PKI: Public Key Infrastructures

PGP: limitations

- PGP's web-of-trust does not reflect the trust model adopted by several organizations, that it is usually hierarchical
- PKIs adopt a hierarchical trust model

PKI

- Developed as part of ISO X.500 framework (Directory Service) but now widespread use beyond X.500
- Also IETF, from 98 has started working groups focussed to PKI issues:
  - IPKI
  - SDSI
  - SPKI

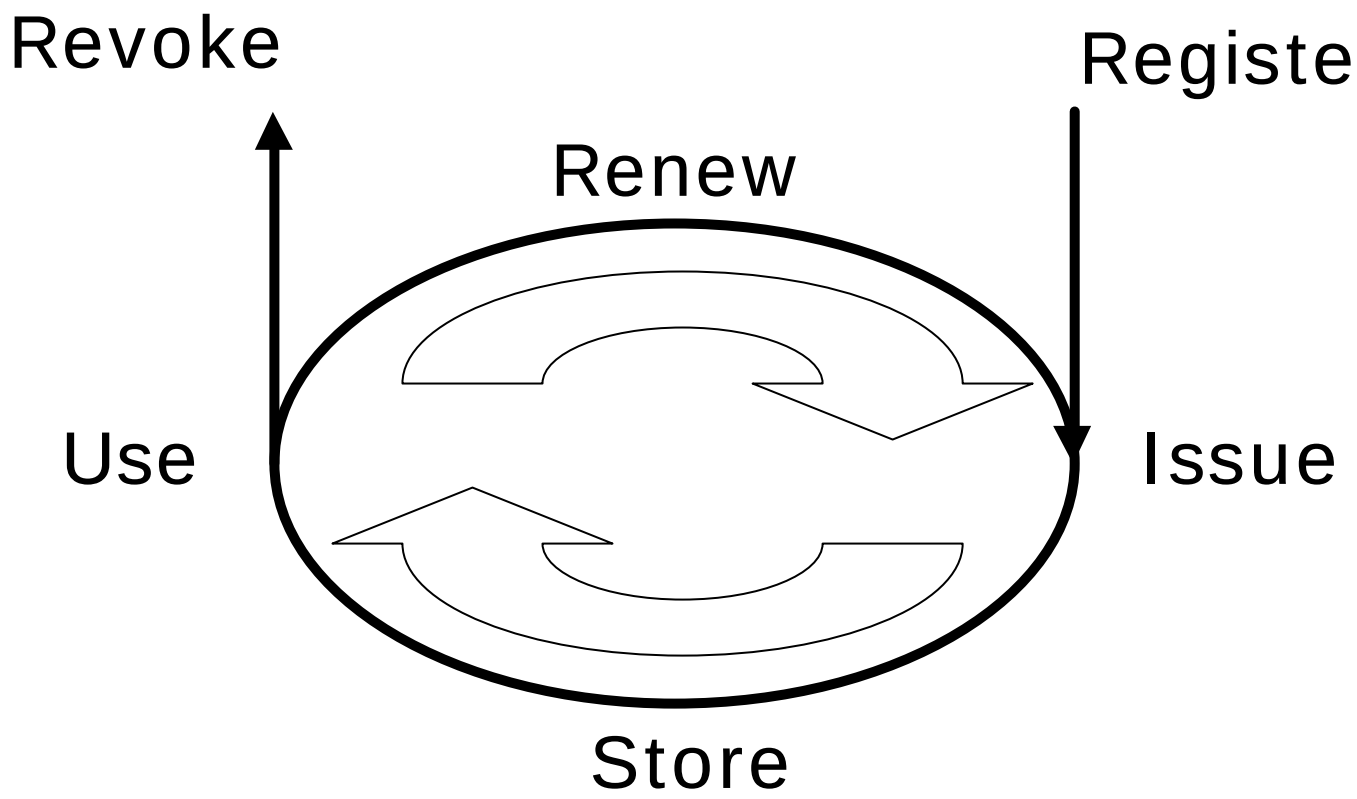
PKI

- There are two basic operations common to all PKIs:
  - Certification:process of binding a public-key value to an individual, organization or other entity
  - Validation:process of verifying that a certification is still valid

