IT-Security 1

Chapter 5: Certificates and Public Key Infrastructures

Prof. Dr.-Ing. Ulrike Meyer

WS 15/16

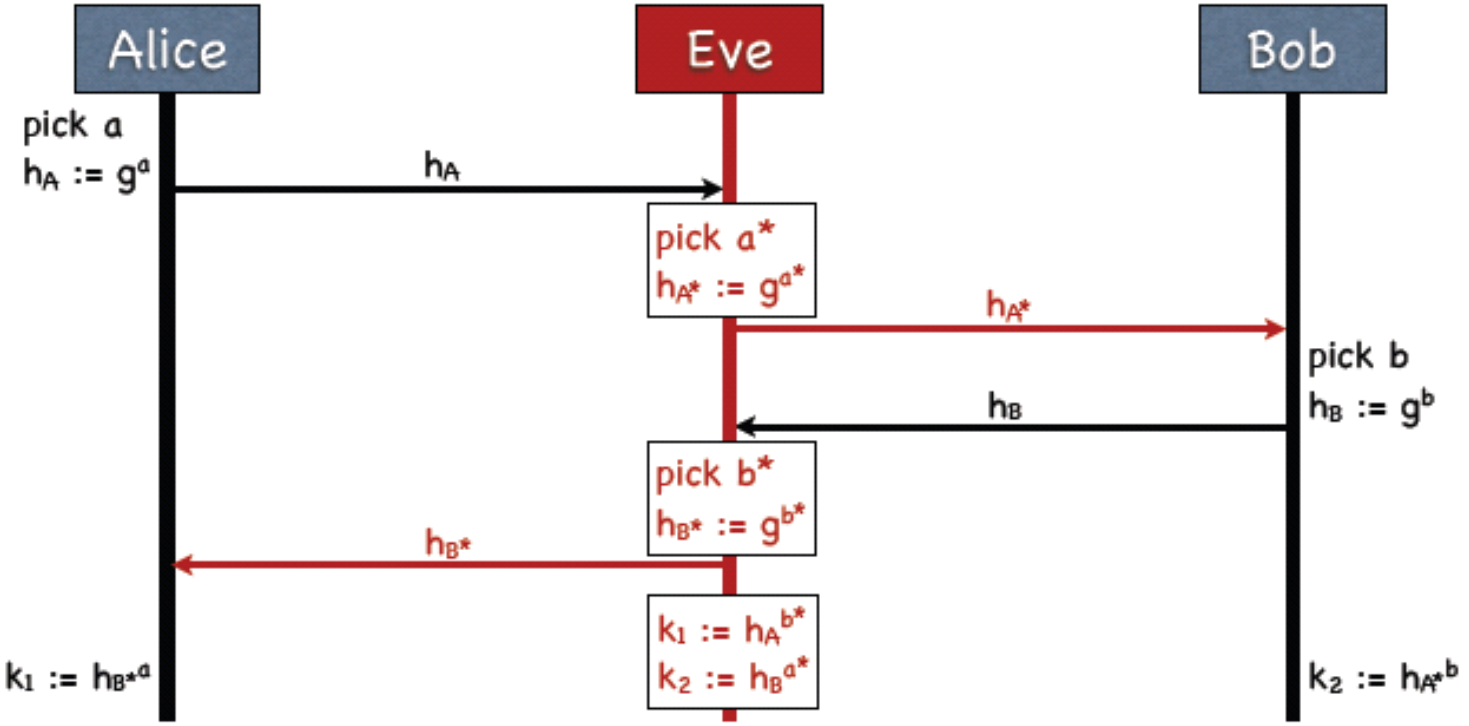
Chapter Overview

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

General Public Key Problem

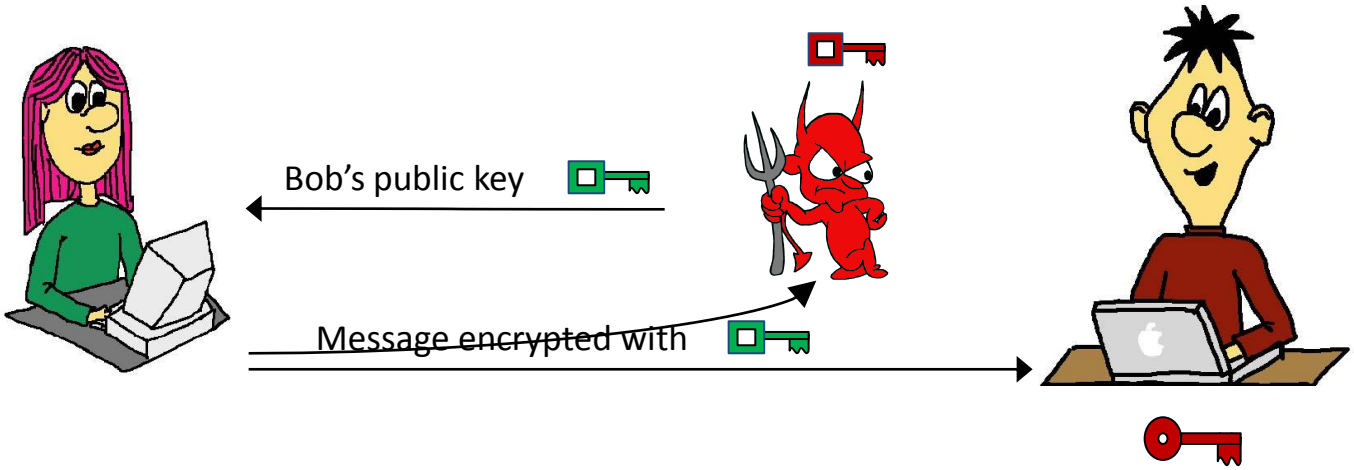
- Public key primitives require an authenticated way to provide / exchange public keys
- This holds for all asymmetric primitives:
  - Diffie-Hellman Key Exchange
  - Public key encryption
  - Digital signatures

