Wednesday June 5, 2013 (9:00am)

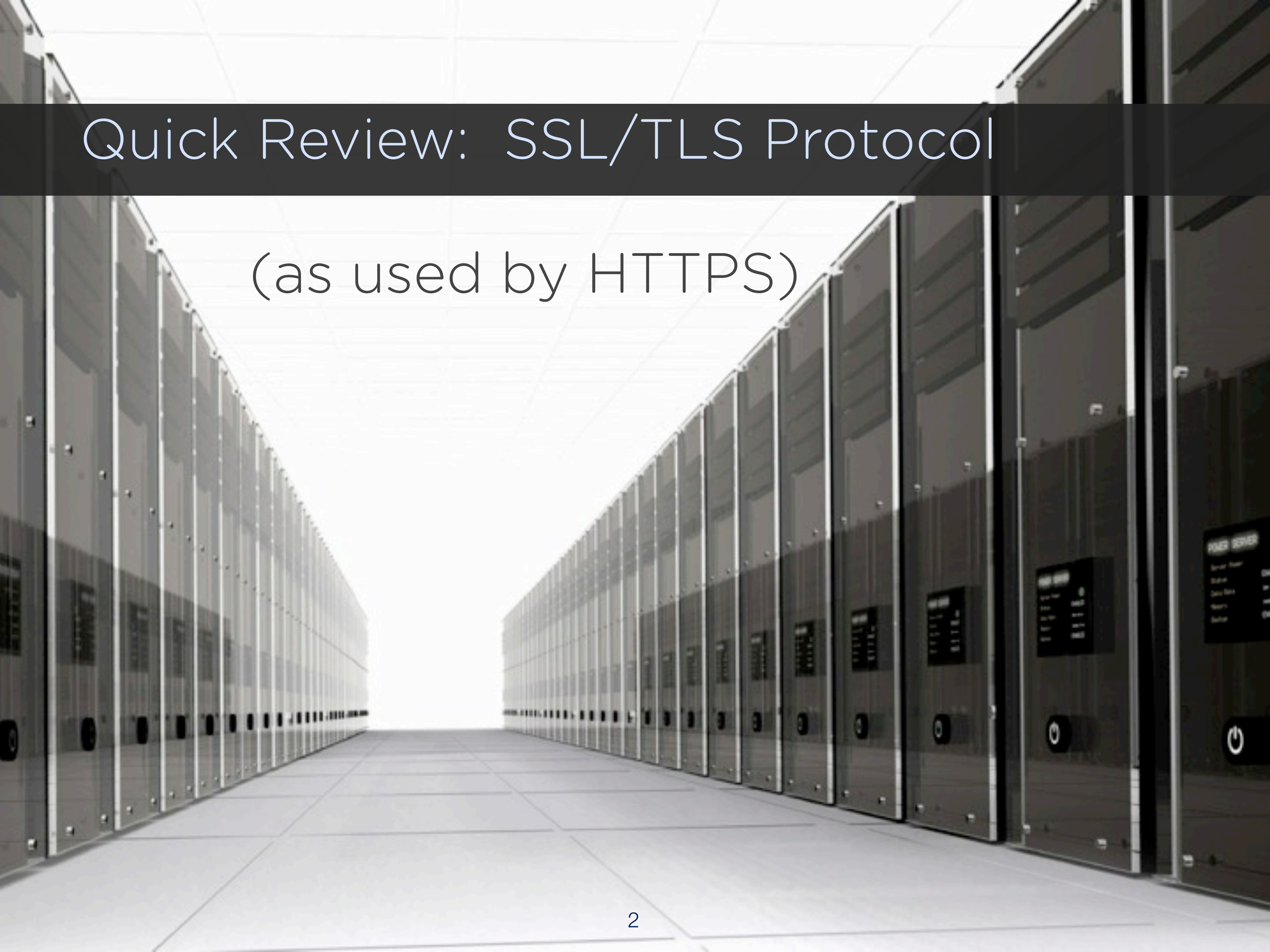
Browser Trust Models:
Past, Present and Future

Jeremy Clark & Paul C. van Oorschot

School of Computer Science
Carleton University, Ottawa, Canada

Quick Review: SSL/TLS Protocol

(as used by HTTPS)




Client
Domain.ca

Domain.ca's
Public Key

Client

Domain.ca

1) Client lists supported versions & ciphersuites
2) Server selects
3) Server sends public key

4a) Client chooses secret value & sends to server, encrypted with server's public key; or

4b) Client & server use Diffie-Hellman to derive secret; server signs values with its public key

5) Shared secret is extracted/expanded into encryption and MAC keys
6) Client MACs previous messages

7) Data is put into records, MAC'd, padded (if applicable), and encrypted

HTTPS (HTTP over SSL/TLS): What can go wrong?

