Introduction into Cyber Security

Chapter 5: Certificates and Public Key Infrastructures

WiSe 18/19

Chair of IT Security

Chapter Overview

- General public key problem
- Trust models
- X.509 certificates
- Certificate revocation

General Public Key Problem

- Public key primitives require an authenticated way to provide / exchange public keys:
  - Diffie-Hellman Key Exchange is vulnerable to a "man-in-the-middle" attack in which the attacker replaces the public DH values Alice and Bob exchange such that Bob ends up sharing a secret key with the attacker and Alice ends up sharing a secret key with the attacker
  - If an attacker can exchange Alice's public encryption key with his own, then he will be able to decipher confidential messages send to Alice
  - If an attacker can exchange Alice's public signature verification key, the attacker can convince others that Alice has signed certain messages.

Trusted Third Parties

- Guarantee authenticity with the help of a trusted third party:
  - Assumption: Alice and Bob are in possession of the public signature key of the trusted third party
  - The trusted third party signs
    - And thereby binds the pairs of public key and identifier
    - i.e. it *issues* certificates for the public keys of Alice and Bob
  - Bob can verify the authenticity of Alice's public key by checking the signature of the trusted third party on Alice's certificate
- A trusted third party that issues certificates is called Certification Authority

Certificates

- A certificate binds the public key of a principal to some identifier (or attribute) of that principal
- A certificate is issued by some issuer, typically called a certification authority (CA)
- A certificate minimally comprises
  - The public key
  - An identifier of the principal
  - The issuer
  - The signature on the hash of the rest of the content of the certificate with the private key of the issuer
- Note:
  - We will discuss a real-world certificate format later on
  - We denote a certificate issued by CA for some principal A by CA<<A>> in the following

Public Key Infrastructures

- Consists of the components to securely distribute public keys, these may include:
  - A certification authority that issues certificates for public keys
  - A repository for retrieving certificates
  - A method for revoking certificates
  - A method for evaluating “chains” of certificates
  - Sometimes a registration authority (RA) is used in addition to the certification authority
    - RA ensures the correctness of the binding between public key and principal and distributes the certificate (and the private key) to the principal
    - CA generates the actual certificate

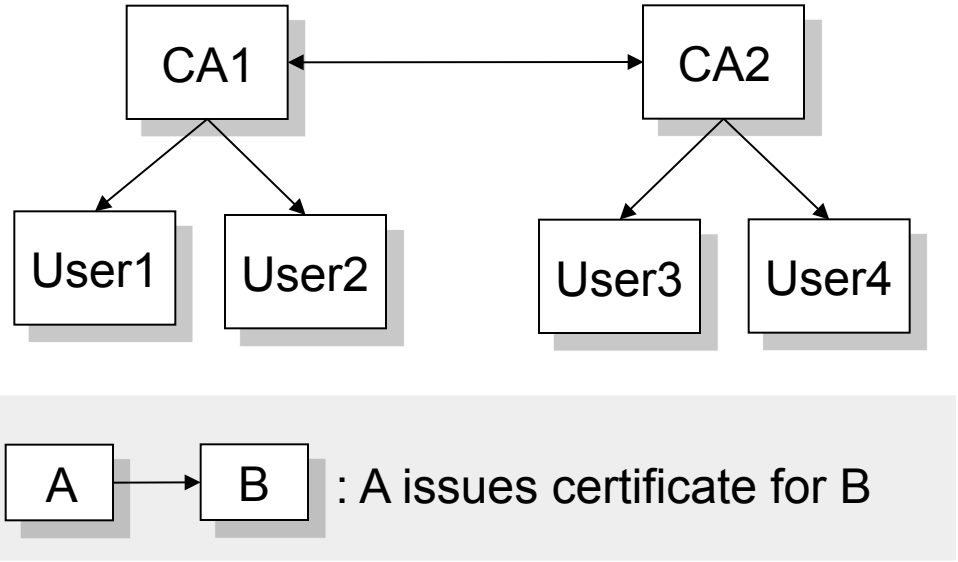
CAs and Trust

- The use of a CA requires
  - Alice and Bob to trust that the CA properly authenticates the principal before issuing the certificate
  - Alice and Bob to obtain the CA's public key in an authenticated manner
- Obviously a single world-wide CA is not realistic as
  - It would have to be trusted by any principal world-wide
  - Any principal would have to store the CA's public key
    - What if the CA's private key is broken and has to be exchanged?
  - Any principal would have to be able to obtain a certificate from that CA in a secure manner
  - …

Some Ways to Use More Than one CA

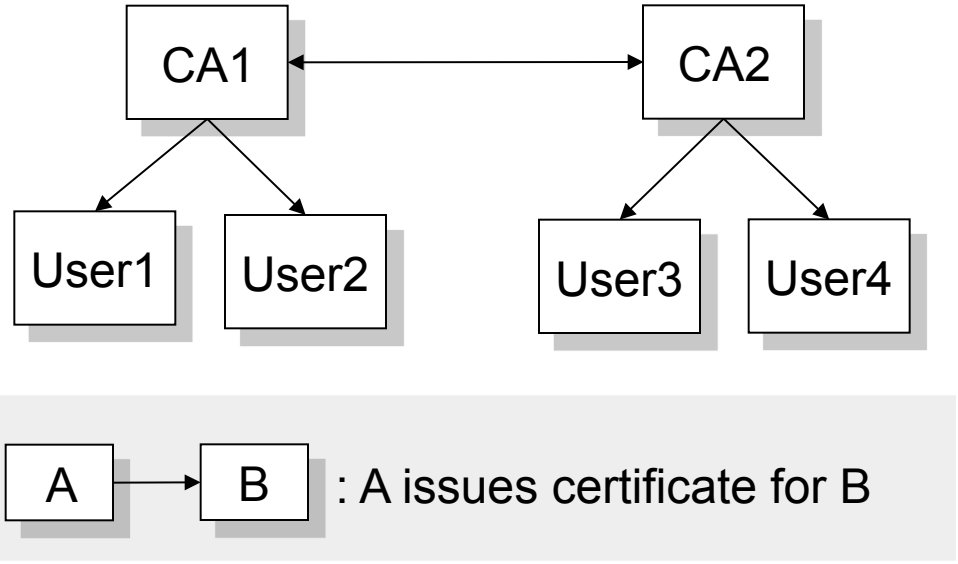
- Oligarchy: Several CAs are used simultaneously. All principles store the public keys of all CAs
  - Used in web browsers: check out the CA certificates pre-installed in your browser!!
- Cross certification: CA1 issues a certificate for CA2 and CA2 issues a certificate for CA1
- CA hierarchy: the CA's are hierarchically structured
  - A "root" CA issues certificates for the CAs on the second level of the hierarchy
  - The CAs on the second level of the hierarchy issue certificates for the CAs on the third level etc.
- Anarchy:
  - Everyone can sign everybody else's key
  - Everyone decides and configures whom they trust

Cross Certification



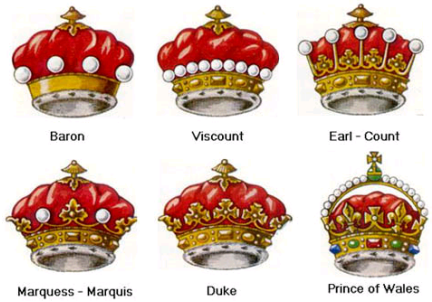
- How can User1 verify the certificate CA2<<User3>> of User3?

