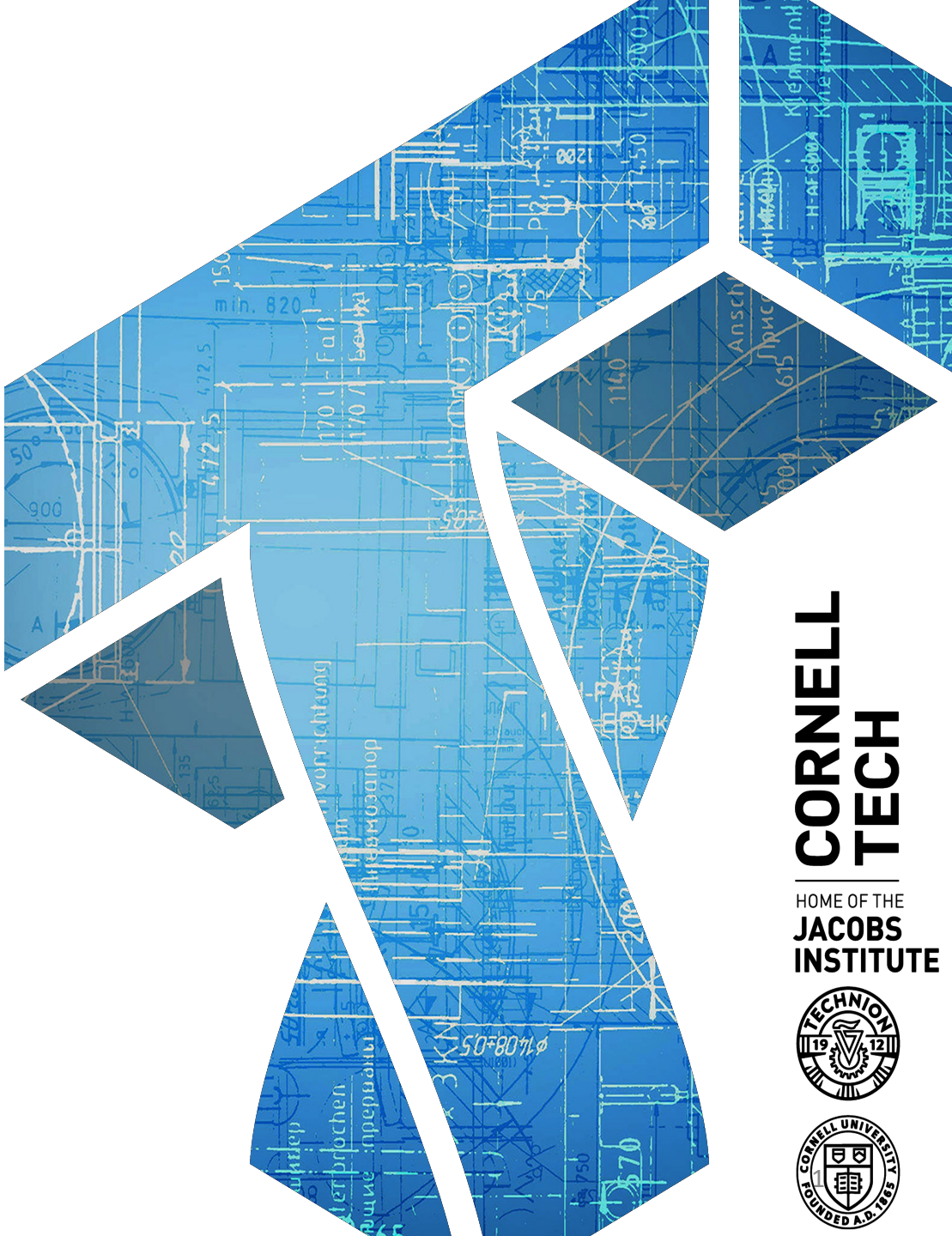
CS 5830
Cryptography



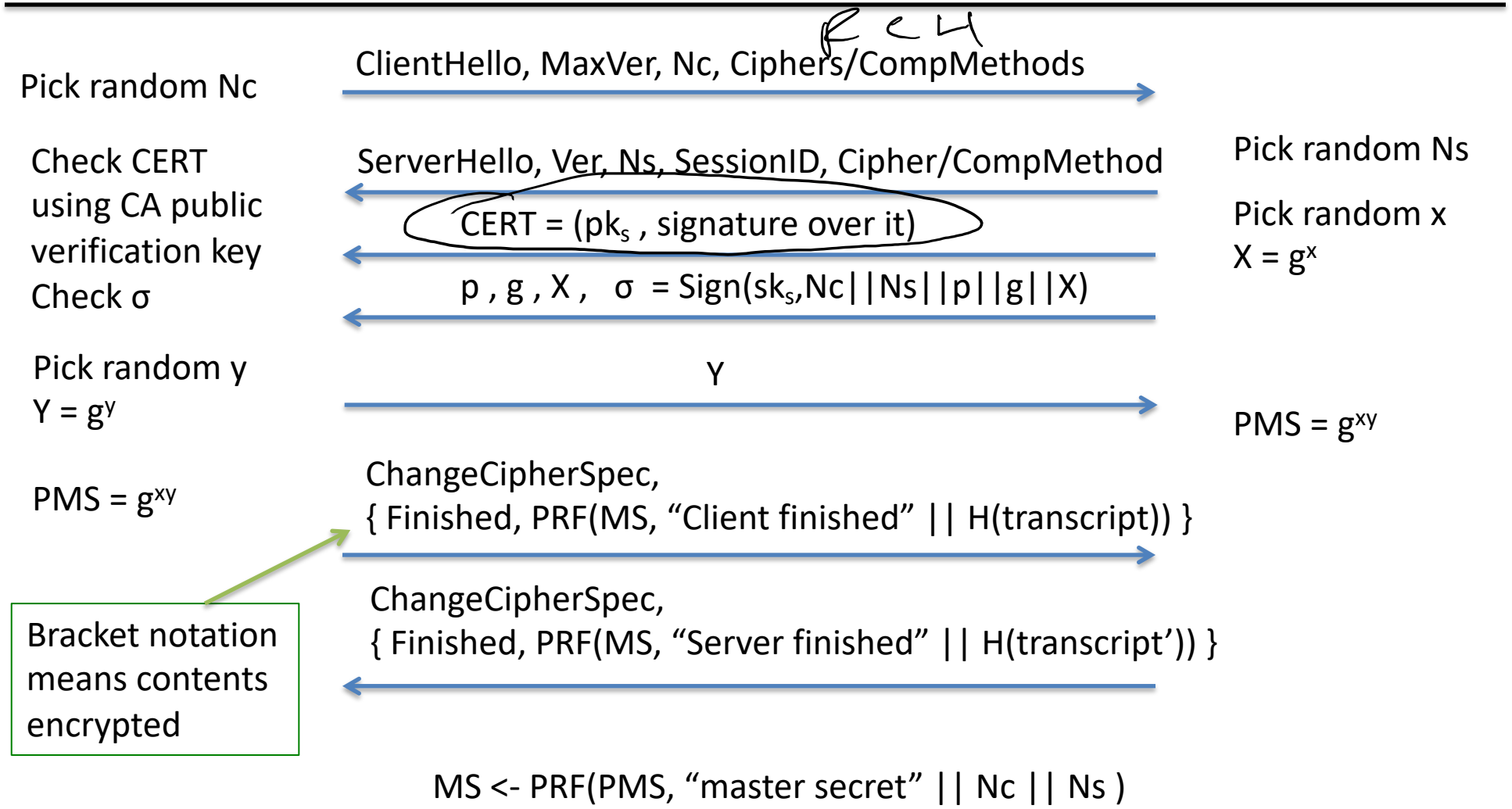
CORNELL TECH

HOME OF THE JACOBS INSTITUTE

TLS handshake for Diffie-Hellman Key Exchange

Client

Server



Apple iOS <7.0.16 signature verification code

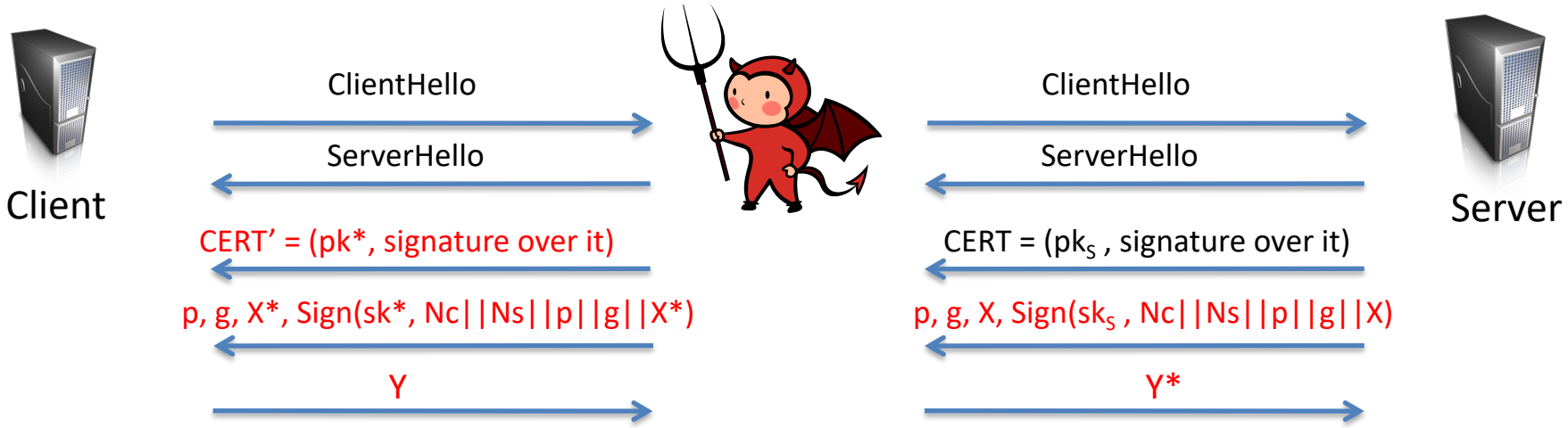
```
    if ((err = ReadyHash(&SSLHashSHA1, &hashCtx)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &clientRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &serverRandom)) != 0)
        goto fail;
    if ((err = SSLHashSHA1.update(&hashCtx, &signedParams)) != 0)
        goto fail;
        goto fail;
    if ((err = SSLHashSHA1.final(&hashCtx, &hashOut)) != 0)
        goto fail;

        err = sslRawVerify(ctx,
                           ctx->peerPubKey,
                           dataToSign,                          /* plaintext */
