51.505 – Foundations of Cybersecurity

Week 12 – PKI

Created by **Pawel Szalachowski** (2017)
Modified by **Jianying Zhou** (2018)

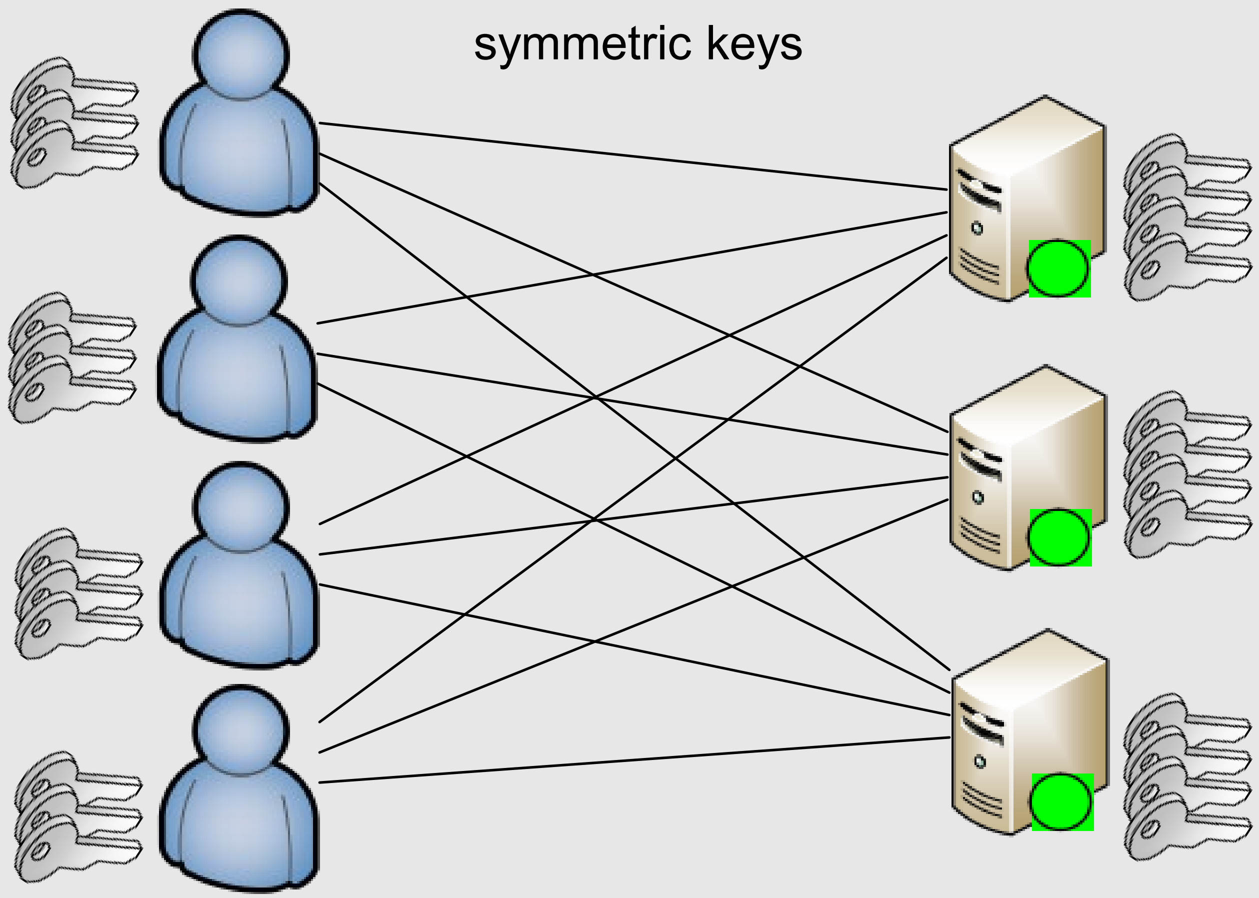
Last updated: 30 Nov 2018

Recap

- Questions on Week 11’s exercises?

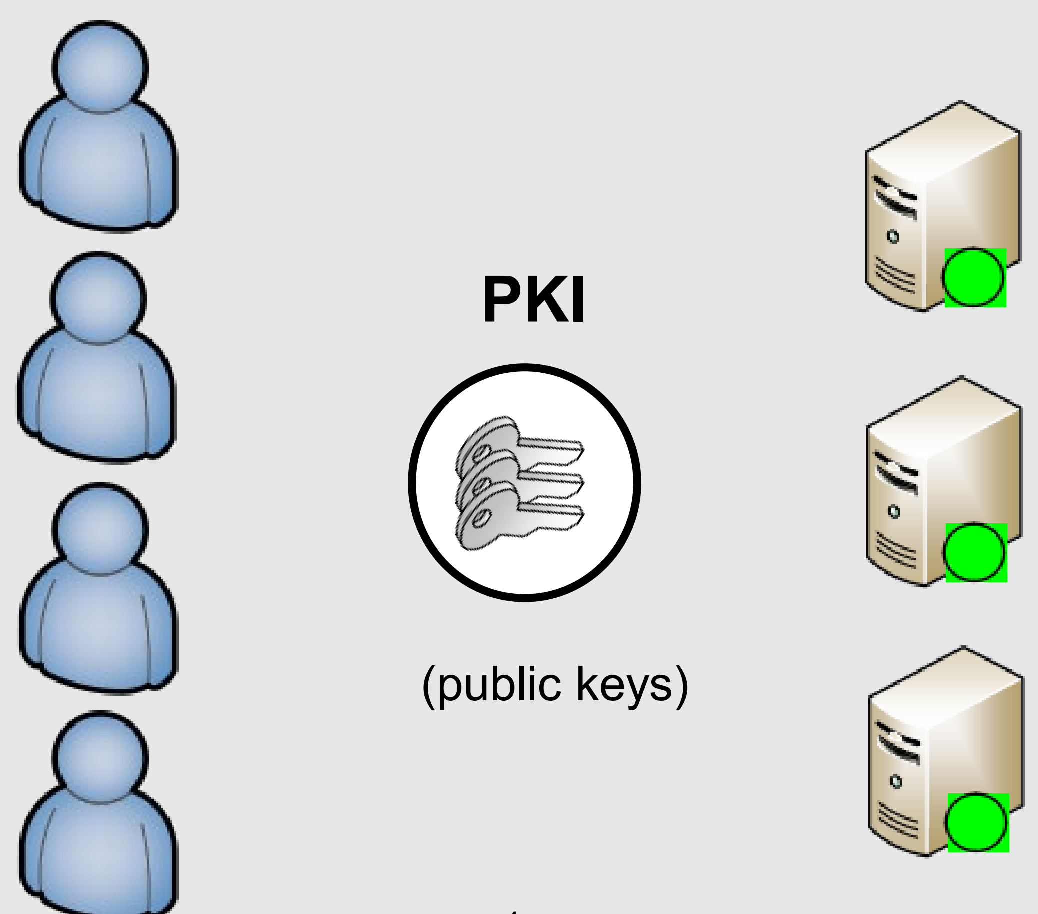
Symmetric Keys (scalability)

- Scalability issues with symmetric crypto
  - ✓ Distribution
  - ✓ Challenges in managing *n* secrets



Public Key Infrastructure (PKI)

- Asymmetric crypto (DH, RSA, …) solves the scalability problems, ... but creates a new one:
  - ✓ **How to ensure that public-key is accessible and authentic?**



Public-Key Infrastructure (PKI)

- An infrastructure that allows to recognize which public key belongs to whom
- There is a central authority, called the *Certificate Authority* (CA).
  - ✓ Everyone trusts the CA and knows its public key.
- Alice to join the PKI
  - ✓ Alice generates public/private key pair (PK_A,PK_A^-) and contacts the CA.
  - ✓ The CA verifies her identity and issues a signed *certificate* that claims that "PK_A belongs to Alice".
  - ✓ Alice can now contact Bob sending PK_A and the certificate.
  - ✓ As Bob trusts the CA, he trusts the certificate.