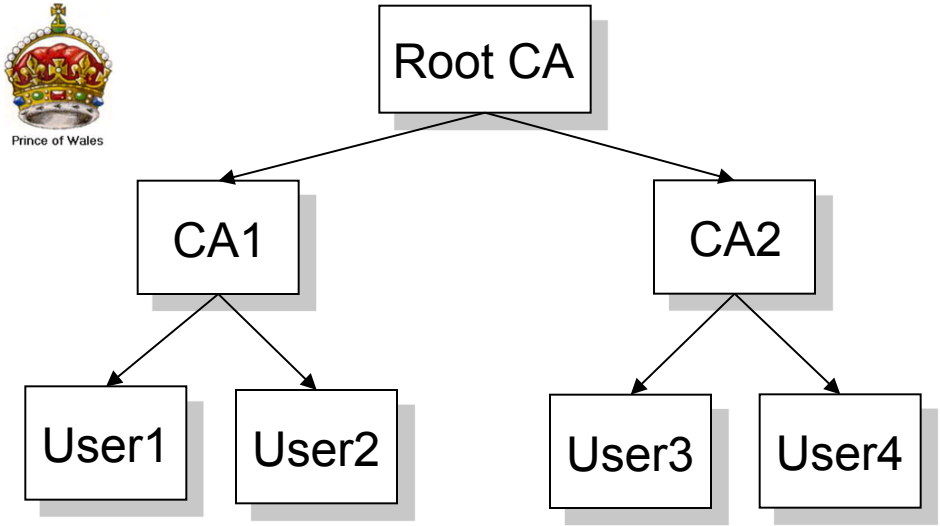
Cross Certification



- User3 presents User1 with the chain of certificates CA1<<CA2>> and CA2<<User3>>
- User1 verifies CA1's signature on CA2's certificate, extracts CA2's key and uses it to verify the signature on CA2<<User3>>

CA Hierarchies (Top down)

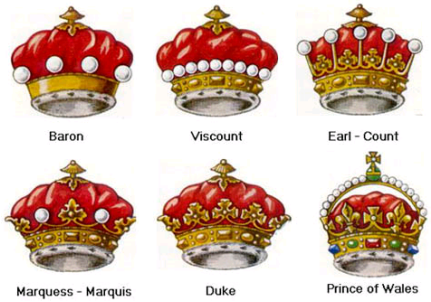


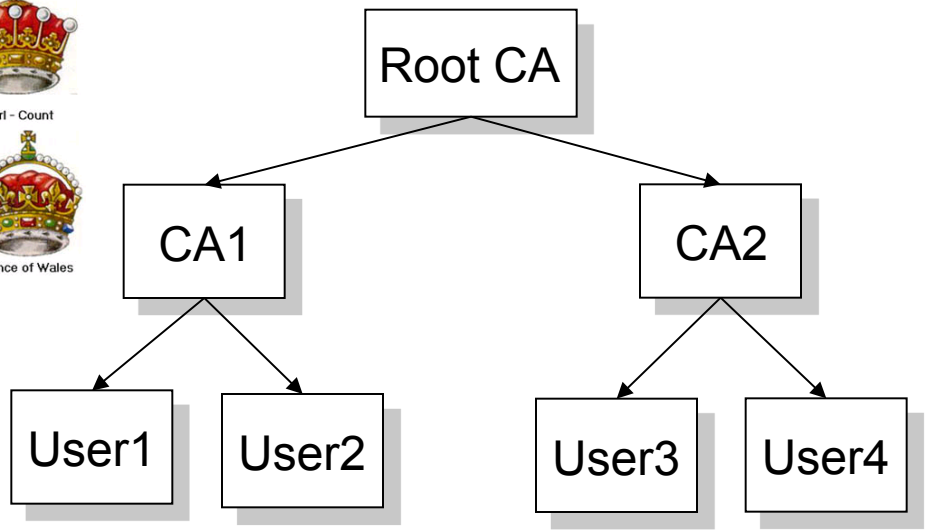


There can be more hierarchy levels of course

- How can User1 verify the certificate CA2<<User3>> of User3?

CA Hierarchies (Top down)

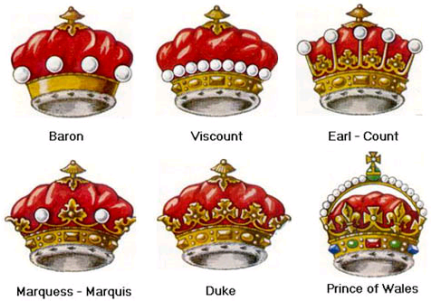


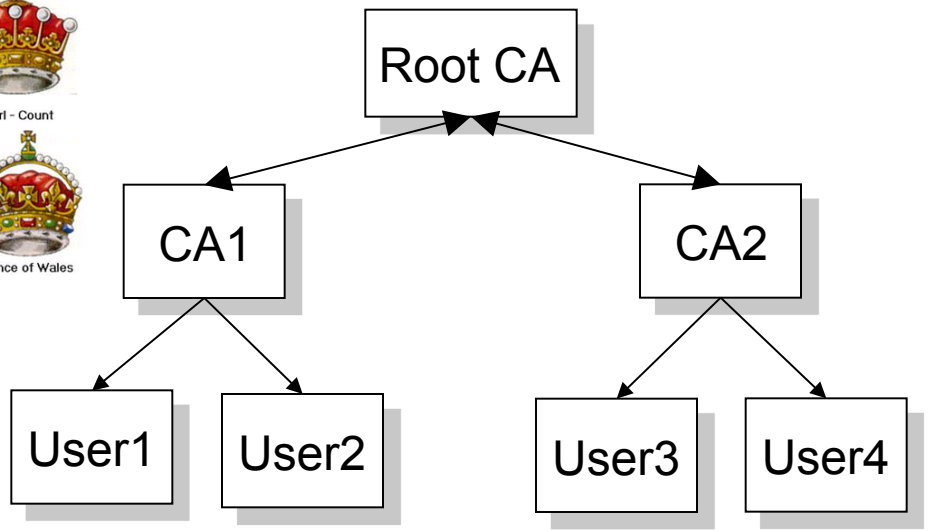


There can be more hierarchy levels of course

- User1 needs to be in possession of the public key of the Root CA
- User1 verifies the signature of the Root CA on CA2's certificate, extracts the public key of CA2 and uses it to verify the signature on User3's certificate

CA Hierarchies (Bottom up)

