COLLEGE OF SOCIAL & BEHAVIORAL SCIENCES
School of Information

SSL/TLS and Certificate Transparency

ISTA 488: Information Privacy with Applications

David Sidi (dsidi@email.arizona.edu)

Warm-up

- None!

Small mention of interesting things

- Modifying torrc to specify exit nodes https://www.torproject.org/docs/tor-manual.html.en
- More EC: send me gpg
- Assignment 2 correction
- Proposals

Hands on: sign my key

- Verify with government-issued identification
- `$ gpg --recv-key EEBA8245`
- `$ gpg --edit-key david@sidiprojects.us`

  (Prompt changes from '$' to 'gpg>')
- `gpg>sign`
- or

  `gpg>tsign`

Hands on: Locally sign a key

```
$ gpg --edit-key <KEY-ID>
gpg>lsign
```

- Question: why do this instead of sign?
- Question: why is there no way to “locally trust?”

```
gpg> tsign
Really sign all user IDs? (y/N) y

pub  4096R/EEBA8245  created: 2014-03-14  expires: 2018-03-14  usage: SC
                     trust: unknown       validity: unknown
 Primary key fingerprint: E622 43FC 0F47 135B 28F0  02C4 8496 9123 EEBA 8245

     David Sidi <david@sidiprojects.us>
     David Sidi <davidsidi@gmail.com>
     David Sidi <dsidi@email.arizona.edu>

This key is due to expire on 2018-03-14.
Please decide how far you trust this user to correctly verify other users' keys
(by looking at passports, checking fingerprints from different sources, etc.)

  1 = I trust marginally
  2 = I trust fully

Your selection? 2

Please enter the depth of this trust signature.
A depth greater than 1 allows the key you are signing to make
trust signatures on your behalf.

Your selection? 1

Please enter a domain to restrict this signature, or enter for none.

Your selection?

Are you sure that you want to sign this key with your
key "Nemo Outis <foo@mamber.net>" (93CDE742)

Really sign? (y/N) y

You need a passphrase to unlock the secret key for
user: "Nemo Outis <foo@mamber.net>"
4096-bit RSA key, ID 93CDE742, created 2017-11-14
```

Hands-on: Upload our public keys to a keyserver

```
$ gpg \
  --keyserver pool.sks.keyservers.net \
  --send-key <YOUR KEY ID>
$ gpg \
  --keyserver pool.sks.keyservers.net \
  --send-key EEBA8245
```

- Why does it make sense to use a pool, rather than a single server?

Hands-on: export your key, and publish it somewhere

- Twitter, Facebook, or your own u.arizona.edu/~janedoe website
- Which key ID should you rely on for people to check your key?
  - None, of, them (three links): Use the full 40 hex digit fingerprint
  - in ~/.gnupg/gpg.conf, include lines

    ```
    keyid-format 0xlong
    with-fingerprint
    ```
- Send me evidence that your key is posted for extra credit

Hands on: Generate a revocation certificate

```
$ gpg \
  --output revoke_compromised.asc \
  --gen-revoke '<KEY ID>'
```

Hands on: Key Discovery

- Key signing parties
- Big Lumber (http://biglumber.com/x/web)
  - Note! http site

Hands-on: Encrypt a message (PKC method)

```
$ echo “sekr3t stuff” > plaintext
$ gpg -e ./plaintext
```

Hands-on: Encrypt a message (symmetric method)

- `$ echo “sqeamish ossifrage” > plaintext2`
- `$ gpg --symmetric ./plaintext2`

Hands-on: Sign a message

- Exercise: Look up how to sign messages in gpg. Create a warrant canary, using this one as an example

Hands-on: Create a detached signature

- Imagine you want to distribute a text file and be sure everyone gets it unmodified
- One thing you can do is create a detached signature, which can be downloaded separately from the text file, and which can be used to verify integrity
- look it up in the man page

Hands-on: set up parcimonie

- Refreshing key information is very important to keep up with revocations
- gpg --refresh-keys happens by default using HKP, which is an unencrypted protocol, and without anonymity

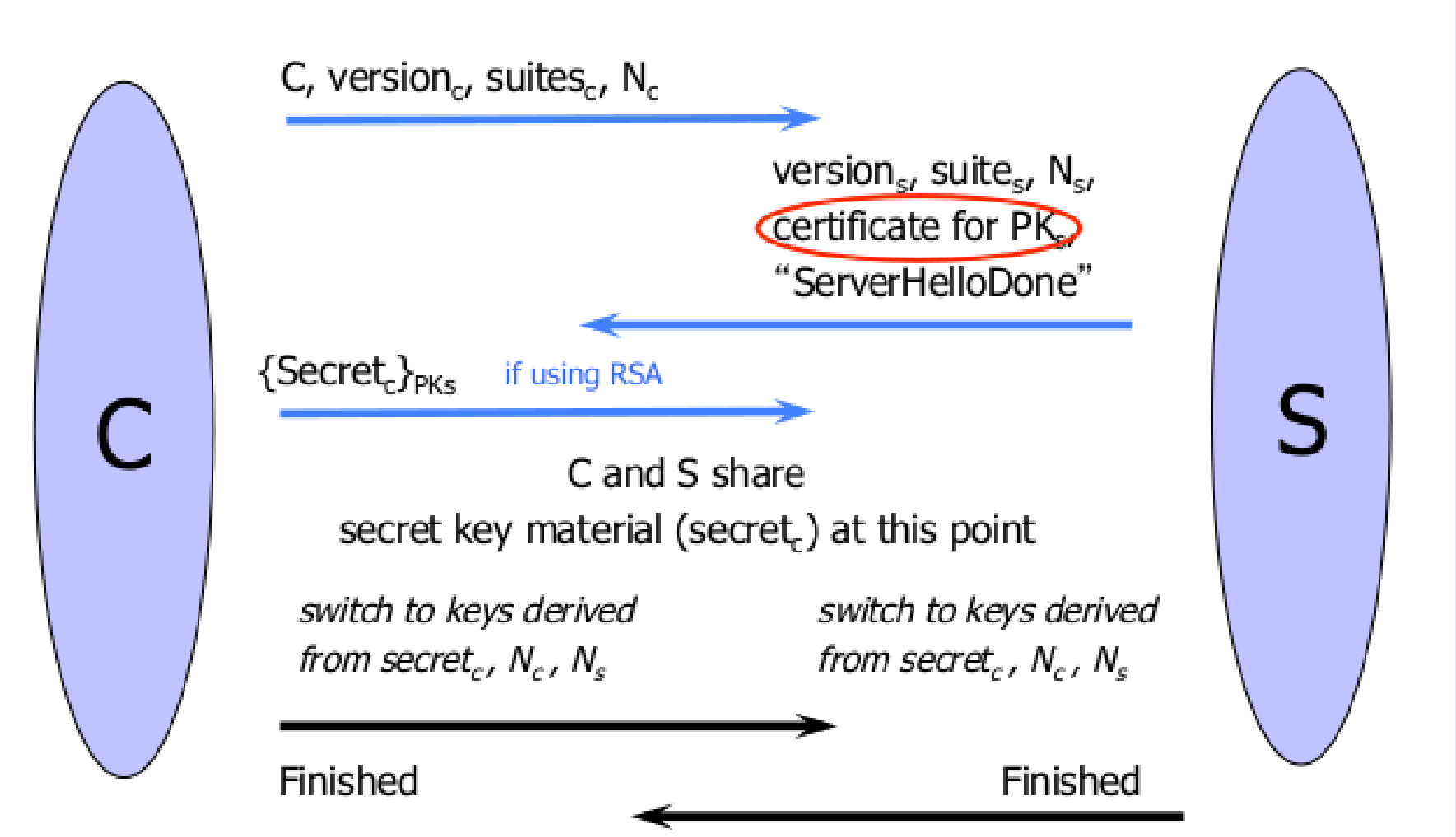
```
$ sudo apt-get install parcimonie

$ parcimonie --verbose
```