Trust and Trust Models

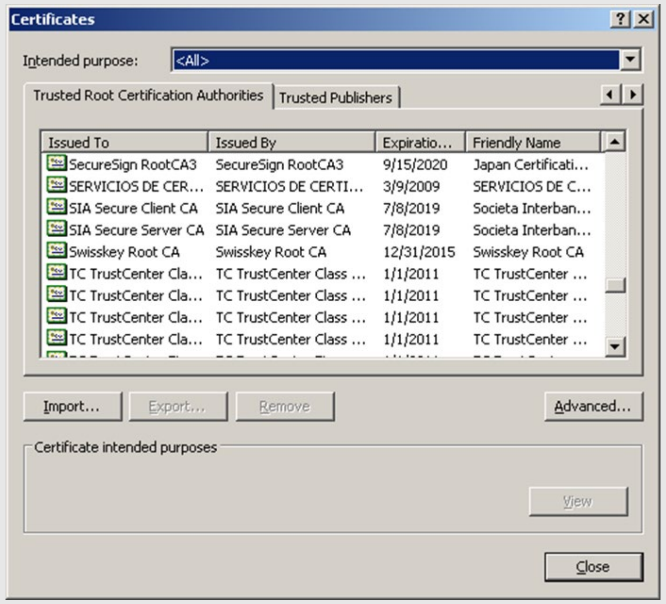
- Needed to solve scalability issues
- Trust Models
  - ✓ Centralized (Monopoly, Oligarchy)
  - ✓ Decentralized (Anarchy)

Trust and Trust Models

- **Monopoly**
  - ✓ Everyone trust a single organization to be the only CA in the world.
  - ✓ The key of that CA is embedded in all software and hardware as the PKI *trust anchor*.
- Problem:
  - ✓ There is no universally trusted entity.
    - Would a bank trust an external CA to issue certificates to its customers?

Trust and Trust Models

- **Oligarchy**
  - ✓ Used in browsers.
  - ✓ A number of organizations set themselves up as trust anchors.
    - – Chances that one of these is corrupted is higher than the chance that a single trust anchor is corrupted.
  - ✓ Product vendor selects amongst them.



- Problem:
  - ✓ Make it easier to dupe users into accepting a trust anchor.
    - – Even knowledgeable users will have a hard time checking the 80+ trust anchors in a current browser.

Trust and Trust Models

- **Anarchy**
  - ✓ Used in PGP.
  - ✓ Each user is responsible for configuring some trust anchors themselves.
    - Meeting people who hand out their certificates.
    - Search through public databases to find a path from one trust anchor to the key you want. (You implicitly trust everyone on that path!)
- Problem:
  - ✓ Not scalable, especially hard on certificate revocation/validation.
  - ✓ Mainly for personal (rather than corporate) use.

Trust and Trust Models

- Trust Models
  - ✓ Centralized (Monopoly, Oligarchy)
  - ✓ Decentralized (Anarchy)

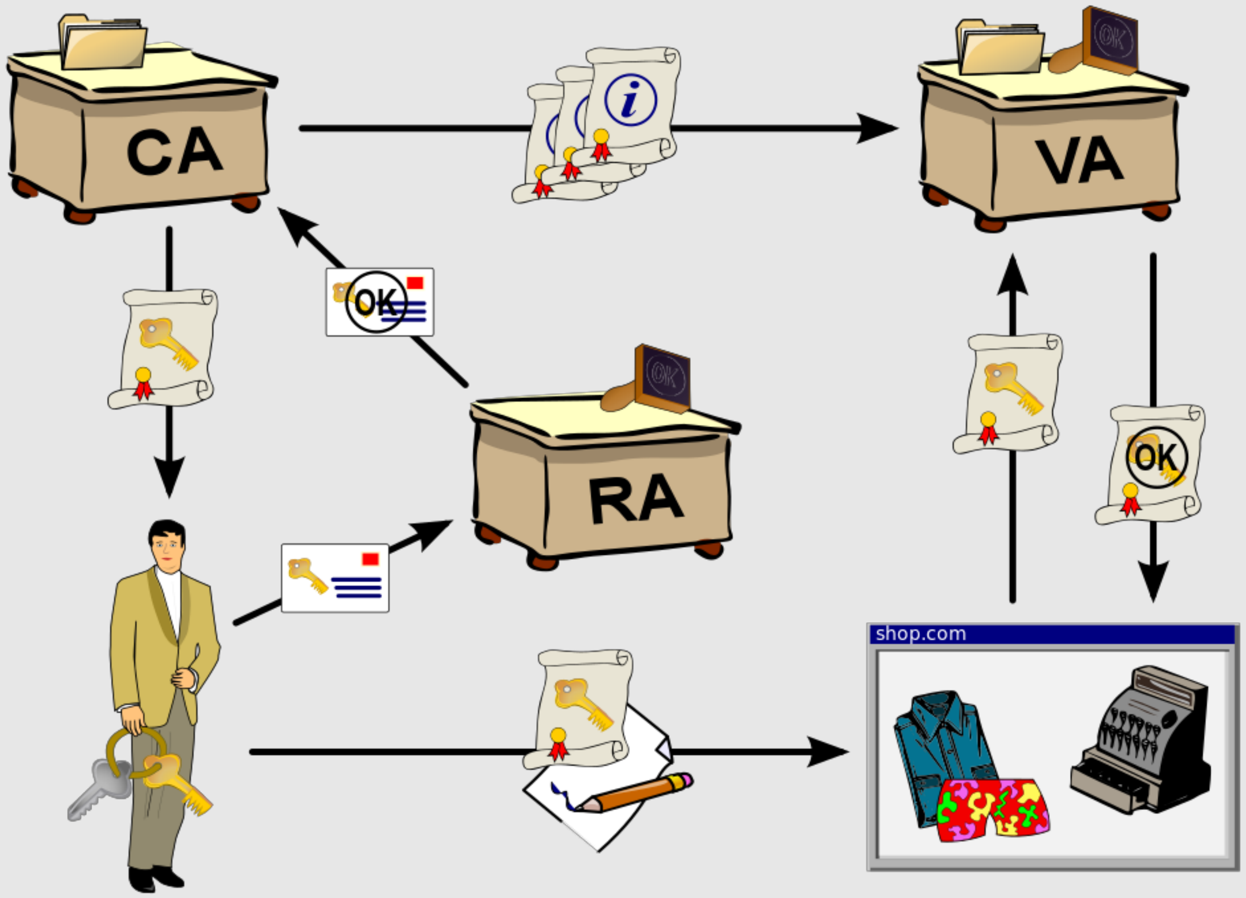
- Impossible to have a universal PKI with a single trusted CA.
- The basic trust relationships are all based on contractual relationships.

PKI Examples

- SSL/TLS
  - ✓ Web (HTTPS), VPN, email, ...
- Credit card organizations
- Enterprises, companies, organizations, ...

Operations

- Registration Authority (RA)
  ✓ verifies identities
- Certificate Authority (CA)
  ✓ issues certificates
- Validation Authority (VA)
  ✓ informs if a certificate is valid



Certificate

- Encoding of a particular data structure **must** be unique
  - ✓ X.509
- Fields
  - ✓ Subject: owner of the certificate
  - ✓ Issuer: issuer (CA) of the certificate
  - ✓ Not Before: the earliest time on which the certificate is valid
  - ✓ Not After: the latest time on which the certificate is valid
  - ✓ Public key: public key of the subject
  - ✓ Signature: signature of the certificate by the issuer's private key
- Other fields like serial number, key usage, algorithm id, ...
- Certificate can be extended for direct authorization.
  - ✓ Useful in a hierarchical CA structure, to limit the power of sub-CAs.

Multilevel Certificates

- For operational reasons certificates form chains
  - ✓ Root certificate (trust anchor)
    - self-signed certificate used for signing other certificates
  - ✓ Intermediate certificate
    - not self-signed, used for signing other certificates
  - ✓ Leaf certificate
    - cannot be used for signing other certificates

