CS 578: CYBER-SECURITY

PART I: ECOSYSTEMS AND APPLICATIONS

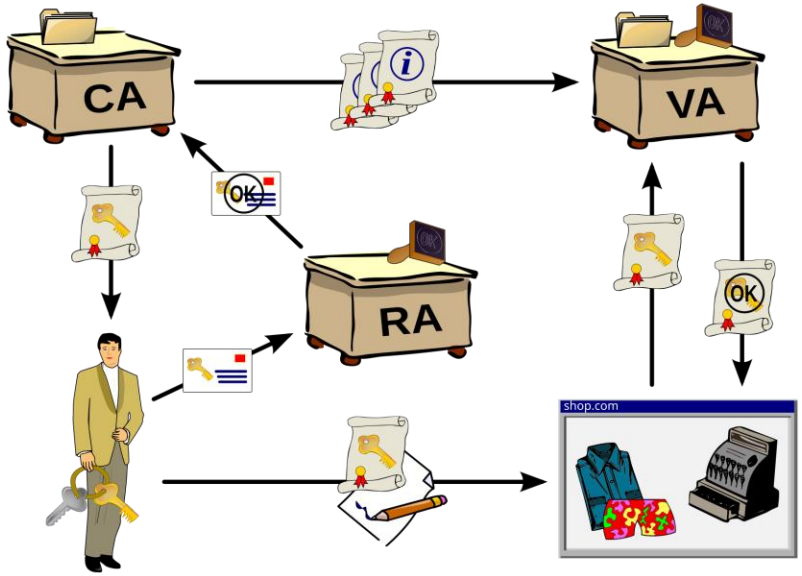
Sanghyun Hong
sanghyun.hong@oregonstate.edu

SAIL
Secure AI Systems Lab

PUBLIC KEY INFRASTRUCTURE (PKI)

PUBLIC KEY INFRASTRUCTURE

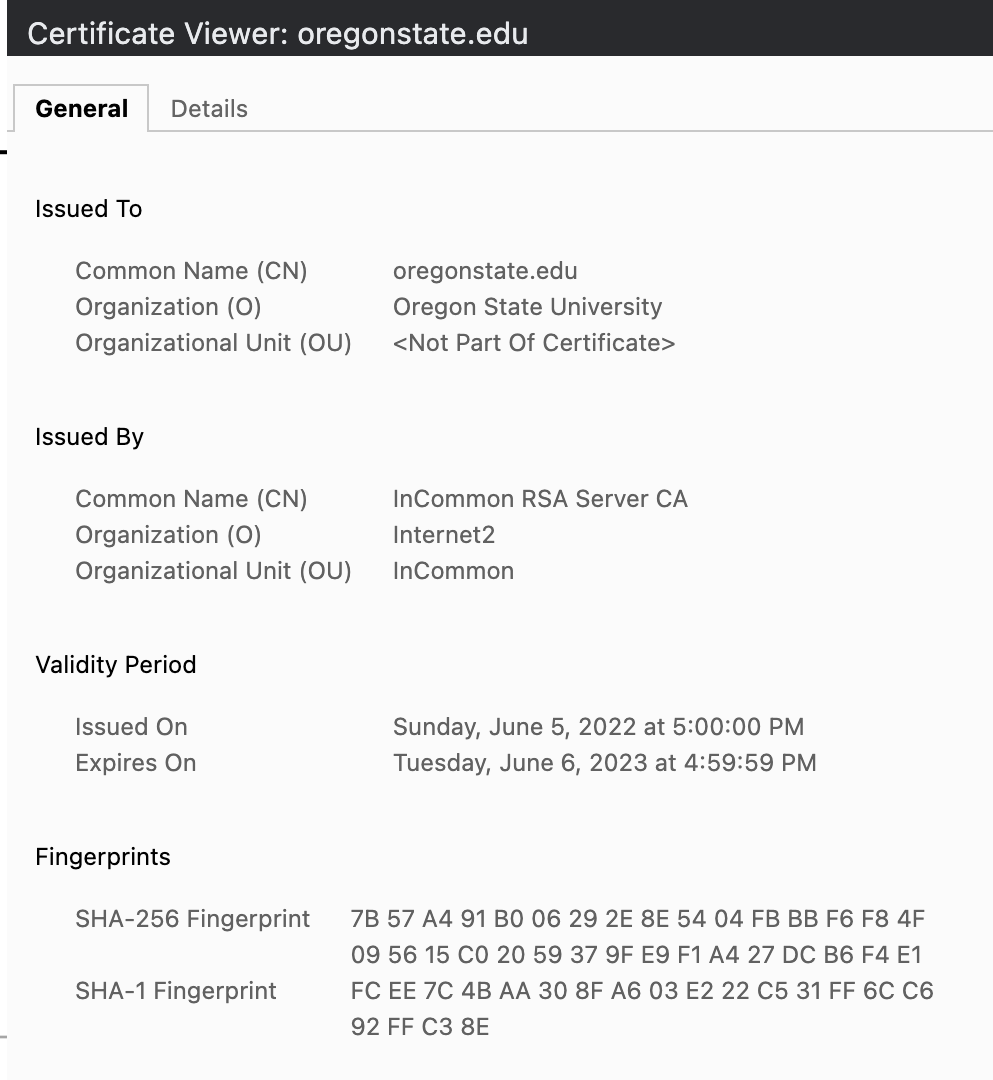
- A collection of
  - Hardware, software, policies, procedures and humans
  - Required to create, manage, distribute, use, store, and revoke digital certificate
- Components
  - RA (registration authority)
  - CA (certificate authority)
  - VA (validation authority): X.509, CRL
  - Others:
    - Central directory
    - Management system
    - Policy



[1]https://en.wikipedia.org/wiki/Public_key_infrastructure

PUBLIC KEY INFRASTRUCTURE

- Digital certificate
  - Entity info (CN)
  - Issuer info (CN)
  - Public key
  - Signature



PUBLIC KEY INFRASTRUCTURE – CERTIFICATE CREATION

- Requester prepares a certificate request
  - Entity information
  - Public key
  - Signature (proving that I have the public key)