SSL/TLS and Certificate Transparency

What is SSL/TLS?

- Secure Sockets Layer and Transport Layer Security protocols
  - Both are for PKI, but they use different cryptosystems. TLS is more recent
  - Set up a secure channel first with asymmetric key, then transfer a secret for symmetric key encryption of the rest of communication
- SSL/TLS is very widely used for internet security
  - “zero configuration” for most
  - see: HTTP over TLS (i.e., HTTPS)

Handshake protocols

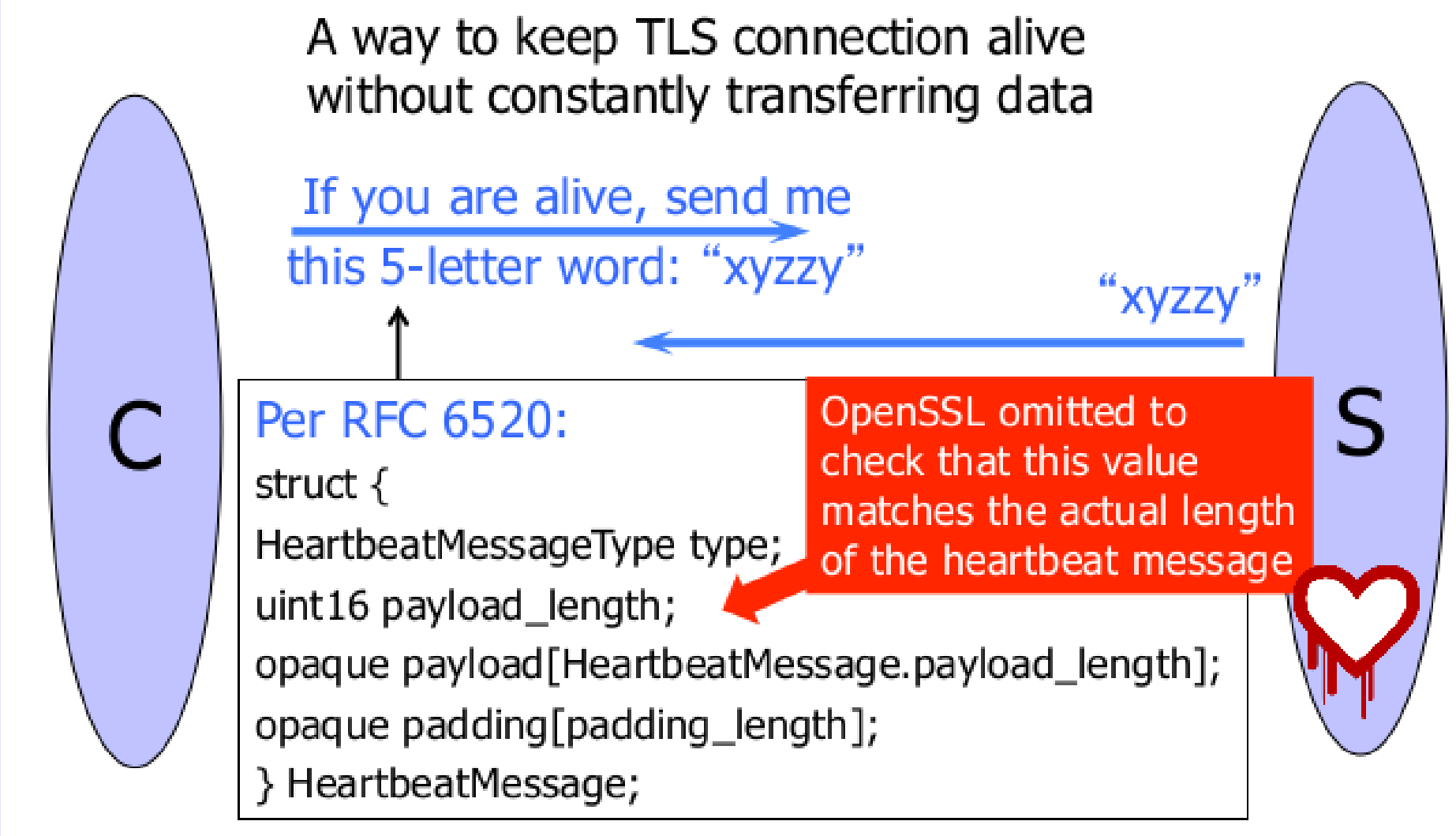


credit: Vitaly Shmatikov

TLS Records

+	Byte +0	Byte +1	Byte +2	Byte +3
Byte 0	Content type			
Bytes 1..4	Version		Length	
	(Major)	*(Minor)*	*(bits 15..8)*	*(bits 7..0)*
Bytes 5..(*m*-1)	Protocol message(s)			
Bytes *m*..(p-*1*)	MAC (optional)			
Bytes *p*..(q-*1*)	Padding (block ciphers only)			