Certificate Revocation

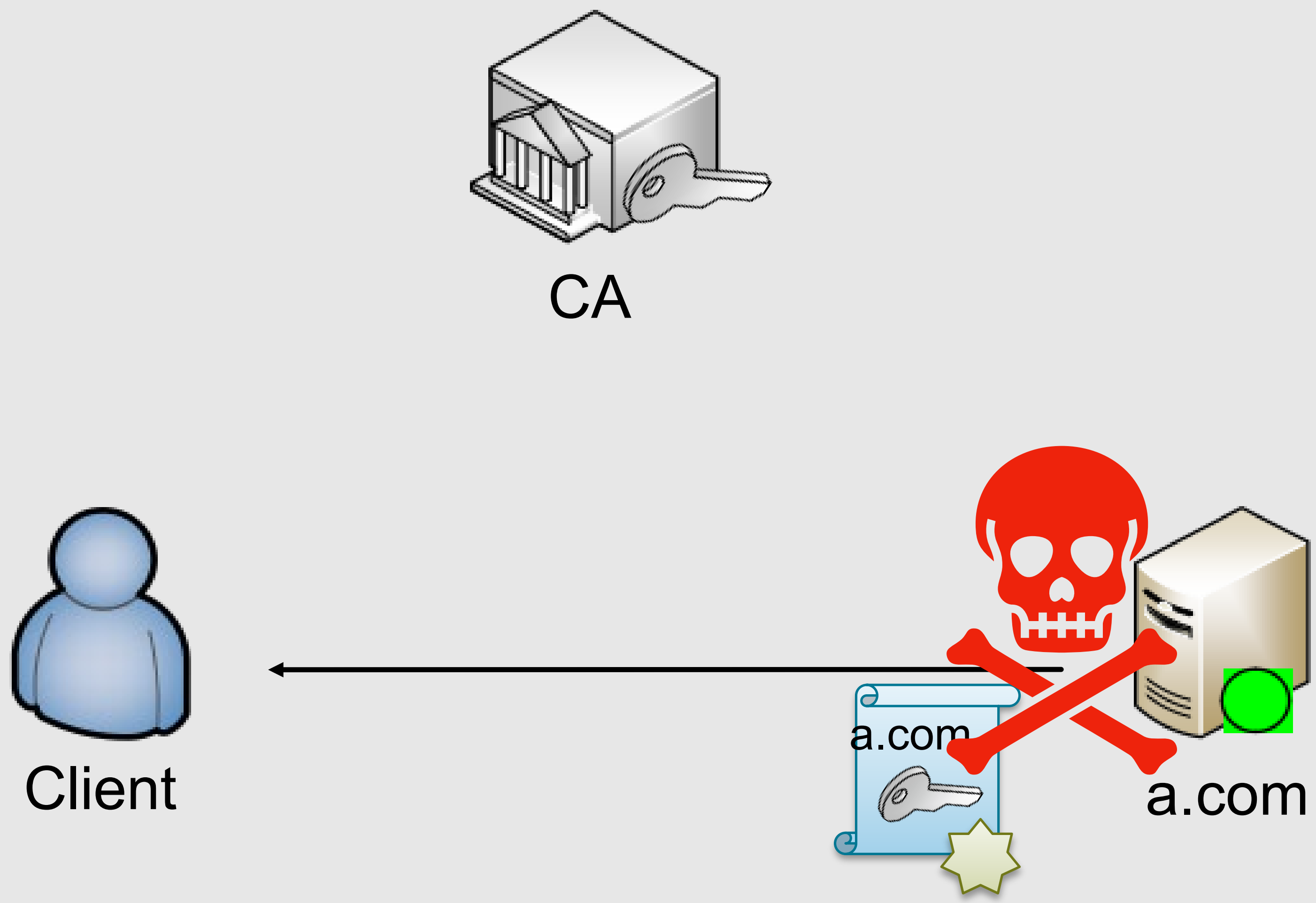
- Sometimes a certificate has to be invalidated (revoked) by the issuing CA.
  - ✓ Reasons for certificate revocation?
  - ✓ How to do this? (One of the hardest problems to solve in a PKI.)
  - ✓ What if root/intermediate certificate has to be revoked? (Collateral damage.)
- <u>Requirements</u>:
  - ✓ Speed of revocation
  - ✓ Reliability of revocations
  - ✓ Overheads
  - ✓ Connectivity
- Possible <u>solutions</u>: revocation list, online verification, fast expiration

Current SSL/TLS PKI Model

- CA knows whether the certificate is valid or not.

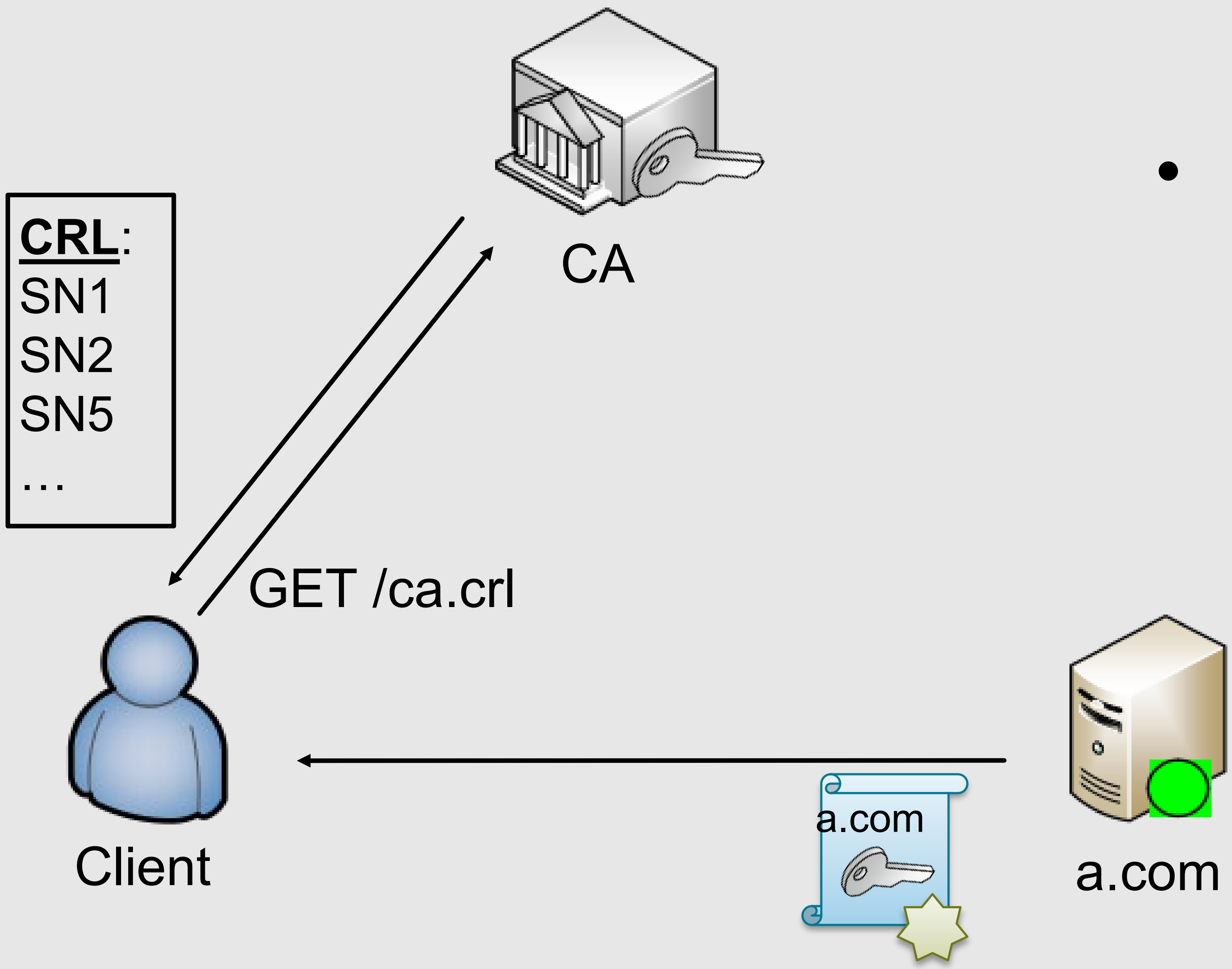
Is the certificate valid?

- *CA*
- *Name*
- *Signatures*
- *Expiration*
- ***Revocation***



Certificate Revocation List (CRL)

