Requirements for Building a PKI

Ryan Sleevi

SAAG @ IETF 109

November 19, 2020

Who this talk is for

You are designing/implementing/using a protocol or format that uses certificates

- On the Internet
- For a generally available service
- That should work “out of the box”
- Interoperably with other vendors
- “Securely”

Who this talk is for

- You want to use the Web PKI, and don’t see any problems
- You want to use the Web PKI, but are unhappy with how it is works
- You’re not sure what the Web PKI is, but need a PKI
- Bonus: You think mutual-TLS is a good idea / easy solution

This is not a “Web PKI is awesome” talk.

This is a “painful lessons from Web PKI, and how you can avoid them” talk.

- Evolution of Internet and Web PKI
- Requirements for Internet PKIs

Evolution of Internet and Web PKI

PKI 0.1: **The** PKI (1988)

- X.509 (11/88), a.k.a X509v1
- An integrated, integral part of the Directory (§ 0.1)
- An authentication specification that also covered password-based authentication and hashing (§ 5)
- Assumptions:
    - There is One Right Way to name things, and it is Distinguished (§ 6.3)
    - There is One True Directory and it is universally accessible and connected (§ 0.4 / § 1.2)
    - Every answer you seek can be found in The Directory (§ 6.5 / § 7)

Internet PKI 0.1: ONs and the IPRA (1989 - 1993)

- Privacy Enhanced Mail (PEM) - RFC 1114 and RFC 1422
- There is no inherent dependency on The Directory
- Approach One (RFC 1114)
  - There is One Right Way to name things, and it's contractually determined by RSADSI (but looks like X.500)
    - But organizations can name things how they want, within their Organizational Notary (ON), which RSADSI will limit to that organization
  - The Directory is not necessary, nor prohibited (§ 3)
  - RSADSI is the co-issuer (runs the infrastructure) for CAs, since security is hard (§ 3.1)
- Approach Two (RFC 1422)
  - There are Many Ways to name things, and the IPRA will maintain a database of hashes to prevent collisions (§ 3.4.2.2)
  - The Directory may not happen (§ 1)
  - ISOC will run the root of trust, the Internet Policy Registration Authority (§ 3.4.2), registering Policy Certification Authorities (PCAs), which will determine CAs that meet those policies. (§ 3.4.2.1)
  - Every answer you seek can be found in the message headers.

Web PKI 0.1: Netscape and SSL (1994)

- Netscape announces SSL in November 1994
- Goal:
  - Rough Consensus and Running Code (but without the IETF)
  - Enable commerce on the web and avoid proprietary lock-in if Microsoft did it first
  - Require minimal changes to existing software.
- Used X.509, because RSADSI said that was a good idea
- No external dependencies
  - Certificates are a flat hierarchy with no certificate chains. Certificates are verified against an embedded list of RSA-licensed CAs. Everything needed is delivered in-band.
- No naming rules
  - Naming was a problem to be solved later.
  - Users should manually inspect the name, and CAs should name things accordingly.
  - Binding to domain names came after it was pointed out as trivially spoofable in Dec ‘94.

(Internet) PKI 1.0: PKIX (1995 - 2008)

- IETF 33: Secure Socket Layer BOF
- IETF 34: First meeting of Public Key Infrastructure (X.509)
- Attempting to use PEM revealed a number of design limitations with X.509
  - Solution: X.509v3 (Finished August 1995)
- No presumed dependency on The Directory
- Many Roots of Trust
- Many ways to name things (X.500, EDI, IP, DNS, RFC 822)
- Goal:
  - Minimal configuration by users
  - Minimal interactivity requirements
  - Minimal cognitive overhead
  - Automatable Certificate Enrollment and Management
- Milestone: RFC 2459 (1999-01)

Web PKI vs Internet PKI: 1995 - 2007

- (Netscape) Web PKI assumed user interactivity; PKIX did not
- Web PKI assumed strong connectivity to Internet; PKIX did not
- Web PKI was focused on browsers (Microsoft vs Netscape) at first, but got messy quickly.
    - IE became part of the OS (and brought the PKI with it), and targeted the whole Internet
    - Netscape broadened protocol support, first with SSL licenses, then Communicator
- Result: Blurry lines between Web and Internet PKIs, the default CAs, and the policy expectations of those CAs
- Early days: Anyone could be a CA, as long as you were trustworthy. Reputation management rather than risk management.
    - Audits came later (~1999 - 2001)