                           dataToSignLen,                  /* plaintext length */
                           signature,
                           signatureLen);
        if(err) {
                sslErrorLog("SSLDecodeSignedServerKeyExchange: sslRawVerify "
                    "returned %d\n", (int)err);
                goto fail;
        }

fail:
    SSLFreeBuffer(&signedHashes);
    SSLFreeBuffer(&hashCtx);
    return err;
```

Meddler-in-the-middle attacks

Suppose authentication vulnerability:
CERT can be forged, Client doesn't check CERT, etc.



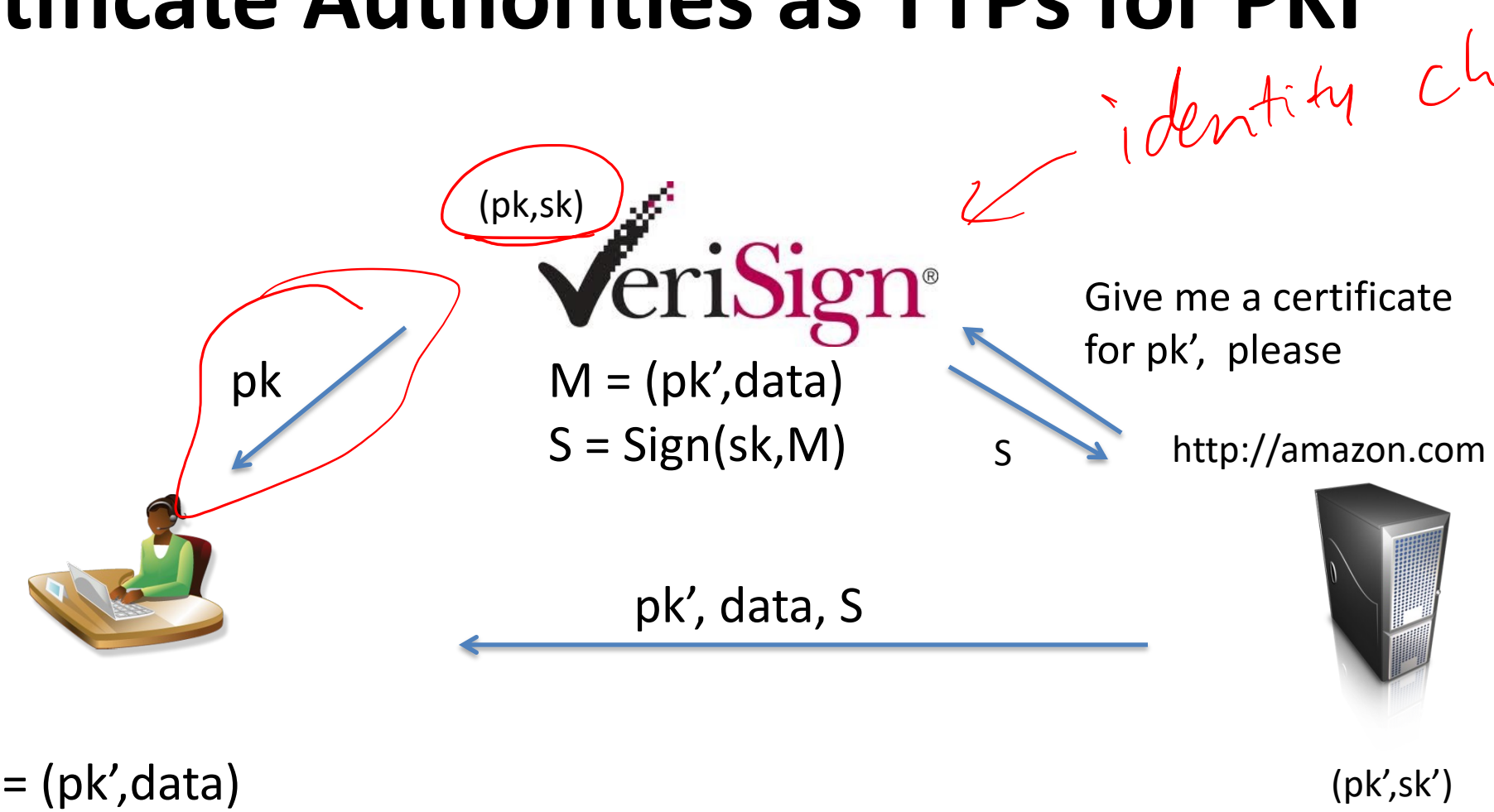
Attacker can choose X*, Y*, so it knows discrete logs
Completes handshake on both sides
Client thinks its talking to Server
All communications decrypted by adversary, re-encrypted and forwarded to server