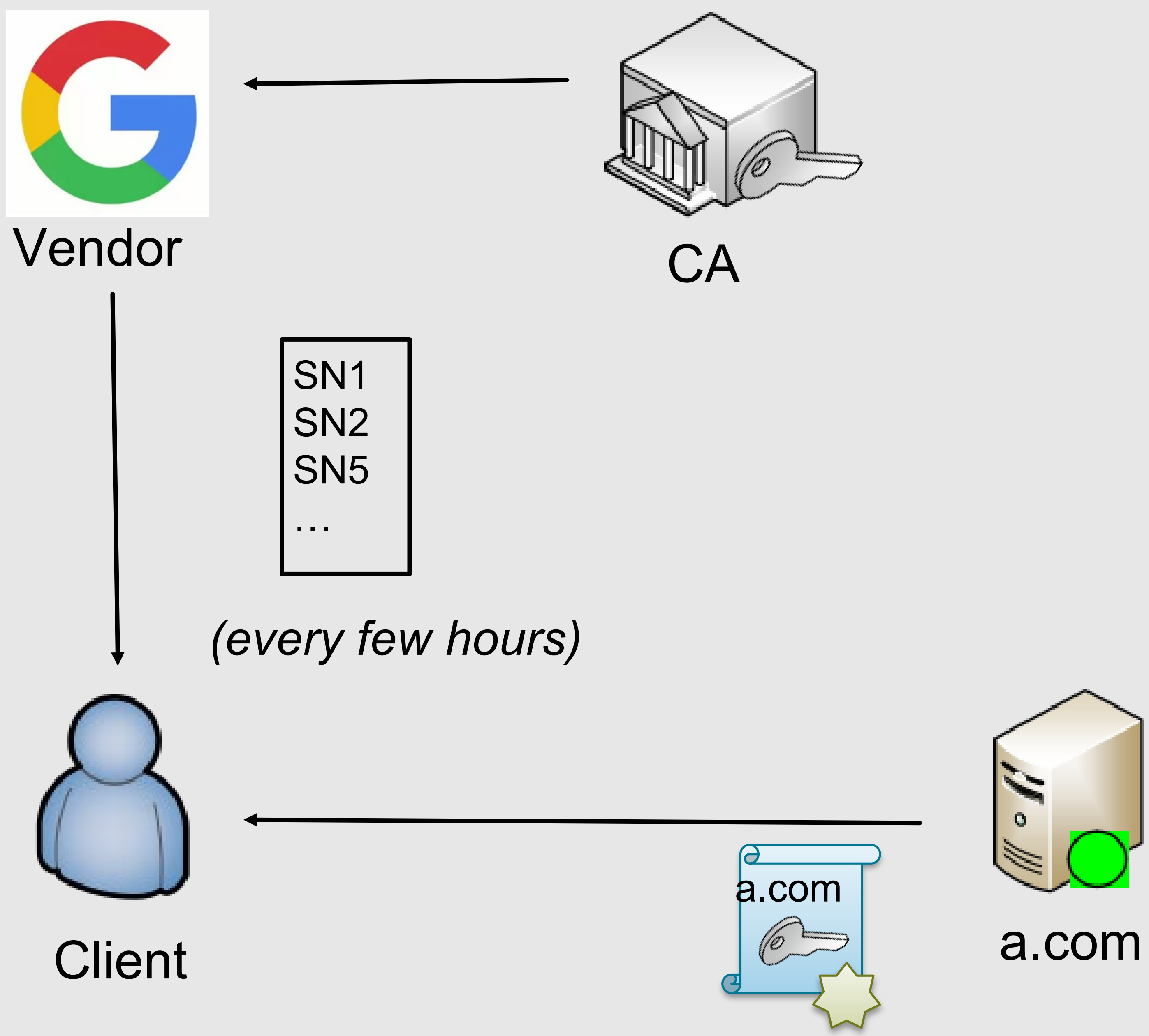
- CRL is a signed list of serial numbers that uniquely identify certificates being revoked.



- **Problems:**
  - ✓ CRLs are big (expensive)
  - ✓ Slow revocation after key compromise/loss

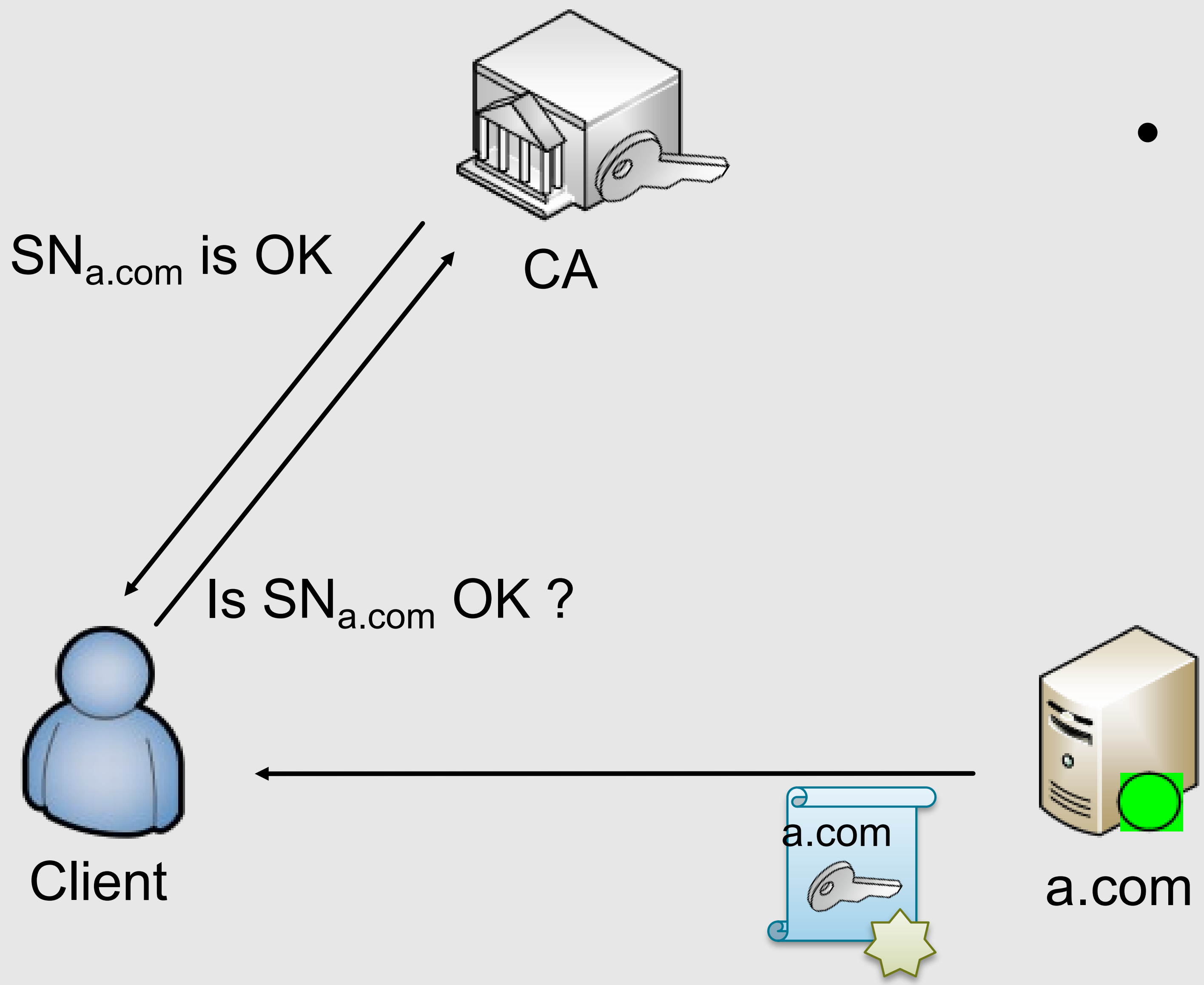
CRLSets

- CRLSets contains a carefully selected collection of revoked-certificate serial numbers published by many different certificate authorities.