heartbeat extension

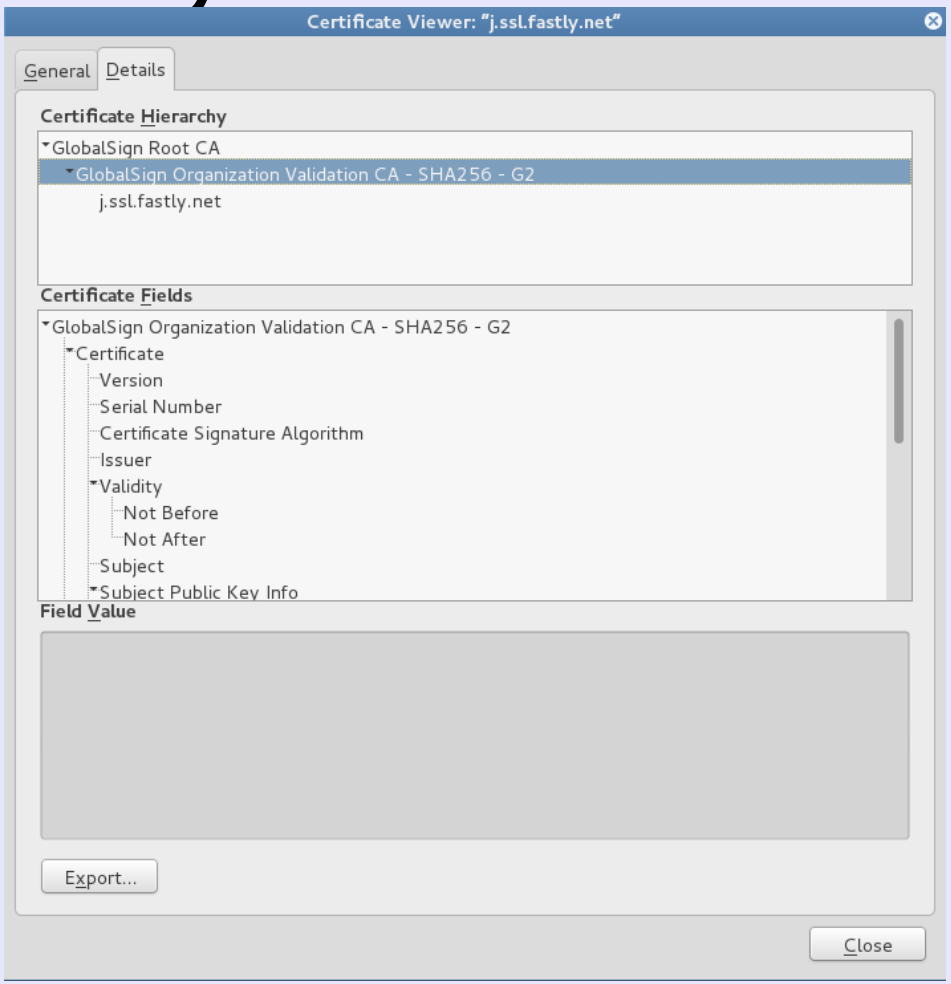


http://heartbleed.com/

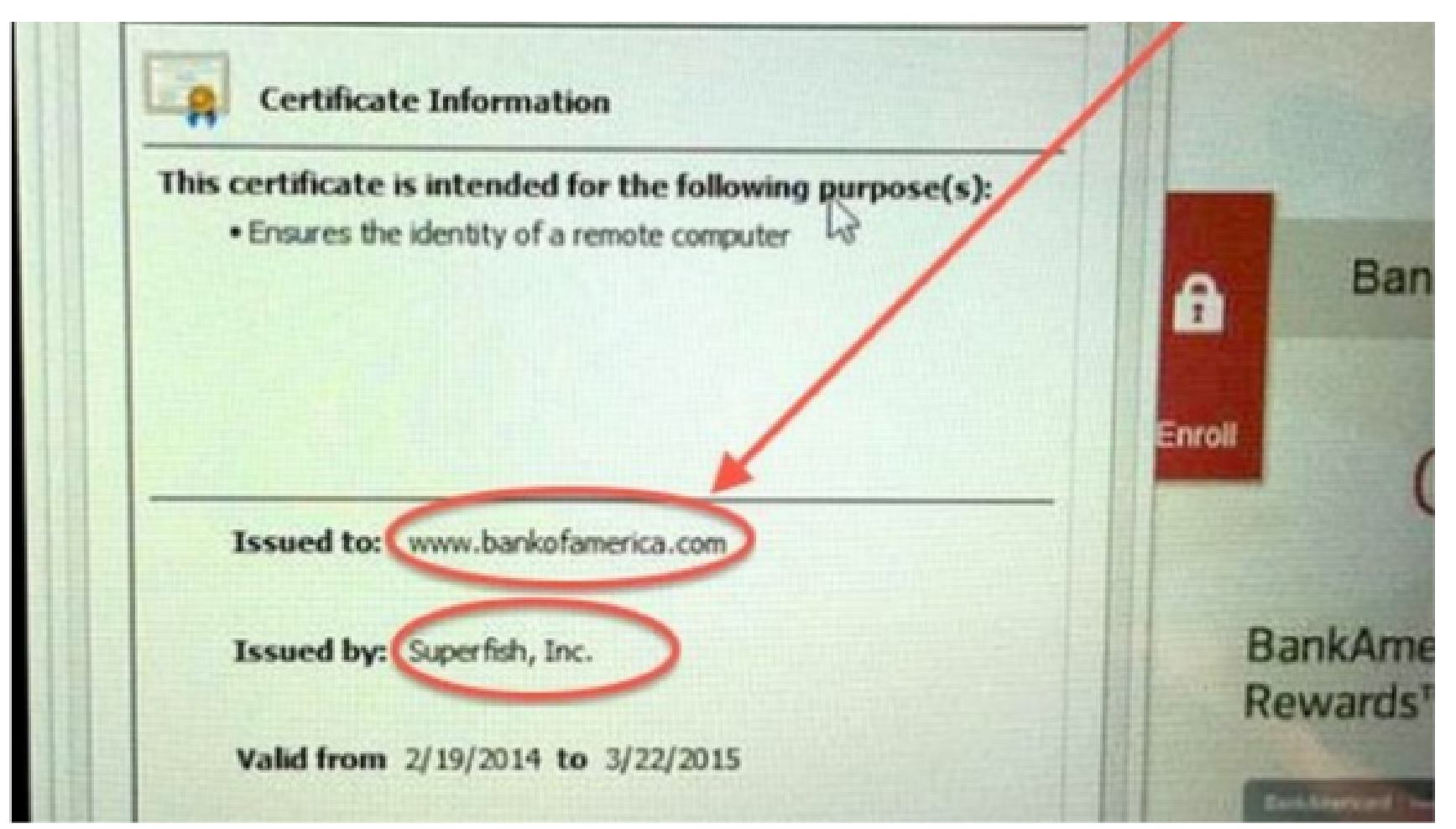
image credit: Vitaly Shmatikov

Certificate authorities form a hierarchy

- CA is a verifier of identity and domain ownership, to avoid problems arising from impersonation
- Root CAs sign certificates for Intermediate CAs, who sign for lower-level CAs: trust chain



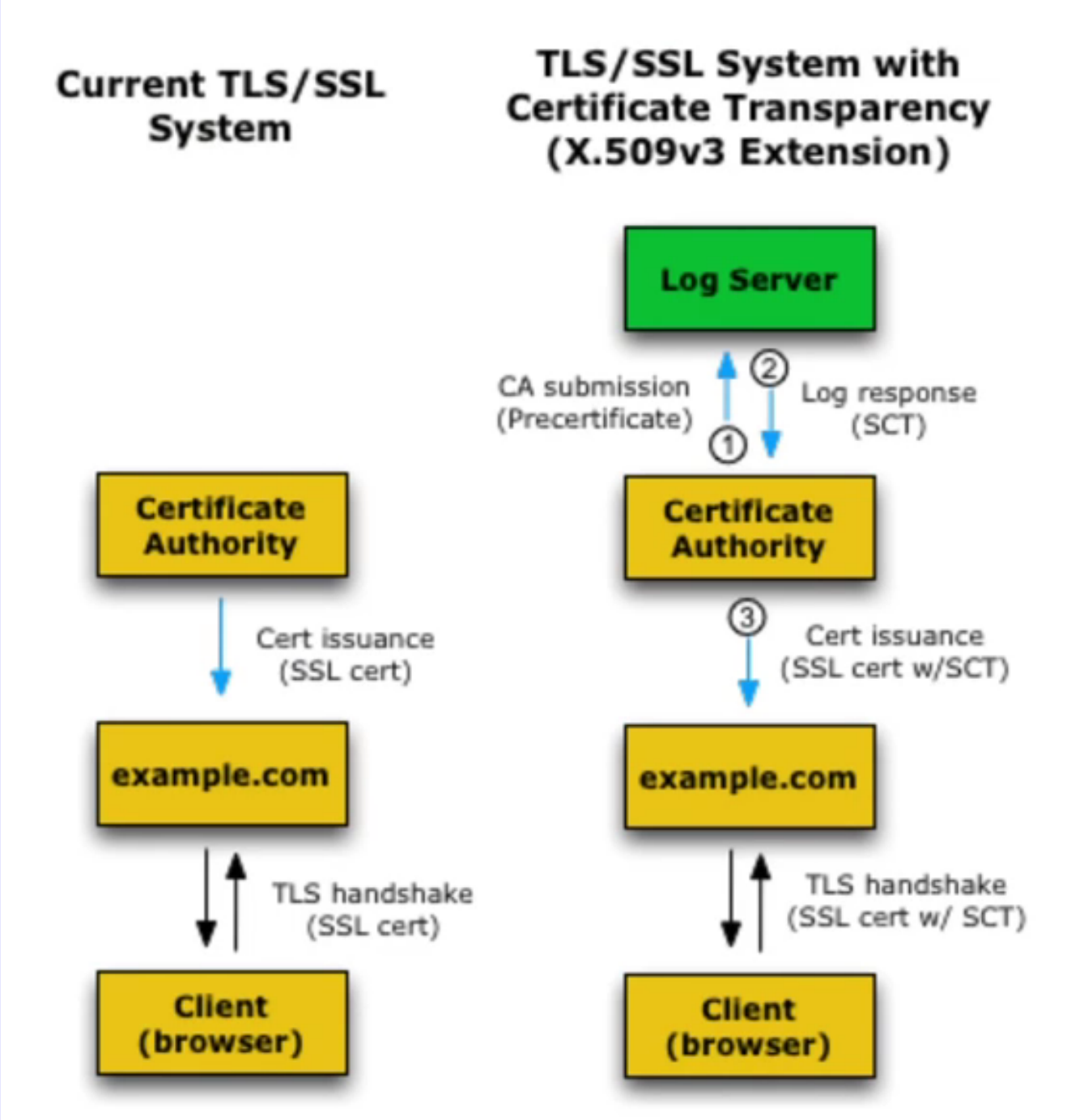
Superfish and the trouble with certs



Much more is wrong:
https://www.eff.org/deeplinks/2010/03/researchers-reveal-likelihood-governments-fake-ssl

Certificate Transparency

- Makes issuance of TLS/SSL certificates publicly auditable
  - cryptographically assured
  - append-only (no deletion, modification, or retroactive insertions)
  - public: log servers advertise their URL and public key
- Notice: **not** about whether the certificate is valid/revoked!
- Open source, anyone can run a log server
- Now *mandatory* for chrome, firefox


Current TLS/SSL System
Certificate Authority
Cert issuance (SSL cert)
example.com
TLS handshake (SSL cert)
Client (browser)
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
1
2
Log response (SCT)
Certificate Authority
3
Cert issuance (SSL cert w/SCT)
example.com
TLS handshake (SSL cert w/ SCT)
Client (browser)