Public-key infrastructure (PKI)

- Trusted third parties (TTPs) that solve that cryptographically bind identity to public key

- Web identity ≈ domain name (cornell.edu)

Certificate Authorities as TTPs for PKI



M = (pk',data)
If Ver(pk,M,S) then
trust pk'

This prevents MitM attacks

```
Certificate:
   Data:
       Version: 1 (0x0)
       Serial Number: 7829 (0x1e95)
       Signature Algorithm: md5WithRSAEncryption
       Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
               OU=Certification Services Division,
               CN=Thawte Server CA/emailAddress=server-certs@thawte.com
       Validity
           Not Before: Jul  9 16:04:02 1998 GMT
           Not After : Jul  9 16:04:02 1999 GMT
       Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
                OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
       Subject Public Key Info:
           Public Key Algorithm: rsaEncryption
           RSA Public Key: (1024 bit)
               Modulus (1024 bit):
                   00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                   33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                   66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                   70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                   16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                   c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                   8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                   d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                   e8:35:1c:9e:27:52:7e:41:8f
               Exponent: 65537 (0x10001)
   Signature Algorithm: md5WithRSAEncryption
       93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
       92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
       ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
       d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
       0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
       5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
       8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
       68:9f
```

cmsx.cs.cornell.edu | InCommon ECC Server CA | USERTrust ECC Certification Authority

Subject Name

Country	US
State/Province	New York
Organization	Cornell University
Common Name	cmsx.cs.cornell.edu

Issuer Name

Country	US
State/Province	MI
Locality	Ann Arbor
Organization	Internet2
Organizational Unit	InCommon
Common Name	InCommon ECC Server CA

Validity

Not Before	Tue, 06 Sep 2022 00:00:00 GMT
Not After	Wed, 06 Sep 2023 23:59:59 GMT

Subject Alt Names

DNS Name	cmsx.cs.cornell.edu

Public Key Info

Algorithm	RSA
Key Size	2048
Exponent	65537
Modulus	C1:A4:6D:B5:68:E9:5A:38:BF:01:E7:93:72:9A:27:2D:07:CC:54:2F:0D:6F:E1...

Basic Constraints

Certificate Authority	No

Key Usages

Purposes	Digital Signature

Extended Key Usages

Purposes	Server Authentication, Client Authentication

Subject Key ID

Key ID	C5:8F:05:72:A3:43:2D:9B:7D:D7:DB:CE:E6:E0:FF:AC:80:31:3A:3D

Authority Key ID

Key ID	E4:B7:CF:CB:0A:94:74:A7:9C:AD:A8:12:04:3A:D0:29:5D:2E:FC:EE

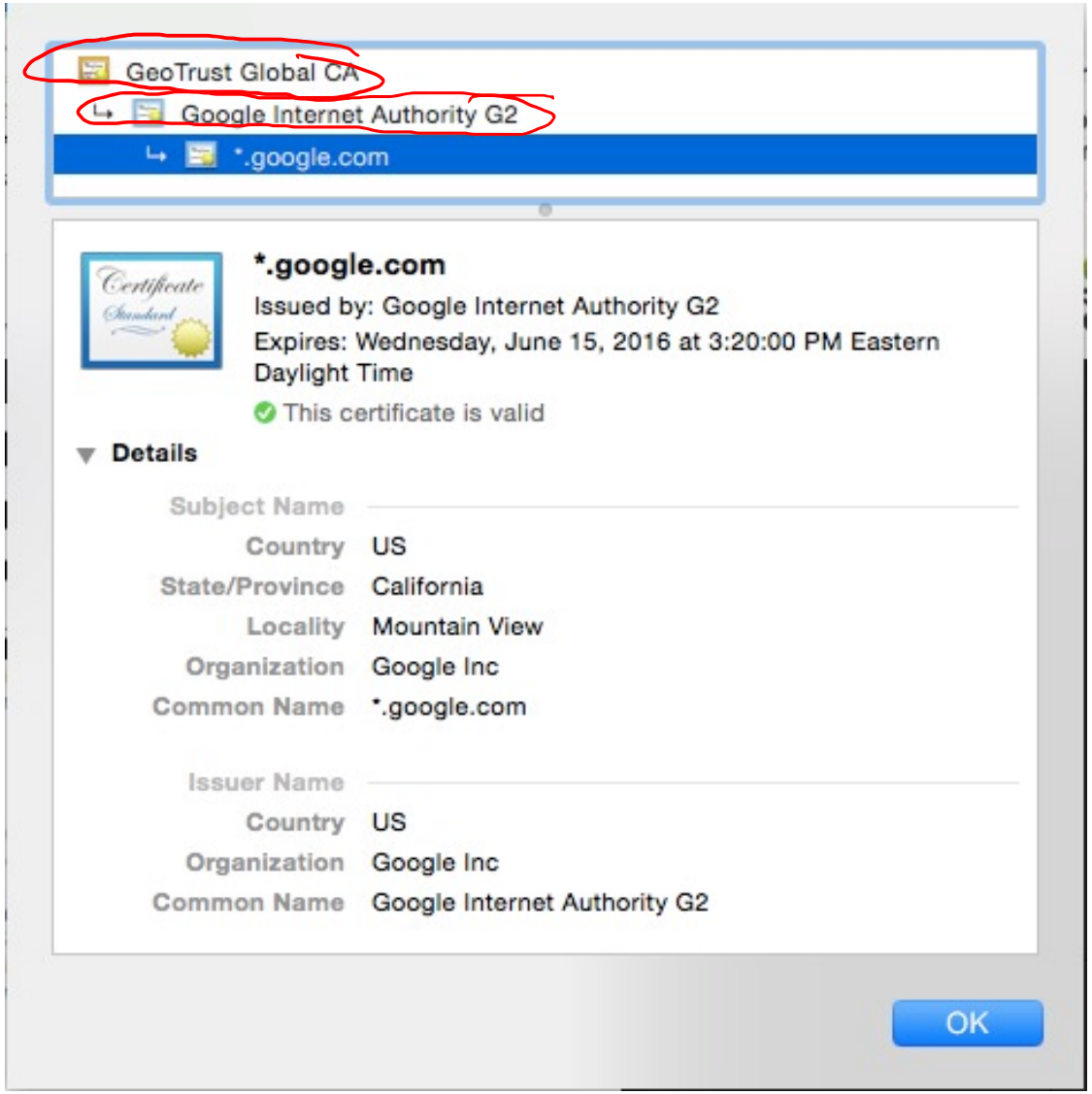
CRL Endpoints

Distribution Point	http://crl.incommon-ecc.org/InCommonECCServerCA.crl

Authority Info (AIA)

Location	http://cert.incommon-ecc.org/InCommonECCServerCA.crt
Method	CA Issuers
Location	http://ocsp.incommon-ecc.org
Method	Online Certificate Status Protocol (OCSP)