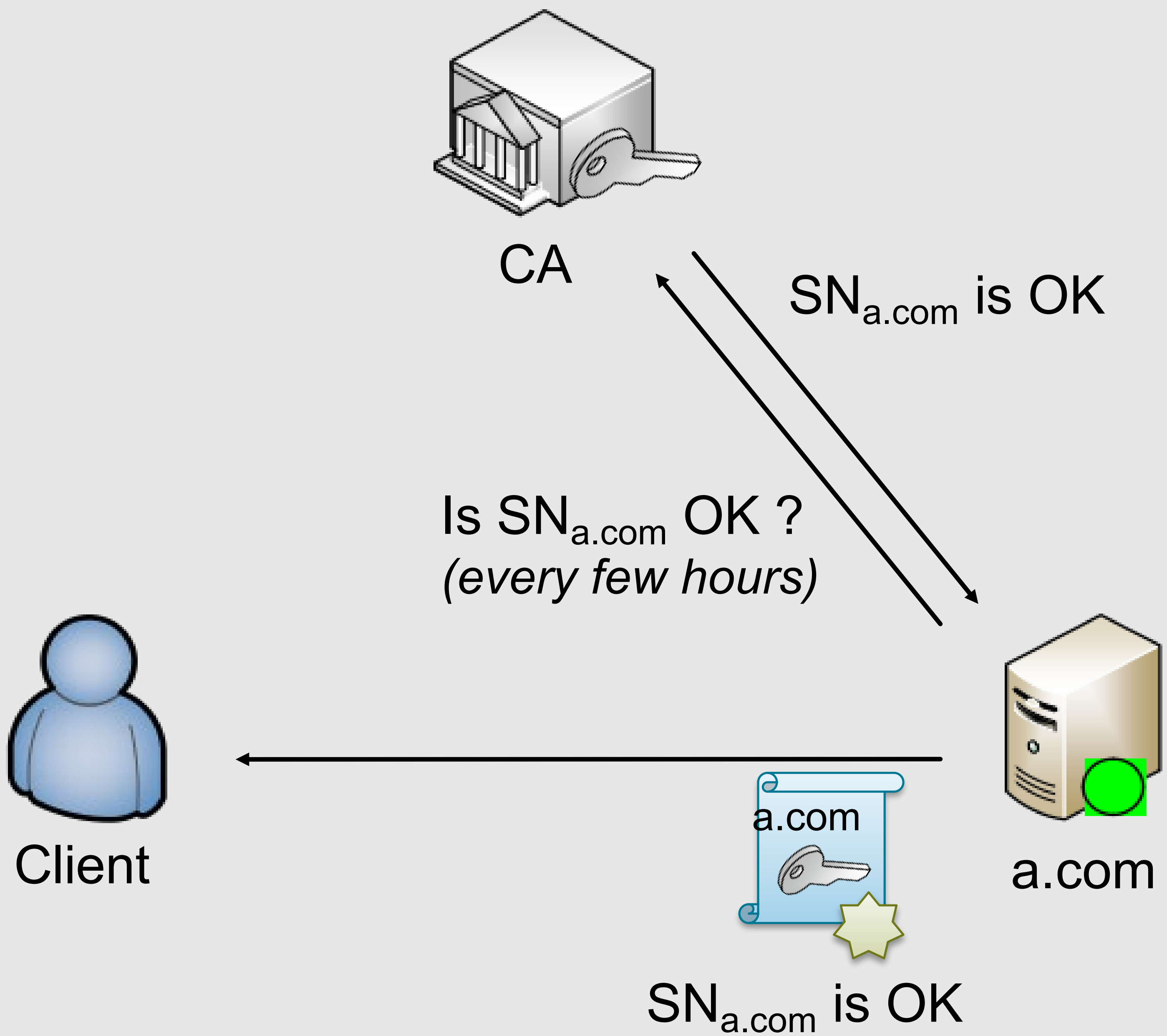
- **Problems:**
  - ✓ CRLSet is max 256KB
  - ✓ Only 0.35% of all revocations are included
  - ✓ Slow revocation after key compromise/loss

Online Certificate Status Protocol (OCSP)



- **Problems:**
  - ✓ Blocking connection (~350ms)
  - ✓ Availability (18% timeouts)

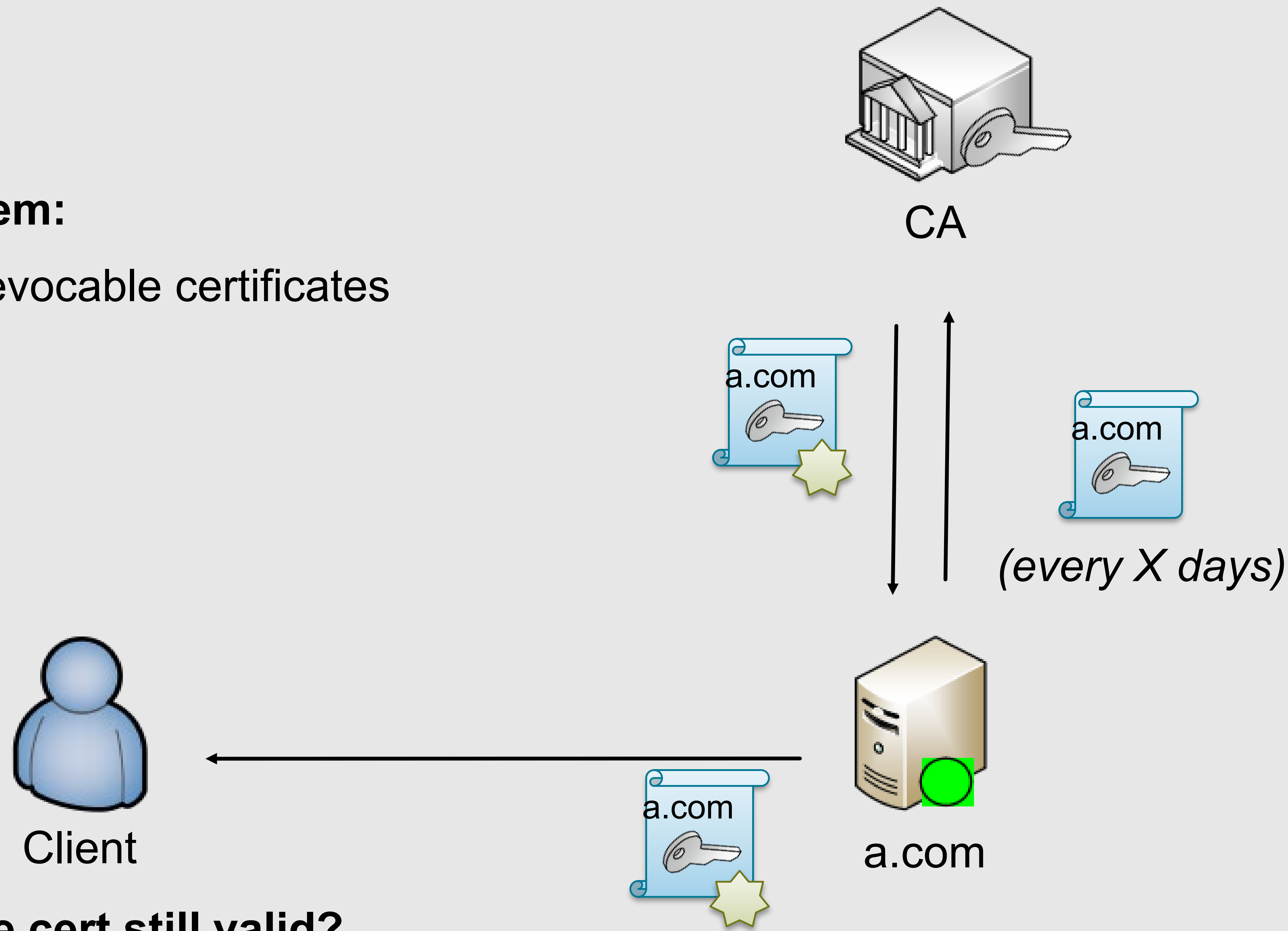
OCSP Stapling



- **Problems:**
  - ✓ Minimal server deployment < 3%
  - ✓ Slow revocation after key compromise/loss

Short-lived Certificates

- **Problem:**
  - ✓ Irrevocable certificates



Is the cert still valid?

Current State

		Desktop Browsers										Mobile Browsers			
		Chrome 44			Firefox	Opera		Safari	IE			iOS	Andr. 4.1–5.1		IE
		OS X	Win.	Lin.	40	12.17	31.0	6–8	7–9	10	11	6–8	Stock	Chrome	8.0
CRL															
Int. 1	Revoked	EV	✓	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	EV	✓	–	✗	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✗	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✓	✗	✗	✗	✗
OCSP															
Int. 1	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	L/W	✗	✓	✓	✓	✗	✗	✗	✗
Int. 2+	Revoked	EV	EV	EV	EV	✗	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Leaf	Revoked	EV	EV	EV	✓	✓	✓	✓	✓	✓	✓	✗	✗	✗	✗
	Unavailable	✗	✗	–	✗	✗	✗	✗	✗	A	✓	✗	✗	✗	✗
Reject `unknown` status		✗	✗	–	✓	✓	✗	✗	✗	✗	✗	–	–	–	–
Try CRL on failure		EV	EV	–	✗	✗	L/W	✓	✓	✓	✓	–	–	–	–
OCSP Stapling															
Request OCSP staple		✓	✓	✓	✓	✓	✓	✗	✓	✓	✓	✗	I	I	✗
Respect `revoked` staple		✗	✓	–	✓	✓	L/W	–	✓	✓	✓	–	–	–	–