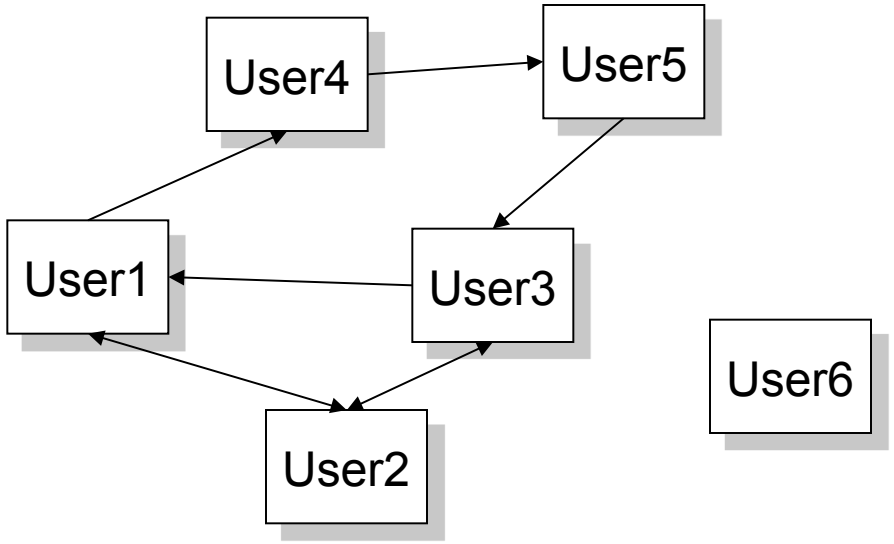




There can be more hierarchy levels of course

- The lower level CAs also sign the upper level CAs
- Advantage: User1 only has to know CA1's public key in advance
- Disadvantage: User1 has to check the signatures on more certificate

Anarchy: e.g. used in PGP



- Obviously hard to find chains if used by a large group
- Left to the choice of user's whom to trust

Certificate Revocation

- Certificates are typically issued for a certain validity period
- However, the private key corresponding to the public key in the certificate may be compromised before the validity period ends
- It is often required to be able to revoke a certificate before it expires
- Consequence: the verifier of a certificate needs to check if
  - the signature(s) on the (chain of) certificate(s) is/are correct
  - the certificate(s) is/are still valid or are already expired
  - the certificate(s) has/have not been revoked e.g. by fetching the appropriate Certificate Revocation Lists

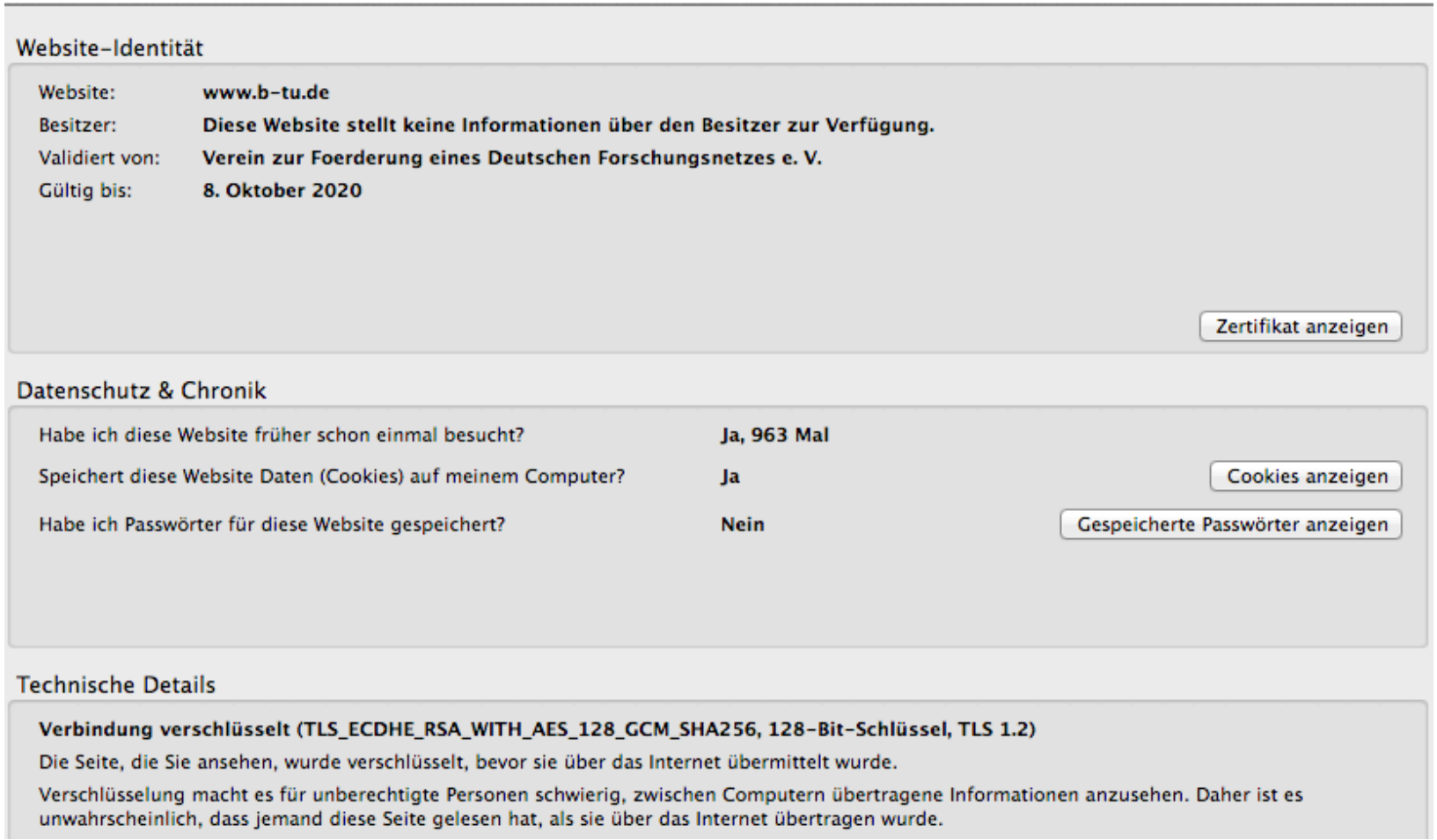
Certificate Revocation with CRLs

- A Certificate Revocation List (CRL) is issued periodically by the CA
- A CRL is a signed list of all revoked certificates which validity period has not expired yet
- The CRL is signed with the private key of the CA

X.509 Certificates and CRLs

- ITU-T standard for public key certificates and certificate revocation lists
- Profile for use on the Internet defined in RFC 5280
- Used e.g. for
  - SSL/TLS in web browsers
  - IPsec
  - In various authentication protocols

Example: TLS with b-tu.de



X.509 Certificates

- Signed Content:
  - Version – The version number
  - Serial Number – An issuer-unique serial number
  - Signature – The signature algorithm identifier (includes hash function)
  - Issuer – Name of the issuer
  - Validity – from: / to: validity period
  - Subject – Name of the subject
  - SubjectPublicKeyInfo – Public key
  - IssuerUniqueID – Unique identifier of the issuer (optional)
  - SubjectUniqueID – Unique identifier of the subject (optional)
  - Extensions
- Additional unsigned content
  - SignatureAlgorithm – Algorithm identifier
  - SignatureValue – Signature on the hash of the content

