CS-3002: Information Security

Lecture # 7: Digital Signature, Public Key Certificates and Public Key Infrastructure

Department of Software Engineering

FAST-NUCES

Overview

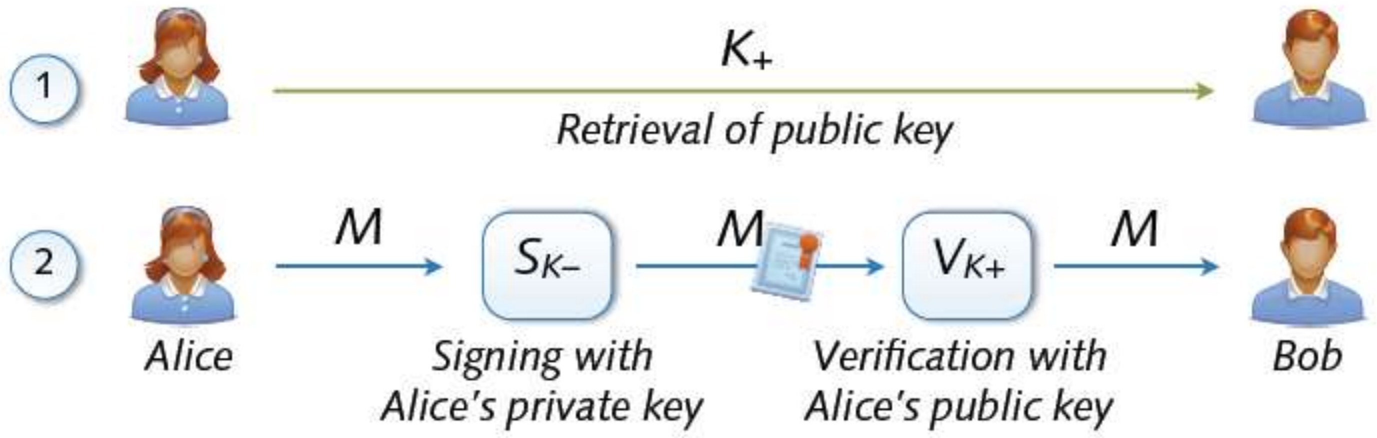
- *What will you learn today*
  - *Digital Signatures*
  - *Public key and Signature*
  - *Public Key Infrastructure*
  - *Identity Based Encryption*

Digital Signature

- **Authentication and Non-Repudiation**
  - Gives a recipient reason to believe that the message was created by a known sender such that they cannot deny sending it
- **Integrity**
  - The message was not altered in transit

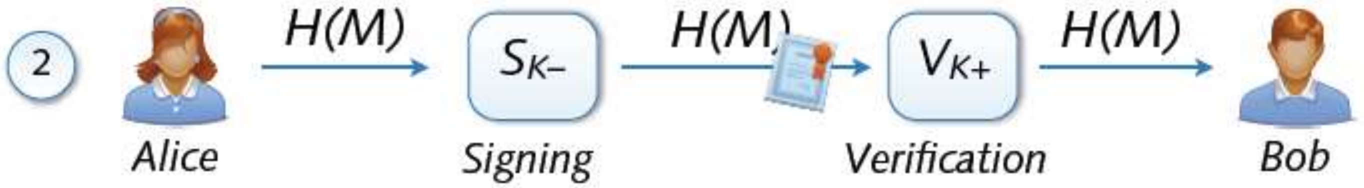
Digital Signature

- **Digital signing: reverse application of public-key system**
  - *Signing = encryption with private key*
  - *Verification = decryption with public key*

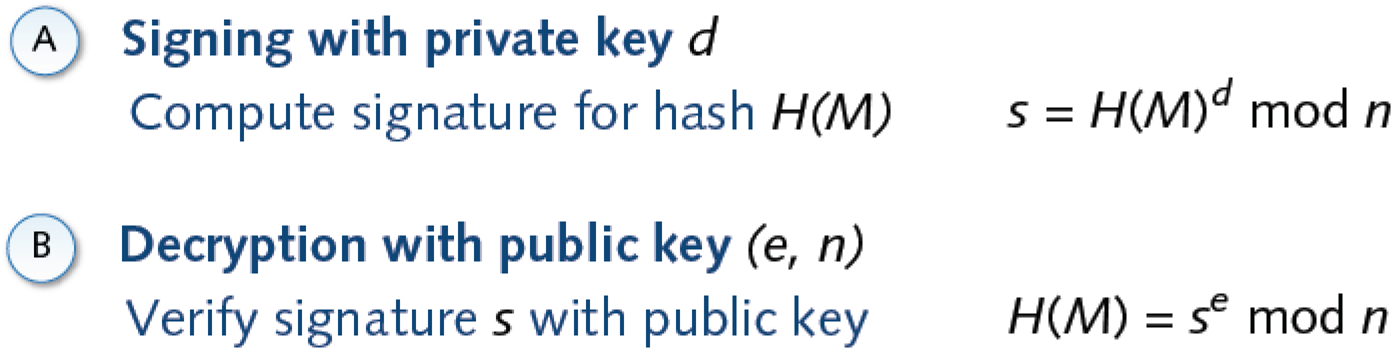

1
K_+
Retrieval of public key
2
M
S_{K-}
M
V_{K+}
M
Alice
Signing with Alice's private key
Verification with Alice's public key
Bob

Signing and Hashing

- **Encryption and decryption of large messages inefficient**
  - Signing of hash *H(M) instead of message M*
  - Verification of message *M using signed hash H(M)*
  - One-way property: hard to find *M' with H(M') = H(M)*
  - Support for signing emails, images, videos, ...