Signed Certificate Timestamp

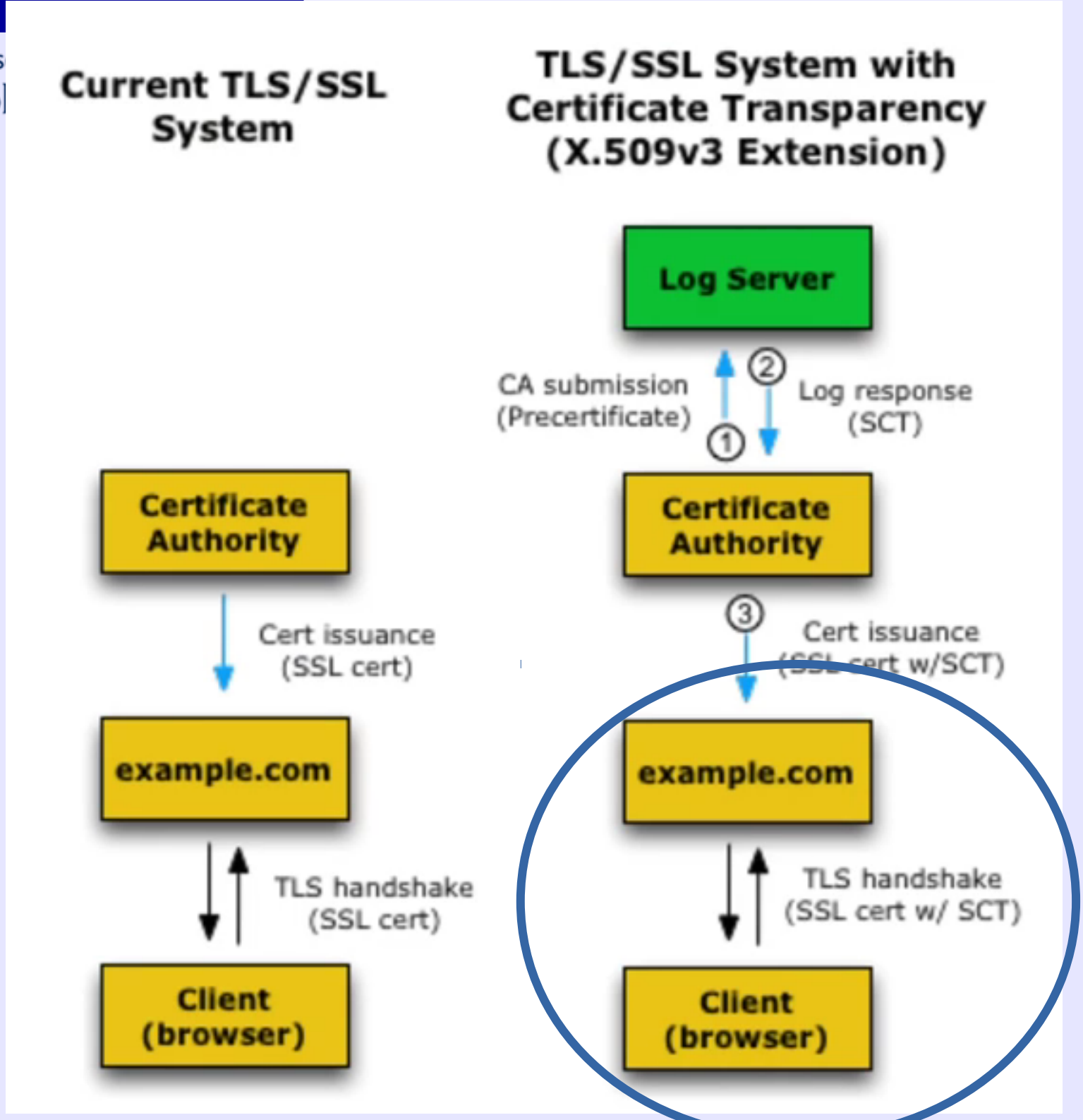
3.2. Structure of the Signed Certificate Timestamp

```
enum { certificate_timestamp(0), tree_hash(1), (255) }
  SignatureType;

enum { v1(0), (255) }
  Version;

  struct {
      opaque key_id[32];
  } LogID;

  opaque TBSCertificate<1..2^24-1>;

  struct {
    opaque issuer_key_hash[32];
    TBSCertificate tbs_certificate;
  } PreCert;

  opaque CtExtensions<0..2^16-1>;
```

- subject of the certificate's name
- issuer's name

- public key of the subject
- validity period
- version number and a serial number

RFC 5280 https://tools.ietf.org/html/rfc5280#section-4.1.2


Current TLS/SSL System
Certificate Authority
Cert issuance (SSL cert)
example.com
TLS handshake (SSL cert)
Client (browser)
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
2
Log response (SCT)
1
Certificate Authority
3
Cert issuance (SSL cert w/SCT)
example.com
TLS handshake (SSL cert w/ SCT)
Client (browser)

Verification of an SCT is part of the TLS handshake

- An extension to the Online Certificate Status Protocol (OCSP) Stapling TLS protocol

```
$openssl s_client -connect sidiprojects.us:443 \
-tls1 -tlsextdebug -status
```

```
======================================
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: C = US, O = Let's Encrypt, CN = Let's Encrypt Authority X3
    Produced At: Nov 18 19:20:00 2017 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: 7EE66AE7729AB3FCF8A220646C16A12D6071085D
      Issuer Key Hash: A84A6A63047DDDBAE6D139B7A64565EFF3A8ECA1
      Serial Number: 0302CD2CAD56657A5F8E57DA8E5F0C1430A1
    Cert Status: good
    This Update: Nov 18 19:00:00 2017 GMT
    Next Update: Nov 25 19:00:00 2017 GMT
```