Example: Diffie-Hellman MiM-Attack



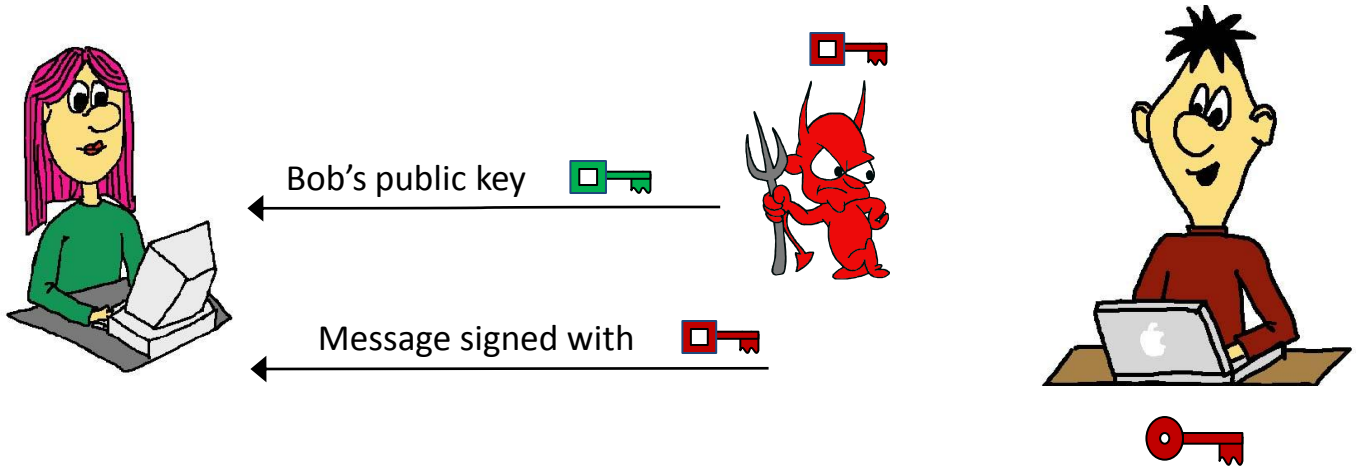
- Attack possible because Alice and Bob have no assurance of the authenticity of the public values received

Example: Public Key Encryption



- If an attacker can exchange Bobs's public encryption key with his own, then he will be able to decipher confidential messages send from Alice to Bob

Example: Public Key Signatures



- If an attacker can exchange Bob's public signature verification key with his own, the attacker can convince Alice that Bob has signed a certain message

Trusted Third Parties

- Solution: guarantee authenticity with the help of a trusted third party
  - Assumption: Alice and Bob are in possession of an authentic copy of the public signature key of the trusted third party
  - The trusted third party signs pairs of public keys and identifiers and thereby binds the keys to the identifiers
  - I.e. the trusted third party 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

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

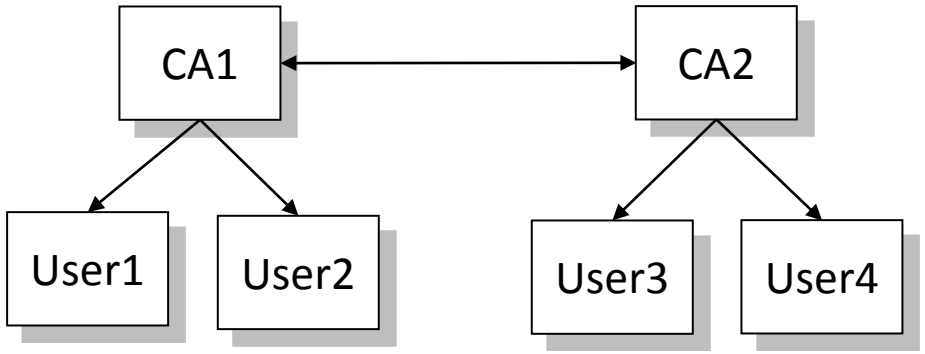
CAs and Trust

- 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

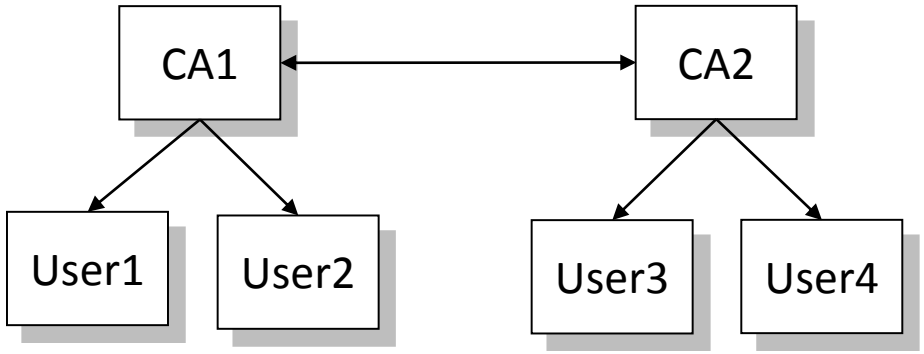
- Oligarchy: Several CAs are used simultaneously. All principles store the public keys of all CAs
- Cross certification: CA1 issues a certificate for CA2 and CA2 issues a certificate for CA1
- CA hierarchy: the CA's are hierarchically structured
  - A "root" CA issues certificates for the CAs on the 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

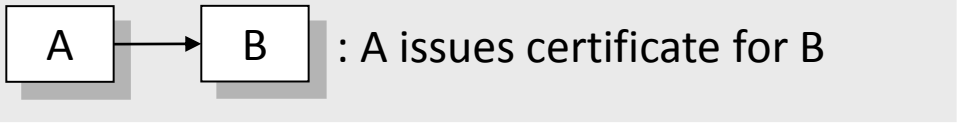
Cross Certification



- How can User1 verify the certificate $<<User3>>_{CA2}$ of User3, assuming that User1 has an authentic copy of CA1's signature verification key?