Signature and RSA

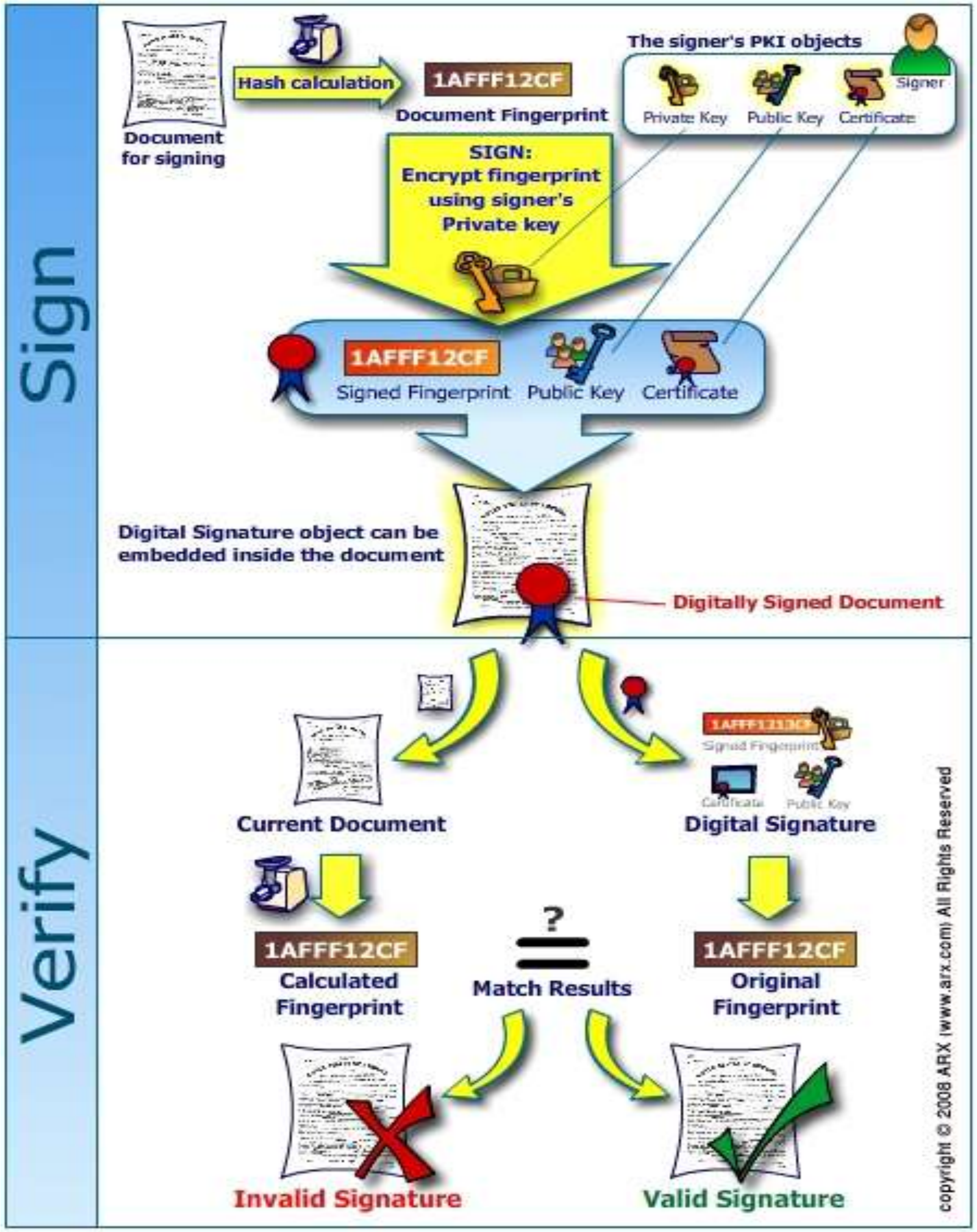


A **Signing with private key** d

Compute signature for hash $H(M)$ $\qquad s = H(M)^d \bmod n$

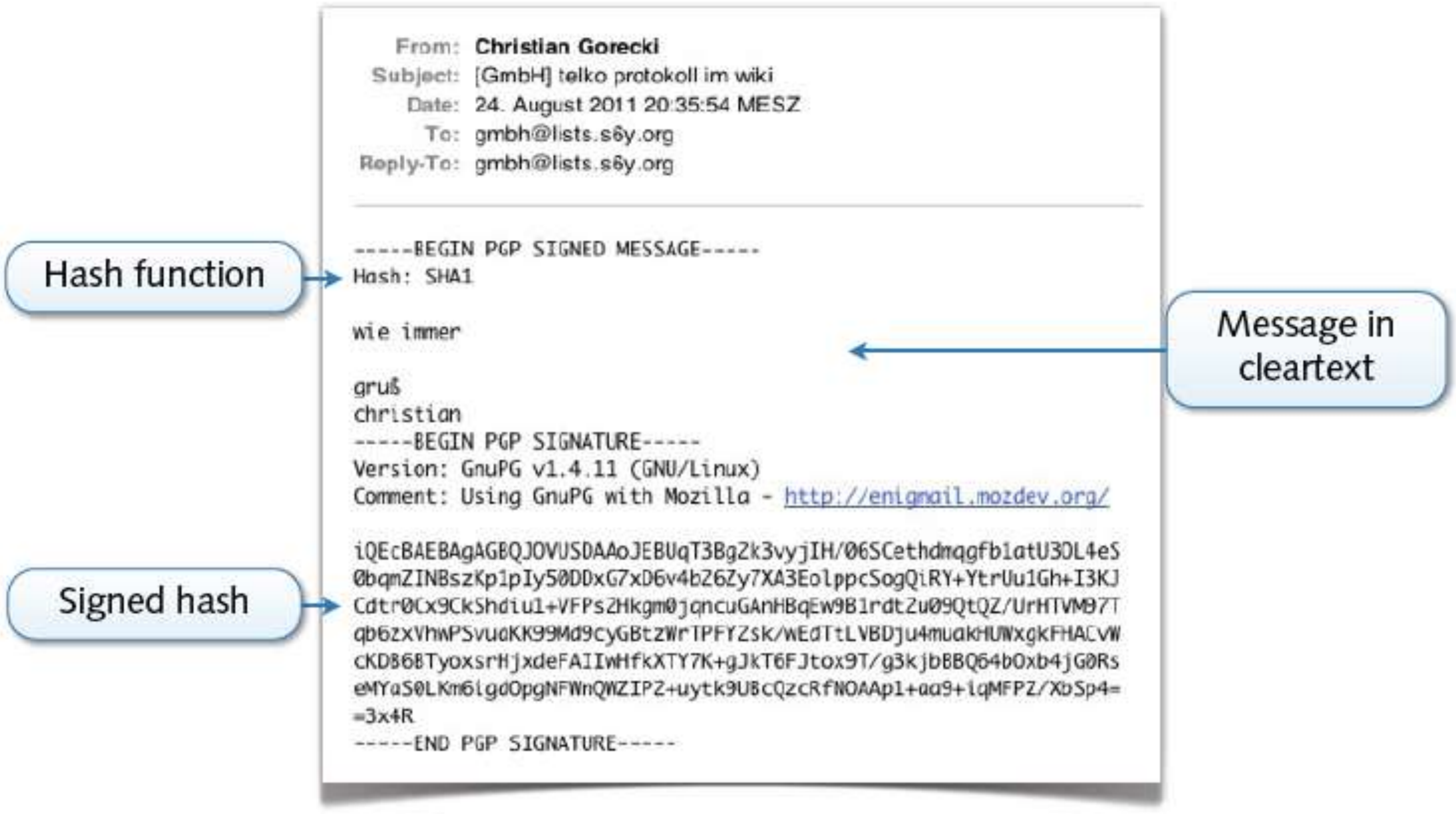
B **Decryption with public key** (e, n)