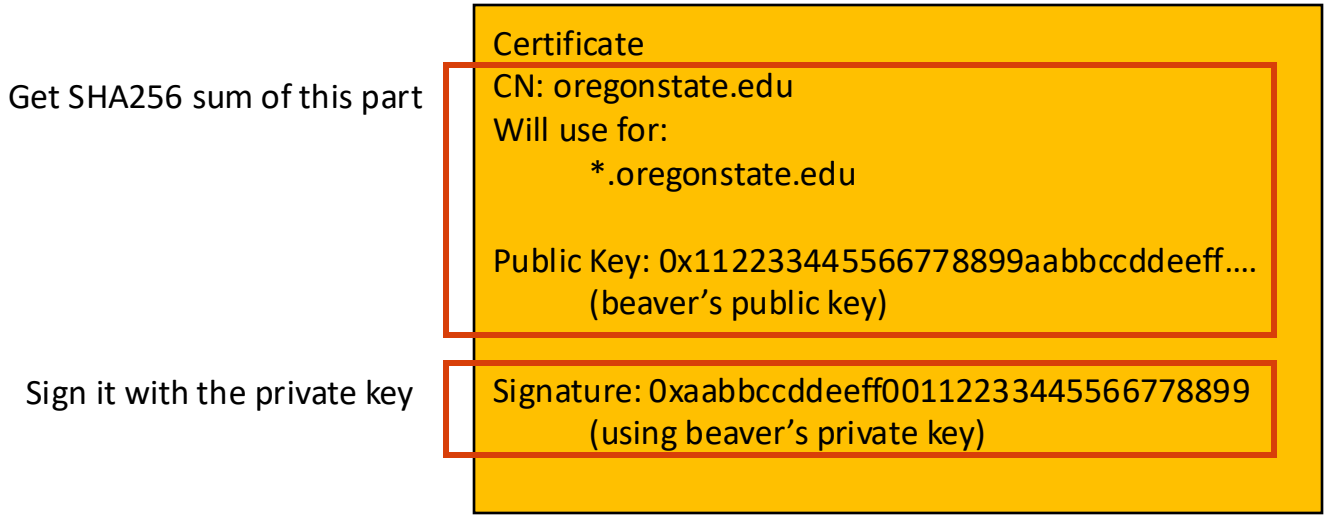
Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x11223344556677889aabbccddeeff….
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(using beaver's private key)

PUBLIC KEY INFRASTRUCTURE – CERTIFICATE CREATION

- Requester prepares a certificate request
  - Entity information
  - Public key
  - Signature (proving that I have the public key)



PUBLIC KEY INFRASTRUCTURE – CERTIFICATE CREATION

- Requester prepares a certificate request
  - Entity information
  - Public key

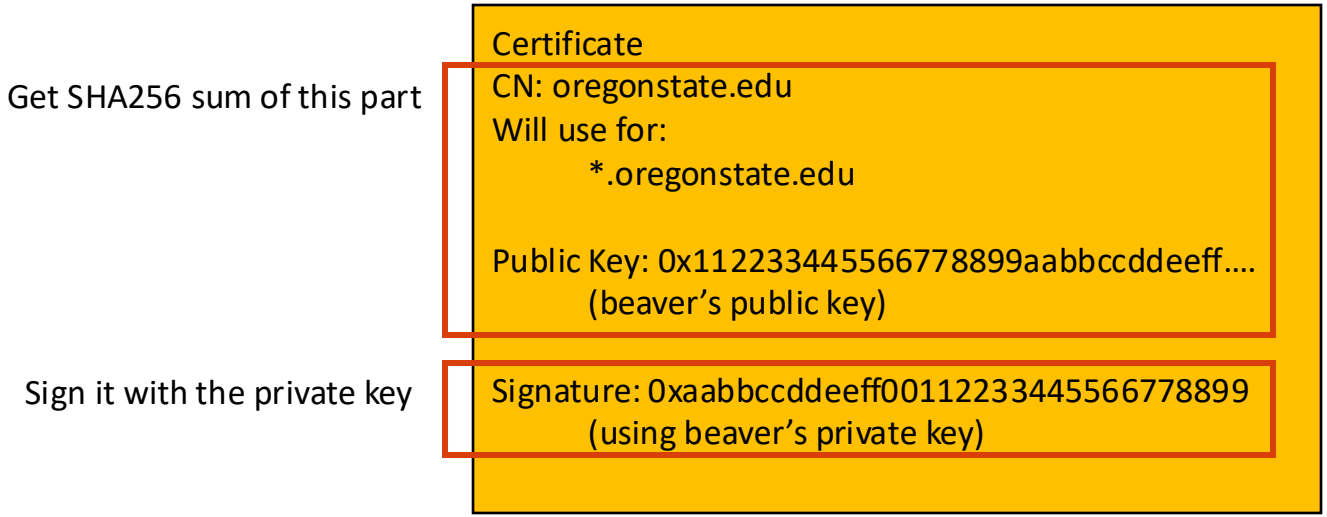
- Issuer (CA) verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer's private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```

PUBLIC KEY INFRASTRUCTURE – CERTIFICATE CREATION

- Issuer (CA) verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer's private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```



PUBLIC KEY INFRASTRUCTURE – CERTIFICATE CREATION

- Requester prepares a certificate request
  - Entity information
  - Public key

- Issuer verifies the requester information, and digitally sign the cert
  - Verify the entity information
  - Get a SHA-256 fingerprint of the certificate
  - Sign the fingerprint (with issuer's private key)

```
RSA_encrypt(private_key, SHA-256(certificate))
```

- Anyone with the public key can verify the result
  - Get issuer's public key from their certificate

CERTIFICATION CREATION DETAILS: STEP 1

- The certificate requesting entity fills
  - Entity information
  - Public Key

- Entity:
  - For google, its *.google.com
  - Can be your website address

- *.secure-ai.systems
  - also has a certificate

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x11223344556677889aabbccddeeff….
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(with beaver's private key)

CERTIFICATION CREATION DETAILS: STEP 2

- The issuer receives the certificate request and verifies:
  - Entity
    - Their identification
    - Owning the target domain name
    - Owning the public key
  - The signature
    - Decrypt the signature with public key
    - It must be the same as SHA256 sum
    - It proves their holding the private key

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x11223344556677889aabbccddeeff….
(beaver's public key)

Signature: 0xaabbccddeeff00112233445566778899
(with beaver's private key)

CERTIFICATION CREATION DETAILS: STEP 2

- The issuer receives the certificate request and verifies:
  - Entity:
    - Their identification
    - Owning the target domain name
    - etc…
  - Then, fill issuer information
    - Issuer information
    - Issuer public key

InCommon®

CN = oregonstate.edu

Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x11223344556677889aabbccddeeff….
(beaver's public key)

Issuer: InCommon RSA
Public Key: 0x2233445566778899011aabbccddeeff

CERTIFICATION CREATION DETAILS: STEP 2

- The issuer receives the certificate request and verifies:
  - Entity:
    - Their identification
    - Owning the target domain name
    - etc…
  - Then, fill issuer information
    - Issuer information
    - Issuer public key
  - and then, sign the certificate
    - Get SHA-256 of the certificate
    - Attach it as a signature!