Cross Certification

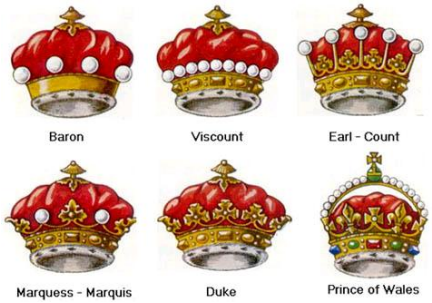


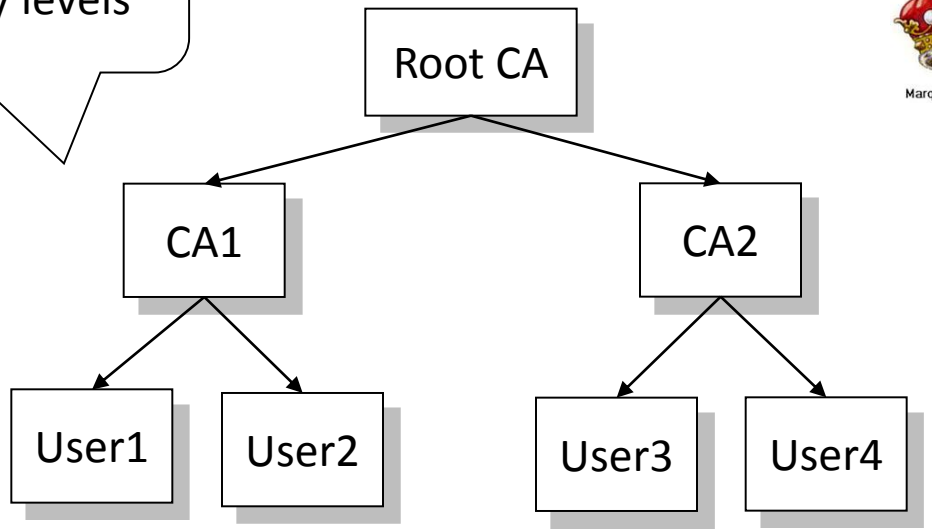


- User3 presents User1 with the chain of certificates $\langle\langle CA2 \rangle\rangle_{CA1}$ and $\langle\langle User3 \rangle\rangle_{CA2}$
- User1 verifies CA1's signature on CA2's certificate, extracts CA2's key and uses it to verify the signature on $\langle\langle User3 \rangle\rangle_{CA2}$

CA Hierarchies (Top down)

There can of course be more hierarchy levels

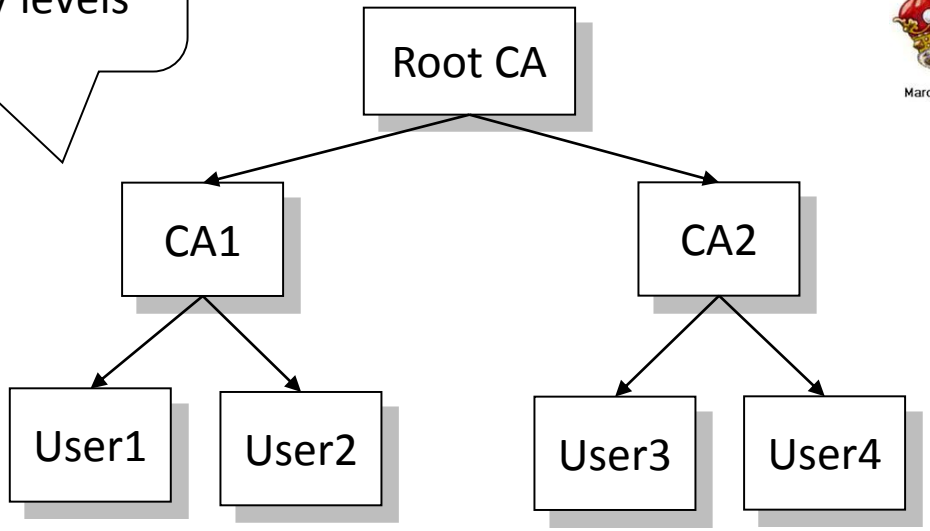




- How can User1 verify the certificate $\langle\langle User3 \rangle\rangle_{CA2}$ of User3?

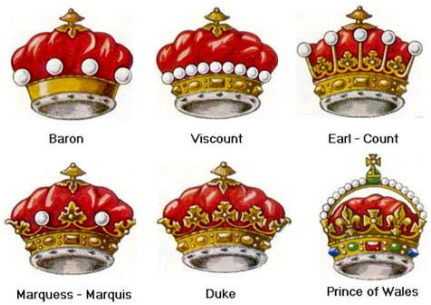
CA Hierarchies (Top down)

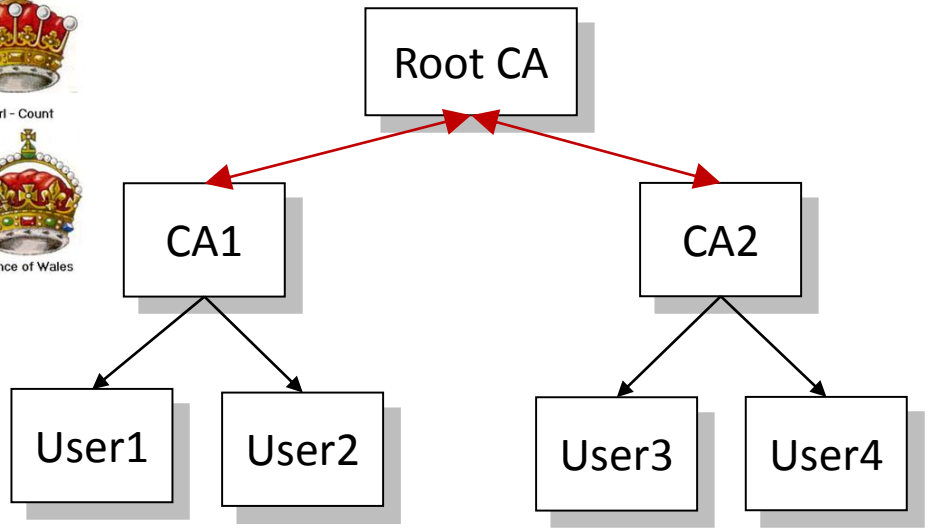
There can of course be more hierarchy levels



- User1 needs to have public key of the Root CA
- User 1 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

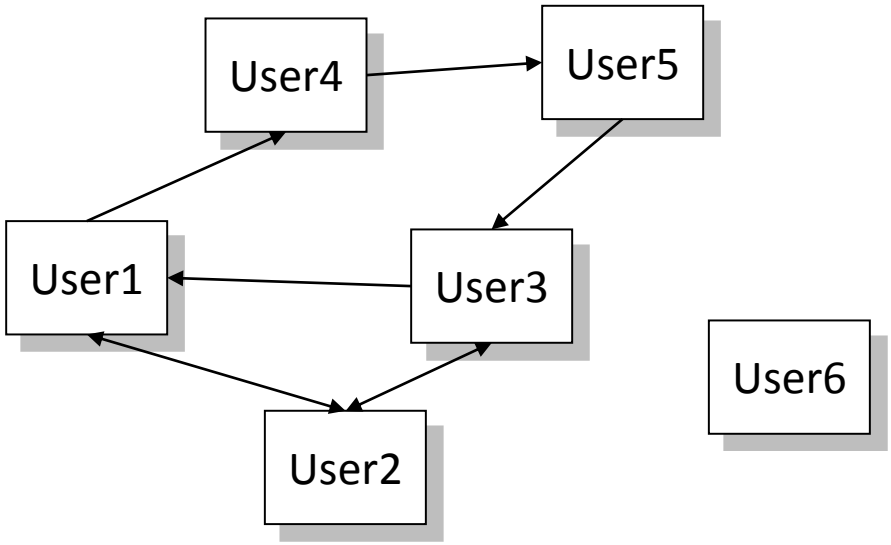
CA Hierarchies (Bottom up)