Examples for Extensions in X.509

- Key Usage – defines the purpose of the key (e.g. signature verification or encryption)
- Authority Key Identifier – e.g. issuer name, serial number of certificate corresponding to the key with which the certificate was signed
- Subject Alternative Name – may be used in addition or instead of the Subject field. May include e.g. an email address, a DNS name, an IP address etc.
- Name Constraints – may be used in CA certificates only. Describes constraints on the subject names for which a CA can issue certificates
- …

X.509 CRLs

- Signed content
  - Version - if present must be v2 or v3
  - Signature - Signature algorithm identifier (includes hash function)
  - Issuer - Name
  - thisUpdate - Time
  - nextUpdate - Time (optional)
  - revokedCertificates - List of revoked certificates
    - userCertificate - Certificate serial number
    - revocationDate - Time
    - crlEntryExtensions - optional extensions
  - CrlExtensions - optional
- SignatureAlgorithm
- SignatureValue

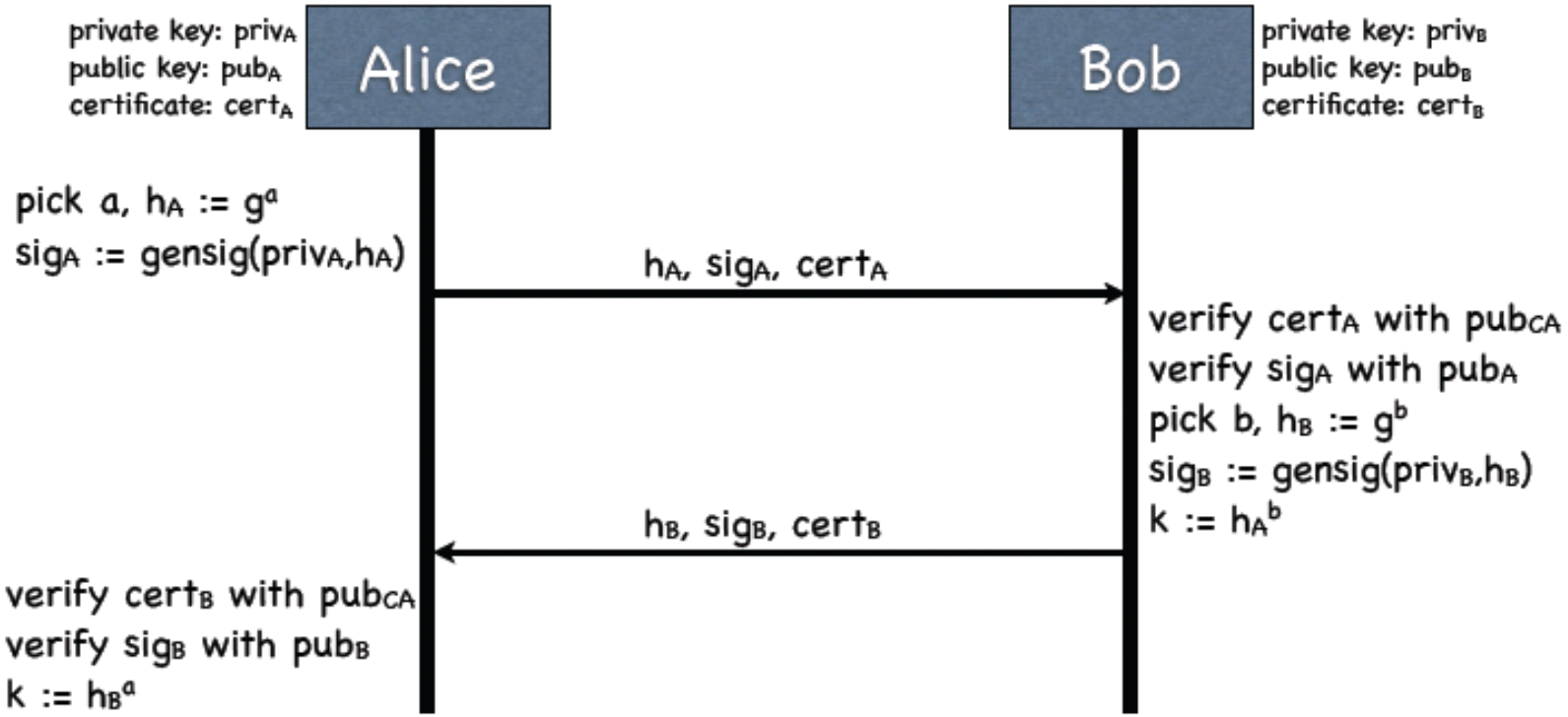
Online Certificate Status Protocol

- The disadvantage of CRLs is that revocation is only as timely as the period used for issuing CRLs
- The OCSP allows applications to explicitly query if certificates are still valid
  - OCSP client sends status request to OCSP responder
    - Includes list of serial numbers of certificates
  - OCSP responder replies with the status for all certificates in the list
    - The status is one of: good, revoked, unknown
    - The response is signed by the responder
- OCSP only defines the content of messages, not their format as OCSP can be carried over LDAP, HTTP, SMTP, etc.