Web PKI 0.9: CA/Browser Forum & EV (2007 - 2011)

- The blurring of the Web PKI (which started with legal names) and Internet PKI (which focused on Internet naming) created conflict
  - CAs unhappy that their competitors “only” validated domain names and automated certificates.
  - Browsers unhappy that CAs were wildly inconsistent in how they named things and what parts of PKIX they supported.
  - Phishing was a big issue. Browsers believed they could solve this with user education and getting them good information.
  - Solution: Extended Validation Guidelines, which defined a new naming scope (EV), which would rely on human factors to interpret (using UI)
- The point at which PKIX (with the focus on automatic validation, minimal user interaction) and Web PKI (with the focus on browser-specific behaviors) began diverging

Web PKI 1.0: Baseline Requirements (2011 - 2014)

- While EV introduced a new name scope, raising the ceiling, there was still concern about what the floor should be.
    - A number of CAs still arguing domain-only certs were bad, and UI was necessary for all certs.
    - Browsers were still keen to have a set of minimum requirements.
- Then DigiNotar happened
    - Shook trust in the whole system
    - Baseline Requirements adopted shortly thereafter, with a controversial version that preserved both domain and organization certificate policies.
- Birth of the Web PKI “as we know it”
    - Policies defined by browsers, based on what browser software does, and what browser software needs, and explicitly told CAs what they could not issue.
    - Deprecation of 1024-bit certificates
        - Adopted 2011-11-22
        - Target: 2013-12-31
    - Deprecation of SHA-1
        - Adopted 2014-10-14
        - Target: 2016-01-01

Web PKI 2.0: Deprecation of Legacy Symantec PKI (2017-

- Symantec: Largest CA. Key material going back to the very first root CA from RSADSI/VeriSign.
  - After a string of operational and security issues, browsers decided to remove trust in the CA
  - Roots were embedded on “virtually” every PKI-enabled device, making the CA the most widely interoperable, most widely desired root CA.
  - Assumed “too big to fail” because of the interoperability issues
- The most visible splitting point between the Web PKI and Internet PKI
  - A single web server providing an API endpoint for a non-updatable device, such as a TV or set-top box, and accessed by Web browsers, could no longer use the same certificate for both. The Symantec certificate needed for legacy would not work with modern, and modern CAs would not work with legacy devices.
    - **Different PKIs for different clients**
  - Such certificates would continue to work in OSes and other clients, such as e-mail, for months to years after, but would not work in browsers.
    - **Different PKIs for different protocols**