- 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 may be compromised before the validity period of the corresponding certificate 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

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

X.509 Certificates

- Signed Content:
  - Version – The version number
  - Serial Number – An issuer-unique serial number
  - Signature – The signature algorithm identifier (includes hash function used)
  - 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 (includes hash function used)
  - 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
  - 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 (includes hash function)
- SignatureValue

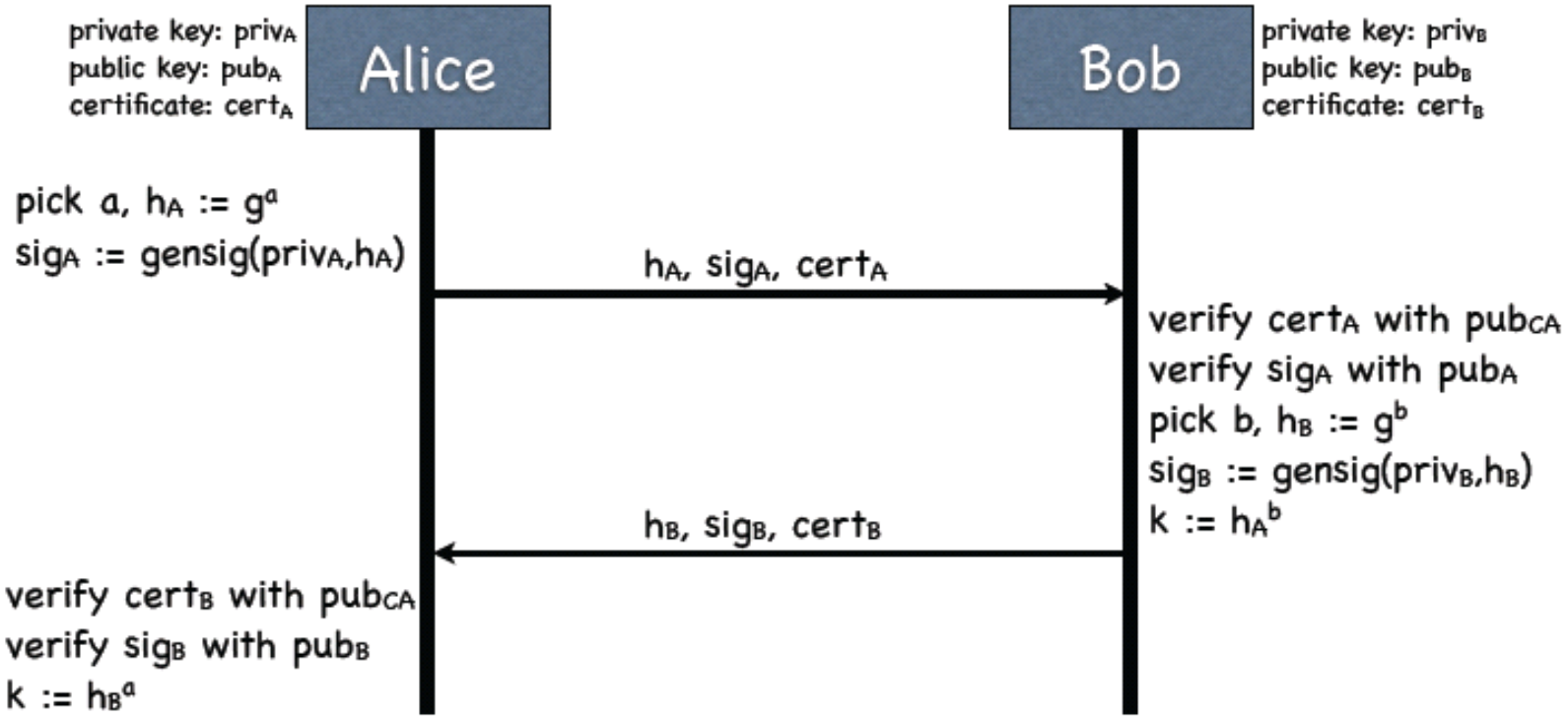
Online Certificate Status Protocol

- 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

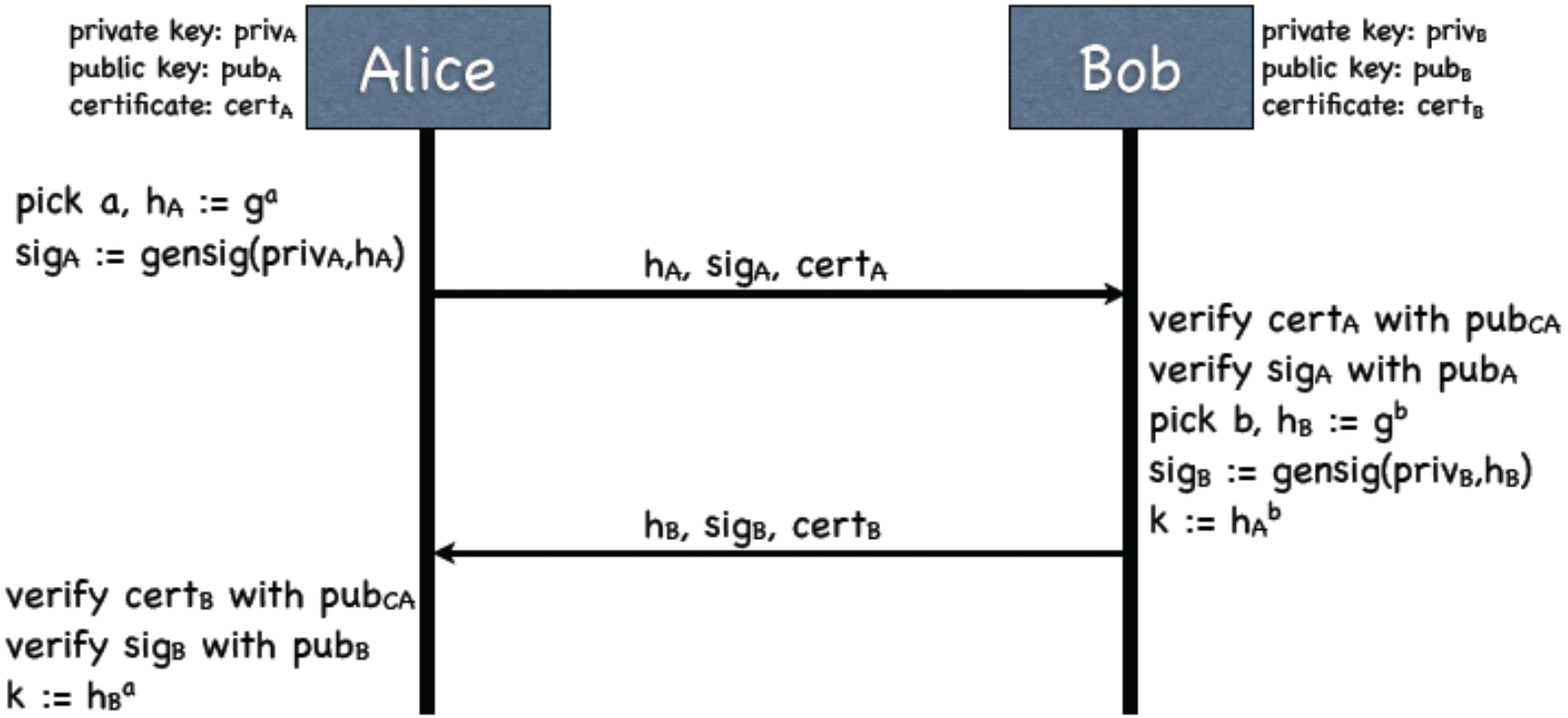
- Lightweight Directory Access Protocol
  - Protocol that specifies how to retrieve certificates and CRLs from a repository storing that information
  - Specifies message formats for
  - reading entries, searching for entries (all clients)
  - adding/deleting entries in the repository (CA only)
- Certificate 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 – First Try



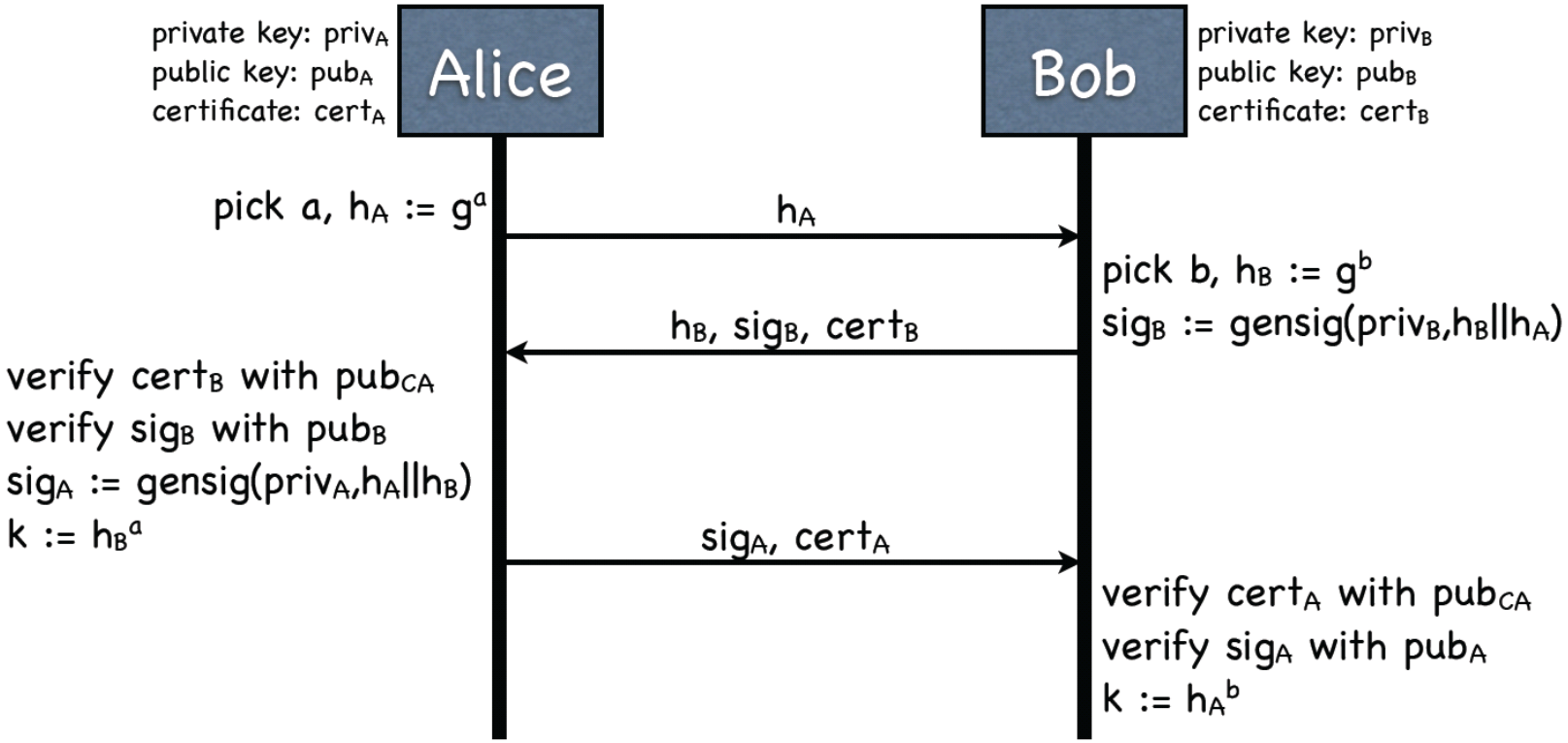
- Problem?

