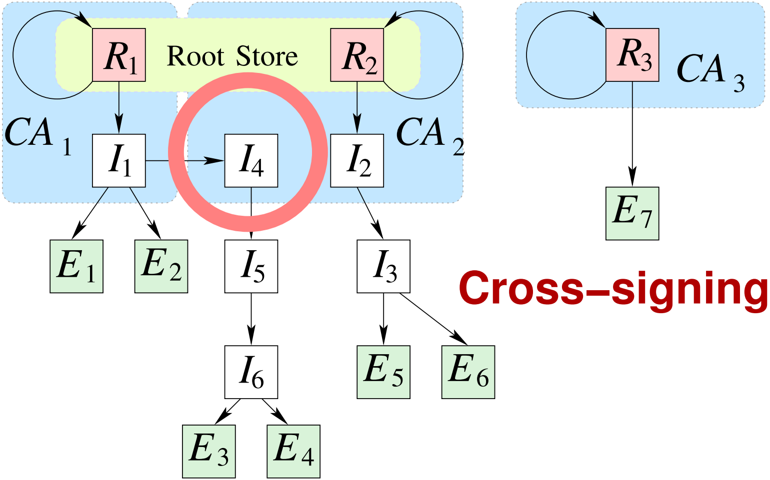
- There are no technical restrictions on what they can sign (e.g., DNS limitations)
- N.B.: DNS restrictions are in the standard, but little used
- The restriction must be supported by the client, too

Hazards of Intermediate Certificates

Some companies/organisations have SSL proxies

- They monitor their employees' traffic
- May make sense in order to avert things like industrial espionage
- However, some CAs have issued intermediate certs to be used as sub-CAs in proxies or added to client root stores
- This allows transparent rewriting of certificate chains– a classic Man-in-the-middle attack
- Worst: the holder of the sub-CA is suddenly as powerful as all CAs in the root store
- Since outing of first such CA, Mozilla requires practice to be disclosed, and stopped

Cross-signing

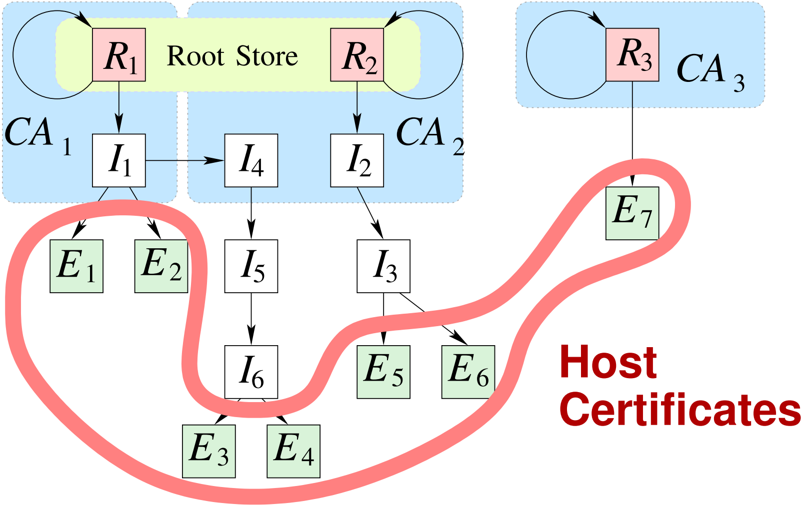


Cross-signing

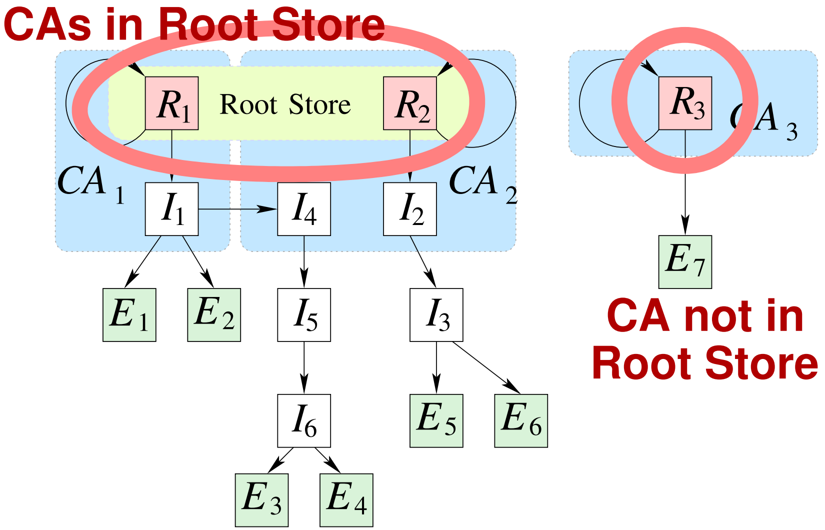
A CA signs a root or signing certificate of another CA