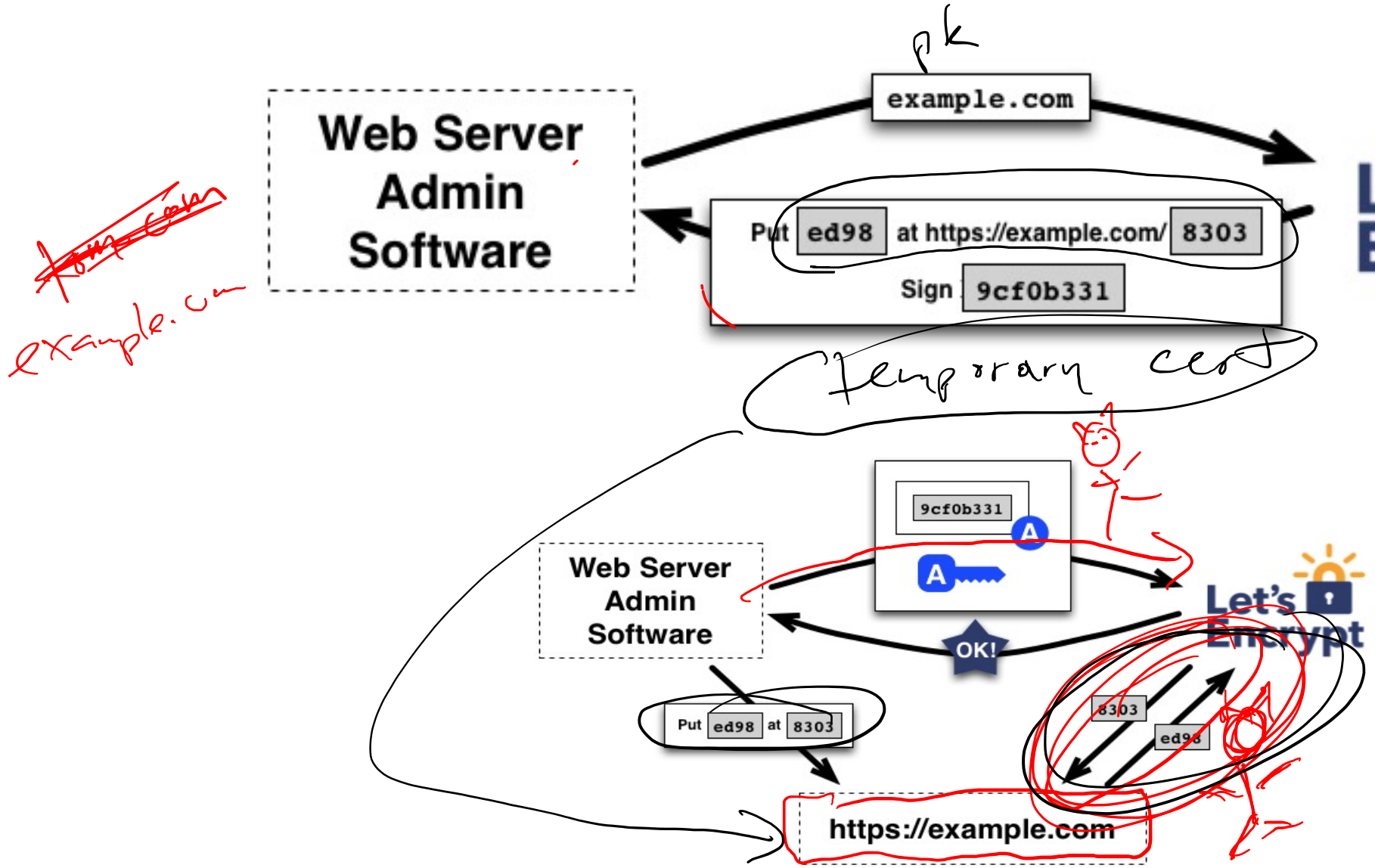
Certificate chains



Identity checks?

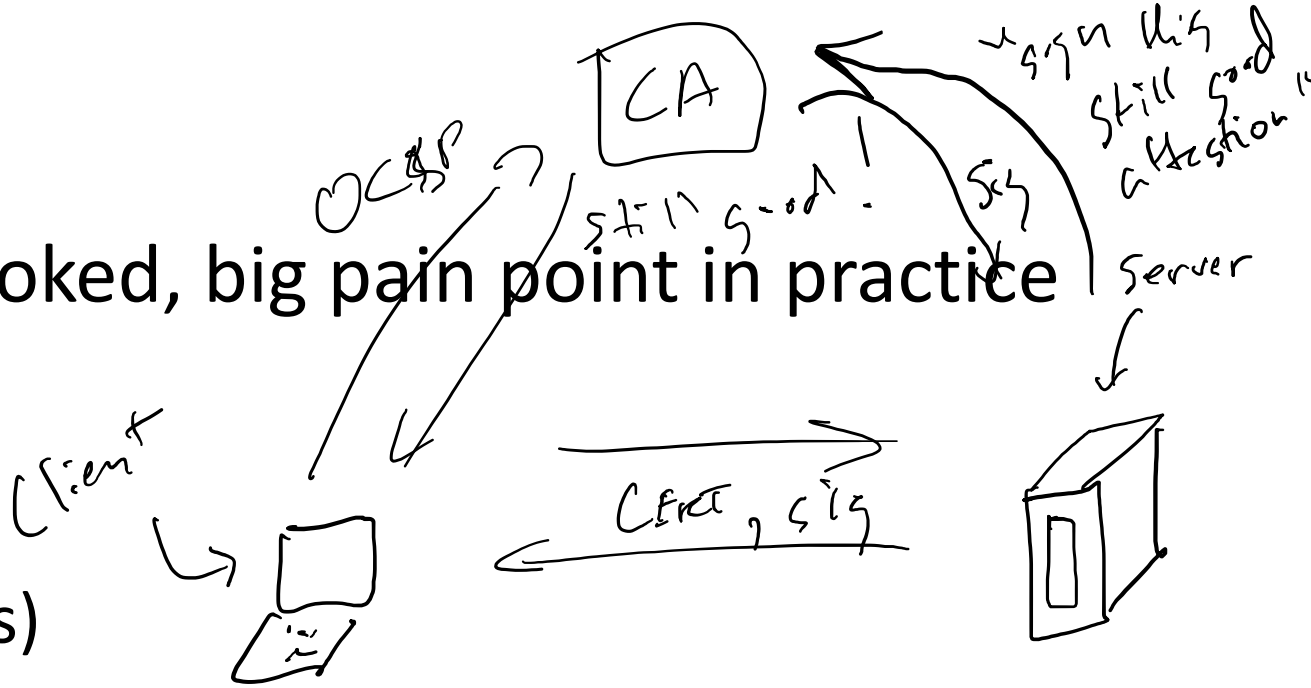
- CA's must check that requestor of cert is who they say they are
- Domain validated
  - Prove ownership of domain
- Extended validation
  - Establish legal identity of requestor
  - Physical presence of website owner
  - Confirm ownership of domain
  - Etc.