Requirements for Internet PKIs

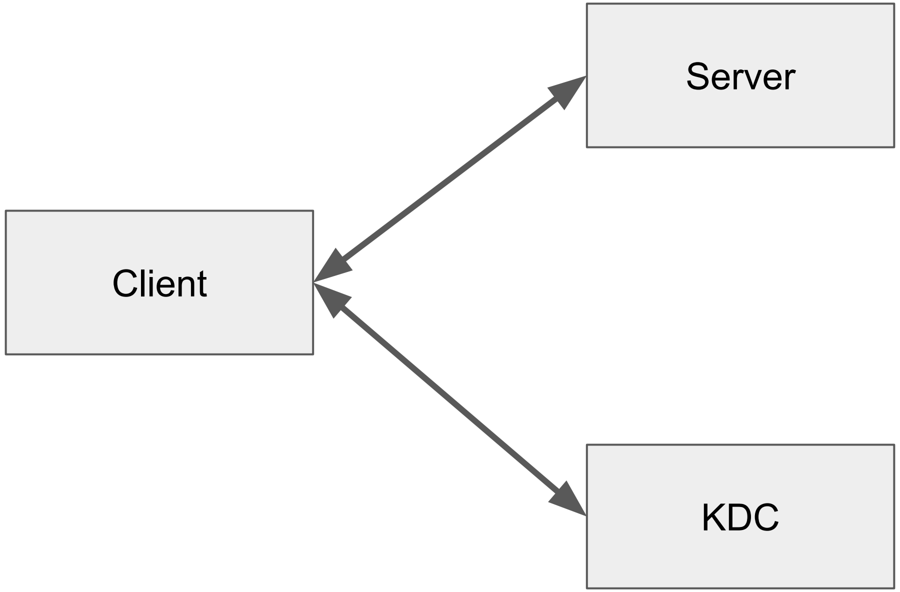
Assumptions

You are designing/implementing/using a protocol or format that uses certificates

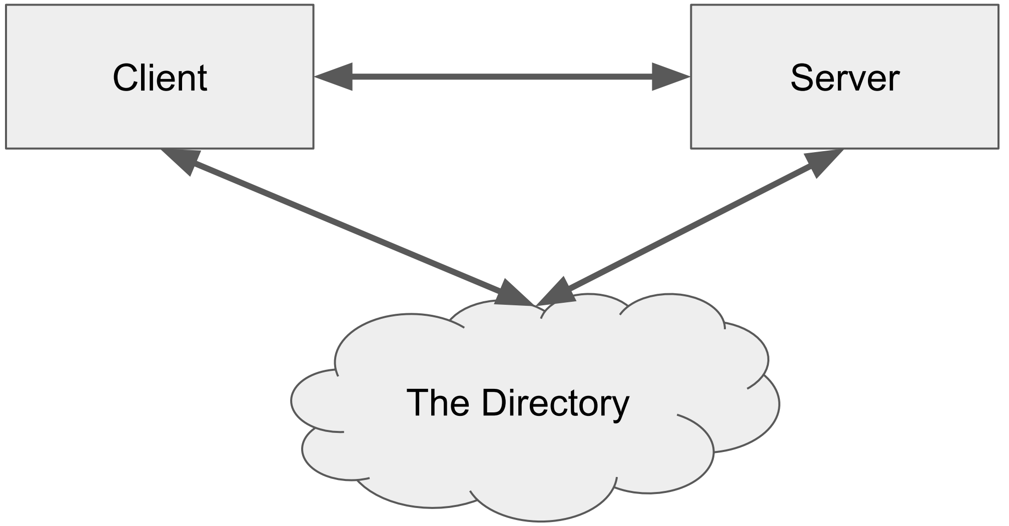
- On the Internet
- For a generally available service
- That should work “out of the box”
- Interoperably with other vendors
- “Securely”

What is your connectivity model?