1) Cryptographic security and TLS protocol itself

2) CA & browser trust model supporting TLS

   A. Certification

   B. Anchoring trust

   C. Transitivity of trust

   D. Maintenance of trust

   E. Indication and interpretation of trust

Overview

This talk will largely be exploring:

3) Enhancements to the CA/B trust model (Certification Authority/Browser)

- specifically: in SSL/TLS as used by HTTPS, how to ensure Domain.ca's public key is authentic & valid
- source: [Clark & van Oorschot] IEEE Symposium S&P 2013, "SSL and HTTPS: Revisiting past challenges and evaluating certificate trust model enhancements"

A Peak Ahead ...

Primitive	Security Properties Offered								Evaluation of Impact on HTTPS										
	A				B		C		Security & Privacy				Deployability				Usability		
	Detects MITM	Detects Local MITM	Protects Client Credential	Updatable Pins	Detects TLS Stripping	Affirms POST-to-HTTPS	Responsive Revocation	Intermediate CAs Visible	No New Trusted Entity	No New Traceability	Reduces Traceability	No New Auth'n Tokens	No Server-Side Changes	Deployable without DNSSEC	No Extra Communications	Internet Scalable	No False-Rejects	Status Signalled Completely	No New User Decisions
Key Pinning (Client History)	○	○	○						●	●		●	●	●	●	●			
Key Pinning (Server)	○	○	○						●	●				●	●	●	●		●
Key Pinning (Preloaded)	●	●	●	●					○	●		●	○	●	●		●	○	●
Key Pinning (DNS)	●	●	●	●					○	●	●		○		●	●	●	○	●
Multipath Probing		●		●								●	●	●		●		●	
Channel-bound Credentials			○						●	●		●		●	●	●	●	○	●
Credential-bound Channels			○						●	●		●		●	●	●	●	○	●
Key Agility/Manifest				●					●	●				●	●	●	●	●	●
HTTPS-only Pinning (Server)					○	○			●	●				●	●	●	●		●
HTTPS-only Pinning (Preloaded)				●	●	●			○	●		●	○	●	●		●	○	●
HTTPS-only Pinning (DNS)				●	●	●			○	●			○		●	●	●	○	●
Visual Cues for Secure POST						●			●	●		●	●	●		●		●	
Browser-stored CRL							●		○	●		●	●	●	●	●	●	●	●
Certificate Status Stapling							●		●	●	●			●	●	●	●	○	●
Short-lived Certificates							●		●	●	●	●		●	●	●	●	●	●
List of Active Certificates							●	●			●	●	●	●		●	●	●	●

Certificate Infrastructure & Trust Model [1]

Some questions related to **Certificate Authorities** (CAs) **& trust**:

* Who is allowed to become a CA? To anchor trust?
* How can this authority be delegated (transitivity of trust)?
* How are certificates revoked (maintenance of trust)?
* How do users interact with certificate info (indication, interpretation of trust)?

Certificate Infrastructure & Trust Model [2]

Issues related to DNs (X.509 **Distinguished Names**), **namespaces**:

* essential TLS attribute related to DN is: **domain name**
    * put in CN (common name) attr. under **Subject**, unless 1 or more domains given in X.509 ext. field: Subject Alt. Name
    * DV/domain-validated certificates assume domain names map to correct server IP address
* CA must validate cert request is from legitimate entity of specified Subject name; but **who controls the name space?**
    * vanilla browser trust model: any (browser-endorsed) CA can issue a browser-acceptable certificate for any site

Certificate Infrastructure & Trust Model [3]