InCommon®

CN = oregonstate.edu

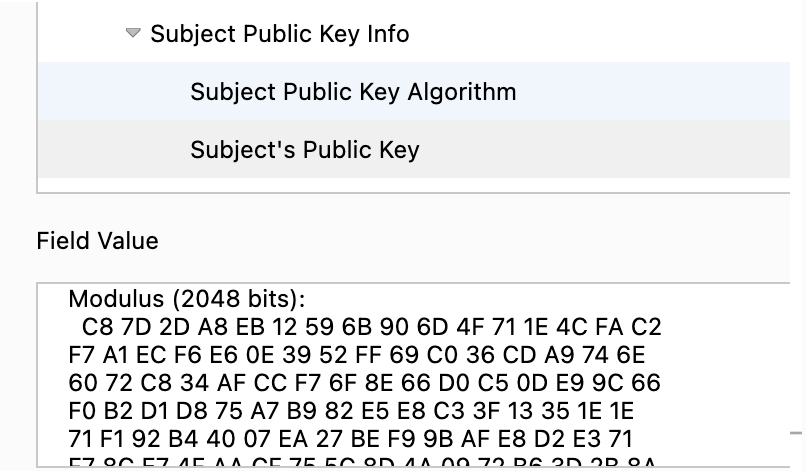
Certificate
CN: oregonstate.edu
Will use for:
*.oregonstate.edu

Public Key: 0x11223344556677889 9aabbccddeeff….
(beaver's public key)

Issuer: InCommon RSA
Public Key: 0x22334455667788990011aabbccddeeff
Signature: 0xffeeddccbbaa00112233445566778899
(InCommon RSA's private key)

PUBLIC KEY INFRASTRUCTURE – CERT

- Now InCommon RSA verified
  - oregonstate.edu is owned by
  - Oregon State University
  - With a specific Public Key



Certificate Viewer: oregonstate.edu

General | Details

Issued To

Common Name (CN)	oregonstate.edu
Organization (O)	Oregon State University
Organizational Unit (OU)	<Not Part Of Certificate>

Issued By

Common Name (CN)	InCommon RSA Server CA
Organization (O)	Internet2
Organizational Unit (OU)	InCommon

Validity Period

Issued On	Sunday, June 5, 2022 at 5:00:00 PM
Expires On	Tuesday, June 6, 2023 at 4:59:59 PM

Fingerprints

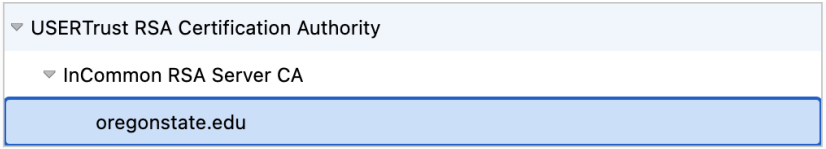
SHA-256 Fingerprint	7B 57 A4 91 B0 06 29 2E 8E 54 04 FB BB F6 F8 4F 09 56 15 C0 20 59 37 9F E9 F1 A4 27 DC B6 F4 E1
SHA-1 Fingerprint	FC EE 7C 4B AA 30 8F A6 03 E2 22 C5 31 FF 6C C6 92 FF C3 8E

PUBLIC KEY INFRASTRUCTURE – CERTIFICATE VERIFICATION

- OSU owns “oregonstate.edu”
  - Verified by InCommon RSA
- Verification of the certificate
  - Use InCommon RSA’s public key
  - Where is it? It is written in InCommon RSA’s certificate
- But InCommon RSA, who will verify their identity?
  - InCommon RSA verifies “oregonstate.edu”
  - Who will verify InCommon RSA?