Free CAs


example.com
Web Server Admin Software
Put ed98 at https://example.com/ 8303
Sign 9cf0b331
Let's Encrypt
9cf0b331
A
A
OK!
Web Server Admin Software
Put ed98 at 8303
8303
ed98
https://example.com

Revocation

- Certificates must often be revoked, big pain point in practice
  - Short expirations
  - CRLs (Certificate revocation lists)
  - OCSP (online certificate status protocol)
    - Client queries CA to check on validity of cert
      - privacy concerns, performance / scalability issues
    - Stapling: server periodically gets fresh, time-stamped OCSP signature from CA. Sends to clients along with certificate



The Web PKI Ecosystem

- http://conferences.sigcomm.org/imc/2013/papers/imc257-durumericAemb.pdf

- ~1800 CAs that can sign *any* domain controlled by 683 organizations

- DigiNotar compromise, GoDaddy bug, Turktrust incident, …

Buggy Domain Validation Forces GoDaddy To Revoke SSL Certificates (threatpost.com)

33

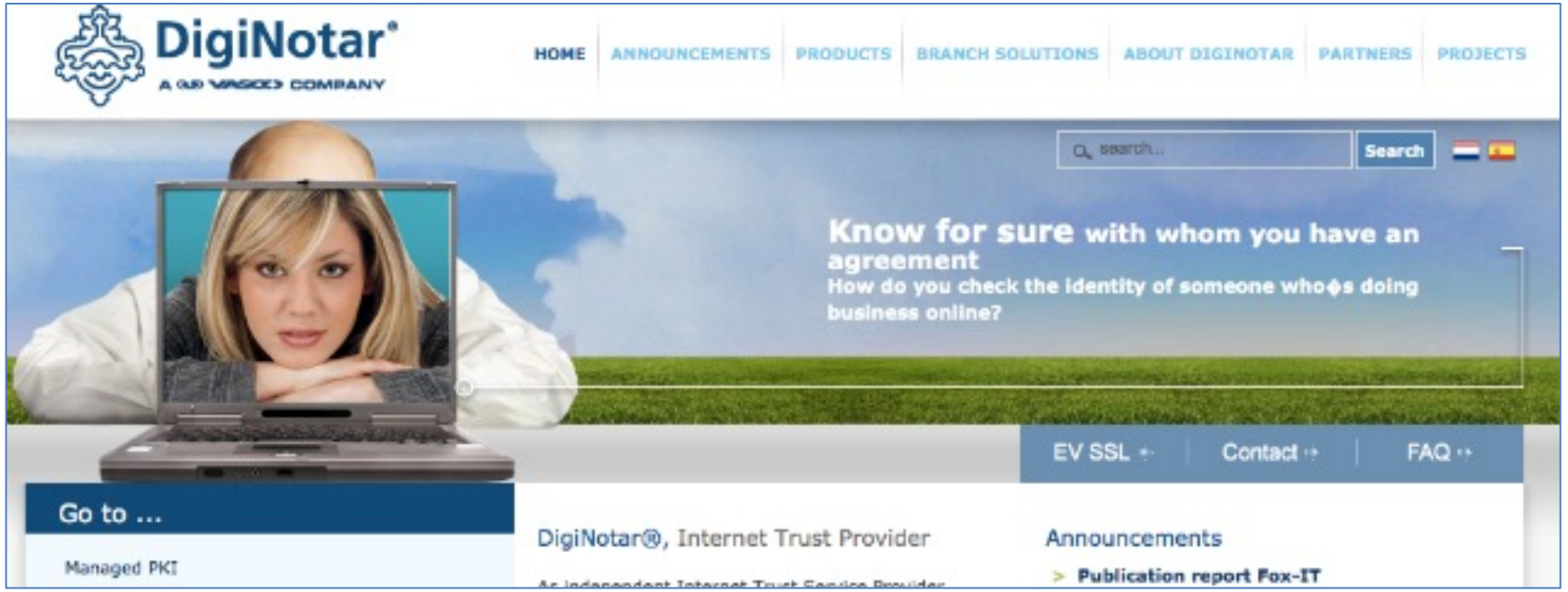
Posted by **BeauHD** on Wednesday January 11, 2017 @09:25PM from the time-to-take-action dept.

msm1267 quotes a report from Threatpost:

> GoDaddy has revoked, and begun the process of re-issuing, new SSL certificates for more than 6,000 customers after a bug was discovered in the registrar's domain validation process. The bug was introduced July 29 and impacted fewer than two percent of the certificates GoDaddy issued from that date through yesterday, said vice president and general manager of security products Wayne Thayer. "GoDaddy inadvertently introduced the bug during a routine code change intended to improve our certificate issuance process," Thayer said in a statement. "The bug caused the domain validation process to fail in certain circumstances." GoDaddy said it was not aware of any compromises related to the bug. The issue did expose sites running SSL certs from GoDaddy to spoofing where a hacker could gain access to certificates and pose as a legitimate site in order to spread malware or steal personal information such as banking credentials. GoDaddy has already submitted new certificate requests for affected customers. Customers will need to take action and log in to their accounts and initiate the certificate process in the SSL Panel, Thayer said.

Turktrust incident

- Turktrust is a CA used by Turkish government
- December, 2012 detected an erroneous CERT for *.google.com signed by Turktrust
  - Gives whoever owned private key ability to MITM most Google websites



Today, Microsoft issued a Security Advisory warning that fraudulent digital certificates were issued by the Comodo Certificate Authority. This could allow malicious spoofing of high profile websites, including Google, Yahoo! and Windows Live.

https://nakedsecurity.sophos.com/2011/03/24/fraudulent-certificates-issued-by-comodo-is-it-time-to-rethink-who-we-trust/

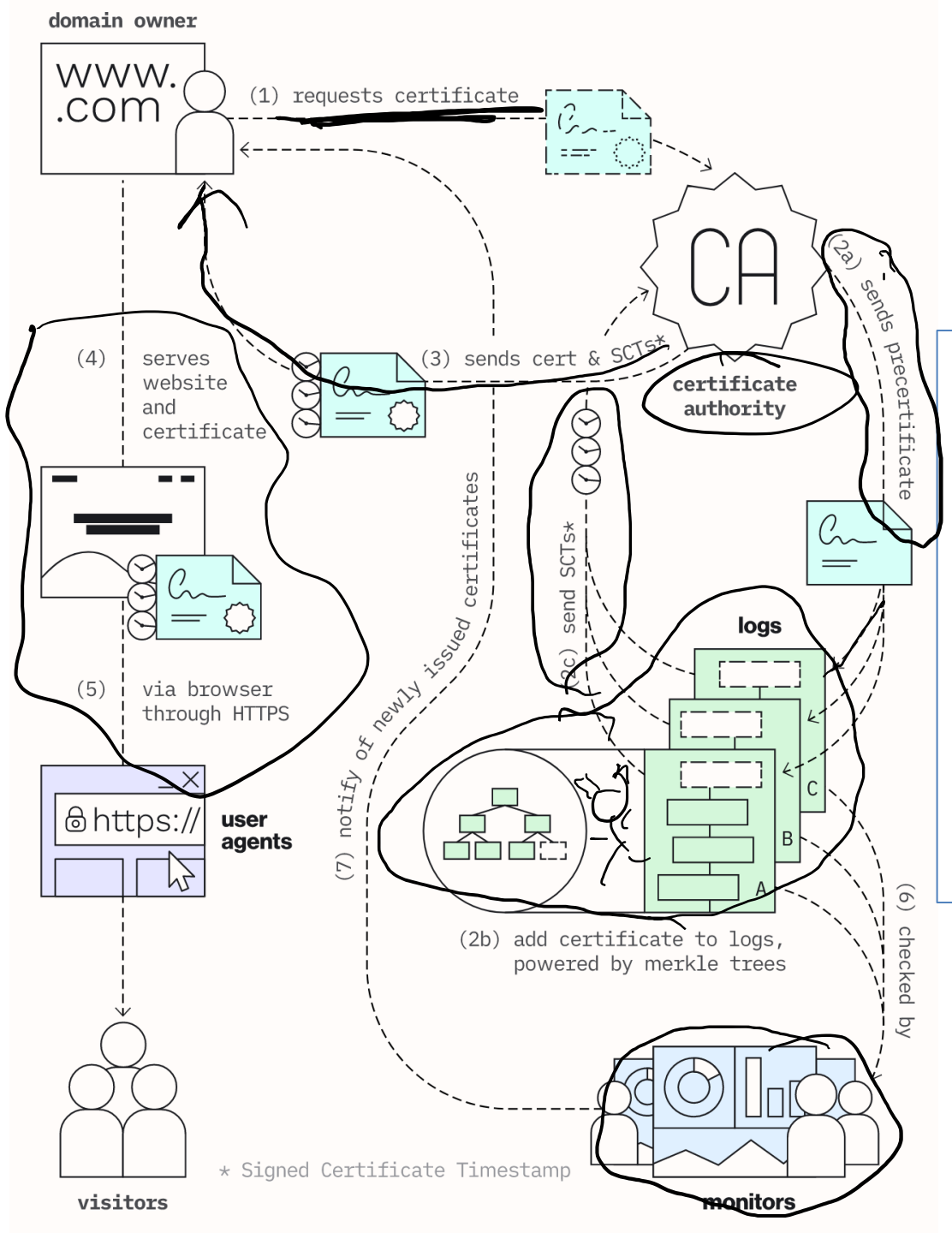
https://technet.microsoft.com/library/security/2524375