OCSP stapling is better than the alternatives

- There are other ways for the client to verify the SCT

3.2. Structure of the Signed Certificate Timestamp

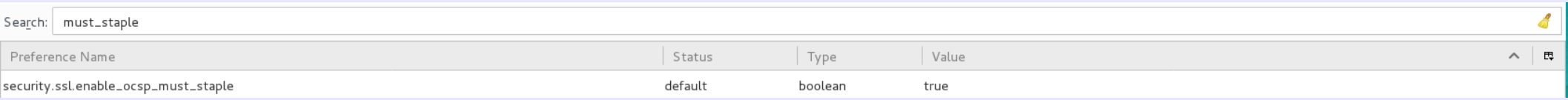
```
enum { certificate_timestamp(0), tree_hash(1), (255) }
  SignatureType;

enum { v1(0), (255) }
  Version;

  struct {
      opaque key_id[32];
  } LogID;

  opaque TBSCertificate<1..2^24-1>;

  struct {
    opaque issuer_key_hash[32];
    TBSCertificate tbs_certificate;
  } PreCert;

  opaque CtExtensions<0..2^16-1>;
```

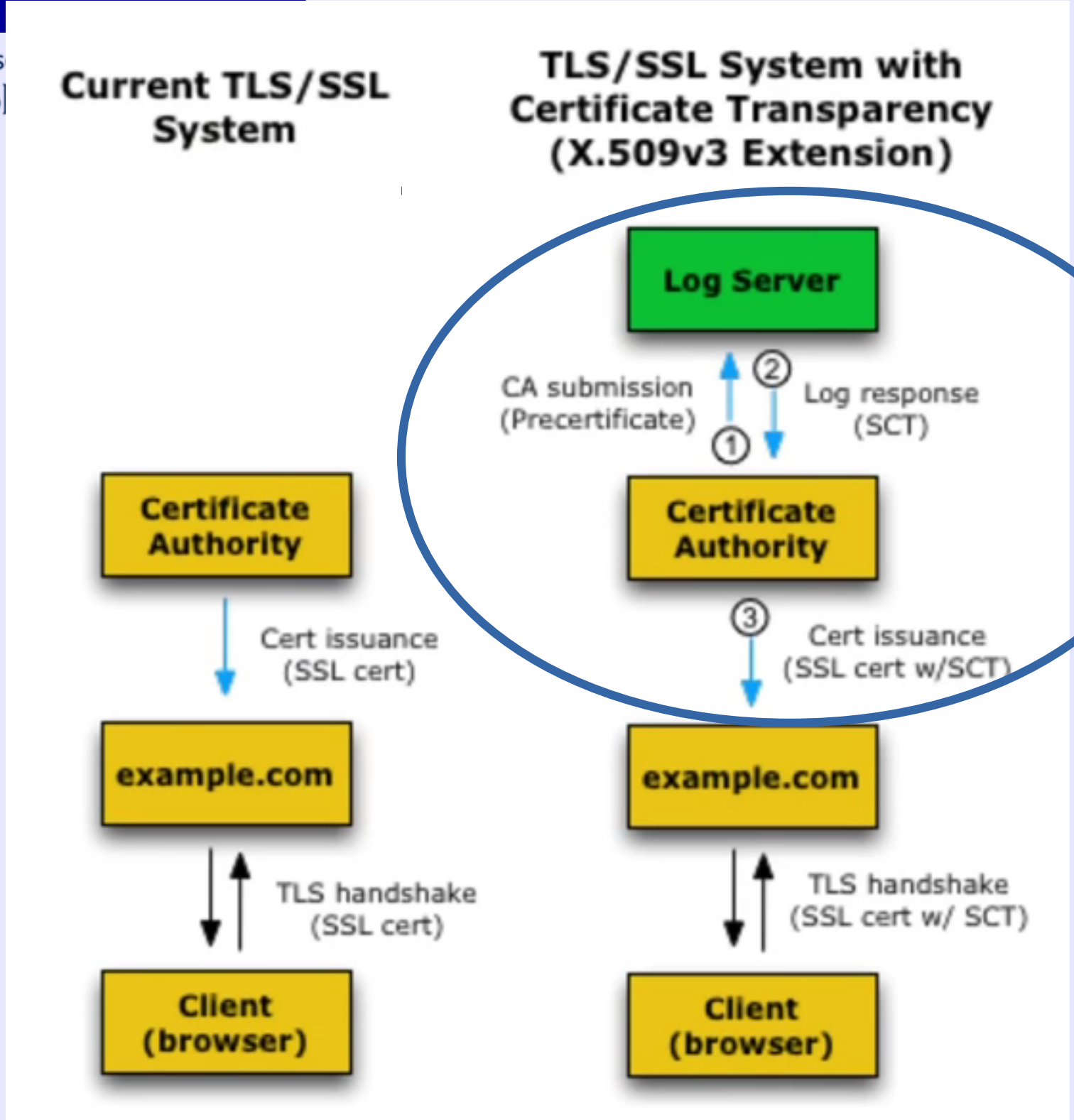
- subject of the certificate's name
- issuer's name

- public key of the subject
- validity period
- version number and a serial number
- **SignedCertificateTimestampList (as extension)**

OCSP stapling is better than the alternatives

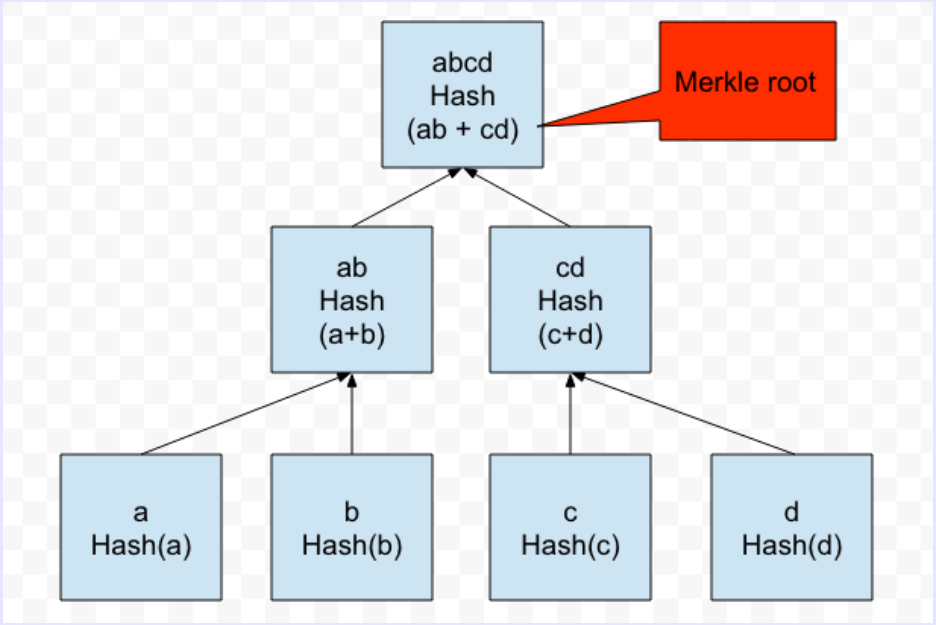
- There are other ways for the client to verify the SCT in the TLS handshake (x509v3 certificate extensions, or TLS extensions)
- OCSP stapling does not require going out to the CA
  - the OCSP request, signed by the CA, is combined with the certificate and sent to the client
  - SCT can be included as part of this stapling
- Why might contacting the CA be a negative thing?