PUBLIC KEY INFRASTRUCTURE – CERTIFICATE VERIFICATION

- "oregonstate.edu"
  - Verified by InCommon RSA Server CA

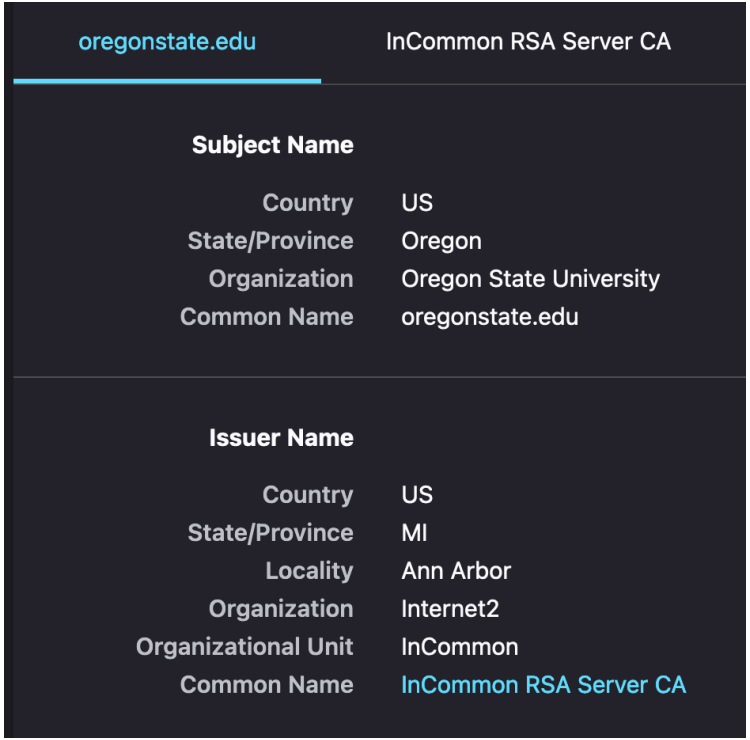


- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Authority

- USERTrust RSA CA
  - Verified by self

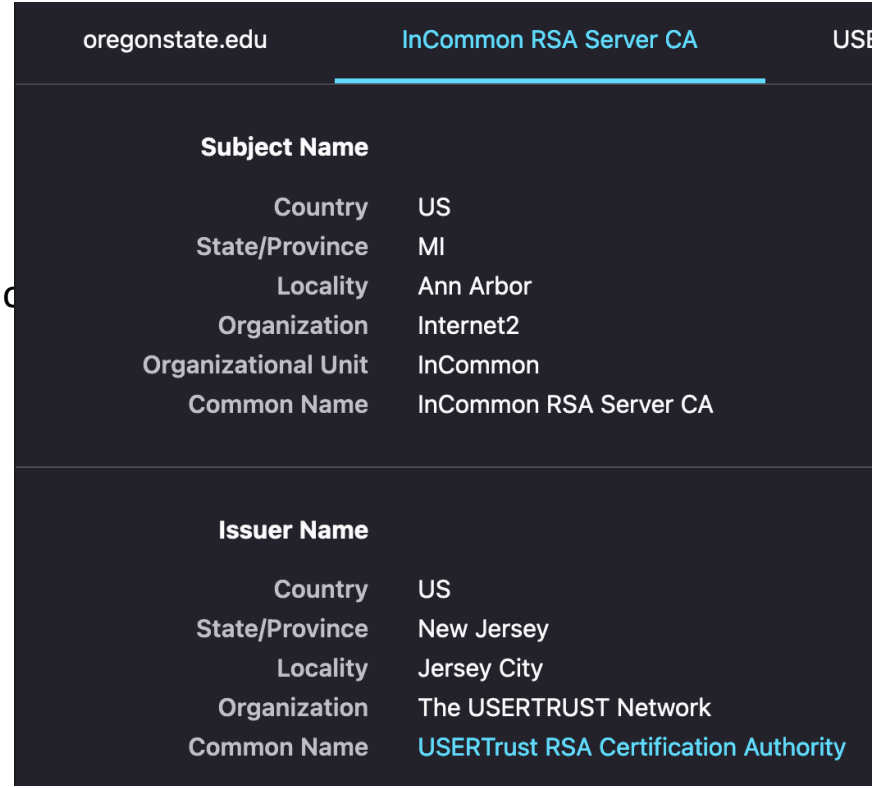
PUBLIC KEY INFRASTRUCTURE – CHAIN OF TRUST

- "oregonstate.edu"
  - Verified by InCommon RSA Server CA
- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Authority
- USERTrust RSA CA
  - Verified by self



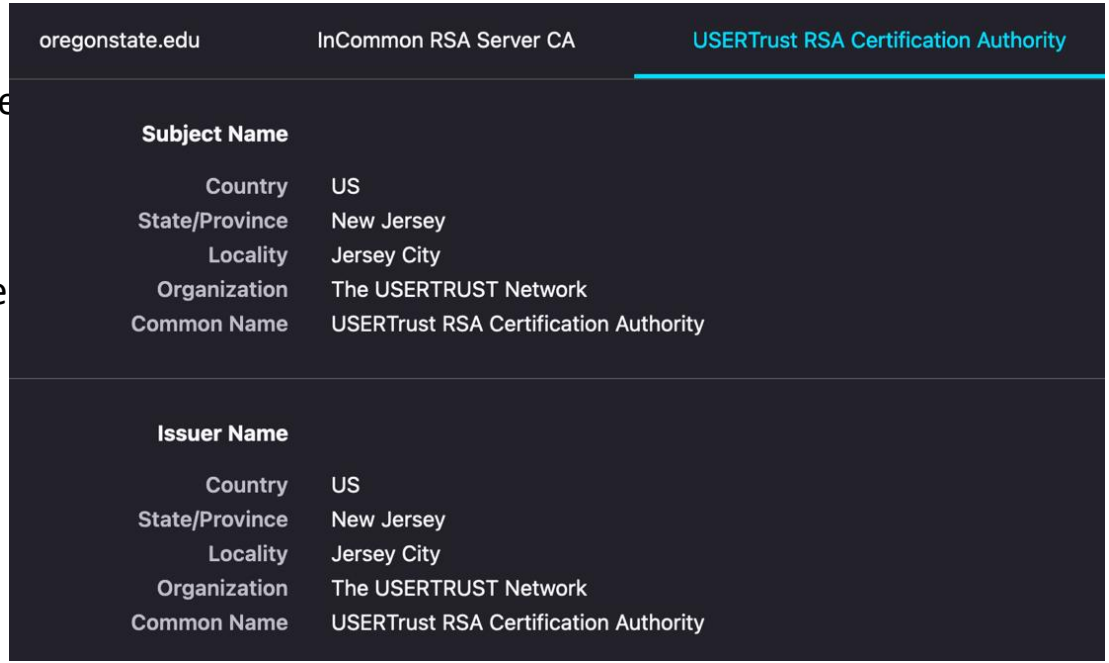
PUBLIC KEY INFRASTRUCTURE – CHAIN OF TRUST

- "oregonstate.edu"
  - Verified by InCommon RSA Server CA
- InCommon RSA Server CA
  - Verified by USERTrust RSA Certificate Autho
- USERTrust RSA CA
  - Verified by self



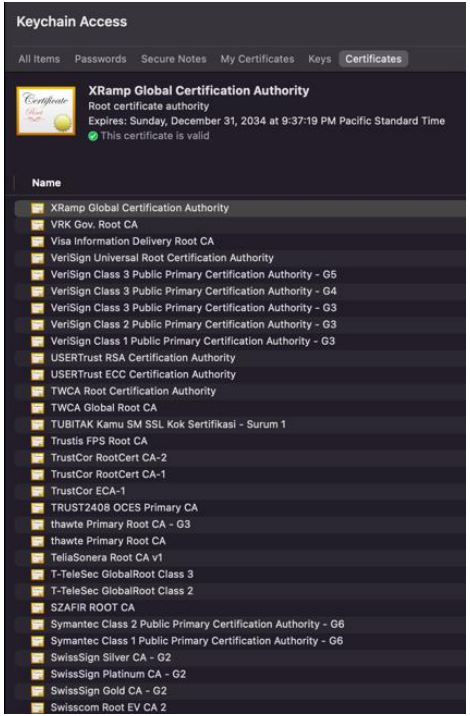
PUBLIC KEY INFRASTRUCTURE – CHAIN OF TRUST