Certificate transparency

https://certificate.transparency.dev/

- Force CAs to log the certificates they sign in a public tamper-evident register
  - ~~Experimental~~ IETF standard

- Effort originally led by Google
  - Chrome requires it for “extra validation” certs
  - DigiCert, Couldflare, has implemented



Embedded SCTs

Log ID	AD:F7:BE:FA:7C:FF:10:C8:8B:9D:3D:9C:1E:3E:18:6A:B4:67:29:5D:CF:B1:0...
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Tue, 06 Sep 2022 07:01:45 GMT
Log ID	7A:32:8C:54:D8:B7:2D:B6:20:EA:38:E0:52:1E:E9:84:16:70:32:13:85:4D:3B...
Name	Cloudflare “Nimbus2023”
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Tue, 06 Sep 2022 07:01:45 GMT
Log ID	E8:3E:D0:DA:3E:F5:06:35:32:E7:57:28:BC:89:6B:C9:03:D3:CB:D1:11:6B:E...
Signature Algorithm	SHA-256 ECDSA
Version	1
Timestamp	Tue, 06 Sep 2022 07:01:45 GMT

Certificate/public-key pinning

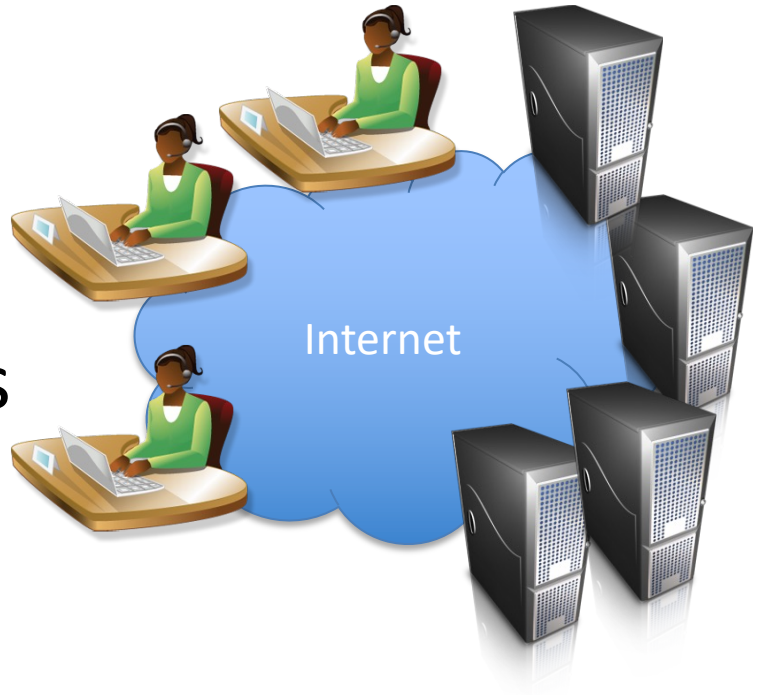
- Client knows what cert/pk to expect, rejects otherwise
  - Pre-install some keys
  - HPKP (HTTP Public Key Pinning)
    - HTTP header that allows servers to set a hash of public key they will use
    - “Trust-on-first-use” example

```
Public-Key-Pins:
    pin-sha256="d6qzRu9zOECb90Uez27xWltNsj0e1Md7GkYYkVoZWmM=";
    pin-sha256="LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=";
    max-age=259200
```

https://developers.google.com/web/updates/2015/09/HPKP-reporting-with-chrome-46?hl=en

Basic empirical questions about security

- Say we know about some attacks
  - Padding oracle attacks, DSA nonce reuse, bad RNGs for RSA keys, etc.
- What are interesting empirical questions related to cryptographic vulnerabilities?
  - How many servers on the internet are currently vulnerable?
  - How many attacks are occurring?



- How can we measure these things?

Internet scanning

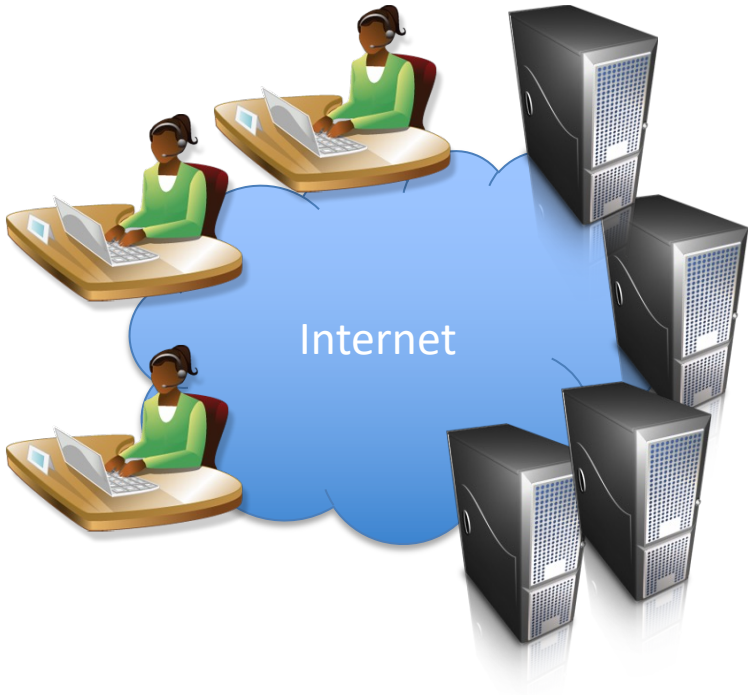
How do we survey the Internet to infer vulnerability status of systems?

Send network packets to IP address, see what response looks like

Example: TCP SYN scan

Horizontal scan: many IPs on 1 port

Vertical scan: many ports on one IP



Basics of network reconnaissance

- Host discovery
  - Narrow broad swath of potential IPs to ones that have hosts associated with them
- Service discovery
  - For a particular host, identify running services
  - E.g., is it accepting SSH connections (22) or HTTP (80)?
- OS fingerprinting
  - Identify the OS software version running
  - E.g., Windows vs Linux?
- Application fingerprinting
  - same at higher level
  - Apache version 1.3 or 2.0+?