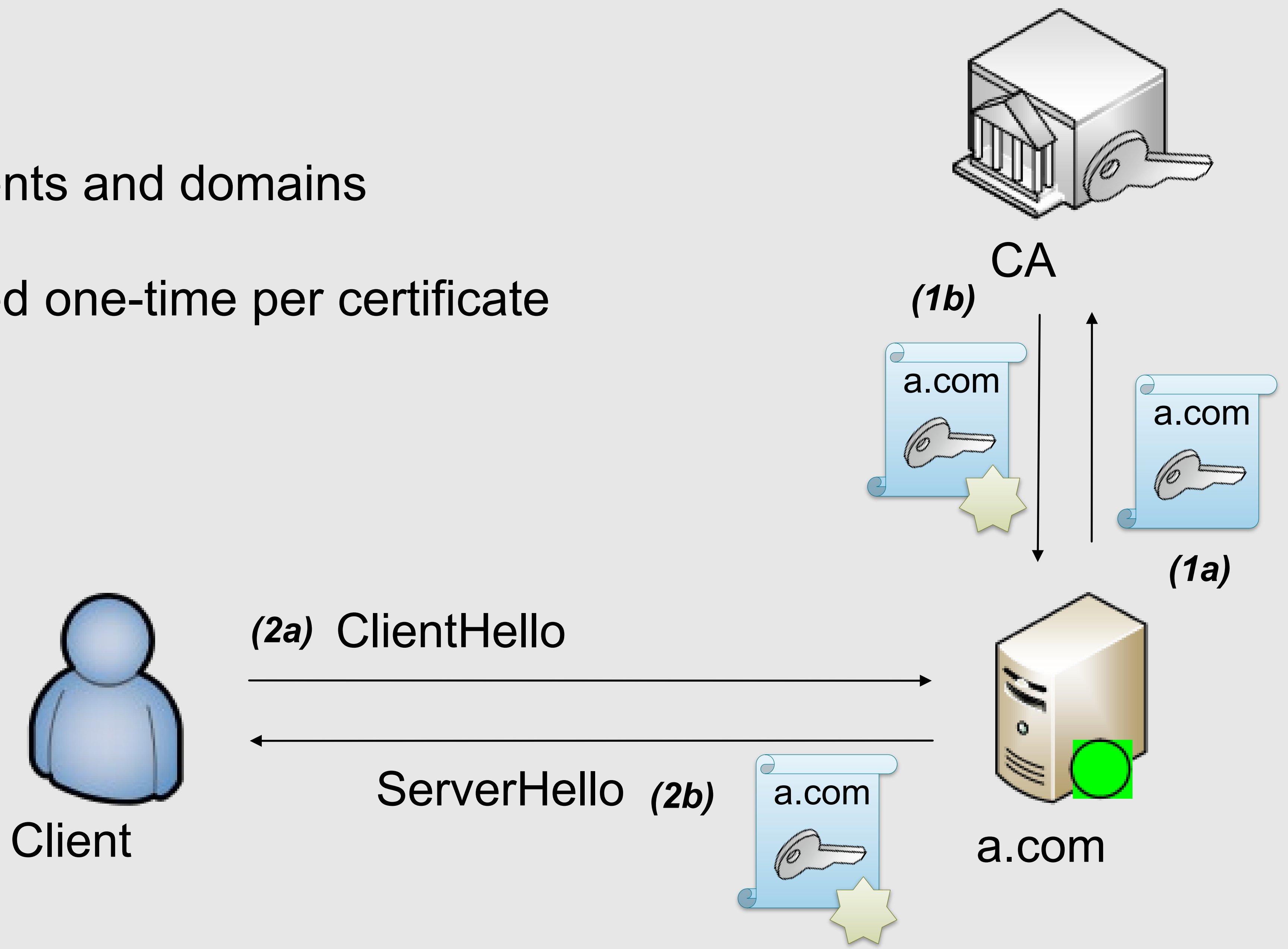
Table 2: Browser test results, when intermediate (Int.) and leaf certificates are either revoked or have revocation information unavailable. ✓ means browser passes test in all cases; ✗ means browser fails test in all cases. Other keys include EV (browser passes only for EV certificates), L/W (browser passes only on Linux and Windows), A (browser pops up an alert), and I (browser requests OCSP staple but ignores the response).

Certificate-Chain Validation

- The root CA certificate is trusted.
- All certificates are valid.
  - ✓ NotBefore < time() < NotAfter
  - ✓ Not revoked (if revocation is supported)
- The leaf certificate is issued for the contacted party.
- Certificates form a *chain of trust.*
  - ✓ 1st certificate is self signed, and *i*th certificate's issuer is *(i-1)*th certificate's subject.
  - ✓ 2nd certificate can be verified with the public key of the 1st one, 3rd certificate can be verified with the public key of the 2nd one,..., *i*th certificate can be verified with the public key of the *(i-1)*th.

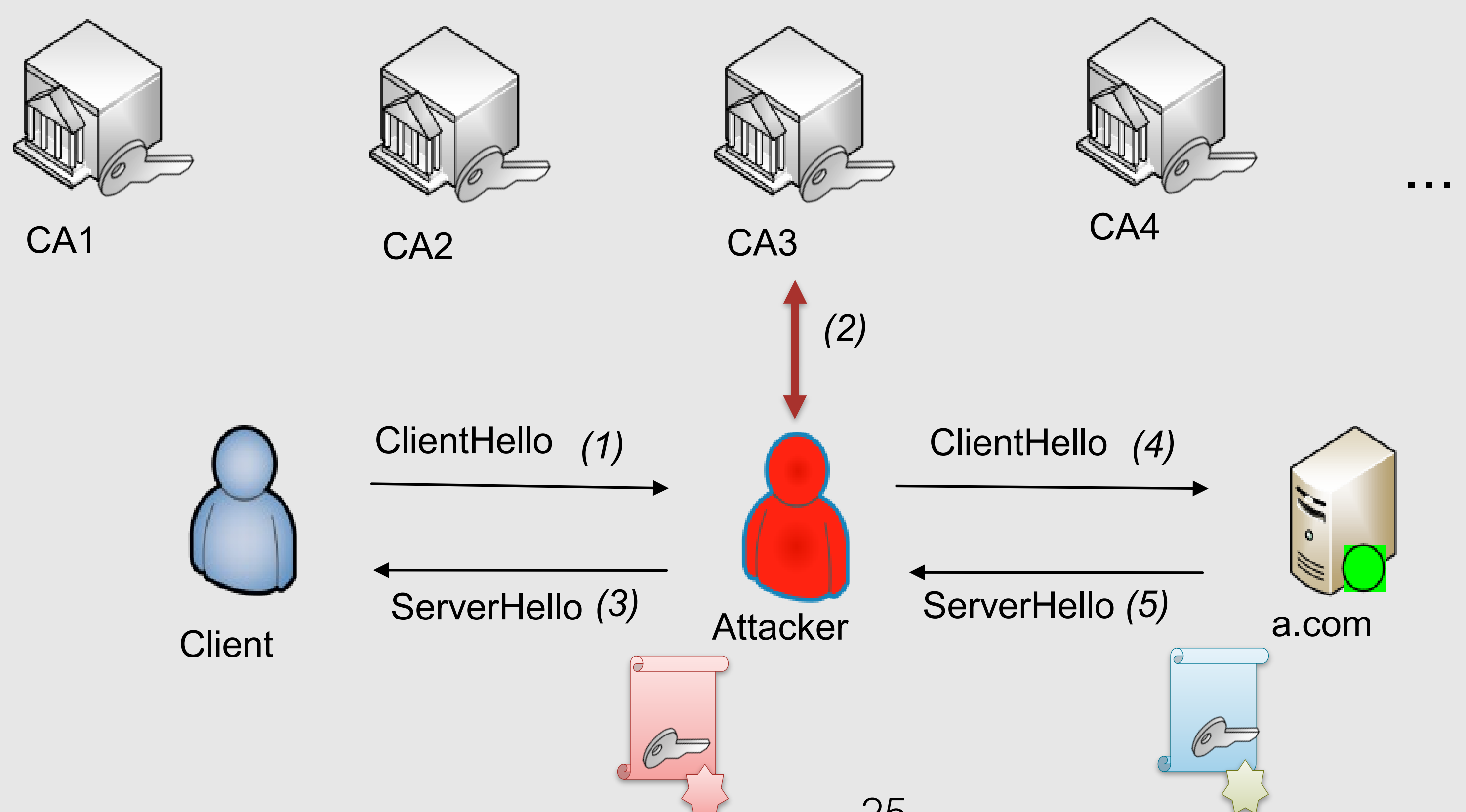
SSL/TLS PKI Model

- SSL/TLS Protocol
- CA is trusted by clients and domains
- Step *(1) is* performed one-time per certificate



SSL/TLS PKI: Weak Authentication

- Certificates signed by single CA
  - ✓ Currently, cannot sign certificate by multiple CAs.
- Weakest-link security with too many *trusted* entities
  - ✓ Current browsers trust ~1500 keys that can issue valid certificates.