- "oregonstate.edu"
  - Verified by InCommon RSA Se
- InCommon RSA Server CA
  - Verified by USERTrust RSA Ce
- USERTrust RSA CA
  - Verified by self



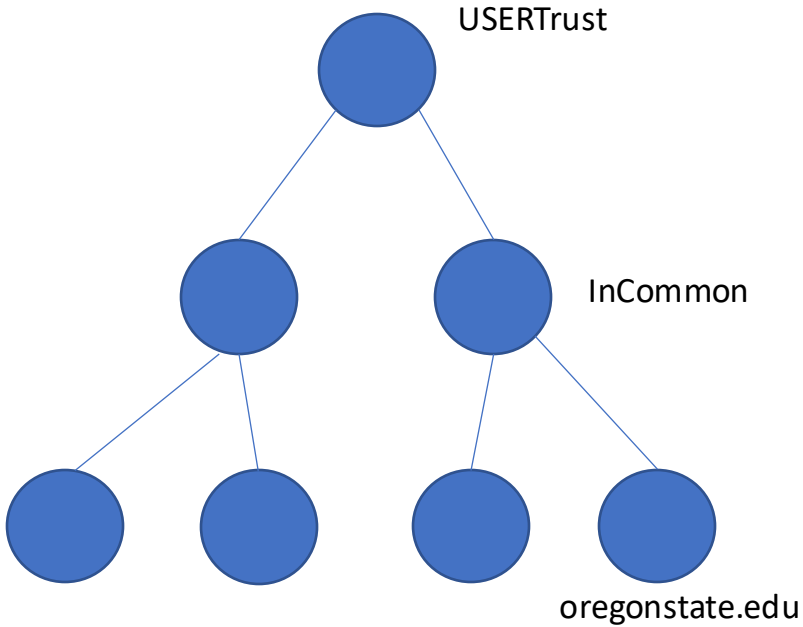
ROOT CERTIFICATE AUTHORITY (ROOT CA ≈ US IN PREV. EXAMPLE)

- Define small set of trustworthy certificate authorities
  - Private companies are authorized by some jurisdiction to run the CA company
    - Google Trust Service (GTS CA)
    - DigiCert
    - Verisign
    - etc..
- Trust their self-signed certificate
  - Stored in almost every computer machines



PUBLIC KEY INFRASTRUCTURE (PKI)

- An Infrastructure that provides public key with certificate chain
- Trust anchor: Root CA
  - Set a small set of entities use self-signed cert
- Verify the certificate chain!
  - Must verify the entire chain



PUBLIC KEY INFRASTRUCTURE – CERTIFICATE REVOCATION

- Make an issued certificate invalid
  - CA is responsible for revocation

- Revocation procedure
  - A certificate holder informs the CA, possibly
    - The certificate is *compromised*
    - The certificate expiration date is approaching
  - CA produces authenticated attestations that the certificate has been revoked
  - CA maintains a certificate revocation list (CRL)
    - Only for the certificates that has re-issued and revoked prior to their expiration date
    - Contains (serial number, time stamp of revocation, reason for revocation, …)
    - Client is responsible for periodically download CRLs

PUBLIC KEY INFRASTRUCTURE – CERTIFICATE REISSUE

- Re-create and replace a certificate
  - CA is responsible for certificate reissue
  - Client is responsible for making a certificate signing request

- Reissue procedure
  - A certificate holder informs the CA, possibly
    - The certificate is *compromised*
    - The certificate expiration date is approaching
  - A certificate holder makes a certificate signing request
    - CA will make a new signature with their private key
    - (Optional) Client can choose a new private/public key pair for the reissue

CERTIFICATE REVOCATION MEASUREMENT – IN THE WAKE OF HEARTBLEED

REVISIT: HANDSHAKE REQUIRES FORWARD SECURITY

- Forward Secrecy / Perfect Forward Secrecy
  - We want to keep all the communication secure
  - Even if the server's private key (i.e., the long-term key) has been breached

- Example of such breaches
  - Heartbleed (https://heartbleed.com/): CVE-2014-0160

IMPACT OF HEARTBLEED ON CERTIFICATE REVOCATION

- Check the trust chain
  - A large increase in the fraction of newly-appearing certificates
  - Many certificates are re-issued in the wake of heartbleed

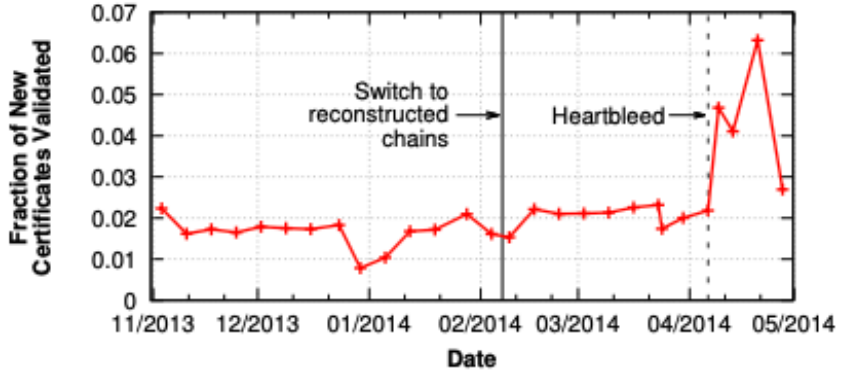


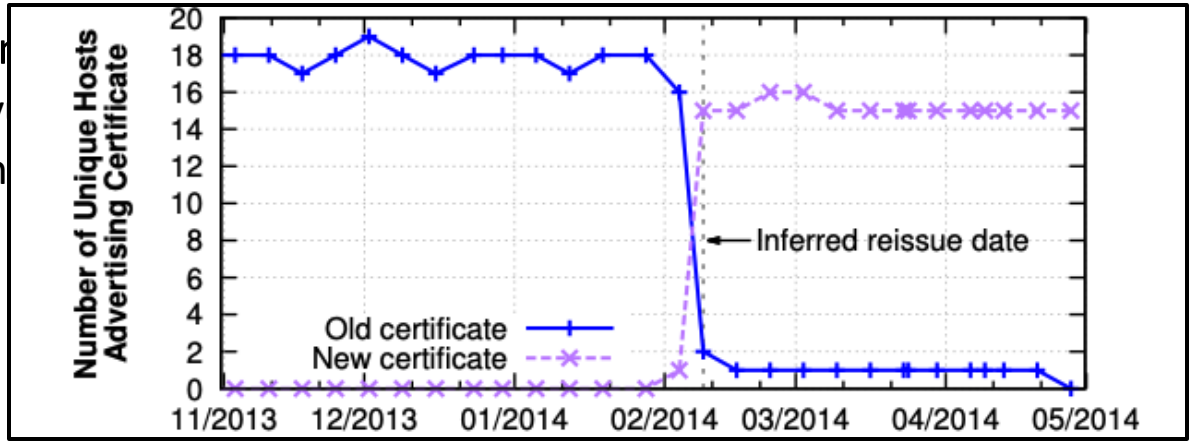
Figure 2: Fraction of new certificates that we could verify for provided (February 5, 2014 and before) and reconstructed (post February 5, 2014) chains.