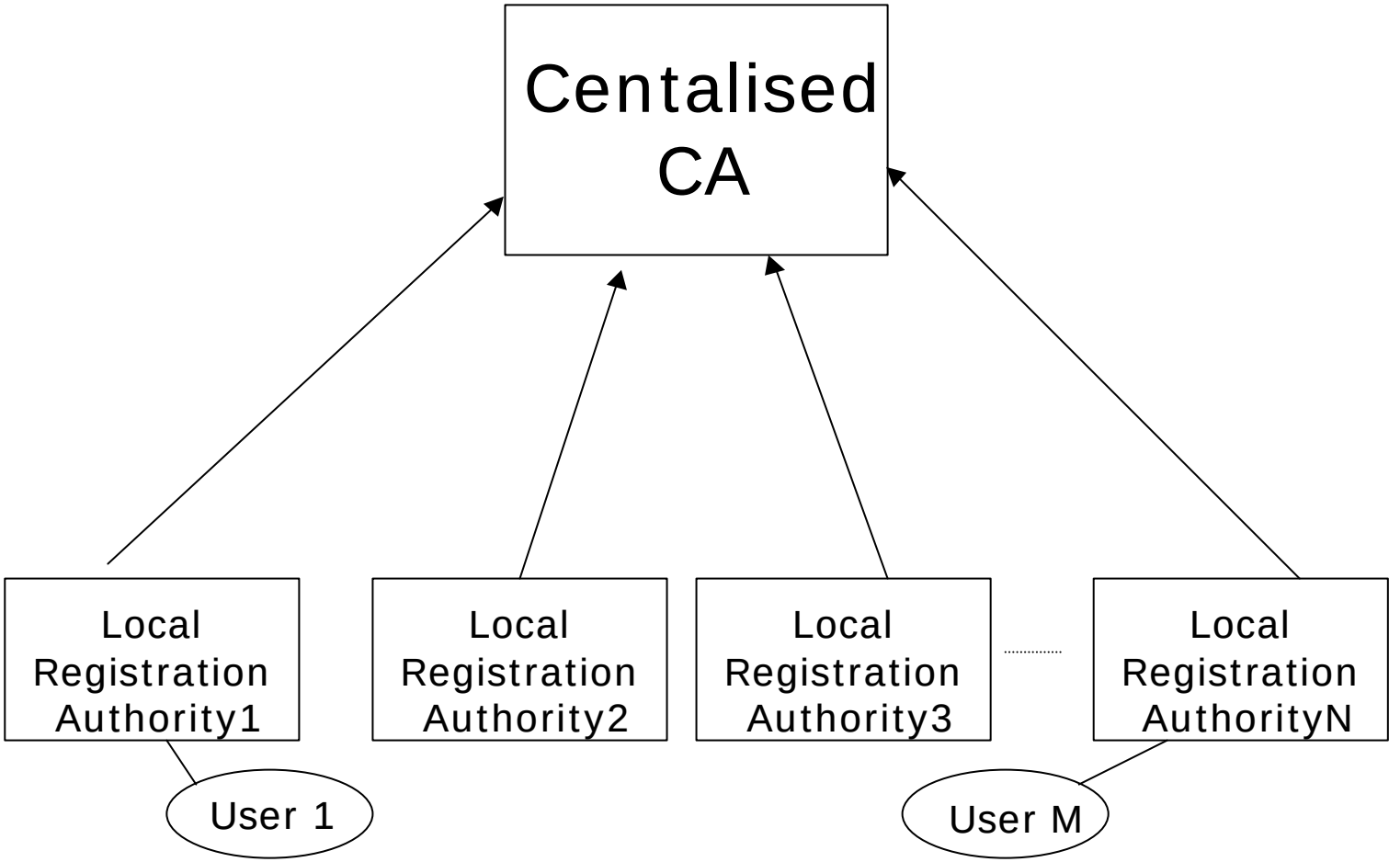
Certification

- Registration
  - Verification of entity's identity by verification of conventional credentials (e.g. passport) $\rightarrow$ Registration Authority
- Issuance
  - Upon verification, issuance of the digital binding public key $\leftrightarrow$ entity (digital certificate) $\rightarrow$ Certification Authority