Issues related to **browsers trust anchors** & **intermediate CAs**:

* browser vendors embed self-signed CA certs (**trust anchors**)
* site certificate is **browser-acceptable** if browser can build a certificate chain leading to trust anchor
* 100s of trust anchors (from somewhat fewer organizations) are augmented by **intermediate CAs** empowered by these
    * ~1500 CA certs from ~650 orgs in ~50 countries are browser-accepted (2010 SSL Observatory estimate)
* intermediate CA cert may be constrained in # of further CAs that it can delegate to, by {pathlen} basic constraint
* intermediate CAs invisible to clients until certs encountered-thus difficult to preemptively know/remove "bad" CA certs

Certificate Infrastructure & Trust Model [4]

A few other background items :

* MITM: view as a type of ``proxy" which breaks the expectation of SSL providing "end-to-end" protection
  * aided by fraudulent but browser-accepted certificates
  * proxy can be set up by various attack vectors (including claimed "government-compelled" certificates)
* validating received site certificate matches URL hostname:
  * current browsers do okay, but errors more common in mobile apps (e.g., Android) displaying HTTPS data, cloud clients, other non-browser software employing HTTPS

Main categories of CA/B trust model enhancements

1. **Detect or Prevent Certificate Substitution Attacks**
   - illegitimate (but browser-accepted) certificates

2. **Detect or Prevent SSL Stripping**
   - active downgrade to HTTP: adversary replaces references to HTTPS sites by HTTP (POST-to-HTTPS)
   - many users ignore security indicators, don't understand warnings, and click through them

3. **Increase reliability of revocation**

8 Properties + 11 Evaluation Criteria

(table columns)

* We now discuss properties + evaluation criteria by which we rate the various new proposals

[refresh]

Primitive	Security Properties Offered								Evaluation of Impact on HTTPS										
	A				B		C		Security & Privacy				Deployability				Usability		
	Detects MITM	Detects Local MITM	Protects Client Credential	Updatable Pins	Detects TLS Stripping	Affirms POST-to-HTTPS	Responsive Revocation	Intermediate CAs Visible	No New Trusted Entity	No New Traceability	Reduces Traceability	No New Auth'n Tokens	No Server-Side Changes	Deployable without DNSSEC	No Extra Communications	Internet Scalable	No False-Rejects	Status Signalled Completely	No New User Decisions
Key Pinning (Client History)	○	○	○						●	●		●	●	●	●	●			
Key Pinning (Server)	○	○	○						●	●				●	●	●	●		●
Key Pinning (Preloaded)	●	●	●	●					○	●		●	○	●	●		●	○	●
Key Pinning (DNS)	●	●	●	●					○	●	●		○		●	●	●	○	●
Multipath Probing		●		●								●	●	●		●		●	
Channel-bound Credentials			○						●	●		●		●	●	●	●	○	●
Credential-bound Channels			○						●	●		●		●	●	●	●	○	●
Key Agility/Manifest				●					●	●				●	●	●	●	●	●
HTTPS-only Pinning (Server)					○	○			●	●				●	●	●	●		●
HTTPS-only Pinning (Preloaded)				●	●	●			○	●		●	○	●	●		●	○	●
HTTPS-only Pinning (DNS)				●	●	●			○	●			○		●	●	●	○	●
Visual Cues for Secure POST						●			●	●		●	●	●		●		●	
Browser-stored CRL							●		○	●		●	●	●	●	●	●	●	●
Certificate Status Stapling							●		●	●	●			●	●	●	●	○	●
Short-lived Certificates							●		●	●	●	●		●	●	●	●	●	●
List of Active Certificates							●	●			●	●	●	●		●	●	●	●

Properties offered by various proposals (not in current HTTPS-CA/B offerings) [1 of 2]

1. Detecting Certificate Substitution (including browser-accepted certificates for subject domains not controlled)

A1: detects MITM
(in general: partial if requires blind TOFU)

A2: detects local MITM
(subset: local DNS cache poisoning, on-path interception)

A3: protects client credential
(protects password or cookie during HTTPS MITM)

A4: updatable pins
(resolve false-reject errors when pinned certs change)

Properties offered by various proposals (not in current HTTPS-CA/B offerings) [2 of 2]