Additional PKI-related Protocols

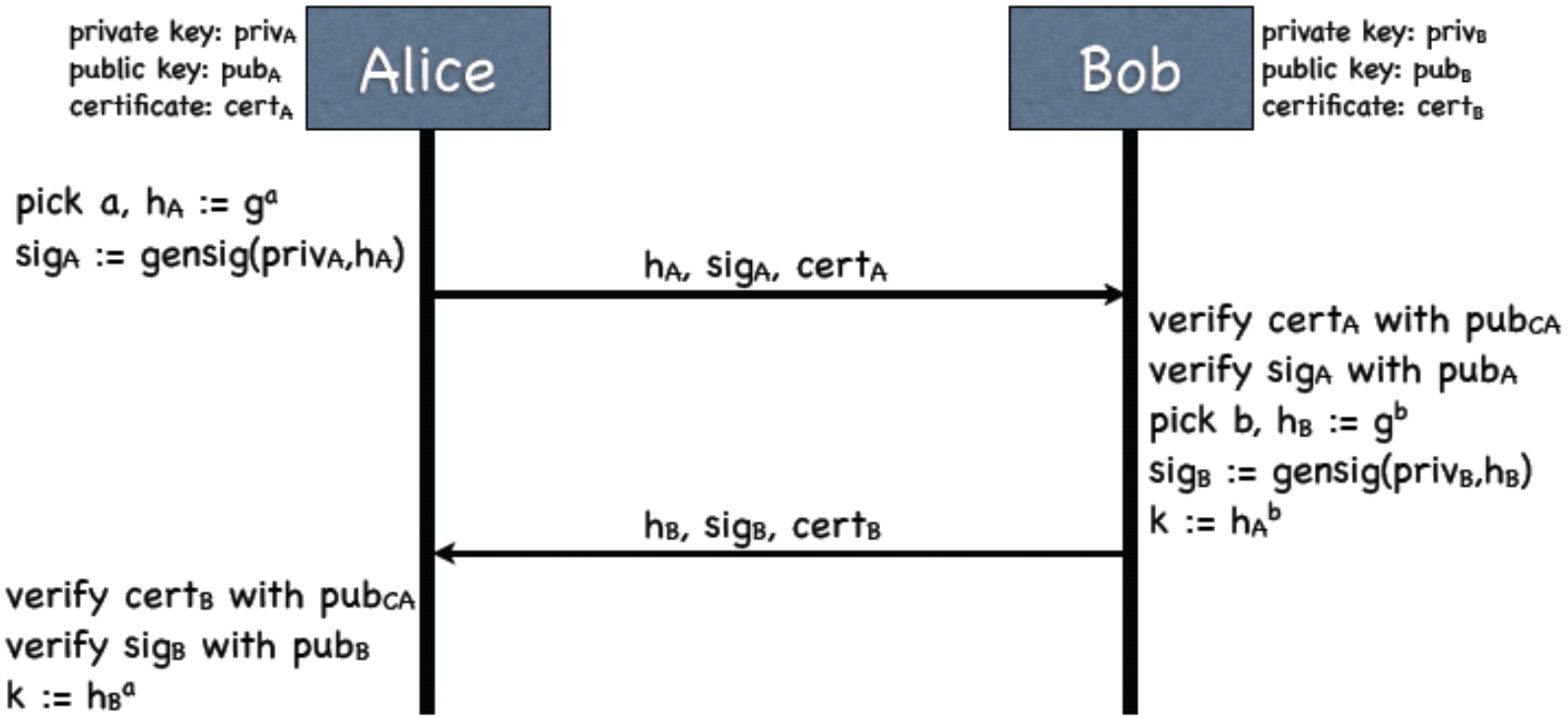
- Lightweight Directory Access Protocol
  - Protocol that specifies how to retrieve certificates and CRLs from a repository storing that information
  - Specifies message formats for
    - reading entries, searching for entries (all clients)
    - adding/deleting entries in the repository (CA only)
- Certificate Management Protocol
  - Specifies message formats for certificate creation, initial distribution, and management
- Certificate Request Message Format
  - Specifies message formats for requesting a certificate for a particular public key from a CA

Using Certificates to Protect Diffie-Hellman



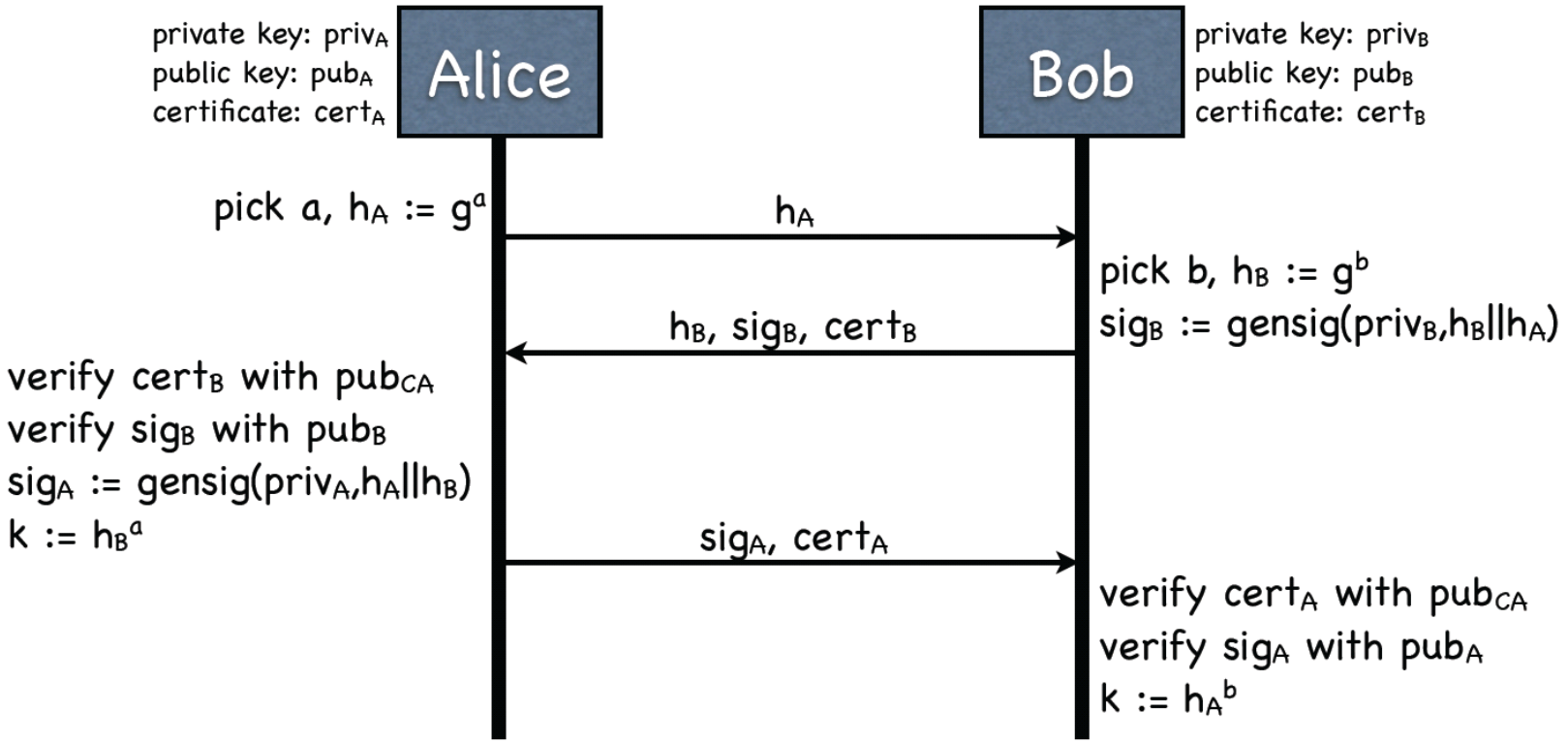
- Problem?