- A special case of intermediate cert
- In a business-to-business model, this makes sense:
  - Two businesses wishing to cooperate cross-sign each other
  - Makes it easy to design business processes that access each others' resources via SSL/TLS
- For the WWW, it completely breaks the root store model
- A new CA can be introduced, subverting control of the root store vendor
- This has happened. CNNIC (Chinese NIC) was cross-signed by Entrust, long before they became part of the root store in Mozilla
- Inclusion of CNNIC caused outrage anyway

End entities in X.509: DNS host name

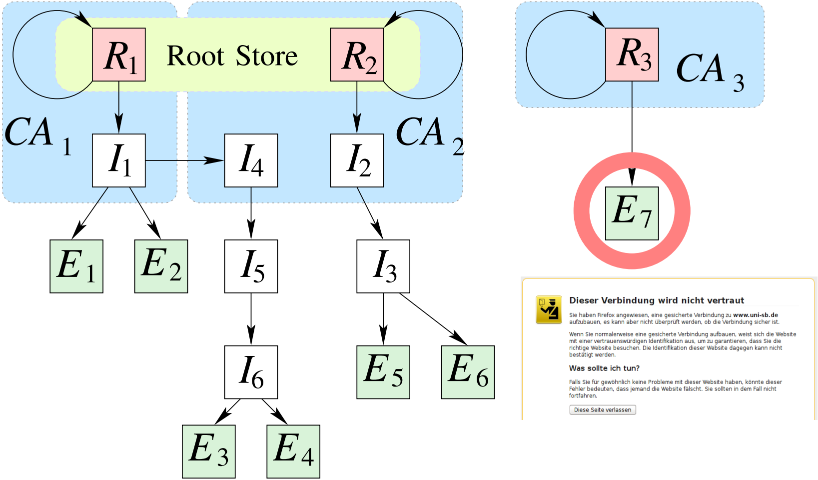


A CA is not in your root store?



One source of WWW errors

Root certificate not in Root Store



Dieser Verbindung wird nicht vertraut

Sie haben Firefox angewiesen, eine gesicherte Verbindung zu **www.uni-sb.de** aufzubauen, es kann aber nicht überprüft werden, ob die Verbindung sicher ist.

Wenn Sie normalerweise eine gesicherte Verbindung aufbauen, weist sich die Website mit einer vertrauenswürdigen Identifikation aus, um zu garantieren, dass Sie die richtige Website besuchen. Die Identifikation dieser Website dagegen kann nicht bestätigt werden.

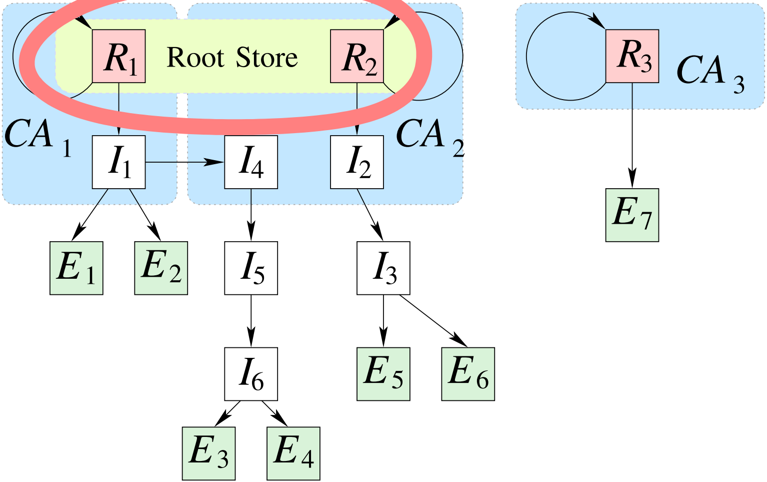
Was sollte ich tun?

Falls Sie für gewöhnlich keine Probleme mit dieser Website haben, könnte dieser Fehler bedeuten, dass jemand die Website fälscht. Sie sollten in dem Fall nicht fortfahren.