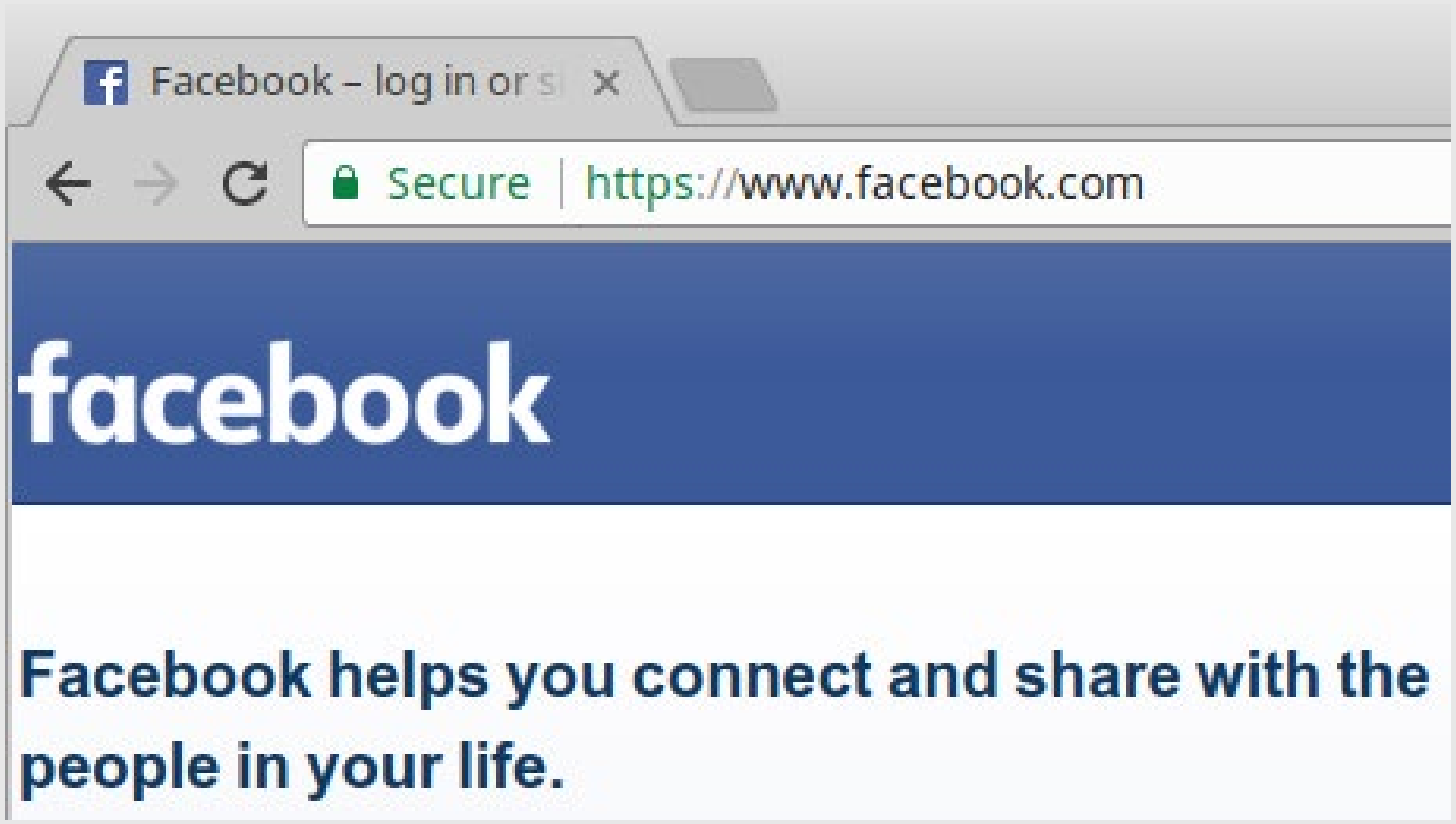


SSL/TLS PKI Problems

- Weakest-link security
- Revocation system is insecure and inefficient
  - ✓ Various schemes
  - ✓ Some CAs are *too-big-to-fail*
- Trust agility
  - ✓ Domains cannot state which CAs are trusted
- Transparency
  - ✓ CAs' actions are not transparent
- Imbalance
  - ✓ CAs have almost unlimited power
- Misconfigurations
  - ✓ SSLv2, weak crypto, NULL cipher suites

SSL/TLS as a Secure Channel

- Secure communication
  - ✓ Client-Server via HTTP**S**



PKI vs Key Server

- **Pros & Cons**
  - ✓ Key server requires everyone online in real time. For PKI, CA need not be online in real time (unless for OCSP).
  - ✓ Key server is a single point of failure. For PKI, CRL database is less security-critical and easier to distribute.
  - ✓ Key server distributes symmetric keys which cannot be used for non-repudiation. With PKI, private key owner cannot deny its digital signatures.
  - ✓ Key server needs the master key in online computer. For PKI, the root CA's private key is rarely used and need not in online computer.
  - ✓ PKI is much more complex than key server, and requires more computational power.

Key Points

- PKI addresses key management problem.
- Trust models (Monopoly, Oligarchy, Anarchy) will decide how PKI is established.
- Certificate revocation is the hardest problem to solve in PKI.

Exercises & Reading

- Classwork (Exercise Sheet 12): due on Fri Nov 30, 10:00 PM
- Homework (Exercise Sheet 12): due on Fri Dec 7, 6:59 PM
- Reading: FSK [Ch18, Ch19, Ch20]
- Final exam (Week 14): Fri 14 Dec, 7:00-9:30 PM (mainly covering Week 5 – Week 12, open-book but no Internet access)

Week 13 Presentations

Slides are available on eDimension.

1. Liu Bowen
   - ✓ **Title**: *Enter the Hydra: Towards Principled Bug Bounties and Exploit-Resistant Smart Contracts* (Usenix Security 2018)
2. Sakshi Sunil Udeshi
   - ✓ **Title:** Turning Your Weakness Into a Strength: Watermarking Deep Neural Networks by Backdooring (Usenix Security 2018)
3. Flavio Toffalini
   - ✓ **Title:** *The Guard's Dilemma: Efficient Code-Reuse Attacks Against Intel SGX* (Usenix Security 2018)
4. Tok Yee Ching
   - ✓ **Title:** *Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation* (NDSS 2018)

Review of Weeks 5-12

Topics

- **Week 5: Symmetric Encryption**
  - ✓ Attacks (COA, KPA, CPA, CCA) & security level
  - ✓ Block ciphers (AES)
  - ✓ Block cipher modes (ECB, CBC, OFB, CTR)

- **Week 8: Hash & MAC**
  - ✓ Merkle-Damgard construction
  - ✓ MD-based hash functions
  - ✓ Hash-based MACs (HMAC)
  - ✓ Cipher-based MACs (CBC-MAC, CMAC)

Topics

- **Week 9: Secure Channel & Randomness**
  - ✓ Order of authentication and encryption
  - ✓ Authenticated encryption (CCM, GCM)
  - ✓ Real randomness vs Pseudo-randomness

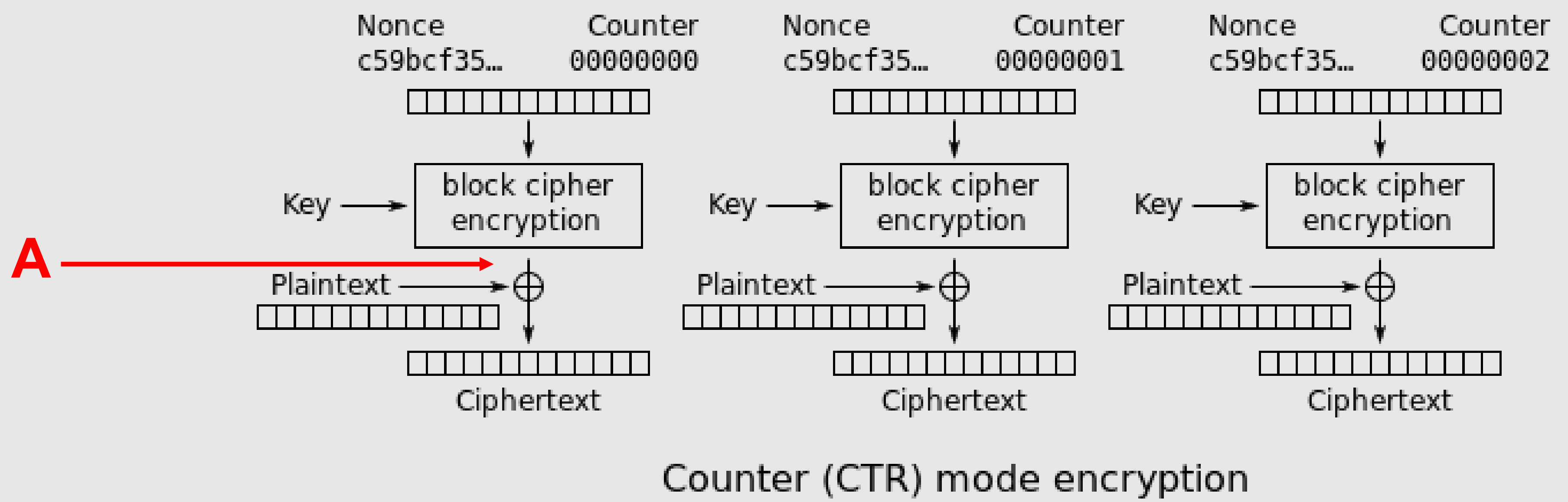
- **Week 10: Public-Key Cryptography**
  - ✓ Computations modulo a prime
  - ✓ (E)GCD algorithms
  - ✓ Cyclic groups & CRT
  - ✓ Diffie-Hellman algorithm
  - ✓ RSA algorithm