Using Certificates to Protect Diffie-Hellman – First Try



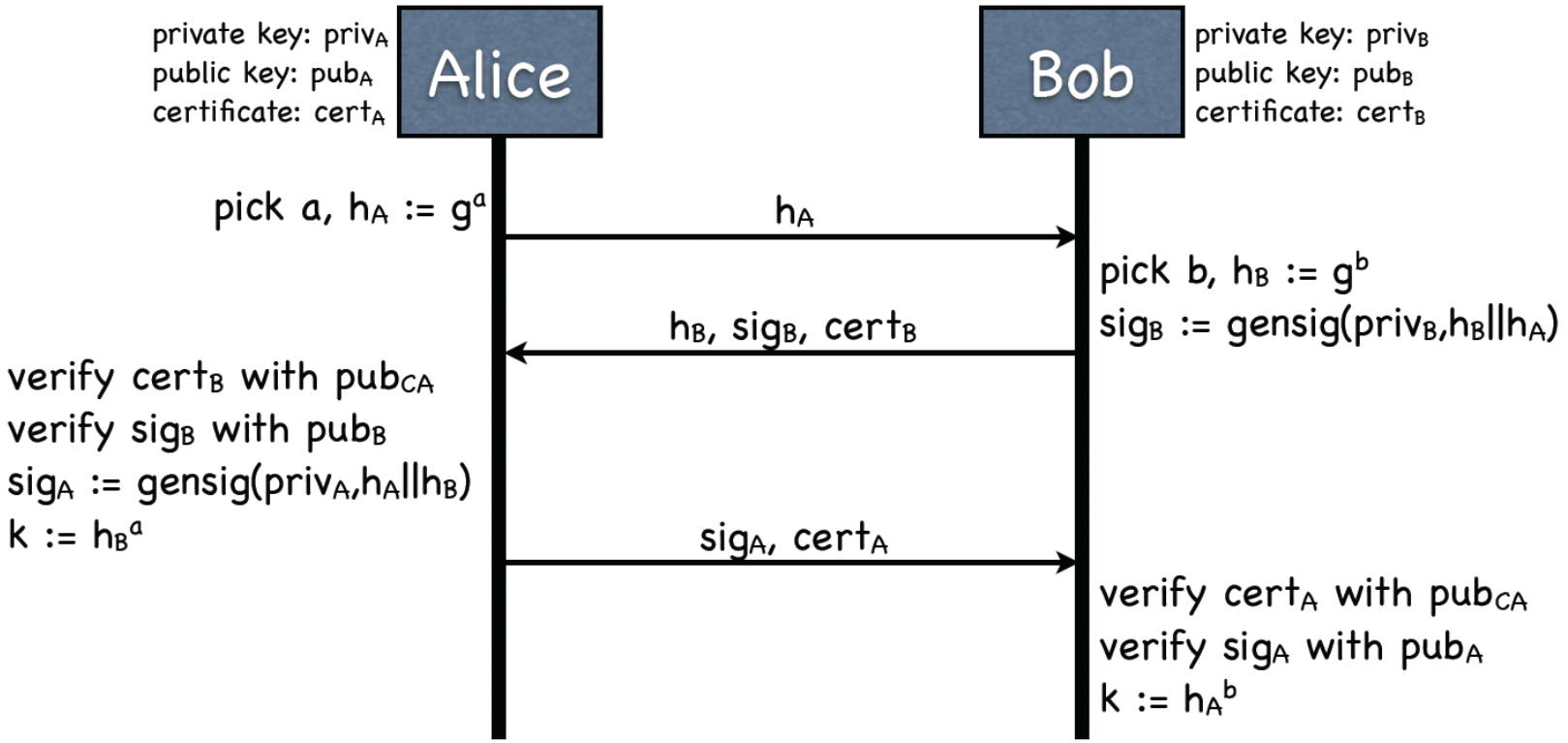
- The signatures are not bound to THIS particular run of the protocol
- An attacker could record e.g. Alice's message and replay it to Bob

Authenticated Diffie Hellman – Second Try



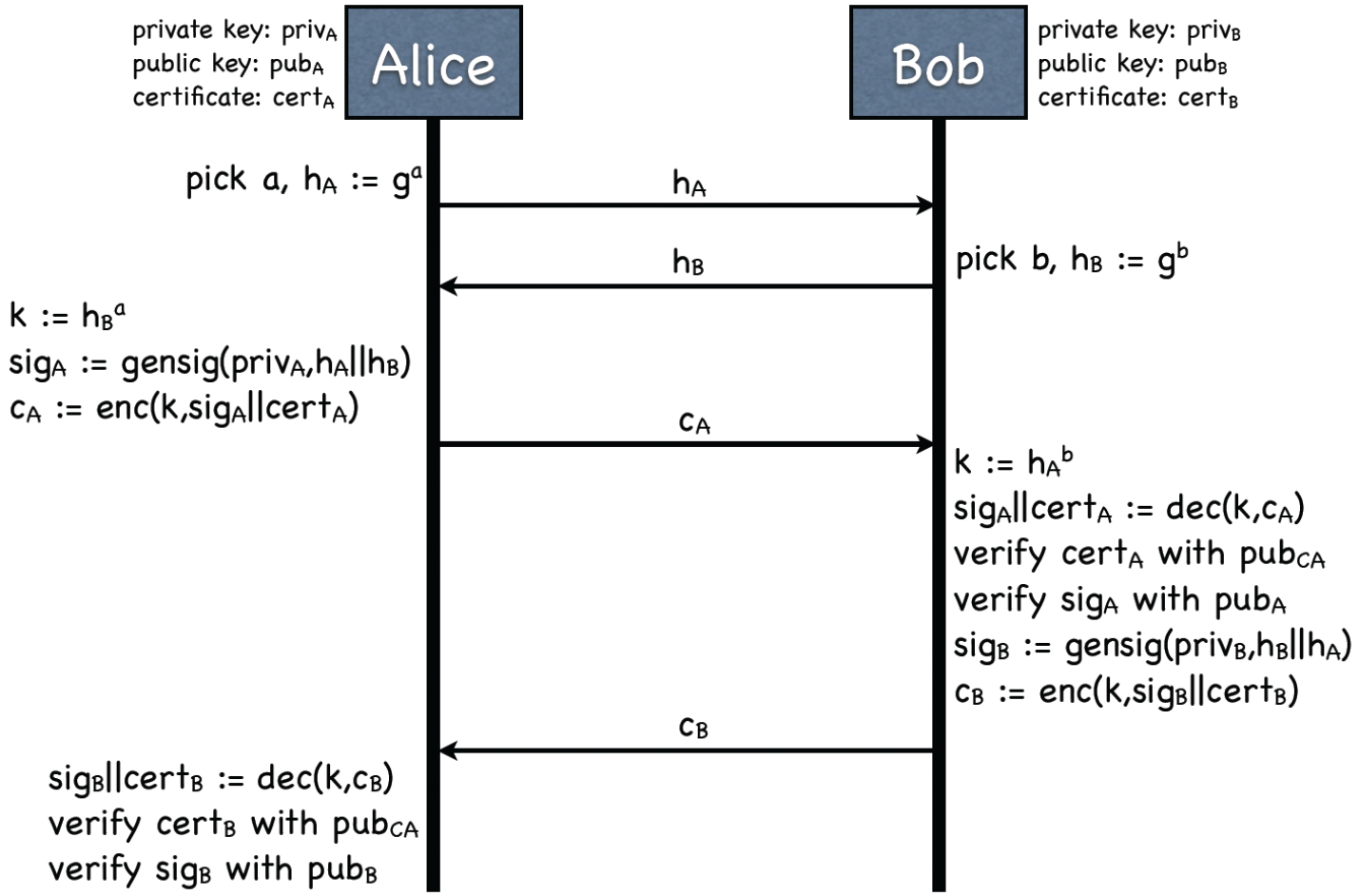
- Both parties sign the public DH values received from each other
- Both parties use public key certificates to exchange their public signature keys
- Now replay is not possible because Alice can check if Bob correctly received her public DH value and vice versa