Diese Seite verlassen

Root Stores Contain CA Certificates

CAs in Root Store



Browser (Client) Root Stores

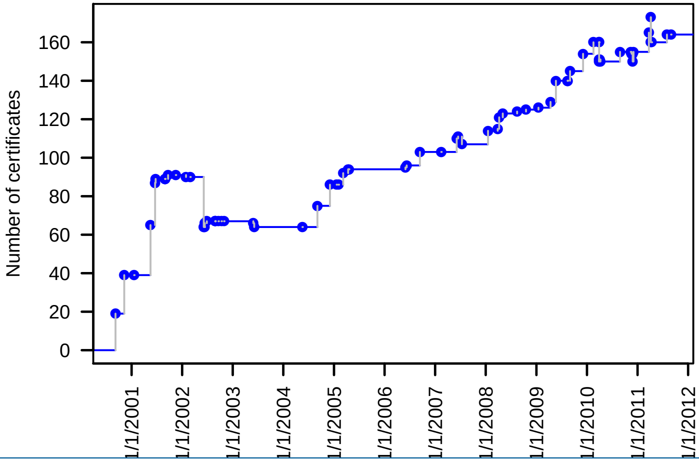
Remember:

- Your browser or your OS chooses the ‘trusted CAs’. Not you.
- All CAs have equal signing authority (there are efforts to change this)
- Any CA may issue a certificate for any domain.
- DNS path restrictions are a possibility; must be set by the CA in their signing cert
- A globally operating CA cannot feasibly set such restrictions in their root cert

The weakest CA determines the strength of the whole PKI. This is also true if the CA is a sub-CA.

Development of Mozilla Root Store

At times, more than 150 trustworthy Root Certificates



Certificate Issuance

How is a certificate issued in practice?

- Domain Validation (DV):
  - Send email to (CA-chosen) mail address with code
  - Confirmed ownership of mail address = ownership of domain
- Extended Validation (EV): require (strong) legal documentation of identity
- Organisational Validation (rare): between DV and EV; less documentation

BTW: Kurt Seifried vs. RapidSSL

How to hijack a Web mailer in 3 easy steps

- Step 1: register e-mail address: `ssladministrator@portugalmail.pt`
- Step 2: ask RapidSSL for certificate for portugalmail.pt, giving this address as your contact
- Step 3: Watch ‘Domain Validation by e-mail probe’ fail

Kurt succeeded. It cost him $<$ 100 USD.

Main failure here:

- Web mailers and CAs have not agreed on ‘protected’ addresses
- This issue is now in Mozilla’s ‘Problematic practices’

Economics and security

PKI is a good area to study dynamics and interplay of economics and security

- Incentive to lower prices → less checks, makes certification cheaper
- Actually not true! Results of a study (2013):
  - Empirical (quantitative) part: the more expensive CAs have more customers
  - Quantitative part: in interviews, customers say they prefer a CA that is ‘too big to fail’ and will never be removed from root stores
  - Indeed, large CAs are difficult to remove from root stores as the Web browser would suddenly show errors for many sites!
- This shows customers behave rationally correct, but different from what designers of security system would have expected

Certificate Revocation

Revocation is crucial—yet often neglected in discussions

- No certificate can be considered valid without a revocation check
- This is because we need confirmation that a certificate is valid *at the moment of interest*, not some time in the past
- Consider this: Milhouse has stolen Bart's private key. Bart notices one day later. Milhouse has a window of one day during which he can impersonate Bart.
- There are several cases when an already issued certificate must be withdrawn. Examples:
  - Corresponding private key compromised
  - Certificate owner does not operate service any longer
  - Key ownership has changed
- In these cases, there are two options: CRLs and OCSP

Certificate Revocation Lists (CRLs)

A CRL is a list of certificates that are considered revoked

- They are (should be) issued, updated and maintained by every CA
  - Certificates are identified by serial number
  - A reason for revocation can be given
  - Every CRL must be timestamped and signed
- There are further entries, like time of next update
- Technically, a browser (client) should download CRL (and update it after the given time), and lookup a host certificate every time it connects to a server

Problems with CRLs

CRLs have a number of problems

- Intermediate certs should be checked, too – induces load and network activity
- There is a time interval between two updates (window for attack)
- CRLs can grow large
  - Response to this: Delta CRLs that contain only latest updates
  - Requires server side support—very rarely used
- Downloads of CRLs can be blocked by a Man-in-the-middle
- For these reasons, browsers have never activated CRLs by default

Online Certificate Status Protocol (OCSP)

OCSP allows live revocation checks over the network

- Query-response model
- Query = lookup of a certificate in a server-side CRL-like data structure
  - Query by several hash values and cert's serial number
  - Replay protection with nonces
  - Query may be signed
  - Does not require encryption
- Response:
  - Contains cert status: `good`, `revoked`, `unknown`
  - Must be signed

Problems with OCSP

There are a number of issues with OCSP:

- Lookups go over the network – induces latency
- OCSP information must be fresh. Not just from CRLs.
- OCSP servers must have high availability
- OCSP can be blocked by a Man-in-the-middle—many browser will ‘soft-fail’ = show no error
- Privacy! OCSP servers know which sites users access
- Browsers ‘accept as good’ if no OCSP response received
- *“[OCSP was] designed as a fully bug-compatible stand-in for CRLs”* – P. Gutmann

OCSP Stapling

Addresses several problems of OCSP

- Problems addressed: latency of lookup, load on CA
- The idea is thus that servers request fresh OCSP ‘proof’ from CA: ‘this certificate is still considered valid’
- This can be done at regular intervals
- The ‘proof’ is ‘stapled’ to the certificate that the server sends in the SSL/TLS handshake
- Reduces load on CA
- Although around for a long time, the idea is only now gaining traction
- Solves privacy problem

New approaches to revocation

In-browser revocation lists:

- Browsers preload a list of revoked certificates for the most common and important domains
- Updates are distributed via the browser’s update mechanism
- This counters the devastating attacks where traffic to the CA is dropped—but the scalability is not good

Short-lived certificates

- Give certificates a very short validity period (1 hour–1 day)
- Replace certificates fast, do not attempt any other revocation
- Works well and gives very clearly defined window of attack
- Problem: certification becomes a frequent and ‘live’ operation—shunned so far for the Web

Revocation: lessons learned

Revocation is crucial—but no silver bullet so far

- It is probably safe to say that CRLs never worked and are of very limited use
- OCSP checks are expensive, too (latency, load)—and not sufficient against an attacker who drops traffic to the CA
- OCSP stapling is an improvement
- Revocation is an unsolved problem

Proposals to enhance X.509

Pinning (TOFU)

Aim: reassurance of a certificate's authenticity

- As a defence against rogue CAs issuing malicious certs
- Idea: client stores information about a host/Web site on first contact
- Most commonly: store the public key of a site
- Use this information to re-identify a site later
- E.g. if public key is suddenly different on next connect: warn user

Pinning assumes a secure first connection

- Thus also known as 'trust-on-first-use'
- Inherent bootstrapping problem

Two pinning variants

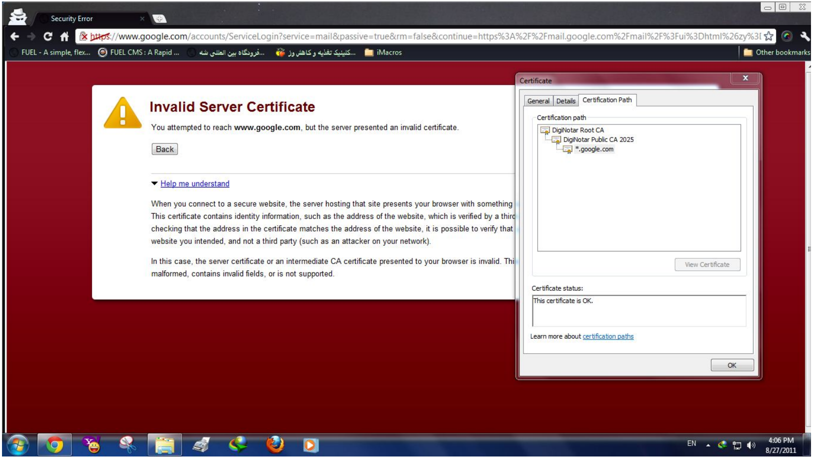
Static pinning

- Preloaded pins:
  Google Chrome, Mozilla Firefox (smallish number)
- User-driven pinning:
  add-ons for browsers that allow users to store and compare public keys of sites

Dynamic pinning

- Idea: communicate helpful information to aid clients with pinning

DigiNotar vs. Iran?



Issues to solve

Depending on the variant, pinning has shortcomings:

- For certain users, secure first contact may not be possible
  - E.g. dissidents in authoritarian countries
- Life-cycle problem
  - Servers may (legitimately) update/upgrade their keys
- Scalability
  - Browsers cannot come preloaded with pins of all sites, and keep them up to date

Public log schemes

Idea of a public log

- Public logs store some information publicly and append-only
- They sign every new entry and establish a ‘history’ of entries
- Public logs are neutral. Their only role is observe and assert their observations by signing them.
- **Certificate Transparency (CT):** logs for X.509
  - **Aim:** make **transparent** who issued certificates to whom, and when
  - **Anyone** can verify logs’ content and/or their correct operation
  - Enables detecting rogue CA issuing certificates for a domain
  - Different logs around the globe, run by different parties
  - **After-the-fact** solution; no direct defence for clients

Certificate Transparency in Google Chrome