Verify signature s with public key $\qquad H(M) = s^e \bmod n$



Sign
Document for signing
Hash calculation
1AFFF12CF
Document Fingerprint
The signer's PKI objects
Signer
Private Key
Public Key
Certificate
SIGN: Encrypt fingerprint using signer's Private key
1AFFF12CF
Signed Fingerprint
Public Key
Certificate
Digital Signature object can be embedded inside the document
Digitally Signed Document
Verify
Signed Fingerprint
Certificate
Public Key
Current Document
Digital Signature
1AFFF12CF
Calculated Fingerprint
?
Match Results
1AFFF12CF
Original Fingerprint
Invalid Signature
Valid Signature
copyright © 2008 ARX (www.arx.com) All Rights Reserved

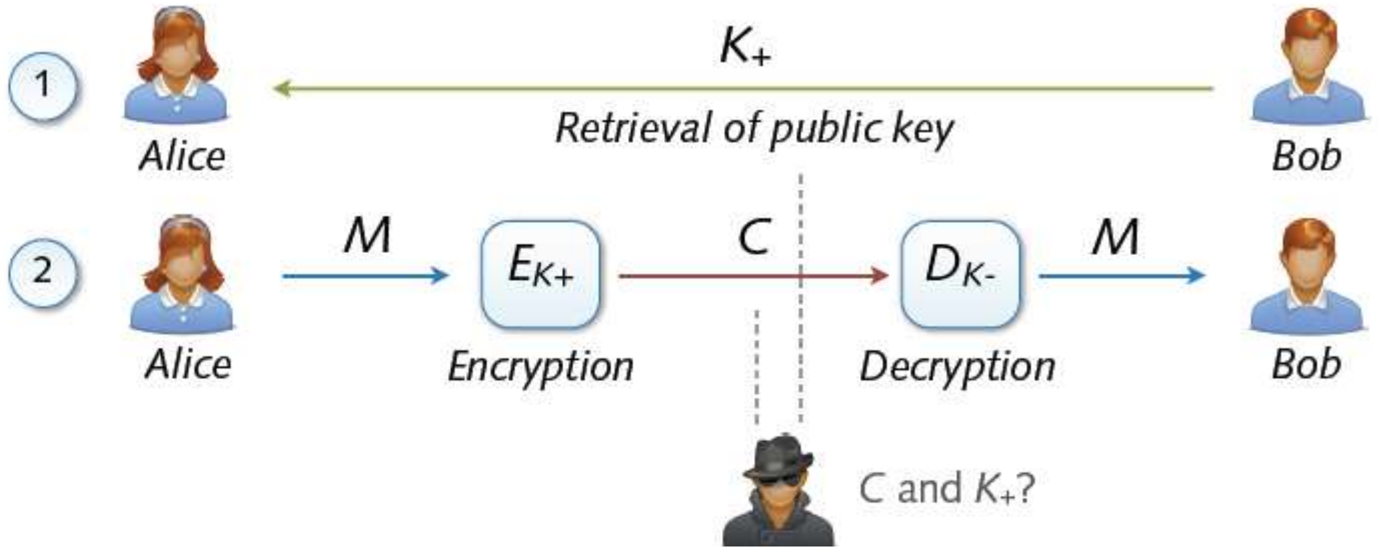
Example: PGP Signature



```
     From: Christian Gorecki
  Subject: [GmbH] telko protokoll im wiki
     Date: 24. August 2011 20:35:54 MESZ
       To: gmbh@lists.s6y.org
 Reply-To: gmbh@lists.s6y.org

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA1

wie immer

gruß
christian
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.11 (GNU/Linux)
Comment: Using GnuPG with Mozilla - http://enigmail.mozdev.org/

iQEcBAEBAgAGBQJOVUSDAAoJEBUqT3BgZk3vyjIH/06SCethdmqgfb1atU3DL4eS
0bqmZINBszKp1pIy50DDxG7xD6v4bZ6Zy7XA3EolppcSogQiRY+YtrUu1Gh+I3KJ
Cdtr0Cx9CkShdiu1+VFPs2Hkgm0jqncuGAnHBqEw9B1rdtZu09QtQZ/UrHTVM07T
qb6zxVhwPSvuaKK99Md9cyGBtzWrTPFYZsk/wEdTtLVBDju4muakHUWxgkFHACvW
cKDB6BTyoxsrHjxdeFAIIwHfkXTY7K+gJkT6FJtox9T/g3kjbBBQ64bOxb4jG0Rs
eMYaS0LKm6igdOpgNFWnQWZIPZ+uytk9UBcQzcRfNOAAp1+aa9+iqMFPZ/XbSp4=
=3x4R
-----END PGP SIGNATURE-----
```

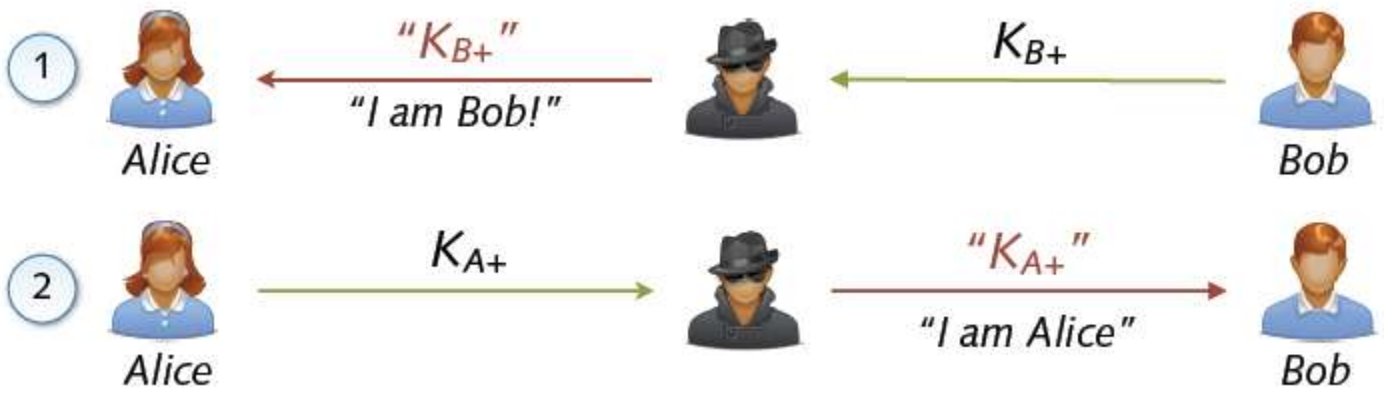
Asymmetric Cryptosystem

- **Asymmetric cryptosystems**
  - Asymmetric encryption and decryption
  - K+ (pk) = public key of Bob K– (sk) = secret key of Bob
  - No secure key exchange necessary