Using Certificates to Protect Diffie-Hellman – First Try



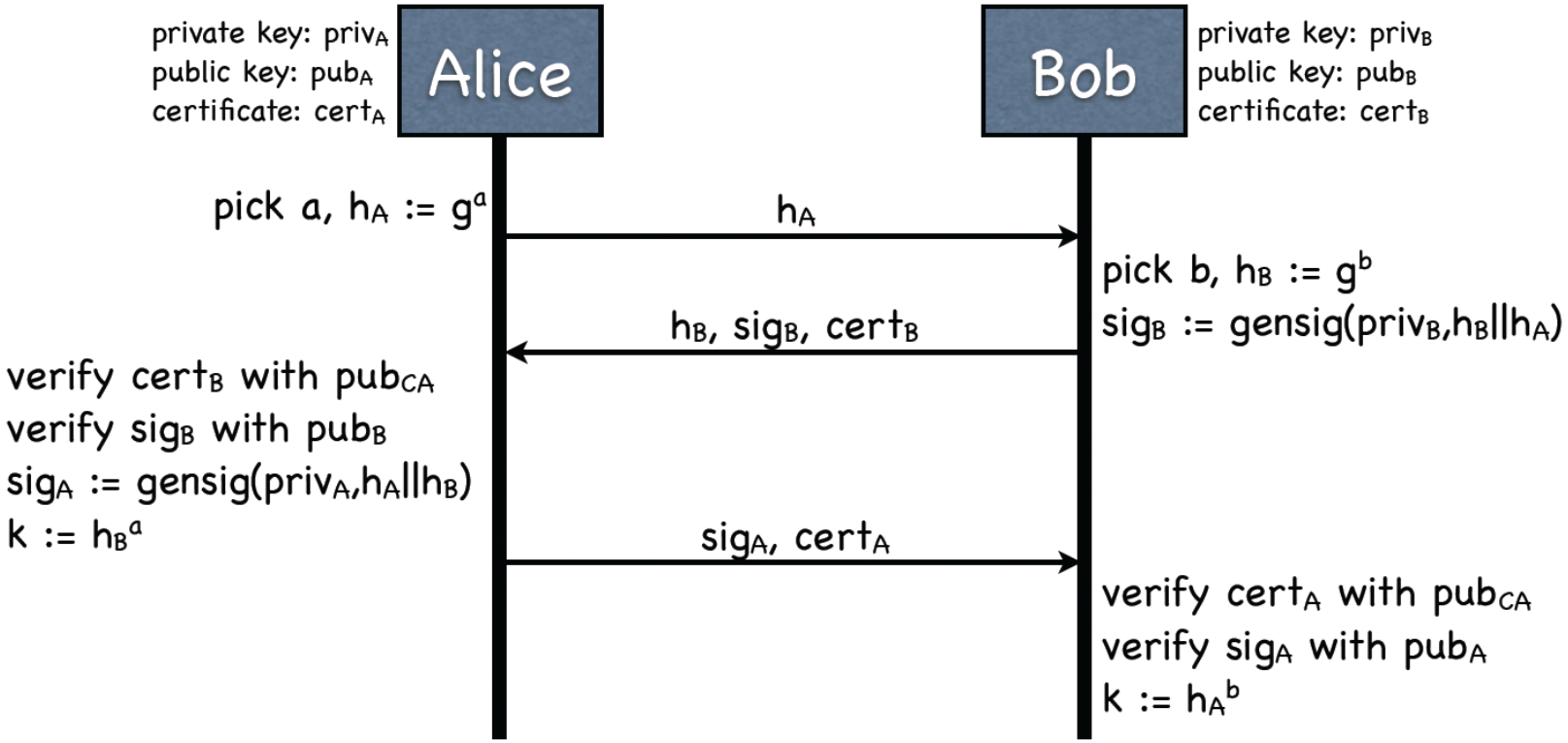
- The signatures are not bound to THIS particular run of the protocol
- An attacker could record e.g. Alice's message and replay it to Bob

Authenticated Diffie Hellman – Second Try



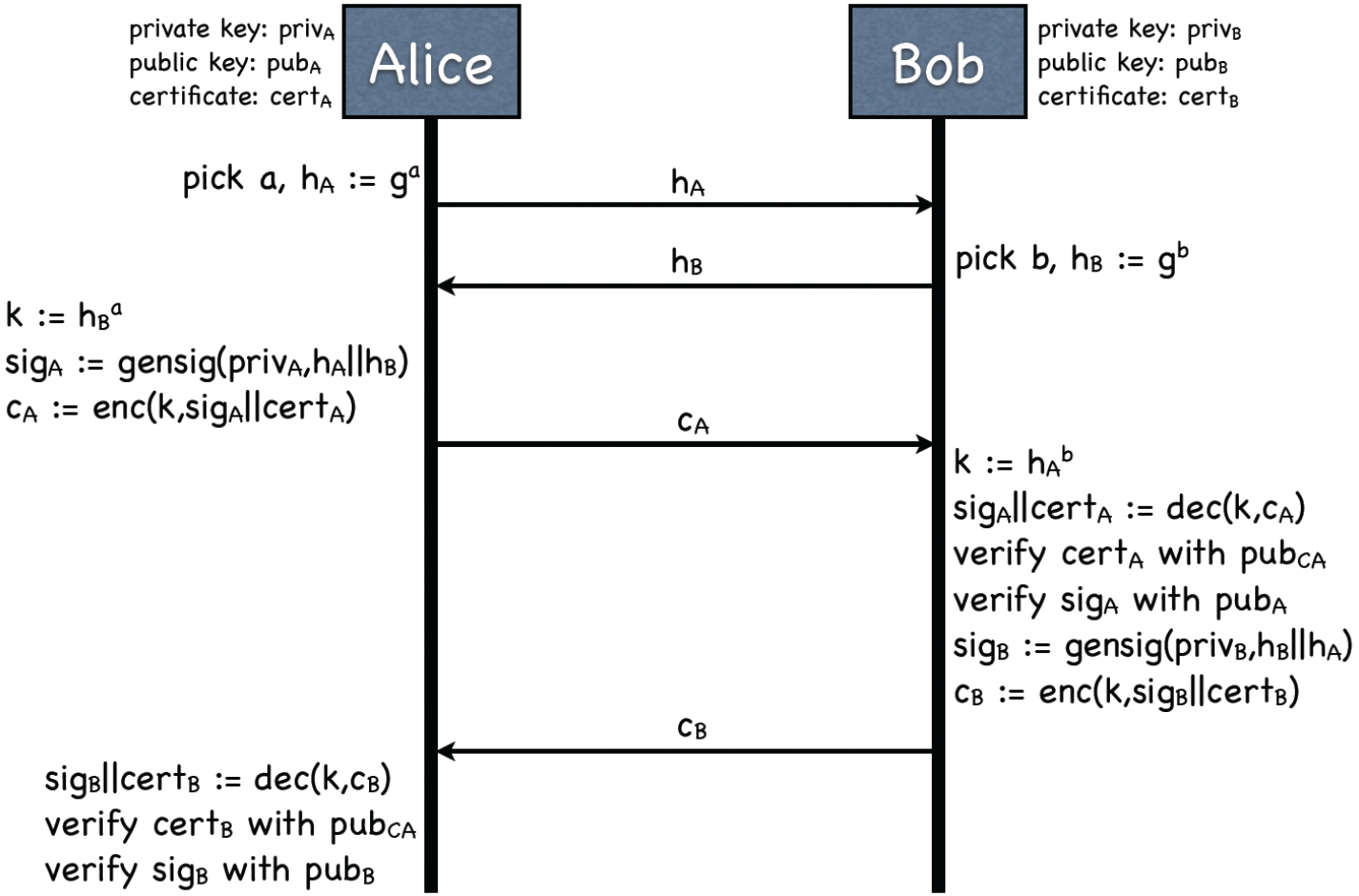
- Both parties sign the public DH values received from each other
- Both parties use public key certificates to exchange their public signature keys
- Now replay is not possible because Alice can check if Bob correctly received her fresh public DH value and vice versa