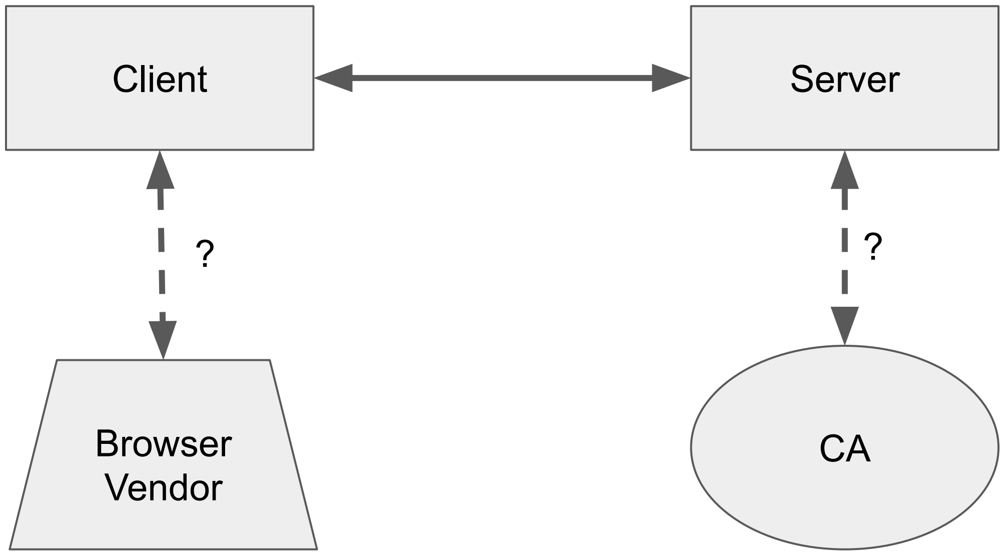
Connectivity - Kerberos



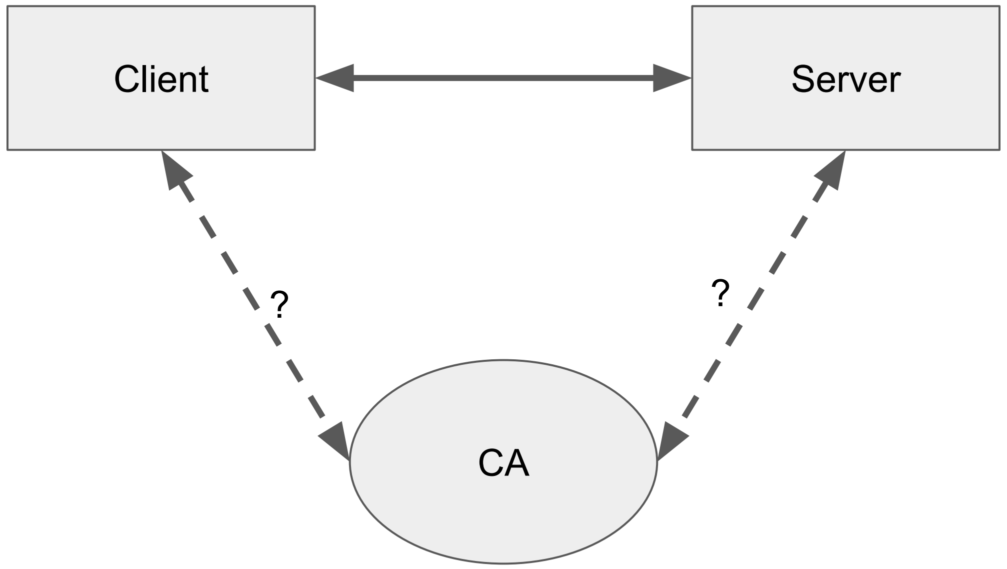
Connectivity - X.509v1



Connectivity - Web PKI



Connectivity - PKIX



Connectivity - Why it matters

- Affects every aspect of both technical design and policy implications
- Example:
    - Client cannot talk to external entities
    - Server cannot talk to external entities
    - Problem: No ability to fetch/deliver revocation data
    - Solution: Short-lived certificates
    - Problem: Certificate lifecycle management, re-evaluating whether third party CAs are safe provisioning and rotation, and think about whether or not trusting a third-party is “safe”
- Example:
    - Client cannot talk to external entities
    - Server **can**, meaning OCSP Stapling is viable
    - Problem: If Client-built path != Server-built path, OCSP responses aren’t usable.
    - Solution: Strict requirements on certification paths to avoid ambiguity (e.g. no x-signing)
- Practical Example: EAP-TLS

What is your naming scheme?

Naming - Questions

- Is your naming scheme context-dependent (civil/legal names) or is it globally unique (DNS names, “The Directory” Distinguished Names)?
- How many naming authorities do you have involved?
    - DNS zones can involve multiple naming authorities.
        - Should that be reflected in the PKI hierarchy via CAs?
        - Should that be addressed via lookup protocols like CAA (RFC 8659)?
    - “RFC 822” (5322) names have local-part @ global-part.
        - Should that be done through a constrained sub-CA?
        - Or should users be able to bring-their-own address without global-part’s explicit approval?
- Even for a domain name, there may be different authorities based on port/service. Should an HTTP server on port 80 be able to obtain a certificate for an SMTP server on port 25?
    - Do you use SRVNames to bind to a particular service?
    - Or do you use dNSName plus extendedKeyUsage to bind to a particular application purpose?

Naming - Questions (continued)

- Are names stable, or do they frequently change?
  - RSADSI contracts required 2Y certs. PEM got RSA to allow shorter-lived (<=1Y) certs.
  - Web PKI CAs started with 1Y. Then went to 10Y. Took 9 years (2011 - 2020) to get CAs back to 1Y certs.
  - Domain names change more frequently than annually! (e.g. BygoneSSL)
    - Is this a problem for lifetimes? For revocation? It depends!
  - Web PKI 0.1 was a quick hack, because “that Kaufman/Eastlake proposal” (RFC 2065/DNSSEC) wasn’t done yet, and required too much to change to be quickly usable
- Are names unambiguously, interoperably machine parsable?
  - Domains: ish! “preferred name syntax” vs underscores.
  - URLs: Real world of running code is unbearably messy

Who can issue certificates?