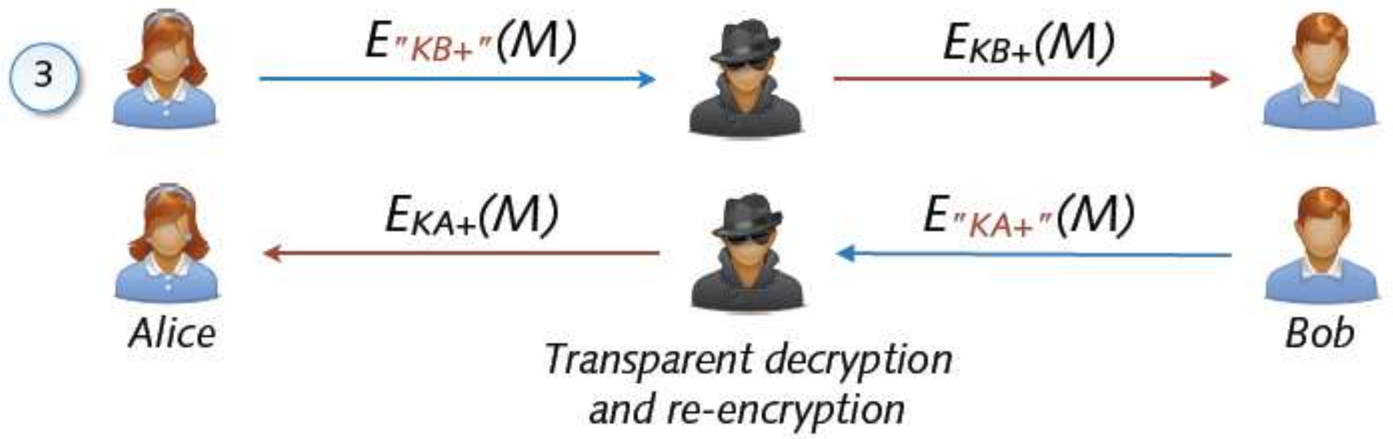
Man in the Middle (MITM)

- **Common attack against asymmetric cryptosystems**
  - Interception of public key exchange by attacker
  - Transparent eavesdropping using forged keys



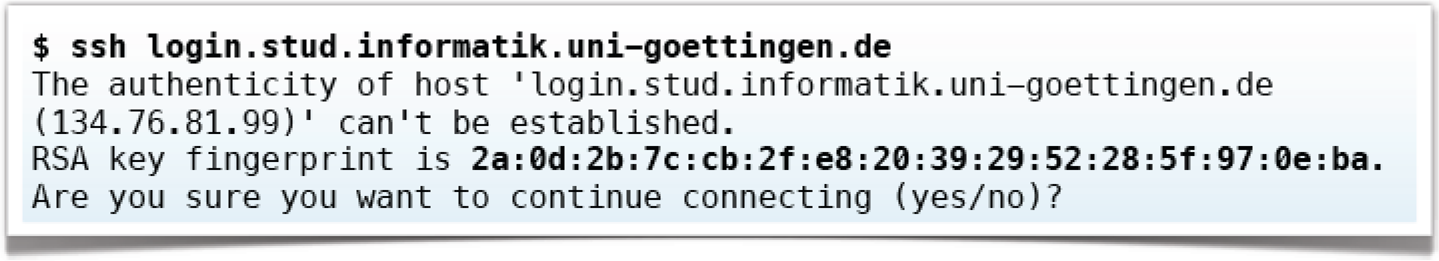
Man in the Middle (MITM)

- **Attacker invisible to both parties**
  - Received data encrypted with correct public key
  - Sent data encrypted with forged public keys



Key Fingerprints

- **Protection against MITM using key fingerprints**
  - Manual comparison of public keys using hash values
  - Storage of approved public keys in database
- **Example: SSH client presents fingerprint for validation**

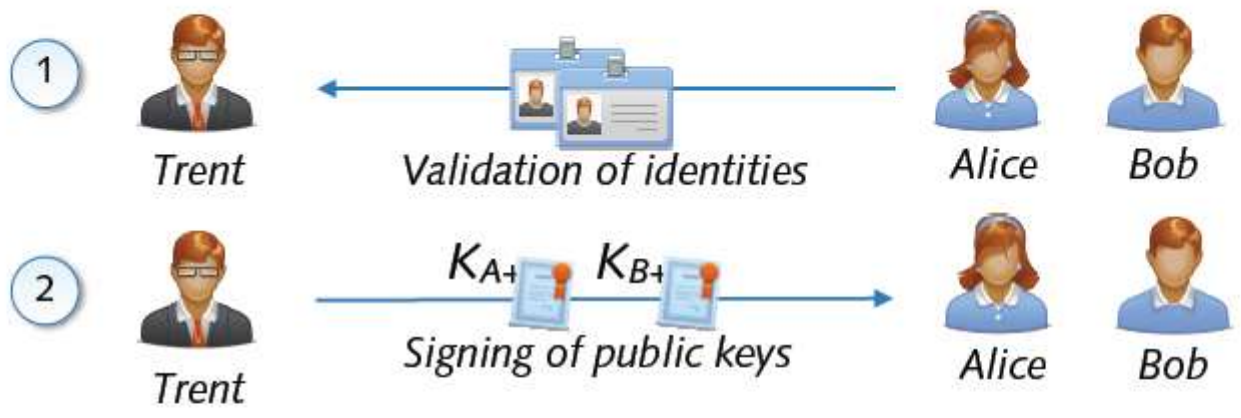


```
$ ssh login.stud.informatik.uni-goettingen.de
The authenticity of host 'login.stud.informatik.uni-goettingen.de
(134.76.81.99)' can't be established.
RSA key fingerprint is 2a:0d:2b:7c:cb:2f:e8:20:39:29:52:28:5f:97:0e:ba.
Are you sure you want to continue connecting (yes/no)?
```

- **Secure exchange of fingerprints required (hen-egg problem)**

Public Key and Signatures

- **Problem: Public keys not linked to identity of user**
- **Solution: Validation and signing of public key by third party**
  - Certification of link between identity and public key



  - Acceptance of signed public keys only → no MITM attack

Public Key Certificates

- Electronic document that uses a digital signature to bind a public key with an identity
  - Information such as the name of a person or an organization, their address, and so forth.
  - The certificate can be used to verify that a public key belongs to an individual.