Topics

- **Week 11: Cryptographic Protocols**
  - ✓ Key negotiation (authenticated DH)
  - ✓ Key distribution (Kerberos)
  - ✓ Entity authentication (symmetric key based & public key based)
  - ✓ Non-repudiation (fairness, TTP involvement)

- **Week 12: PKI**
  - ✓ Trust models
  - ✓ Digital certificates & revocation

W5-HW6: An adversary observes the communication encrypted using CTR mode with the same fixed nonce. The nonce is hardcoded, so it is not included in the ciphertext. The adversary knows the 16-byte ciphertext ***C*** and ***C'***, and the plaintext ***P*** corresponding to C. What information, if any, can the adversary infer about the plaintext ***P'*** (corresponding to C')?



- Because using the same nonce,

$$P \oplus A = C$$
$$P' \oplus A = C'$$

- Then $P \oplus P' = C \oplus C'$

- If the adversary knows *C, C'* and *P* are know, it can easily recover *P'*.

W8-HW4: Suppose message *a* is one block long. Suppose that an attacker has received the ***MAC t*** for ***a*** using CBC-MAC under some random key unknown to the attacker. Explain how to forge the MAC for a two-block message of your choice. What is the two-block message that you chose? What is the tag that you chose? Why is your chosen tag a valid tag for your two-block message?

- As *a* is one-block message,

  $$MAC_K(a) = E_K(a) = t$$

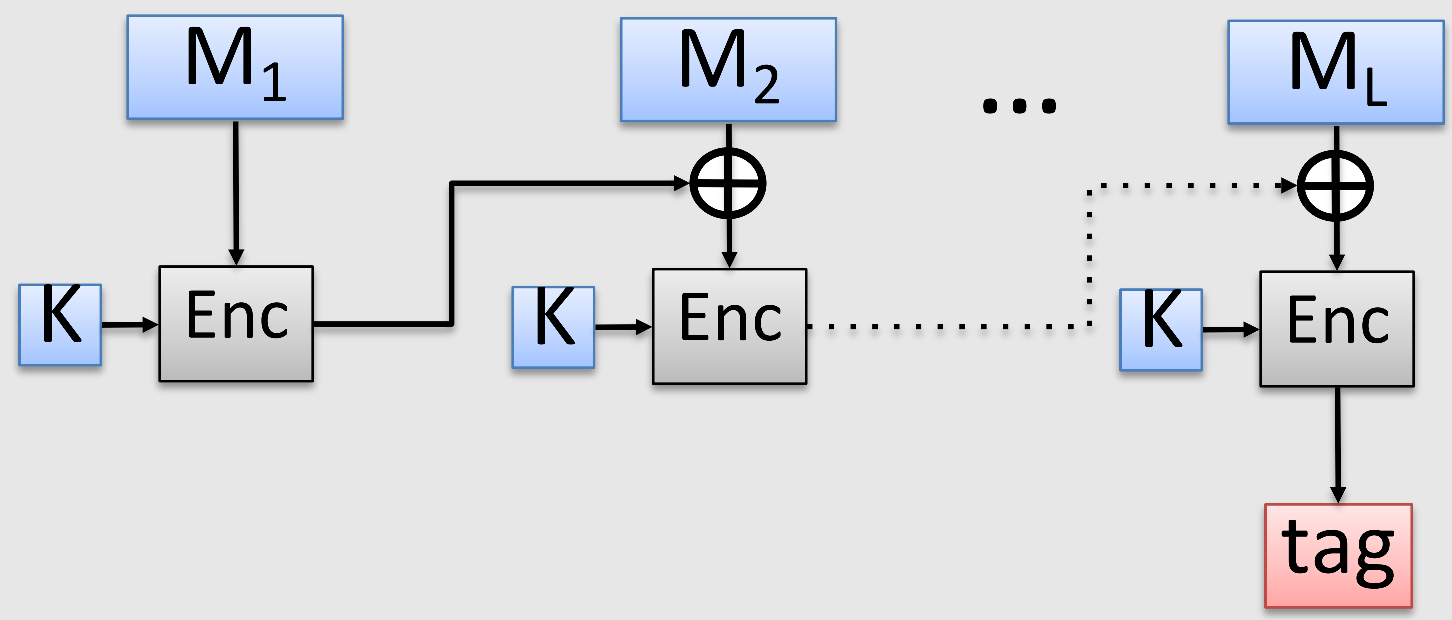
- The two-block message to be chosen is:

  $$M = a \,||\, (t \oplus a)$$

- The tag can be calculated as follows:

  $$\begin{aligned} MAC_k(M) &= MAC_k(a \,||\, (t \oplus a)) \\ &= E_K((t \oplus a) \oplus t) \\ &= E_K(a) \\ &= MAC_K(a) \\ &= t \end{aligned}$$

- The chosen tag *t* is valid for the chosen message *M* as shown in the above steps.



$C_1 = Enc(K, M_1)$
$C_2 = Enc(K, M_2 \oplus C_1)$
$C_3 = Enc(K, M_3 \oplus C_2)$
...
$C_L = Enc(K, M_L \oplus C_{L-1})$
$tag = C_L$

W10-HW1: Proof of *lcm(a,b) = ab/gcd(a,b)*

- By taking the prime power decomposition of *a* and *b*, we have

  $a = p_1^{c1} \times p_2^{c2} \times \ldots \times p_k^{ck}$

  $b = p_1^{d1} \times p_2^{d2} \times \ldots \times p_k^{dk}$

  where each of the p_i are distinct primes and each of the c_j and d_ℓ are non-negative integers.

- The important part of this trick is that we write both *a* and *b* as a product of the same primes, even if some of the powers are zero. By definition:

  $lcm(a,b) = p_1^{max(c1,d1)} \times \ldots \times p_k^{max(ck,dk)}$

  $gcd(a,b) = p_1^{min(c1,d1)} \times \ldots \times p_k^{min(ck,dk)}$

- We notice that $max(c_i,d_i) + min(c_i,d_i) = c_i+d_i$

  $lcm(a,b) \times gcd(a,b) = p_1^{c1+d1} \times \ldots \times p_k^{ck+dk}$

  $= (p_1^{c1} \times p_1^{d1}) \times \ldots \times (p_K^{cK} \times p_K^{dK})$

  $= a\ b$

- Therefore *lcm(a,b) = ab/gcd(a,b)*

W11-HW2: Fair Non-repudiation Protocol Using Off-line TTP

- **IEEE CSFW'97** (simplified)

1. A → B: C, EOO_C
2. B → A: EOR_C
3. A → B: K, EOO_K

IF 4. B → A: EOR_K **THEN** stop

ELSE {3'. A → TTP: K

4'. B ← TTP: K, con_K

5'. A ← TTP: con_K**}**

- K – message key defined by A
- C=Enc_K(M) – cipher text of M
- EOO_C, EOR_C – evidence of origin and receipt of C
- EOO_K, EOR_K – evidence of origin and receipt of K
- con_K – evidence of confirmation of K
- **Problem:** B may not be able to terminate the protocol run timely without breaching fairness.
- Further improve ?

- **Evidence**

✓ B receives **EOO_C, EOO_K** if TTP is not involved, otherwise **EOO_C, con_K**

✓ A receives **EOR_C, EOR_K** if TTP is not involved, otherwise **EOR_C, con_K**

End of Slides for Week 12