Authenticated Diffie Hellman – Second Try



- Problem: Attacker Eve can make Alice believe that she is exchanging a key with Eve and not Bob
  - Eve just has to exchange the Bob's public key and signature with her own

Secure Authentic Diffie-Hellman



- Bob and Alice confirm that they both agreed upon the correct key by using it to encrypt their certificate and their signature

So…

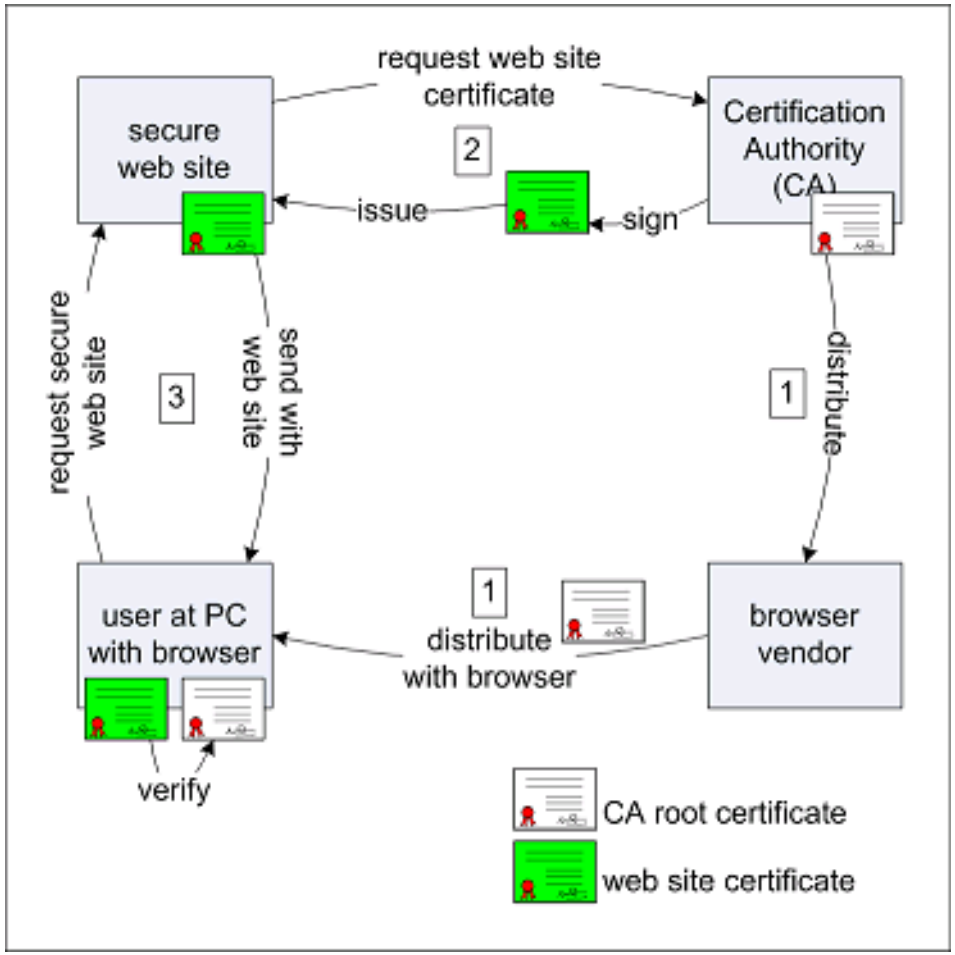
- … certificates can be used to distribute public keys in an authenticated way
- The content of a certificate is hashed and then signed with the private key of a certification authority
- In Chapter 3 we discussed hash functions and saw that a tools are available online that find collisions in MD5
- What are the consequences of the MD5 attack for faking certificates?

Typical Way of Obtaining Certificates

- User creates private key
- User creates a Certificate Signing Request
  - Containing
    - User identity
    - Domain name
    - Public key
- CA processes the Certificate Signing Request
  - Includes
    - Validating user identity
    - Validating domain ownership
    - Signs and returns the certificate
- User installs private key and certificate e.g. on a web server

Distribution of Web Site Certificates

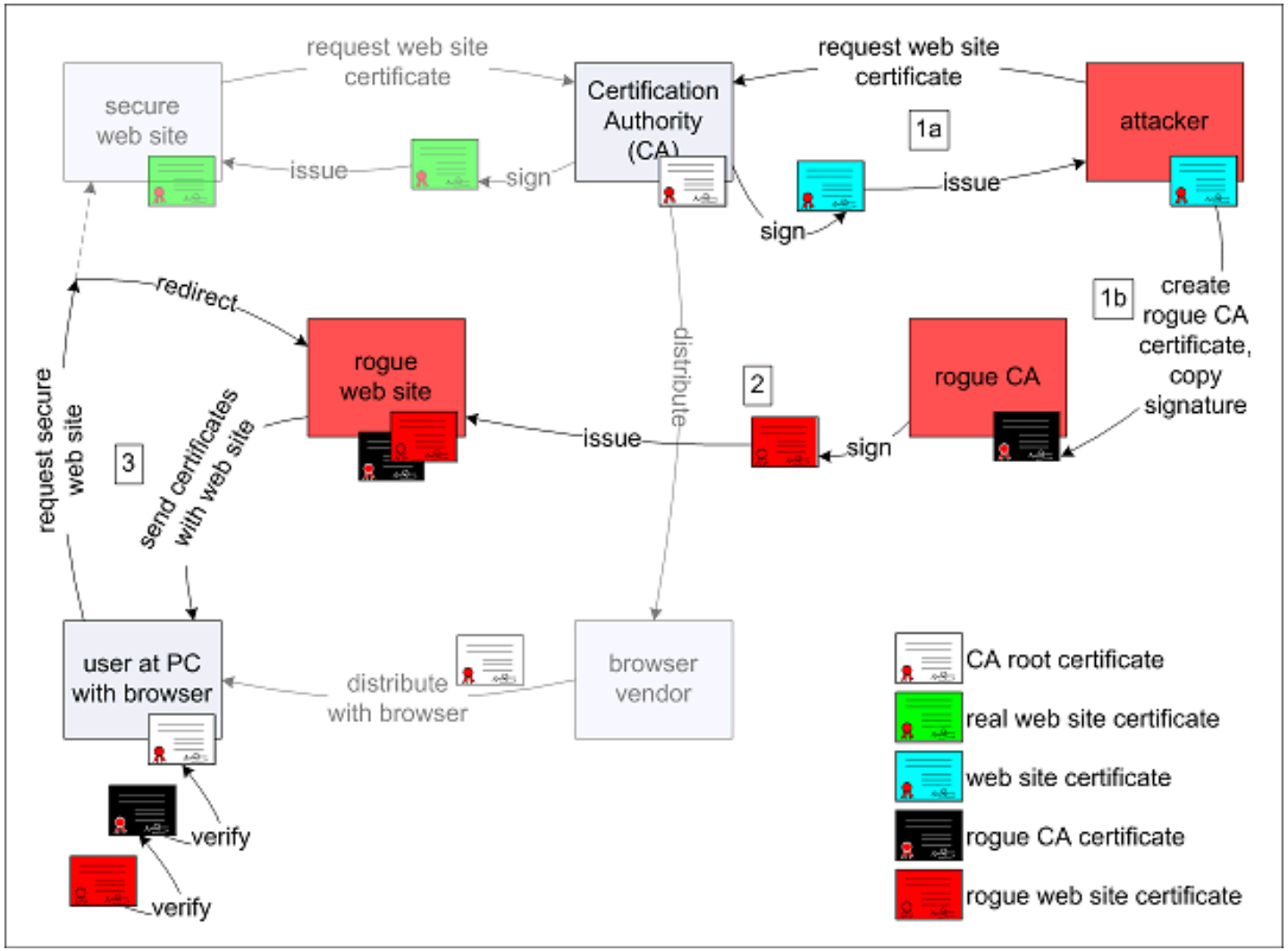
- CA root certificates are distributed via browser vendors
- Web servers can request certificates from a CA
- Users can verify certificate presented by web site with the pre-installed CA certificate