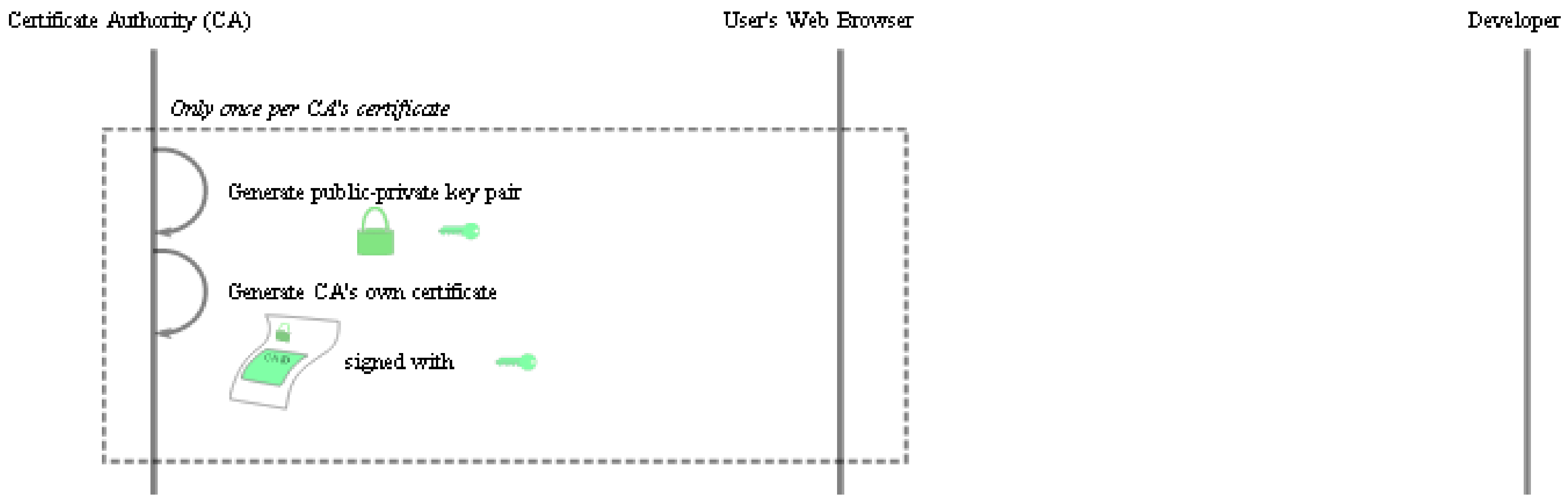
Two Types of Signature on a Certificate

- In public key infrastructure (PKI) scheme
  - Signature will be of a certificate authority (CA).
- In web of trust scheme
  - Signature is of either the user (a self-signed certificate) or other users ("endorsements").
  -

In either case, the signatures on a certificate are attestations by the certificate signer that the identity information and the public key belong together.

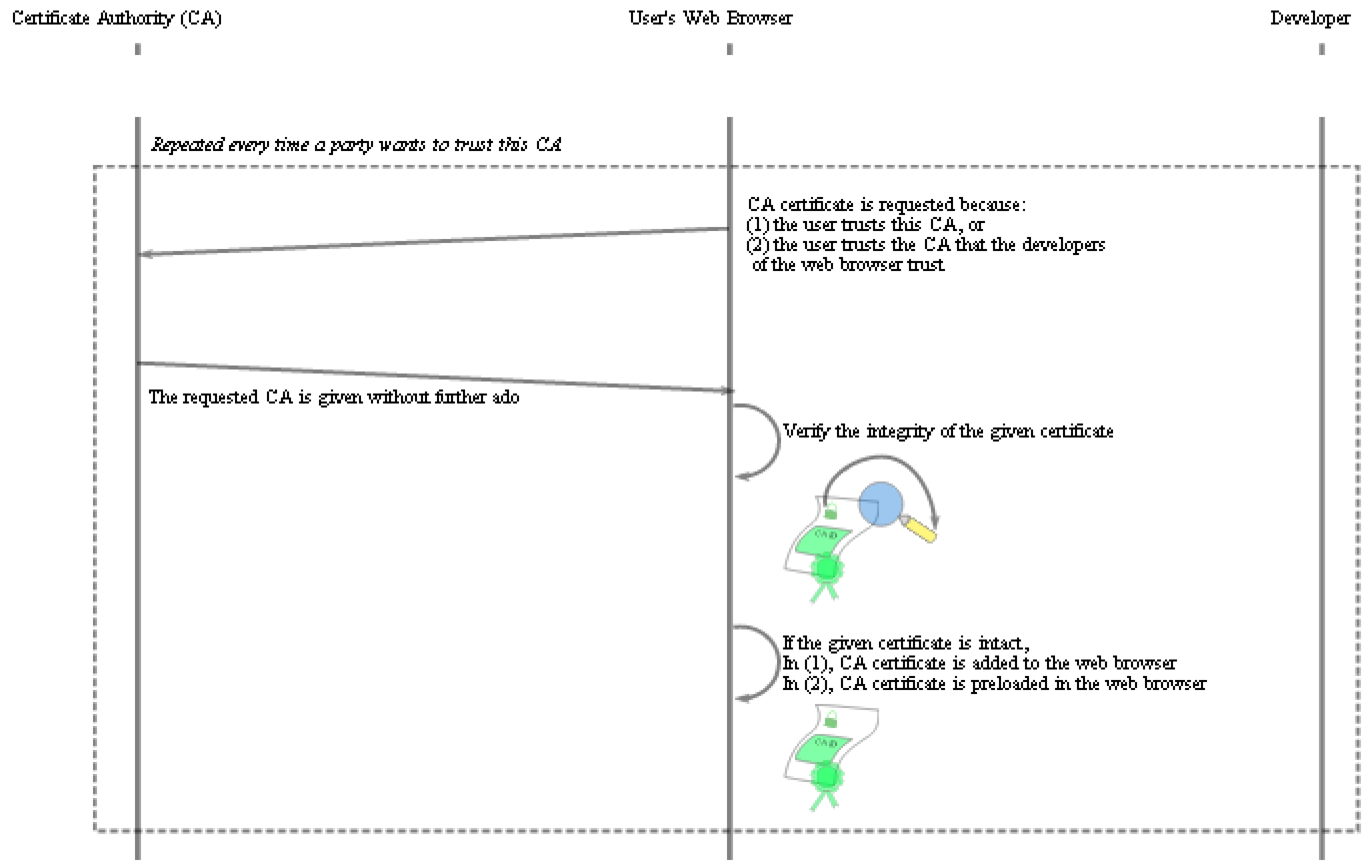
*from wiki

Certificate Creation (Step 1)


Certificate Authority (CA)
User's Web Browser
Developer
Only once per CA's certificate
Generate public-private key pair
Generate CA's own certificate
signed with

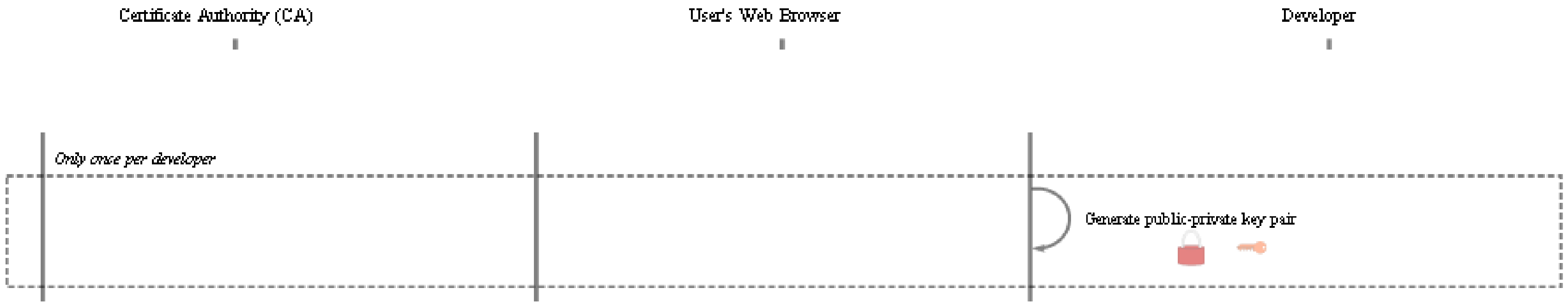
*from wiki

Certificate Creation (Step 2)