2. Detecting TLS Stripping (downgrading HTTPS to HTTP)

B1: detects TLS stripping
(even if HTTPS request doesn't reach true server)

B2: affirms POST-to-HTTPS
(deters POST over HTTP: enforces or uses security indicator)

3. PKI Improvements

C1: responsive revocation
(even when CRLs, OCSP responses unavailable)

C2: intermediate CAs visible
(every one visible to user at any time)

Evaluation Criteria for Impact on HTTPS [1 of 3]

1. Security & Privacy

SP1: No New Trusted Entity
(partial if existing trusted party does more)

SP2: No New Traceability
(re: parties aware of sites visited over HTTPS)

SP3: Reduces Traceability
(eliminates such parties, e.g., OCSP responders)

SP4: No New Authentication Tokens
(e.g., pins, signed OCSP responses)

Evaluation Criteria for Impact on HTTPS [2 of 3]

2. Deployability

D1: No Server-Side Changes
(partial if server changes needed, but not to code)

D2: Deployable without DNSSEC
(not widely deployed yet)

D3: No Extra Communications
(new rounds which block completion of connection)

D4: Internet Scalable
(could support enrolment of all HTTPS servers)

Evaluation Criteria for Impact on HTTPS [3 of 3]

3. **Usability** (as determinable without user studies)

U1: No False Rejects
(user needn't distinguish attacks vs. FR of legitimate certs)

U2: Status Signalled Completely
(vs. user not knowing why HTTPS "succeeded")

U3: No New User Decisions
(decisions automated; no new cues or dialogues)

Primitives (table rows)

* Next: the 16 primitives extracted from the various new proposals for enhancing the CA/B model

* [primitives vs. actual proposals - see later]

[refresh]

Primitive	Security Properties Offered								Evaluation of Impact on HTTPS										
	A				B		C		Security & Privacy				Deployability				Usability		
	Detects MITM	Detects Local MITM	Protects Client Credential	Updatable Pins	Detects TLS Stripping	Affirms POST-to-HTTPS	Responsive Revocation	Intermediate CAs Visible	No New Trusted Entity	No New Traceability	Reduces Traceability	No New Auth'n Tokens	No Server-Side Changes	Deployable without DNSSEC	No Extra Communications	Internet Scalable	No False-Rejects	Status Signalled Completely	No New User Decisions
Key Pinning (Client History)	○	○	○						●	●		●	●	●	●	●			
Key Pinning (Server)	○	○	○						●	●				●	●	●	●		●
Key Pinning (Preloaded)	●	●	●	●					○	●		●	○	●	●		●	○	●
Key Pinning (DNS)	●	●	●	●					○	●	●		○		●	●	●	○	●
Multipath Probing		●		●								●	●	●		●		●	
Channel-bound Credentials			○						●	●		●		●	●	●	●	○	●
Credential-bound Channels			○						●	●		●		●	●	●	●	○	●
Key Agility/Manifest				●					●	●				●	●	●	●	●	●
HTTPS-only Pinning (Server)					○	○			●	●				●	●	●	●		●
HTTPS-only Pinning (Preloaded)				●	●	●			○	●		●	○	●	●		●	○	●
HTTPS-only Pinning (DNS)				●	●	●			○	●			○		●	●	●	○	●
Visual Cues for Secure POST						●			●	●		●	●	●		●		●	
Browser-stored CRL							●		○	●		●	●	●	●	●	●	●	●
Certificate Status Stapling							●		●	●	●			●	●	●	●	○	●
Short-lived Certificates							●		●	●	●	●		●	●	●	●	●	●
List of Active Certificates							●	●			●	●	●	●		●	●	●	●

V1: Key pinning (client history)

* browser remembers last browser-acceptable public key from a given site; warns if changed
* detects substitution attacks (if previously visited), even if substitute is browser-acceptable
* what to pin:
  * single public key
  * entire certificate chain
  * predicate over specified certificate attributes
* CertLock (Soghoian-Stamm) pins issuing CA country; Certificate Patrol (Firefox extension) pins entire chain

V2: Key pinning (server-assisted)