Current TLS/SSL System
TLS/SSL System with Certificate Transparency (X.509v3 Extension)
Log Server
CA submission (Precertificate)
2
Log response (SCT)
1
Certificate Authority
Certificate Authority
3
Cert issuance (SSL cert)
Cert issuance (SSL cert w/SCT)
example.com
example.com
TLS handshake (SSL cert)
TLS handshake (SSL cert w/ SCT)
Client (browser)
Client (browser)

Log servers use Merkle Hash Trees to keep track of the certificates

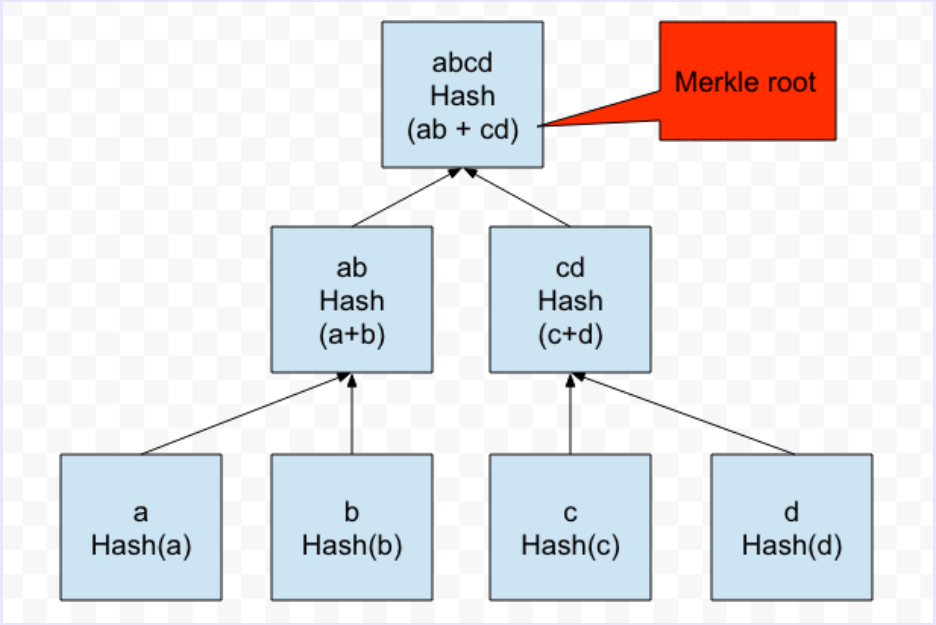
- Binary tree
- Calculated from the leaves: combine children's hashes to get the parent hash
- Can check integrity of a whole lot of hashes by checking one hash!
- All changes are auditable



credit

Log servers use Merkle Hash Trees to keep track of the certificates

- Can catch CA's that are adding and removing illicit certificates
- Can catch cheating log servers
- Not enough to just calculate the root value to audit the log once new hashes are added. Why not?