Certificate Authority (CA)
User's Web Browser
Developer
Repeated every time a party wants to trust this CA
CA certificate is requested because:
(1) the user trusts this CA, or
(2) the user trusts the CA that the developers of the web browser trust
The requested CA is given without further ado
Verify the integrity of the given certificate
If the given certificate is intact,
In (1), CA certificate is added to the web browser
In (2), CA certificate is preloaded in the web browser

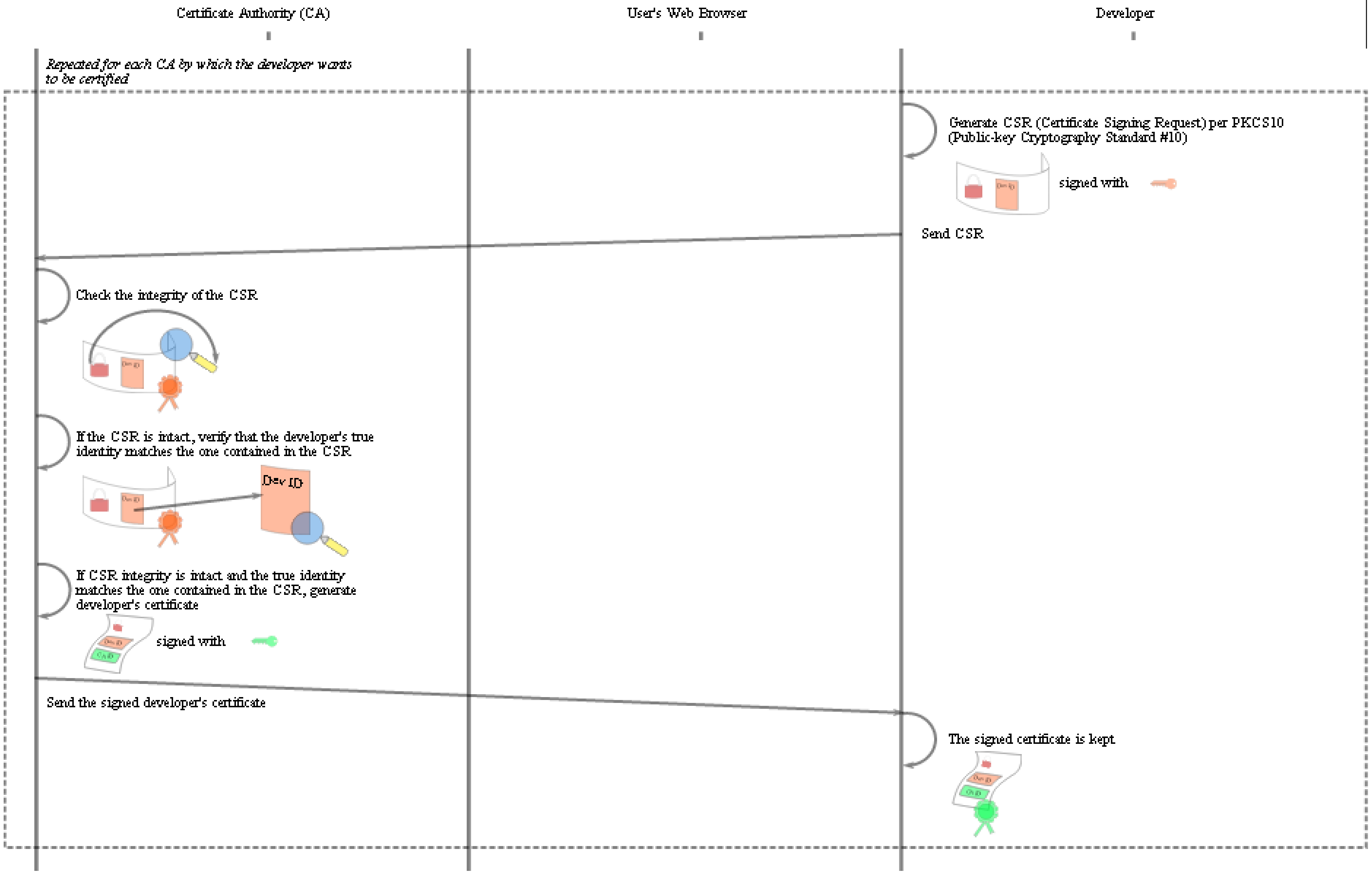
*from wiki

Certificate Creation (Step 3)



*from wiki

Certificate Creation (Step 4)



Repeated for each CA by which the developer wants to be certified

Generate CSR (Certificate Signing Request) per PKCS10 (Public-key Cryptography Standard #10)