* server can specify (in HTTPS header or TLS extension) which certificate attributes to pin, for how long
* HPKP (Google):
  * servers specify a set of (CA, server) public keys, one of which must be present each TLS session
* TACK (Perrin-Marlinspike):
  * servers each manage a TACK key used to sign server's certificates

V3: Key pinning (preloaded)

* pre-configure a list of pins within browser, from browser vendor or other parties
* avoids issue of blind TOFU (e.g., in V1, V2)
* Google Chrome currently:
    * pins some certificates for its own domains, others on request

V4: Key pinning (DNS)

* DNS-based Authentication of Named Entities (DANE)
  * proposes servers pin their public key in their DNSSEC record
  * clients cross-check it

V5: Multi-path probing

* cross-check if certificate that client receives matches independent observers
  * detects local substitution unless all traffic to host tampered
* Perspectives (CMU)
* refined by Convergence (Marlinspike), also DoubleCheck (Columbia)
  * more general crowd-sourcing/trust delegation architecture (objective + subjective)
  * DoubleCheck probes using Tor
* more generally: cross-check any collection of certificate data
  * SSL Observatory, ICSI Notary, Certificate Transparency (Google)
* other subjective trust assertion mechanisms (by crowd-sourcing or delegated authority):
  * Omnibroker, Monkeysphere, YURLs, S-Links

V6: Channel-bound credentials

* passwords, cookies made to functionally depend on specifics of HTTPS connections
  * e.g. channel-bound cookies (USENIX 2012) cryptographically bind authentication value in cookies to site-specific "origin-bound certificate"
    * semi-persistent browser key pair generated on the fly for mutually-authenticated TLS session conveying OBC-dependent cookie
    * requires no user action (no new UI elements)
    * revised: channel ID

V7: Credential-bound channels

* prevent credential theft via MITM
* same goal as V6, but by reversing that idea
  * V6 has server accept credential if properly bound to semi-persistent client certificate
  * here client accepts server certificate based on its binding to client credential
  * assumes pre-shared password
  * DVCert (GerogiaTech): server uses PAKE-based protocol to show knowledge of client password

V8: Key manifest / Key agility

* part of functionality of pinning/multi-path probing
* changes in legitimate server certificates are difficult to distinguish from attacks, so use either
  a) key manifest (flexible list of possible-keys), or
  b) key agility update mechanism for new certificates, e.g.,
    - sign new certificate with old key; or
    - link certificate changes via master secret
* examples: server-assisted pinning, TACK, DANE, DVCert
* Sovereign Keys (Eckersley): servers publish long-term signing keys to certify service keys via a form of cross-signature

V9-V11: HTTPS-only pinning (server, preloaded, DNS)

* addresses TLS stripping - above primitives don't since begin only on HTTPS connection request, which client never gets
* configure domains to only support TLS, inform clients with pin communicated by server: in request headers or TLS extension, by a browser pre-load, or through a DNS record
* ForceHTTPS and its refinement HSTS (server-initiated pins)
* Chrome 22 has over 100 HTTPS-only pins (preloaded)
* some browser extensions like HTTPS Everywhere redirect to HTTPS version of designated sites using a domain whitelist
* SSR proposal (2006) has a site designated as HTTPS-only in its DNSSEC-signed DNS record

V12: Visual cues for secure POST

* to address some TLS stripping attacks, for sites POSTing login credentials from HTTP to HTTPS site
* new persistent security cue signals if form POSTs to HTTP or HTTPS
* SSLight browser extension:
    * green-yellow-red traffic light in login forms

V13: Browser-stored CRLs

* revocation remains problematic: unreliable, fails open
* 4 main methods (V13-V16: respective improvements)
  * CRLs and OCSP (both currently used in CA/B model)
  * short-lived certificates
  * trusted directories
* Browser-stored CRLs
  * vendor (vs. client) periodically fetches CRL distribution point or OCSP responder data, sends update to browser

V14: Certificate status stapling

* modifies distribution of OCSP responses
* certificate holders periodically acquire a signed, timestamped status report, to include with certificates during TLS setup
* Example: OCSP-stapling (RFC)
  * current RFC: only server certificates vs. full chain