Scanners

- NMAP (network map tool)
  - De-facto standard for network reconnaissance, testing
  - Numerous built in scanning methods
- Can use to scan *entire* IPv4 address space

```
[10/26/21 10:27:55 ~/Dropbox/work/teaching/cs6431-fall2021/slides-staging$ nmap -sT 192.168.1.0/28
Starting Nmap 7.92 ( https://nmap.org ) at 2021-10-26 10:28 EDT
Nmap scan report for Fios_Quantum_Gateway.fios-router.home (192.168.1.1)
Host is up (0.0024s latency).
Not shown: 992 closed tcp ports (conn-refused)
PORT     STATE    SERVICE
22/tcp   filtered ssh
53/tcp   open     domain
80/tcp   open     http
443/tcp  open     https
4567/tcp filtered tram
8022/tcp filtered oa-system
8080/tcp open     http-proxy
8443/tcp open     https-alt

Nmap done: 16 IP addresses (1 host up) scanned in 18.93 seconds
10/26/21 10:28:42 ~/Dropbox/work/teaching/cs6431-fall2021/slides-staging$
```

Internet-scale scans

Fewer than 2^{32} IP addresses: ~3,706,452,992
Some ranges can be excluded:

- Private addresses (e.g., 192.168.0.0/16)
- Multicast addresses
- …

Nmap is not particularly fast for horizontal scans:

Nmap –sT on target IP:port takes ~1 seconds

117 years for horizontal scan of all addresses? Need to speed up

“Insane” custom configuration: 116 days

Internet-scale scans

Nmap can nevertheless be scaled up

EFF SSL observatory: 2010 scan of ports 443
3 servers and 3 months of time

P's and Q's paper: 2011 scan of port 443 & 22:
25 hours from 25 EC2 Micro instances

Some scans

- 2011 scan (Heninger et al.)
  - 12,828,613 certificate chains recovered
  - 5,656,519 distinct RSA public keys
  - 6,241 distinct DSA public keys
  - Found many *factorable* RSA keys, DSA keys signing with repeat randomness

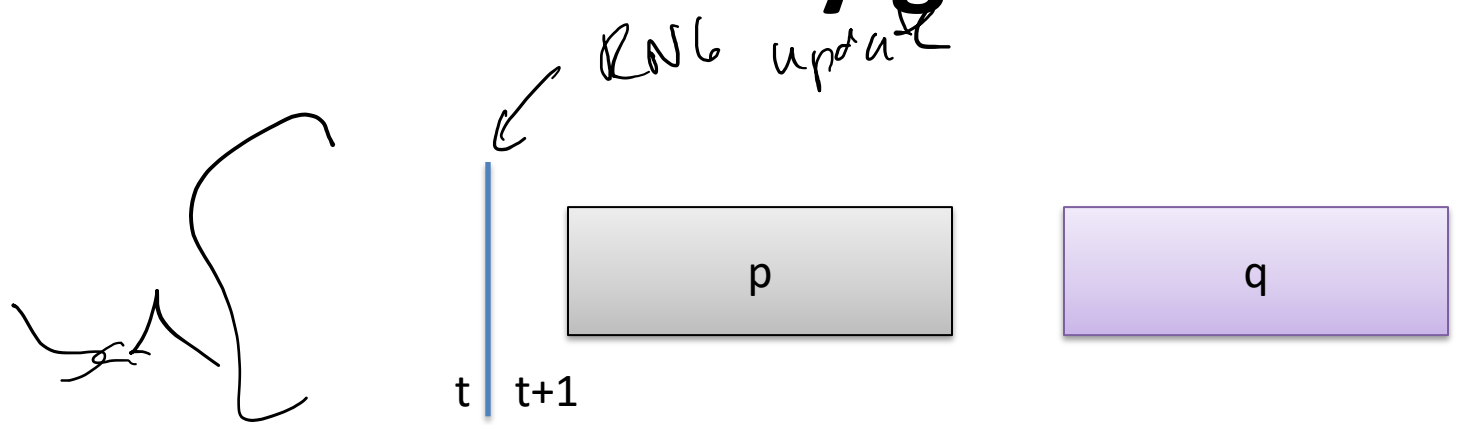
Weak RSA keys

[Heninger et al. 2012]

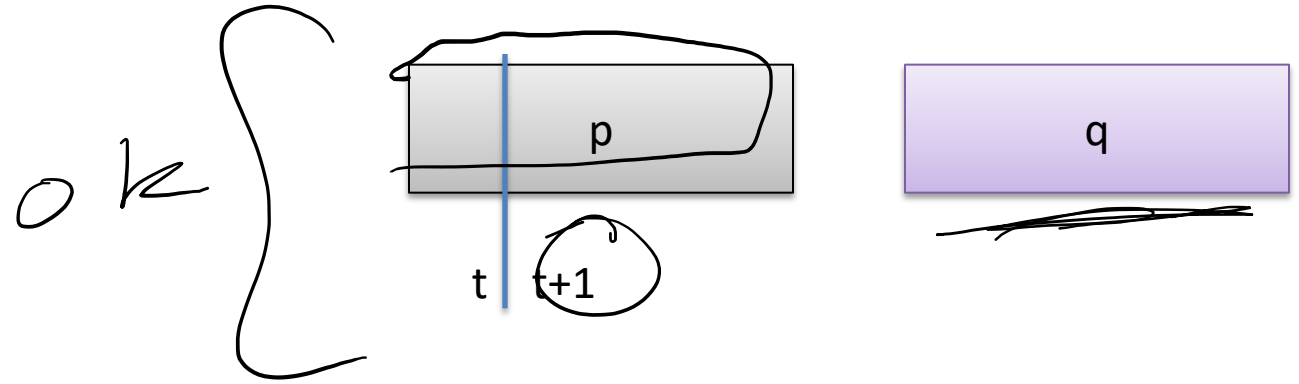
- Recall: (pk,sk) = ((N,e),(N,d)) for N = pq
- Repeat keys (may or may not be problem)

- Factoring is hard for large key sizes (>=1024)
  - But: bad randomness causes bad p,q generation
    - Ex: N1 = p * q N2 = p * q'
    - Compute GCD of large integers in milliseconds
    - Use Bernstein's all-pairs GCD to scale up

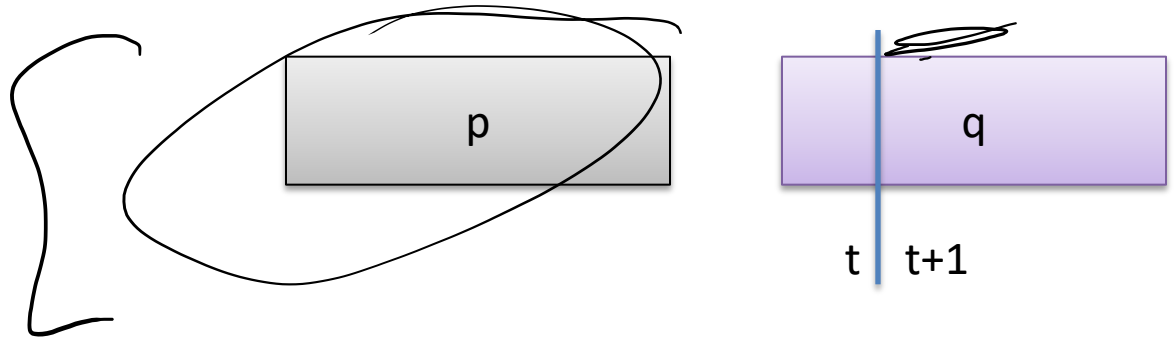
RNGs and RSA key generation



Both primes the same



Both primes different



Shared prime

Weak keys

	Our TLS Scan	
Number of live hosts	12,828,613	(100.00%)
. . . using repeated keys	7,770,232	(60.50%)
. . . using vulnerable repeated keys	714,243	(5.57%)
. . . using default certificates or default keys	670,391	(5.23%)
. . . using low-entropy repeated keys	43,852	(0.34%)
. . . using RSA keys we could factor	64,081	(0.50%)
. . . using DSA keys we could compromise		
. . . using Debian weak keys	4,147	(0.03%)
. . . using 512-bit RSA keys	123,038	(0.96%)
. . . identified as a vulnerable device model	985,031	(7.68%)
. . . model using low-entropy repeated keys	314,640	(2.45%)