Send CSR

Check the integrity of the CSR

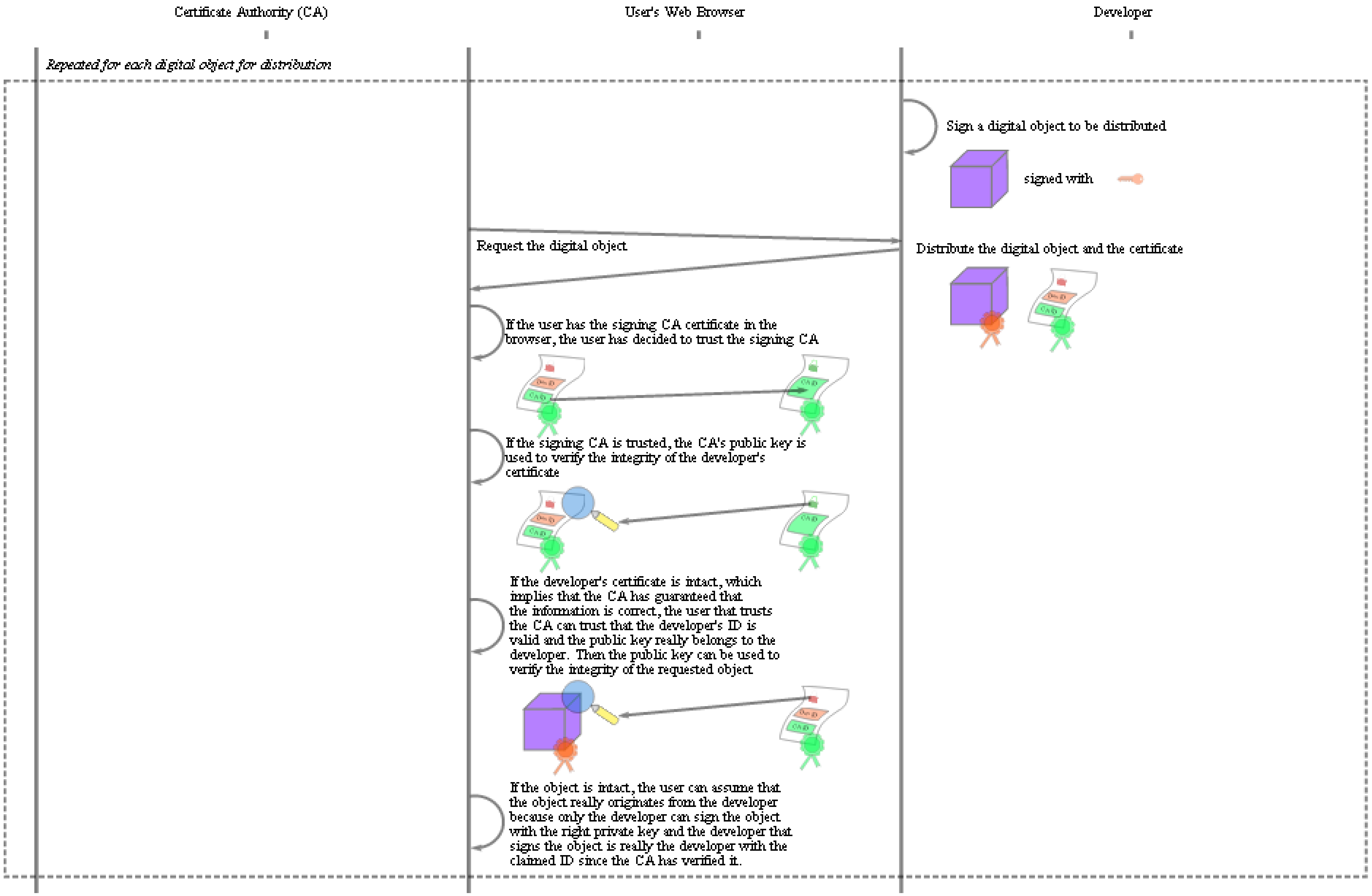
If the CSR is intact, verify that the developer's true identity matches the one contained in the CSR

If CSR integrity is intact and the true identity matches the one contained in the CSR, generate developer's certificate

Send the signed developer's certificate

The signed certificate is kept

Certificate Creation (Step 5)



Repeated for each digital object for distribution

Sign a digital object to be distributed

Request the digital object

Distribute the digital object and the certificate

If the user has the signing CA certificate in the browser, the user has decided to trust the signing CA

If the signing CA is trusted, the CA's public key is used to verify the integrity of the developer's certificate

If the developer's certificate is intact, which implies that the CA has guaranteed that the information is correct, the user that trusts the CA can trust that the developer's ID is valid and the public key really belongs to the developer. Then the public key can be used to verify the integrity of the requested object

If the object is intact, the user can assume that the object really originates from the developer because only the developer can sign the object with the right private key and the developer that signs the object is really the developer with the claimed ID since the CA has verified it.
]

Contents of Typical Digital Certificate

- **Serial Number**: Used to uniquely identify the certificate.
- **Subject**: The person, or entity identified.
- **Signature Algorithm**: The algorithm used to create the signature.
- **Signature**: The actual signature to verify that it came from the issuer.
- **Issuer**: The entity that verified the information and issued the certificate.
- **Valid-From**: The date the certificate is first valid from.
- **Valid-To**: The expiration date.
- **Key-Usage**: Purpose of the public key (e.g. encipherment, signature, certificate signing...).
- **Public Key**: The public key.
- **Thumbprint Algorithm**: The algorithm used to hash the public key certificate.
- **Thumbprint**: The hash itself, used as an abbreviated form of the public key certificate.

*from wiki

Vendor defined classes

VeriSign uses the concept of classes for different types of digital certificates

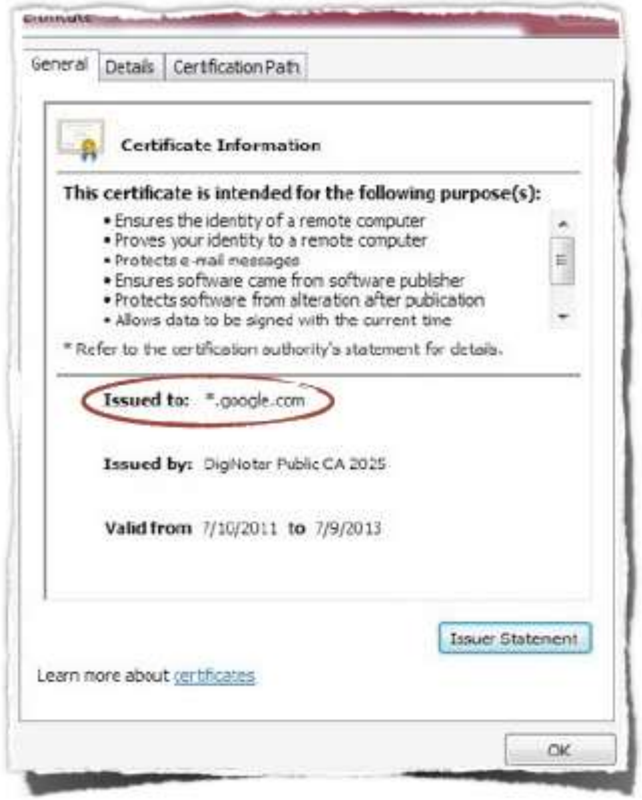
- **Class 1** for individuals, intended for email.
- **Class 2** for organizations, for which proof of identity is required.
- **Class 3** for servers and software signing, for which independent verification and checking of identity and authority is done by the issuing certificate authority.
- **Class 4** for online business transactions between companies.
- **Class 5** for private organizations or governmental security.

Other vendors may choose to use different classes or no classes at all as this is not specified in the PKI standards.

*from wiki

No more MITM?

- **Case: Forged Google certificate**
  - Issued by legitimate CA
  - Valid for *.google.com
  - Used by unknown holder
  - Reported by Iranian users
- **Large-scale attack against CA**
  - Break-in at CA DigitNotar
  - 539 forged certificates



Certificate Information

This certificate is intended for the following purpose(s):

- Ensures the identity of a remote computer
- Proves your identity to a remote computer
- Protects e-mail messages
- Ensures software came from software publisher
- Protects software from alteration after publication
- Allows data to be signed with the current time

* Refer to the certification authority's statement for details.