V15: Short-lived certificates

* renew certificates frequently, to limit exposure vs. long-lived certificates
  * revoke by simply failing to renew
* Example (W2SP 2012):
  * 4-day lifespan = common OCSP response caching time
  * combined with browser-stored CRL and (server-assisted) key pinning

V16: List of active certificates

* trusted directories could publish a publicly searchable list of certificates (valid certificates, or historical)
* could be implemented for HTTPS as whitelist of every TLS certificate: all servers and CAs, including intermediate CAs
    * revoke by removal from list
* allows domain owners to detect fraudulent certificates
* **no full proposal** but related: Certificate Transparency (Google)
    * CT log: public record of site certificates, for discovery of suspicious certificates (vs. an authoritative whitelist)
    * no removal for revocation; site certificates only

Summary & Questions

Primitive	Detects MITM	Detects Local MITM	Protects Client Credential	Updatable Pins	Detects TLS Stripping	Affirms POST-to-HTTPS	Responsive Revocation	Intermediate CAs Visible	No New Trusted Entity	No New Traceability	Reduces Traceability	No New Auth'n Tokens	No Server-Side Changes	Deployable without DNSSEC	No Extra Communications	Internet Scalable	No False-Rejects	Status Signalled Completely	No New User Decisions
	Security Properties Offered								*Evaluation of Impact on HTTPS*										
	A				B		C		*Security & Privacy*				*Deployability*				*Usability*		
Key Pinning (Client History)	○	○	○						●	●		●	●	●	●	●			
Key Pinning (Server)	○	○	○						●	●				●	●	●	●		●
Key Pinning (Preloaded)	●	●	●	●					○	●		●	○	●	●		●	○	●
Key Pinning (DNS)	●	●	●	●					○	●	●		○		●	●	●	○	●
Multipath Probing		●		●								●	●	●		●		●	
Channel-bound Credentials			○						●	●		●		●	●	●	●	○	●
Credential-bound Channels			○						●	●		●		●	●	●	●	○	●
Key Agility/Manifest				●					●	●				●	●	●	●	●	●
HTTPS-only Pinning (Server)					○	○			●	●				●	●	●	●		●
HTTPS-only Pinning (Preloaded)				●	●	●			○	●		●	○	●	●		●	○	●
HTTPS-only Pinning (DNS)				●	●	●			○	●			○		●	●	●	○	●
Visual Cues for Secure POST						●			●	●		●	●	●		●		●	
Browser-stored CRL							●		○	●		●	●	●	●	●	●	●	●
Certificate Status Stapling							●		●	●	●			●	●	●	●	○	●
Short-lived Certificates							●		●	●	●	●		●	●	●	●	●	●
List of Active Certificates							●	●			●	●	●	●		●	●	●	●

Extra Slides: Comments on some specific proposals

HPKP and TACK

Send (via HTTP header or TLS handshake) the attributes about your certificate chain you want pinned.

Trust-on-first-use
Server-side changes
Denial-of-service
No new authority

Browser Preloads

Certificate attributes are pinned in a preloaded list, maintained by the browser vendor.

Resolves trust-on-first-use
Minimal server participation
Not scalable to millions of servers
Requires increased trust in your browser

DANE

Certificate attributes are pinned in a DNS record for your domain and distributed with DNSSEC

Resolves trust-on-first-use
Setting record scales to the internet
Distributing records: DNSSEC scalability has been debated
Records could be stapled into TLS connection
Requires increased trust in DNS system
Could be used with self-signed certificates

Perspectives & Convergence

Third party notaries relay information about the certificate they see for a domain.

No server-side changes
Performance penalty and needs high reliability
Domains may have multiple certificates (load-balancing)
Privacy issues
Trust agility: a pro or a con?

Certificate Transparency

Certificate authorities publish server certificates in an append-only log. Sites monitor the log for fraudulent certificates and report them for revocation

Detection rather than prevention
Increased visibility
Similarities to a notary: performance, tracing, etc.
Differences: one authority, sites can staple logs
To reject unlogged certificates, full CA opt-in
Relies on revocation

Predictions?

Short-term:

Pre-loading the browser with pins
(and HTTPS-only status, and revocation info)

Long-term:

DNS-pinning (e.g., DANE) and Certificate Transparency