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 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…

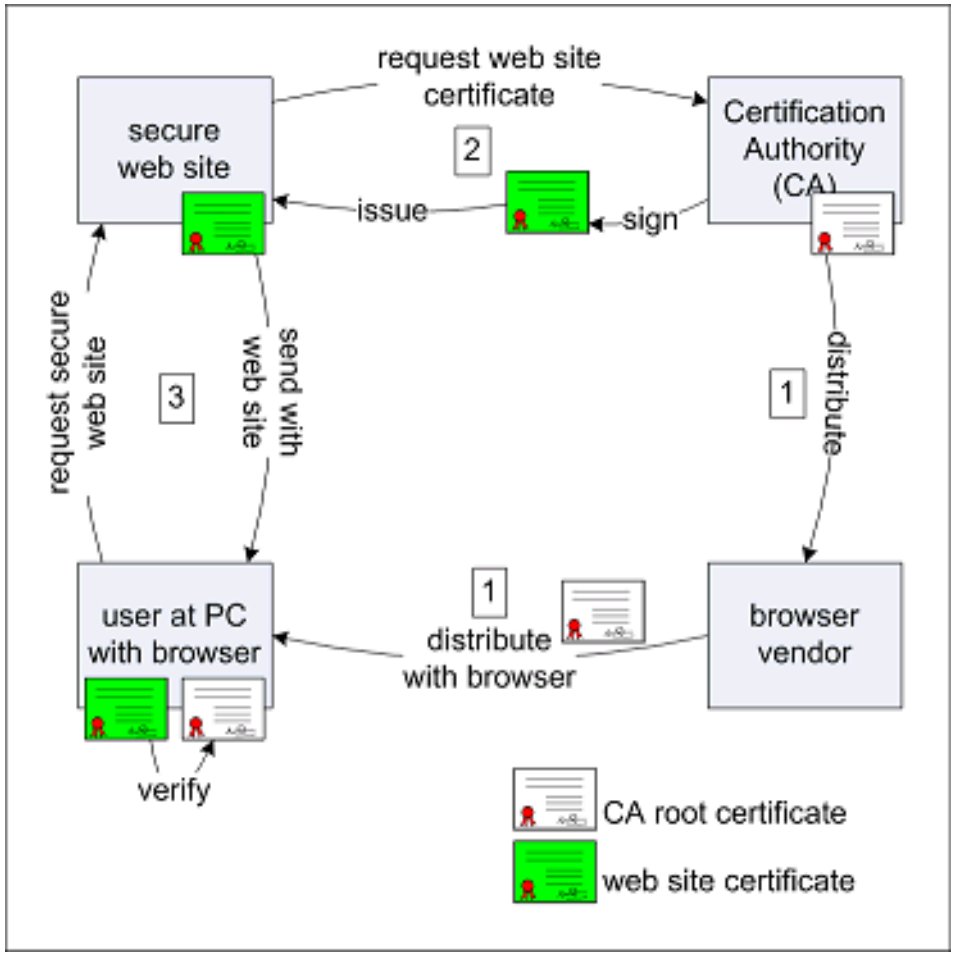
- … certificates can be used to distribute public keys in an authenticated way
- The content of a certificate is hashed and then signed with the private key of a certification authority
- In Chapter 3 we discussed hash functions and saw that a tool is available online that finds collisions in MD5
- What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

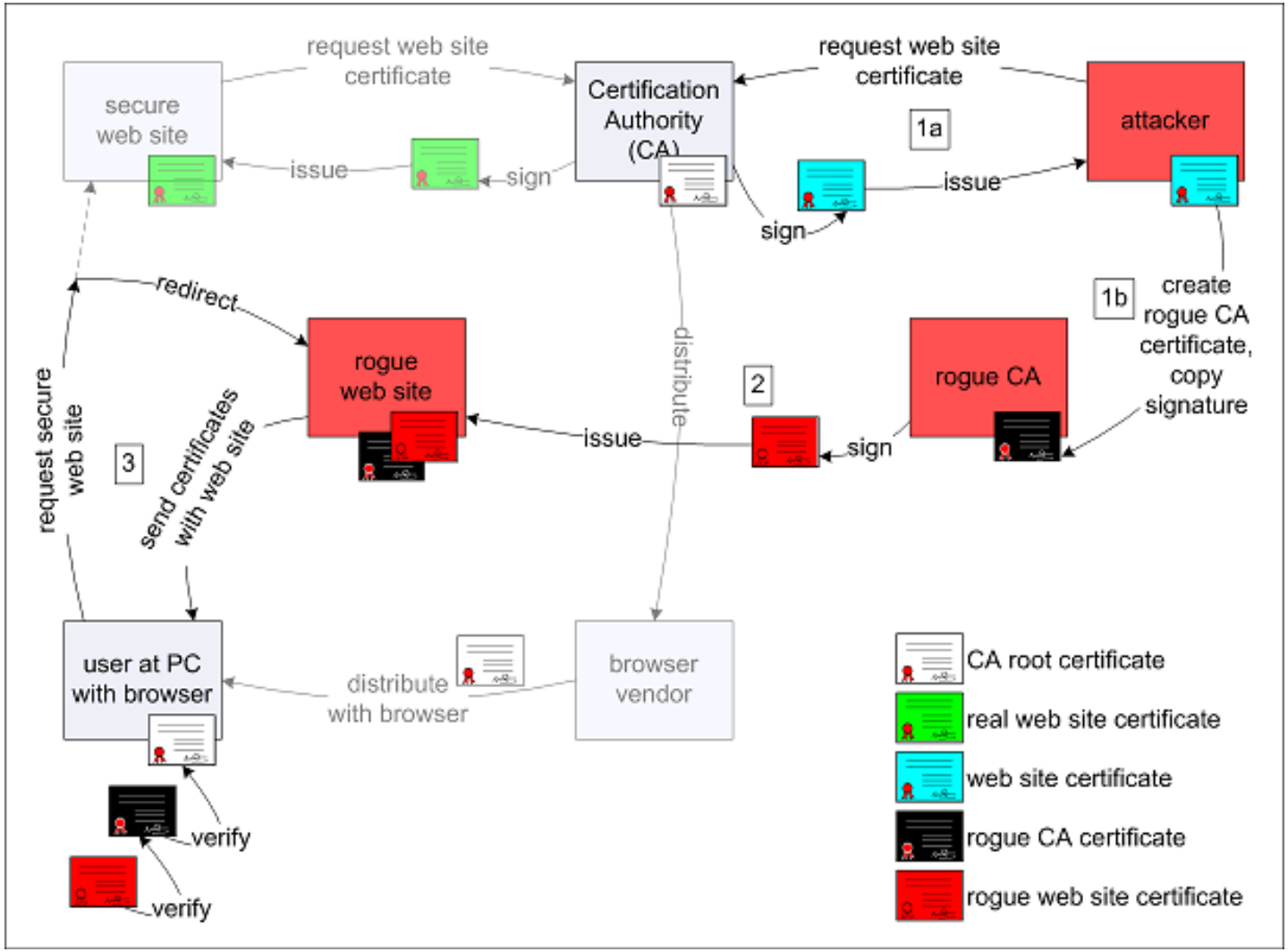
- CA root certificates are distributed via browser vendors
- Web servers can request certificates from a CA
- Users can verify certificate presented by web site with the pre-installed CA certificate