credit

Walk through: auditing a log addition

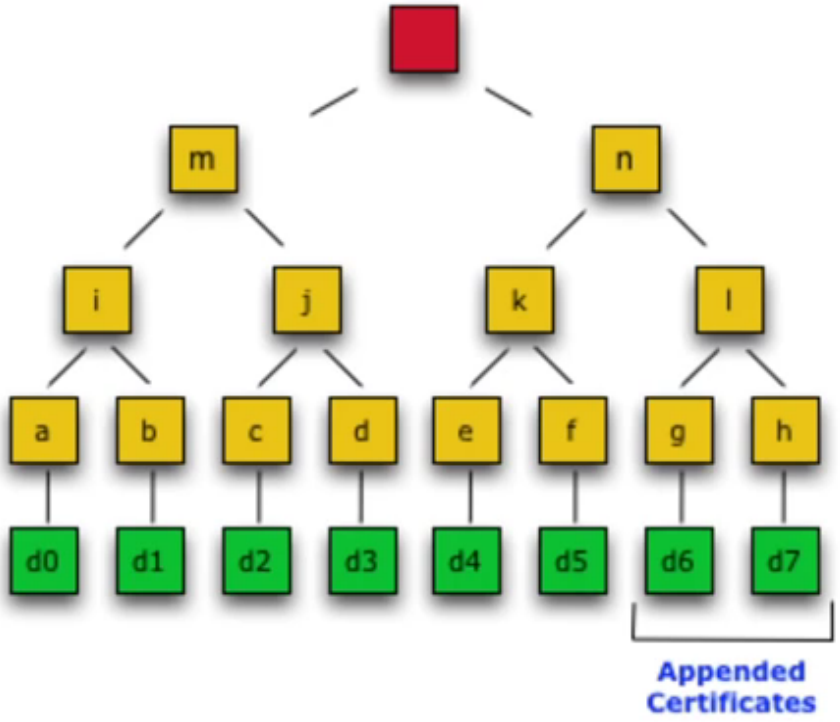


Figure 3

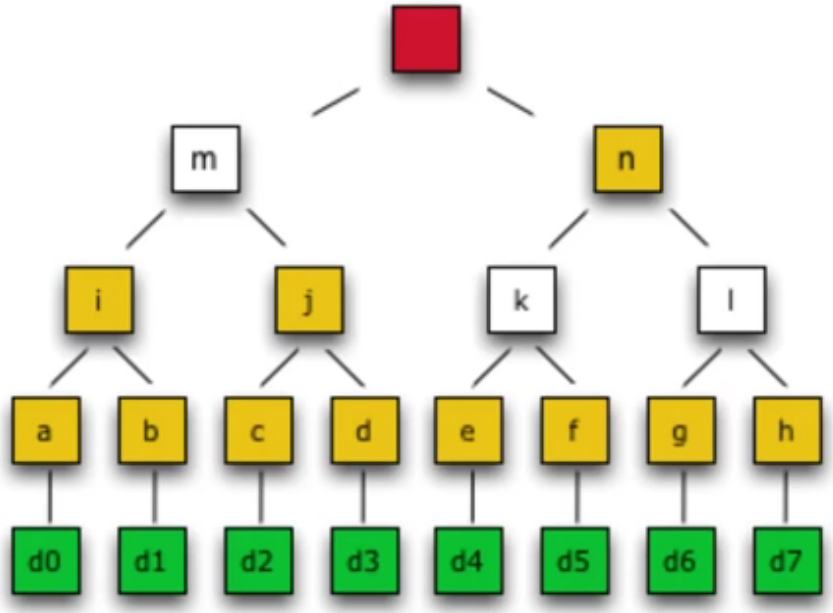
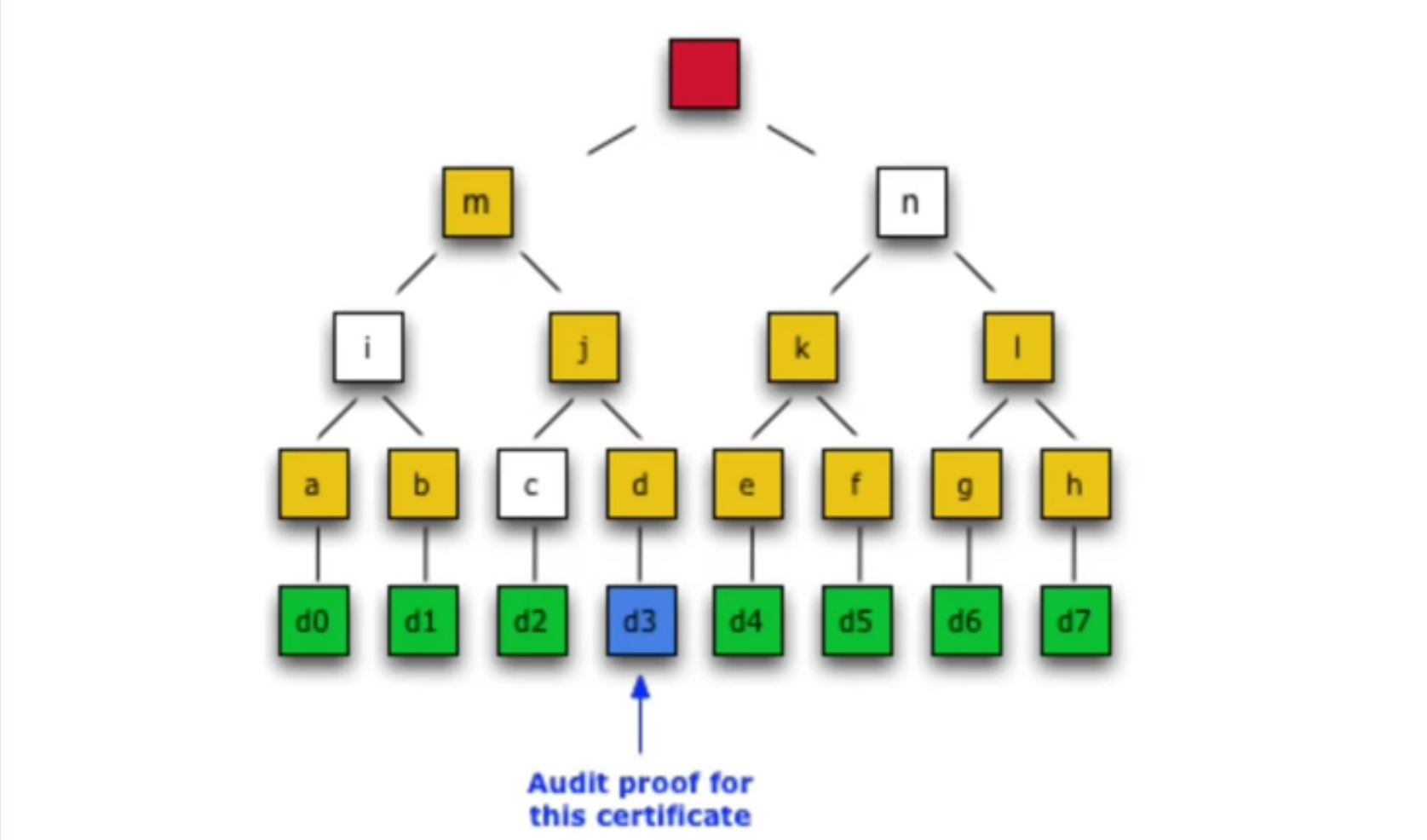


Figure 4

Walk through: Auditing for presence of a particular certificate



Log servers are still centralized in practice

- In theory, anyone can run a log
- In practice, there are only a few
  - Digicert: the first
  - Google: their idea; they run the big ones

```
$curl ct.googleapis.com/icarus/ct/v1/get-sth
{"tree_size":148531007,
"timestamp":1511196824947,
"sha256_root_hash":"bRmJZDeJZIs/WTOYZ3pA+MyJuOEZ9m+XGZIRU9fnViI=
",
"tree_head_signature":"BAMASDBGAiEAk+md3GDvKIPyuQ27UnLdDhKoVB5hn
zVDA8ZX1Dkx/JgCIQCDmYMAi6oqpAXk+LV/vIKwfrhyaCNrXl7N37moFv/BfA=="
}
```

- Use crt.sh to search manually from the browser. Certspotter can help you monitor your domains (https://sslmate.com/certspotter/)

Other ways to fix TLS

- Using GPG, with monkeysphere
  - http://web.monkeysphere.info/
- Flexible trust model of WoT used for PKI
- Problem: goes out to the keyserver for failing requests