INFERRING HEARTBLEED VULNERABILITY

- Check if a website admin revoked or reissued their certificate
  - It has been running a vulnerable OpenSSL version
  - It has not supported the Max Fragment Length

INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Check if a website admi
  - It has been running a v
  - It has not supported th



- Vulnerability analysis
  - Certificate birth: the date of the first scan (observing a host advertising the cert)
  - Certificate death: the last date that the number of advertising the cert > 10% of the max advertising that they observed before
  - Certificate reissue:
    - If the cert dies and they observe a new cert with the same CN within 10 days
    - and If at least one IP address switch from the old cert to the new cert
  - Certificate revocation: when the cert's serial number appears in any CRLs

INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Birth, death, reissue, and revocation
  - The number of birth > the number of death
  - A large spike in all four events in the wake of Heartbleed
  - On average, 29 revocations per day before Heartbleed, but this jumps to 1,414 after it

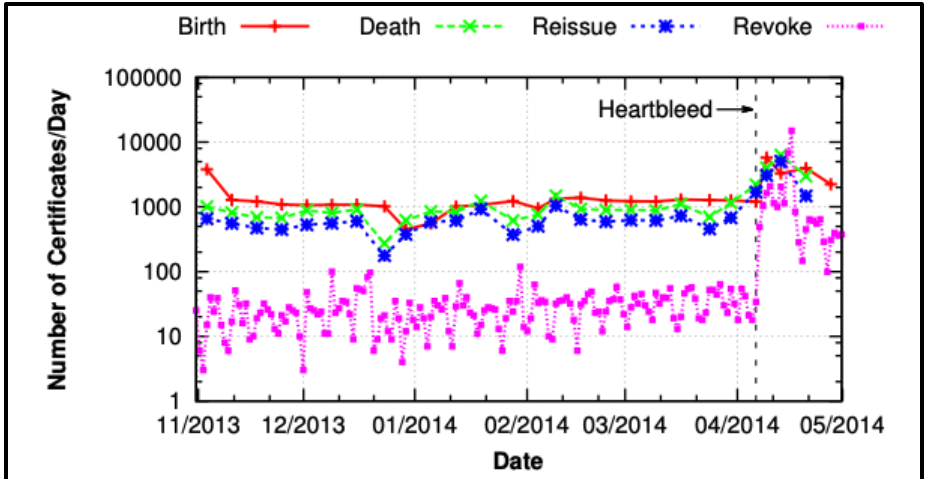


Figure 5: Number of certificate birth, deaths, reissues, and revocations over time. Note the log scale on the y-axis.

INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Prevalence after the wake
  - The vulnerability has been reduced to less than 10%
  - But that does not mean that our Internet is safe

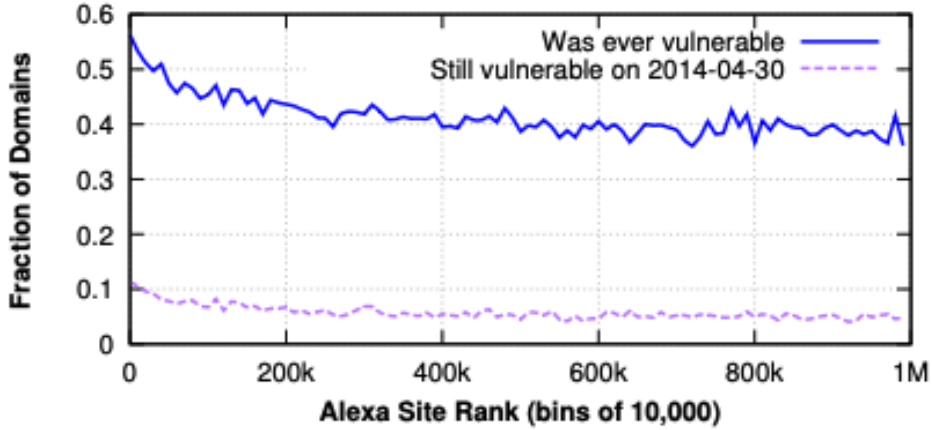
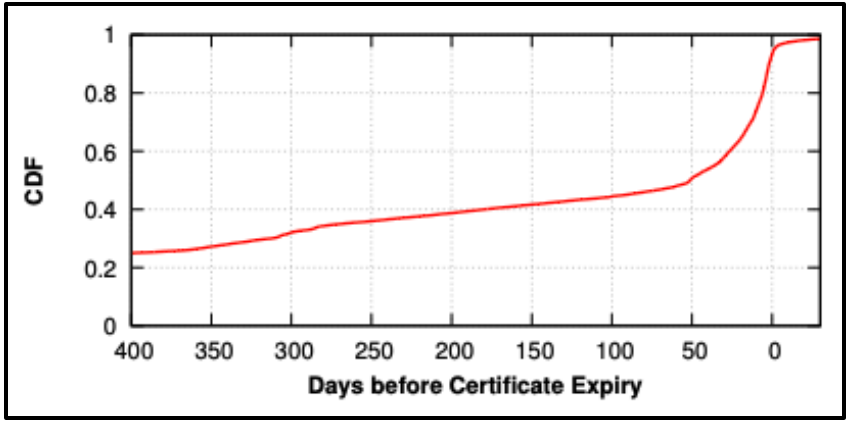
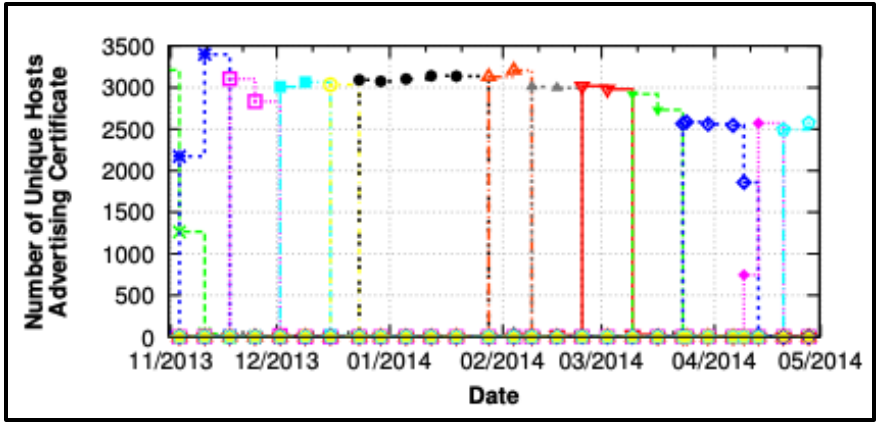


