**Ecosystem**

- HTTPS for all sites, unencrypted HTTP deprecated
  - Alternatively: Anyone who wants a cert should be able to get a cert when they want a cert
- .onion and DV?
- HTTPS being the default mode of browsers; HTTP connections specifically marked as untrusted
- Full logging to Certificate Transparency of all issued certificates
  - Alternatively: Some other mechanism, related or unrelated to CT
- Automatic, public handling of CA misissuance (ala Safebrowsing)
- Automatic issuance of certificates using standard APIs (like ACME)
- Automatic, fully functioning support for OCSP stapling in servers
- Removal of support for Extended Validation Certificates (UI)
- Root Store auto-update across clients

- Require OCSP-Staple or short lived certs
- On the client side, solve the clock problem. Roughtime?
- Increase usage of name constraints, so that fewer CAs are authorized for the whole web

**Policy**

- Browser approval required for every cross-sign or root transfer
- No “withdrawing” a root from inclusion, only key destruction
- Mad-lib style normalized form of audit reports (for machine reading)
- Mad-lib style normalized form of CP/CPS (for machine reading)
- Browser dictated trusted auditors?
- Separate compliance reporting and security reporting for CAs, both public

## CA Auditing

- 100% auditing of certificates for technical compliance with relevant standards
- Automated and ongoing checking of CA CRL/OCSP/ACME/test sites for compliance
- Every party responsible for domain validation is identified and audited
- Every creation of a sub-CA key requiring an audited ceremony
- Key destruction ceremony auditing
- Key transfer reporting requirements
    - Transfer of key alone
    - Acquisition of CA business

## CA/Browser Forum Documents

- Elimination of the Network Security requirements (outdated, not best practice, counterproductive in some cases, like requiring AV)
- Standards related to S/MIME certificates
- Adopt Code of Conduct

**Standards**

- CAA fully deployed and implemented
- * CAA options for indicating methods of validation that can be used (e.g. WHOIS  information versus Domain Authorization Document)
- Support for EdDSA (Curve25519, maybe Curve448)
  - Prehash variants?
- Support for post-SHA-2 hash function
- Multiple hash functions widely supported
- Post-quantum?
- CA Involvement and Implementation of further Standards work
- Faster turnaround time for adding new PKI features (key types, extensions, etc.) to the CA ecosystem
- No more parameterized algorithms.
  - Ed25519 = good

- ECDSA & RSA-PKCS1 = bad
- RSA-PSS = don’t even go there.
- One key type => one set of parameters => one OID.
- If we need, e.g., a SHA-3 Ed25519, define Ed25519v2 = EdDSA(curve25519, SHA-3-512) or whatever.

## CA Restrictions/Limitations

- Profile of CRL (must include reason codes, must use issuerDistributionPoints, must not use extensions)
- Profile of OCSP (must not use nonce, must not use extensions)
- CAA required, deployed, and enforced
- Certificate lifetimes reduced to 13 months within 3 years, and 3-6 months in 5 years
- Certificate lifetimes converging on 90 days
- Reuse of cached information limited by the method used to obtain it
  - Ex: File based auth limited to 5/15 min. Legal document based limited to N months.
- All CAs support CT information delivered in OCSP responses (perhaps via profile of standard API?)

- Limit the binding of one key to one name (e.g. rekey = new cert & new name, no sharing keys among CA certs)
- Every ‘online’ subCA being constrained with pathLen:0 and limited to issue certificates for a single purpose
- No unnecessary bloat in certificates (LogoType, User Notice/CPS text)
- No hydra-headed multi-path certificates - better guidance for the signing and issuance of intermediates and cross-signs, one cert for each unique key+name tuple, etc
- Restrictions / prohibitions on revocation (objective = key compromise only, not subjective like content filtering/censoring/malware)
- Name-matching must be byte-for-byte identical, no canonicalization needed