Issuance

- DNSSEC
  - Pro: Every zone can be independently managed
  - Con: Every zone has to be practically managed
- Web PKI
  - Pro: There’s a set of CAs that can issue for any domain
  - Con: There’s a set of CAs that can issue for any domain
  - Assumption: DNS is still “always” consulted by clients, so DNS/BGP have to lie/be coerced.
- PKIX
  - Pro: Everyone can federate their PKIs using cross-signing and bridges so you can be selective in who you trust
  - Con: Everyone ends up federating their PKIs through cross-signing and bridges, making it unclear who you actually trust

Who is the Policy Authority?

Policy Authorities

- Common policy is necessary for security and interoperability
  - If I can't say X is not secure, because you disagree, I don't have security.
  - If I can say X is not secure, but you disagree, we don't have interoperability.
- Every PKI needs a new policy, and every policy needs a new PKI
  - certificatePolicies OIDs aren't sufficient. A CP OID that says "I won't issue SHA-1", with a signed SHA-1 hash, is both nonsense and "working as intended".
  - Cannot rely on network effects, because network effects in security penalizes the first mover and favors the late mover
- "Someone" who can answer these questions
  - In general, in order for something to work "out of the box", the vendor shipping that product needs to be happy with the policy, which generally means they need to be setting or able to set that policy, for all of the CAs that will work out of the box.

What is your certificate profile?

Certificate Profile

- When everything is extensible, nothing ends up being extensible
  - Draft-iab-protocol-maintenance
- RFC 5280 is the starting point, not the end point
- The PKI is intrinsically part of, and depends on, the protocol being used
  - Ex: Revocation vs Expiration depends on connectivity
  - Ex: Acceptable CRL / OCSP sizes depends on bandwidth
  - Ex: Acceptable protocols (LDAP, HTTP, CRL) depend on purpose
  - Ex: Naming authority determines how many CAs your PKI can expect, which affects how much time is spent verifying the PKI, and how much work attackers can cause you to needlessly do
  - Ex: Trust anchor management depends on updatability and connectivity

Certificate Profile

- Profile is not just about how “a” certificate looks, but must also address the overall hierarchy
- Do you allow cross-signing? Mesh and bridge CAs?
- How deep can a certificate chain be?
- How many possible paths can a certificate have?
- How will you determine which path to verify? (RFC 4158)

How will you audit this?

Audits

- Do you like checkboxes or do you like descriptions?
  - Checkboxes (e.g. ISO 17021/17065)
    - Good for interoperability/compliance testing. Add a test case for every edge case.
    - Bad for security testing. Requires you to consider every possible “bad” thing and add a test to make sure it doesn’t happen.
  - Descriptions (e.g. ISAE 3000)
    - Good for security. Describe the system and controls for how they ensure they meet the requirements. Flexibility in how the security goal is achieved.
    - Bad for interoperability/compliance testing. Good intentions don’t always lead to good results.
  - Either way, it will be specific to your protocol and PKI needs and requirements.

Audits

- Who will perform these audits?
  - Do you perform them?
  - Does the policy authority perform them?
  - Does some independent third-party perform them?
  - If using a third-party, who determines what third parties are acceptable?
  - And who decides what acceptable third-party to use?
- Who defines the criteria for the audits?
  - See above complexity
- Are you more interested in the past or the future?
  - Which is more important: The auditor verifying that they have complied and have evidence of compliance, or verifying that they're likely to comply, by having good processes in place?
- How will you verify audits?
  - Are they machine-readable or human-readable?

Conclusions

- There is not a single Internet PKI. PKIs must be tailored to the use case, clients, and protocol.
    - This means restricting what RFC 5280 permits.
    - RFC 5280 provides the tools necessary to build a car, a pool, or a house. Be careful you don’t build a floating RV, because the maintenance costs are terrible.
- It’s possible to build interoperable PKIs that are simple, easy to use, and work out of the box
    - However, it is not trivial, nor cheap, to do.
- Ensure every single byte is critically necessary for your use case
    - Every piece of extensibility introduces new challenges to interoperability, compliance, and security.
    - GREASE is hard to apply to PKIs.
- Consider protocols that support multiple PKIs
    - Allows for both versioning and migration. Don’t make Netscape’s mistake.