Figure 6: Fraction of domains that have at least one host that was ever vulnerable to Heartbleed as a function of Alexa rank, as well as domains that continued to be vulnerable at the end of the study.

INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Certificate Reissues
  - Generally, 50% of the certificates are re-issued within 60 days
  - A site may periodically reissue certificates as a matter of policy, *e.g.*, google.com





INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Certificate Reissues
  - Generally, 50% of the certificates are re-issued within 60 days
  - A site may periodically reissue certificates as a matter of policy, *e.g.*, google.com

- Heartbleed induced reissues
  - The date of reissue was on and after the Heartbleed
  - > 60 days left until the expiration date
  - No two other issues for the certs with the same common name (CA) before Heartbleed

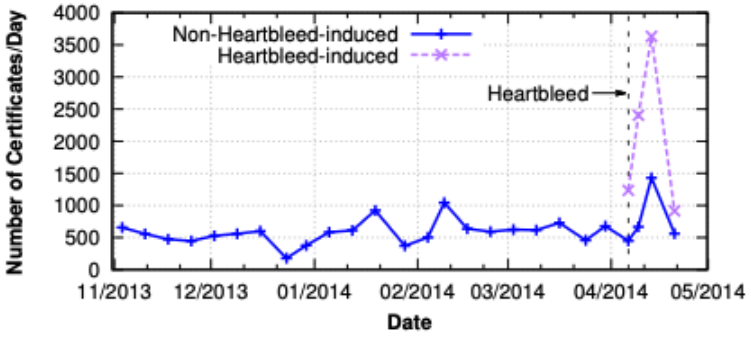


Figure 9: Number of Heartbleed-induced and non-Heartbleed-induced certificate reissues over time.

INFERRING HEARTBLEED VULNERABILITY – CONT'D

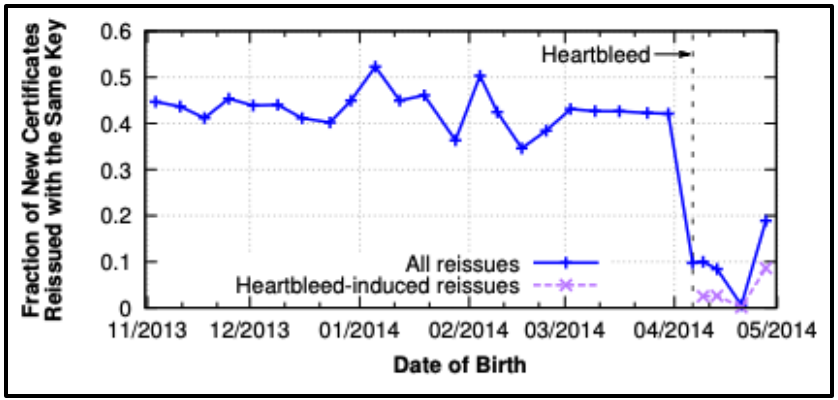
- Certificate Reissues
  - Generally, 50% of the certificates are re-issued within 60 days
  - A site may periodically reissue certificates as a matter of policy, *e.g.*, google.com

- Heartbleed-vulnerable certificates (that should have been reissued)
  - The date of birth was before Heartbleed
  - The certs won't expire 1 mo after the event
  - The certs were from the vulnerable hosts
  - Results
    - 107,712 vulnerable certificates
    - 26.7% has been re-issued before "1 mo after Heartbleed"
    - 73.3% has not been re-issued

INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Certificate Reissues
  - Generally, 50% of the certificates are re-issued within 60 days
  - A site may periodically reissue certificates as a matter of policy, *e.g.*, google.com

- Heartbleed-vulnerable certificates (that should have been reissued)
  - The date of birth was before Heartbleed
  - The certs won't expire 1 mo after the event
  - The certs were from the vulnerable hosts
  - Results
    - There are certs are re-issued with the *same key*



INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Certificate revocation
  - There is a spike in revocation after the wake of Heartbleed
  - Most of the domains that will revoke their certs in direct response to Heartbleed
  - There are "dips" in the revocation rate (particularly during the weekend)