Issued to: *.google.com

Issued by: DigiNotar Public CA 2025

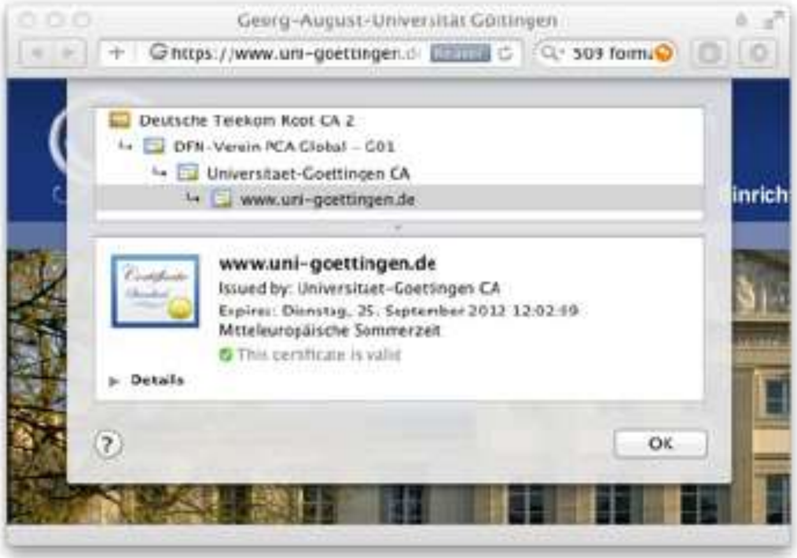
Valid from 7/10/2011 **to** 7/9/2013

Learn more about certificates

Reported on August 28, 2011

Public Key Infrastructure (PKI)

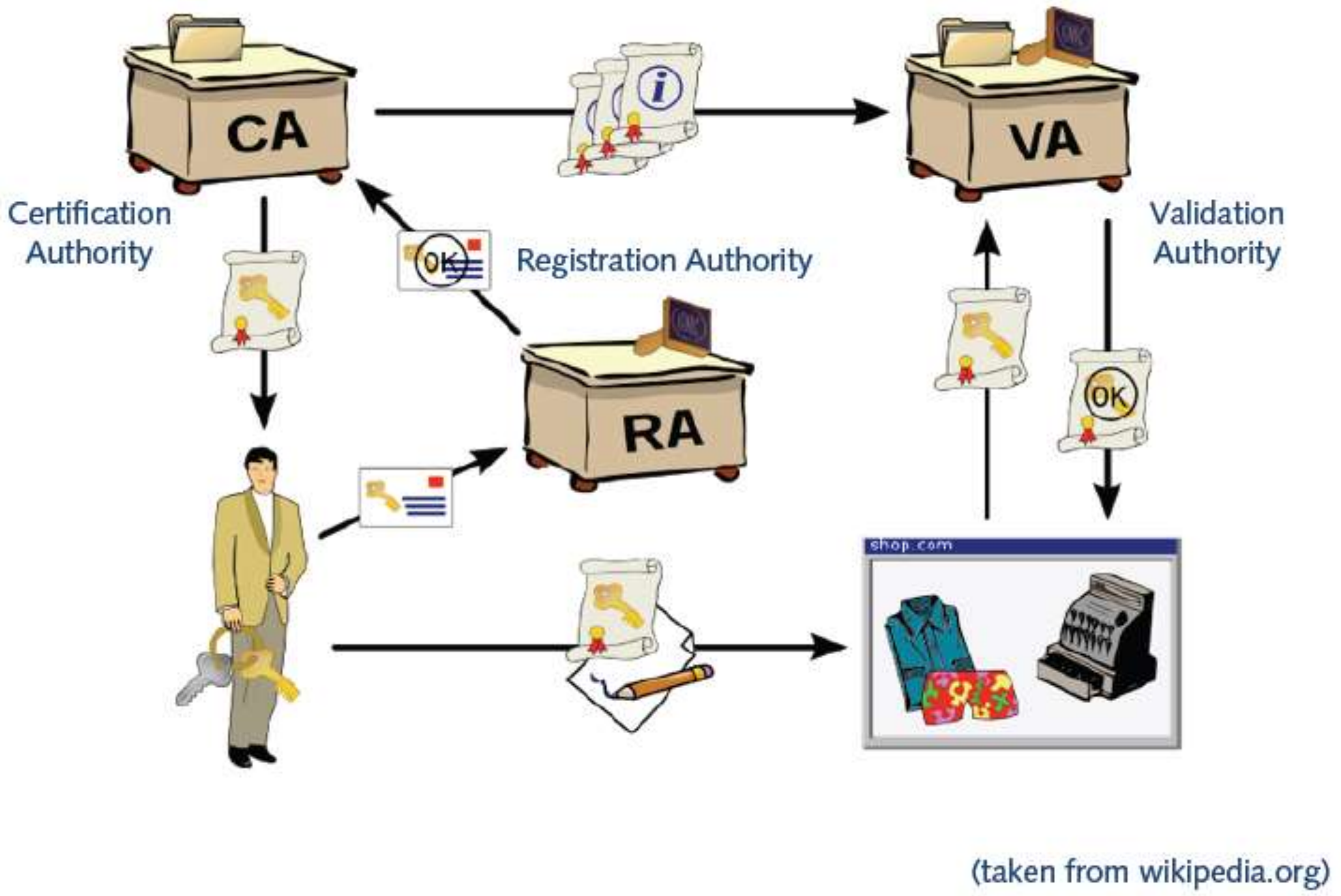
- **Public-key infrastructure** (**PKI**) is a set of hardware, software, people, policies, and procedures needed to create, manage, distribute, use, store, and revoke digital certificates.
- **Management of trust using public-key cryptography**
  - Digital certificates (signatures) on keys, attributes, ...
  - Certificate authorities (CA) as trusted parties
  - Chain of trust with multiple layers
- **Different architectures**
  - Hierarchical PKI,
    - e.g. X.509 standard
  - Web of trust,
    - e.g. PGP software


Georg-August-Universität Göttingen
https://www.uni-goettingen.d
509 form
OK

Deutsche Telekom Root CA 2
↳ DFN-Verein PCA Global – G01
↳ Universitaet-Goettingen CA
↳ www.uni-goettingen.de

www.uni-goettingen.de
Issued by: Universitaet-Goettingen CA
Expires: Dienstag, 25. September 2012 12:02:59 Mitteleuropäische Sommerzeit
This certificate is valid
▶ Details

Roles In PKI



*from wiki

Roles In PKI

- **Certification Authority (CA)**
  - Trusted third party that binds public keys with respective user identities
- **Validation Authority (VA)**
  - The user identity must be unique within each CA domain. The third-party Validation Authority (**VA**) can provide this information on behalf of CA.
- **Registration Authority (RA)**
  - The binding is established through the registration and issuance process, which, depending on the level of assurance the binding has, may be carried out by software at a CA, or under human supervision. The PKI role that assures this binding is called the Registration Authority (**RA**). ***The RA ensures that the public key is bound to the individual to which it is assigned in a way that ensures non-repudiation.***

*from wiki

Acknowledgements

Material in this lecture are taken from the slides prepared by:

- Prof. Dr. Konrad Rieck (Uni-Göttingen)