End of 2008

- A group of researcher showed as a proof of concept that
  - They made a regular, browser pre-installed root CA hash, sign and issue a web server certificate to them
  - They constructed a second certificate with the same MD5 hash
- Even worse
  - The second certificate could be constructed to be a CA certificate, i.e. it could further be used to sign other certificates

Overview Of Attack and Potential Consequences

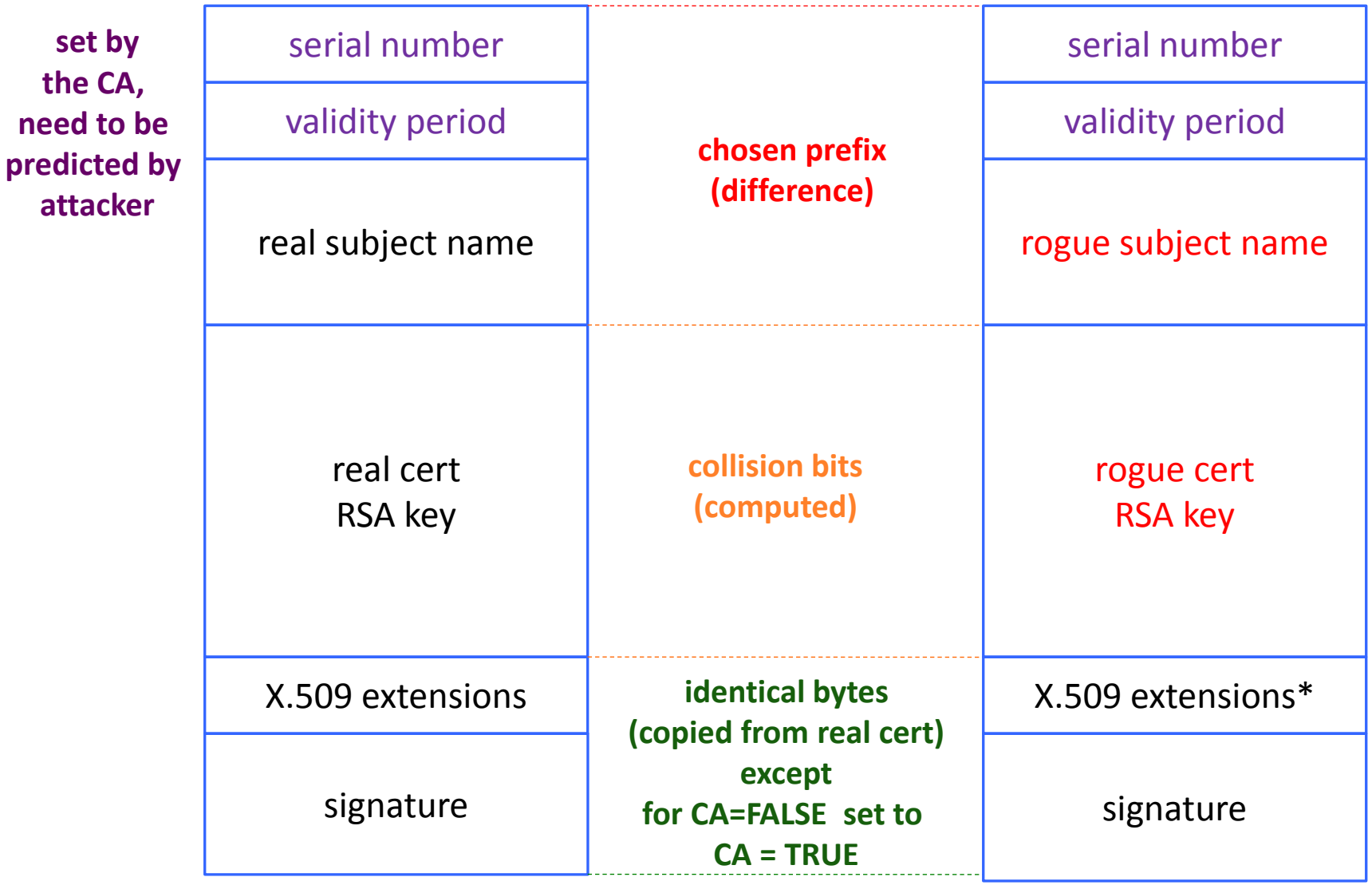


MD5 Collision Attack

- Given any P, P’ a pair C,C’ can be computed such that

$$MD5(P \,||\, C) = MD5(P' \,||\, C')$$

Approach to Find Collisions



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"

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
  - Verisign.co.jp
- All of them moved to other hash functions by March 2009

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
  - Figures taken from there!