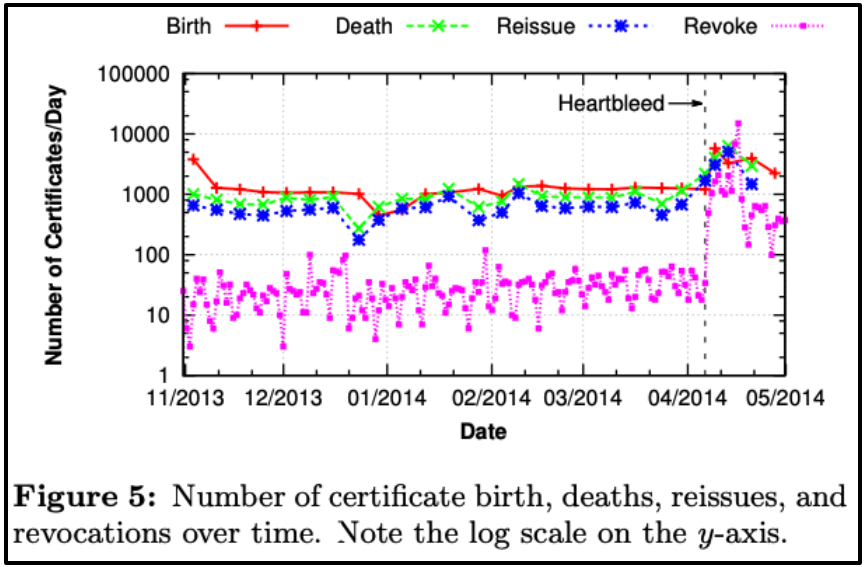
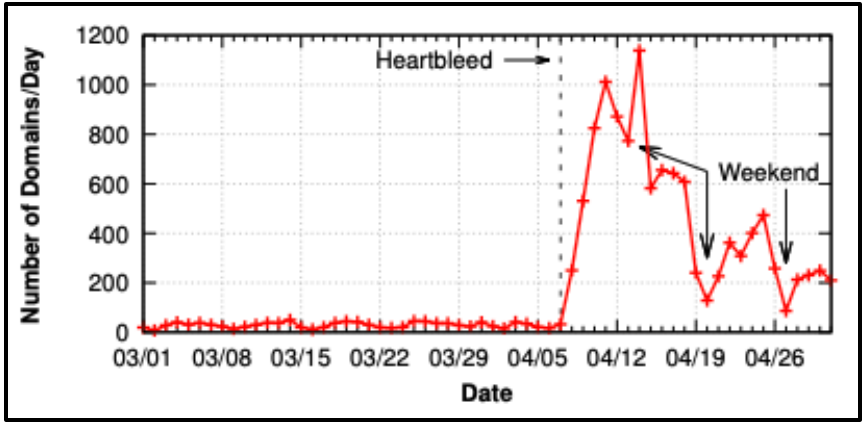
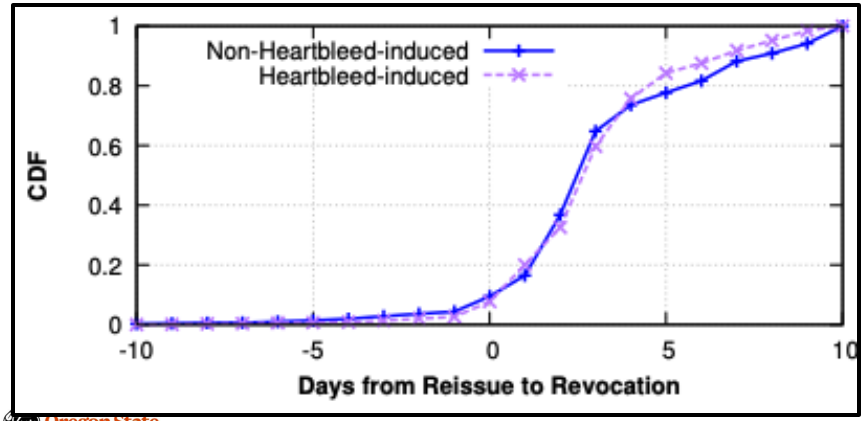


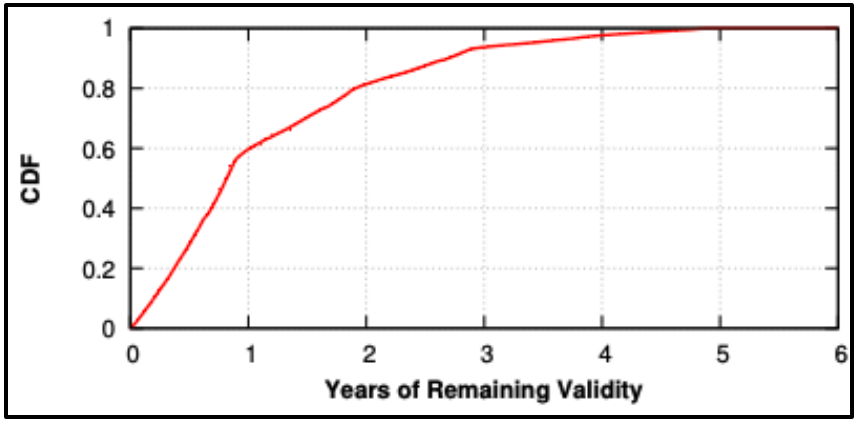
Figure 5: Number of certificate birth, deaths, reissues, and revocations over time. Note the log scale on the y-axis.



INFERRING HEARTBLEED VULNERABILITY – CONT'D

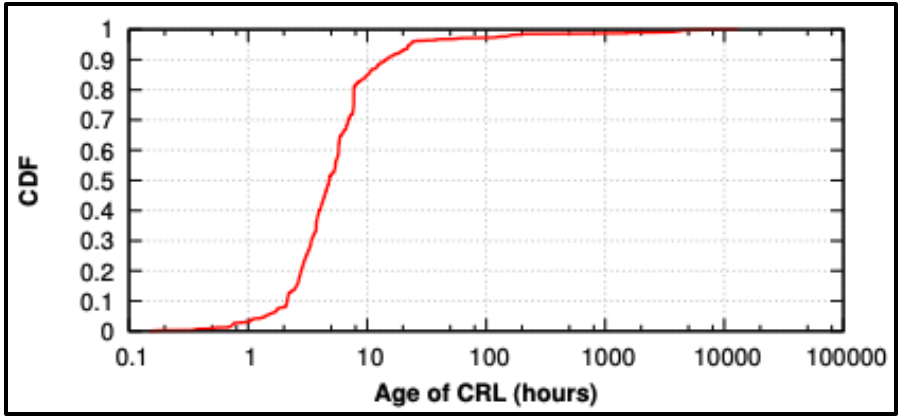
- Certificate revocation
  - 40% of the retired certificates are revoked (2-3% of the total revocation)
  - Revocation and reissue do not happen at the same time… what? (speed difference)
  - Reissues have been done, but revocations were not done (the long term vuln.)





INFERRING HEARTBLEED VULNERABILITY – CONT'D

- Certificate revocation lists (CRLs)
  - An increase in the number of revocation reasons "key compromised" (0.4 -> 1.18%)
  - 85% of the times, 85% of the revocations are available on clients after 10 hours
    - CAs could revoke certificates as often as every few hours
    - The delay was because of the human-in-the-loop
    - It's unclear about the client's impact



Thank You!

Sanghyun Hong

https://secure-ai.systems/courses/Sec-Grad/current

SAIL
Secure AI Systems Lab