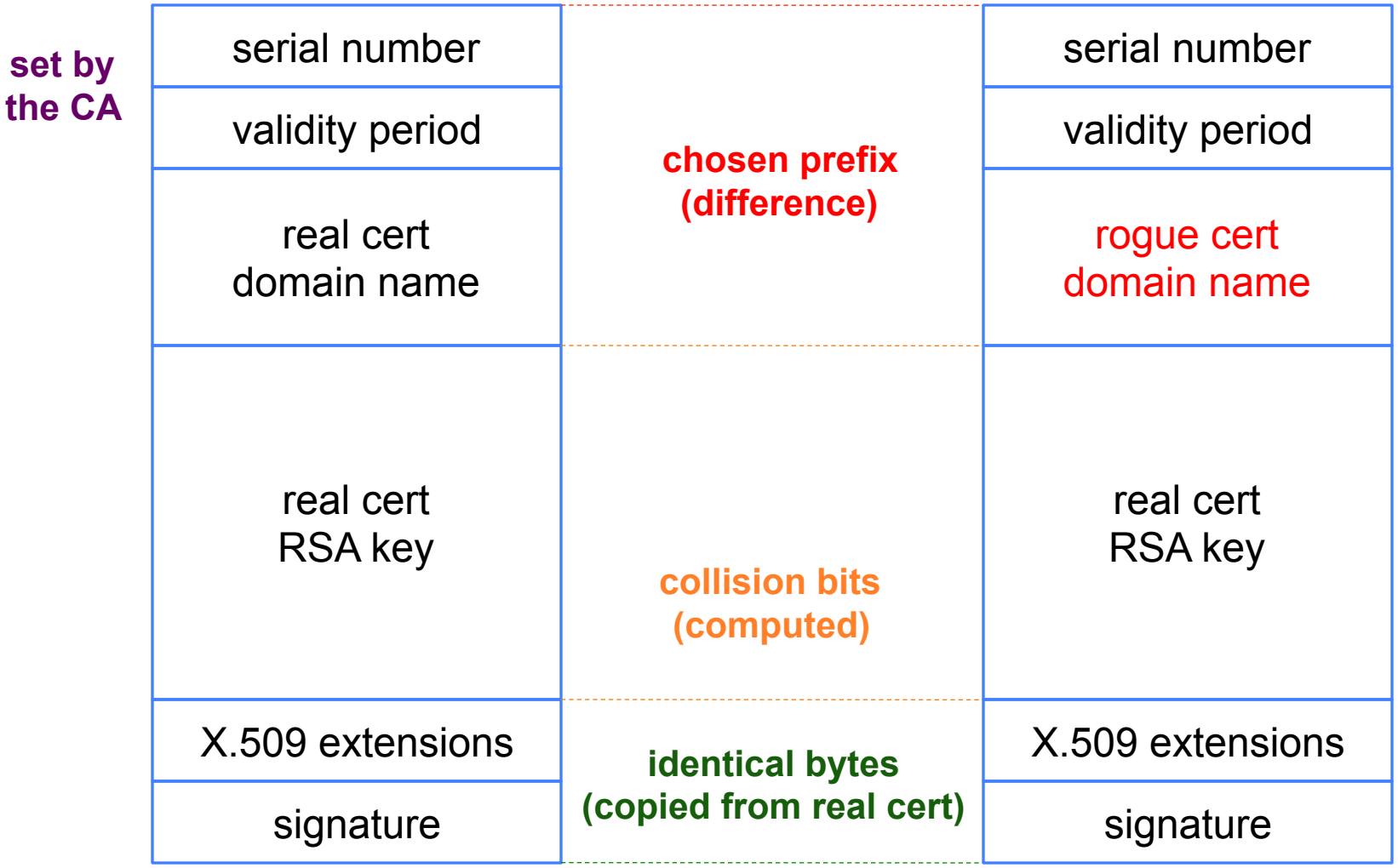
End of 2008

- A group of researchers showed as a proof of concept that
  - they were able to construct two certificates with the same MD5 hashes
  - And make a regular, browser pre-installed root CA hash, sign and issue these certificates for them
- Even worse
  - The certificate they obtained is a CA certificate, i.e. it could further be used to sign other certificates

Overview Of Attack and Potential Consequences



Approach to Find Collisions



Example: Vulnerable CAs in 2008

- Of 30 000 website certificates
  - 9 000 were signed with MD5
  - 97% of them were issued by RapidSSL
- CAs that still used MD5 in 2008
  - RapidSSL
  - FreeSSL
  - TrustCenter
  - RSA Data Security
  - Thawte
  - Versign.co.jp
- All of them seem to have moved to other hash functions by March 2009

The Authors State:

- " With optimizations the attack might be done for $2000 on Amazon EC2 in 1 day"
  - EC2 = Elastic Compute Cloud
- "We want to prevent malicious entities from repeating the attack:
  - We are not releasing our collision finding implementation or improved methods until we feel it's safe
  - We've talked to the affected CAs: they will switch to SHA-1 very, very soon"

References and Further Reading

- Kaufmann et al., Chapter 15
- RFC 5280: X.509 Certificates and CRLs
- RFC 2560: Online Certificate Revocation Protocol
- RFC 2559: X.509 Public Key Infrastructure Operational Protocols - LDAPv2
- RFC 4210: Certificate Management Protocol (CMP)

- Consequences of MD5 attacks
  - http://www.win.tue.nl/hashclash/rogue-ca/#secPres