From [Heninger et al. 2012]

Internet-scale scans

- How to scale up scans and make them faster/cheaper?
- Use a “horizontal scanner”
- Leonard & Loguinov 2010: 24 hour horizontal scans
  - Requires kernel modifications (on Windows)
- ZMap tool improved to 45 minute IPv4 scans

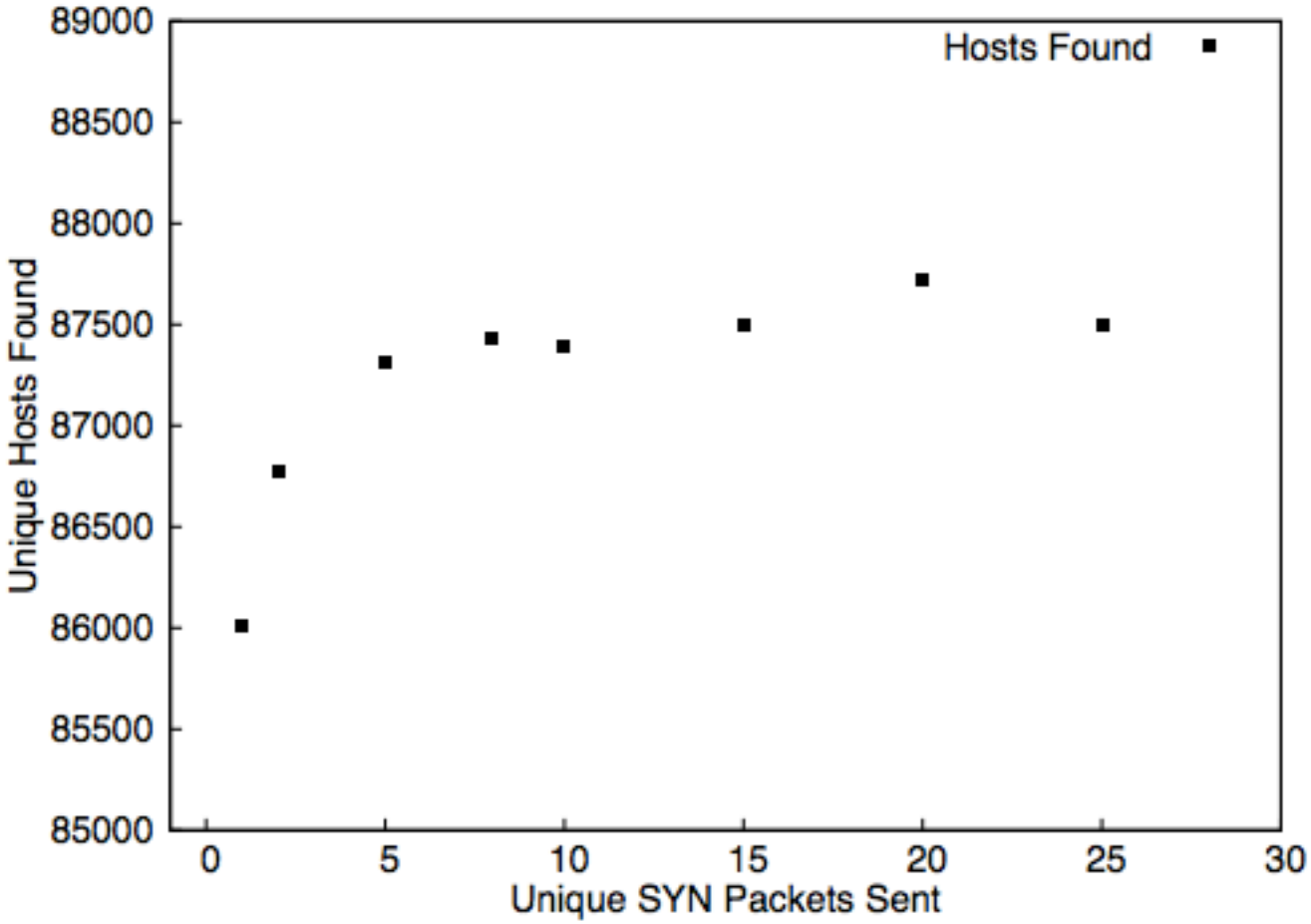
ZMap's design

- Random target ordering
  - What would be a problem with serial order?
  - Choose random generator g in $\mathbf{Z}_p$ for $p = 2^{32} + 15$, and walk the cycle $g, g^1, g^2, \ldots$. Can stop when one hits g again
- No per-connection state
- No retransmission
  - Send fixed number of probes (usually 1)
  - Don't try again
- Send as quickly as CPU/link allow
  - No Kernel mods necessary
  - Use raw socket and send ethernet frames directly

Performance

- Full horizontal IPv4 scan in 45 minutes with entry-level server and gigabit ethernet
- Scan rate up to 1.4 million packets / sec doesn’t affect hit rate
- Coverage of ~98% with just one probe:
  - For smaller sample size, ramp up # of probes, look for plateau
- Much better than Nmap (97% coverage at 117 days)

Coverage estimation



What to do with a fast scanner?

- Track protocol use over time (e.g., uptick in HTTPS)
- Find vulnerable hosts
  - Custom probe module to look for remotely exploitable vulnerabilities
  - Empirical analysis of cryptographic weaknesses
- Build a search engine (Censys follow-up paper)
  - https://censys.io/

Some scans

- 2011 scan (Heninger et al.)
  - 12,828,613 certificate chains recovered
  - 5,656,519 distinct RSA public keys
  - 6,241 distinct DSA public keys
  - Found many *factorable* RSA keys, DSA keys signing with repeat randomness
- 2013 scan (Durumeric et al.)
  - Almost all CA's using RSA keys for signing certs
  - ~8 million unique certificates across ~30 million hosts

Scanning now a widely used tool

- DROWN attacks (Bleichanbacher downgrade against SSLv2):
  - 30% of TLS servers on IPv4 are vulnerable
- Dual EC fingerprinting
  - Found 720 servers using BSAFE-Java
- LogJam attacks (Downgrade to export-level crypto in TLS)
  - 7% of Alexa Top Million
- Many further measurement studies: TLS CA ecosystem, SMTP server ecosystem, SCADA control systems, FTP servers, …

Client-side analysis: Android app ecosystem

- [Fahl et al. 2012] study:
  - Download 13,500 popular Android apps (from Play store)
  - 1,074 have code that accepts any TLS certificate, or any hostname
    - Used static analysis to identify such code
    - Manual analysis: ~40% are vulnerable to MITM attack
- [Georgiev et al. 2012] study
  - "For example, Amazon's Flexible Payments Service PHP library attempts to enable hostname verification by setting cURL's CURLOPT_SSL_VERIFYHOST parameter to true. Unfortunately, the correct, default value of this parameter is 2; setting it to true silently changes it to 1 and disables certificate validation."
- [Oltrogge et al. 2021] still lots of problems in this ecosystem
  - https://saschafahl.de/static/paper/eveandmallory2021.pdf

Summary

- Web PKI relies on various trust assumptions
  - Can be undermined in many ways

- Digital signature schemes power PKI and verifying identities:
  - unforgeability under chosen message attack
  - RSA based schemes PKCS#1 1.5, FDH, PSS

- Internet scanning, code analysis used to assess state of cryptography configurations