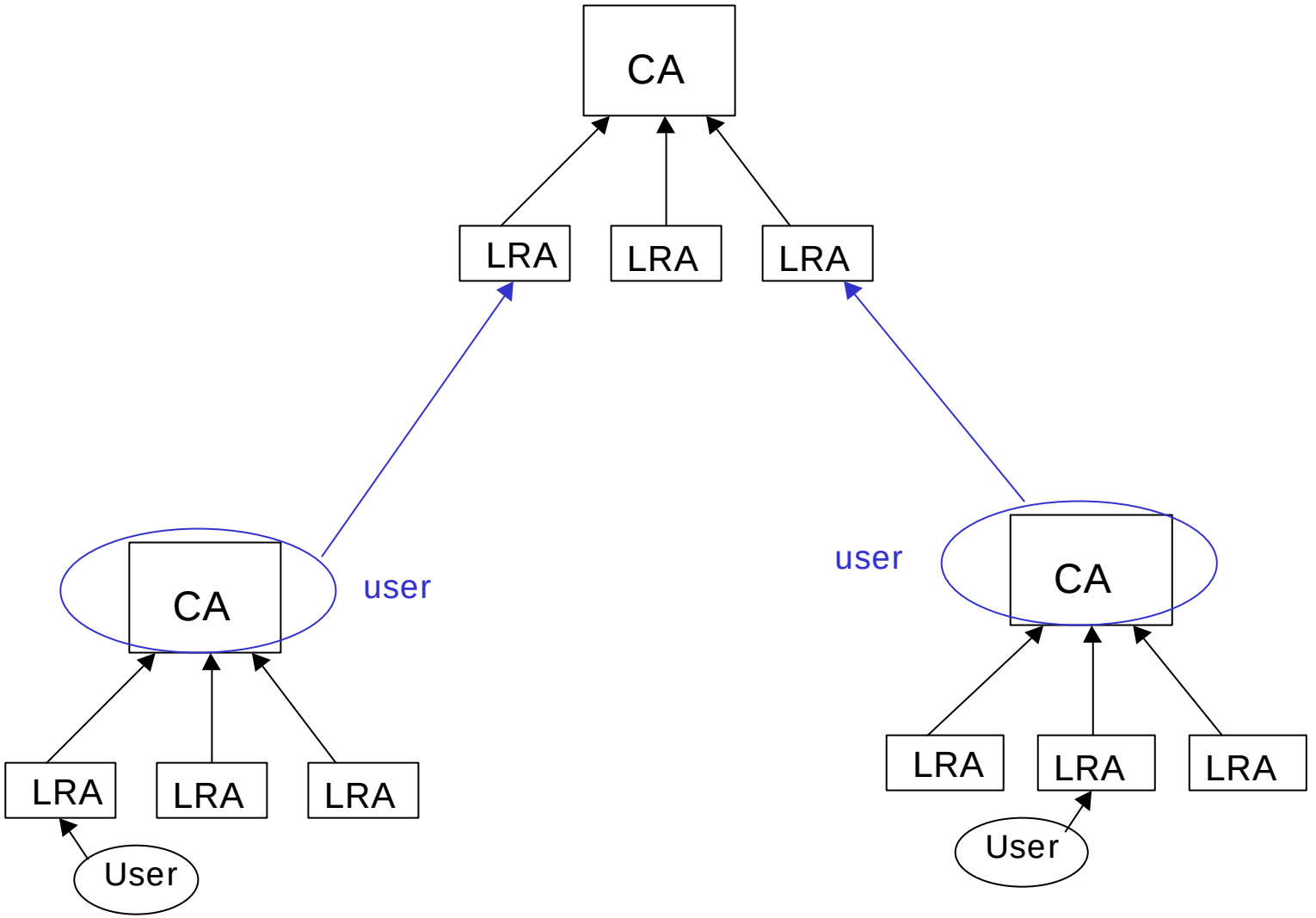
Life Cycle of a User Certificate



Registration: LRA



Distributed Registration



Registration

- The functions are separated for pratical and security reasons

Registration:

  - user presents his conventional credentials (i.e. passport) and public-key to the LRA in an authenticated manner
  - user must prove that he knows the corresponding private key (self-signed request)
  - the LRA validates and forwards to CA for certification
  - Integrity of public key must be preserved

Registration

Registration:

- LRA distribute authenticated CA's public key to users
- LRA register directly herself to the CA
- Generation and exchange of paper evidence stating that:
  - user has authorised the CA to issue a certificate for that public key *(for the CA)*
  - CA has issued a certificate for the given public key *(for the user)*

Issuance

Issuance:

- CA verifies that public has not been registered before in her domain
- CA generates certificate relying on information sent by LRA
- CA publishes certificate on a directory server and may send it to the user
- CA can operate off-line, thus more secure

Digital Certificates

A certificate:

- is the form in which a PKI communicates public key information
- a binding between a name and a public key and/or other attributes
- signed by a certificate issuer (CA)

Digital Certificate: i.e. X.509

```
Certificate          ::=    SIGNED { SEQUENCE  {
    version     [0]          Version DEFAULT v1,
    serialNumber             CertificateSerialNumber,
    signature                AlgorithmIdentifier,
    issuer                   Name,
    validity                 Validity,
    subject                  Name,
    subjectPublicKeyInfo     SubjectPublicKeyInfo,
    issuerUniqueIdentifier   UniqueIdentifier OPTIONAL,
    subjectUniqueIdentifier  UniqueIdentifier OPTIONAL,
    extensions               Extensions OPTIONAL,
} }
```

Digital Certificate: es. X.509

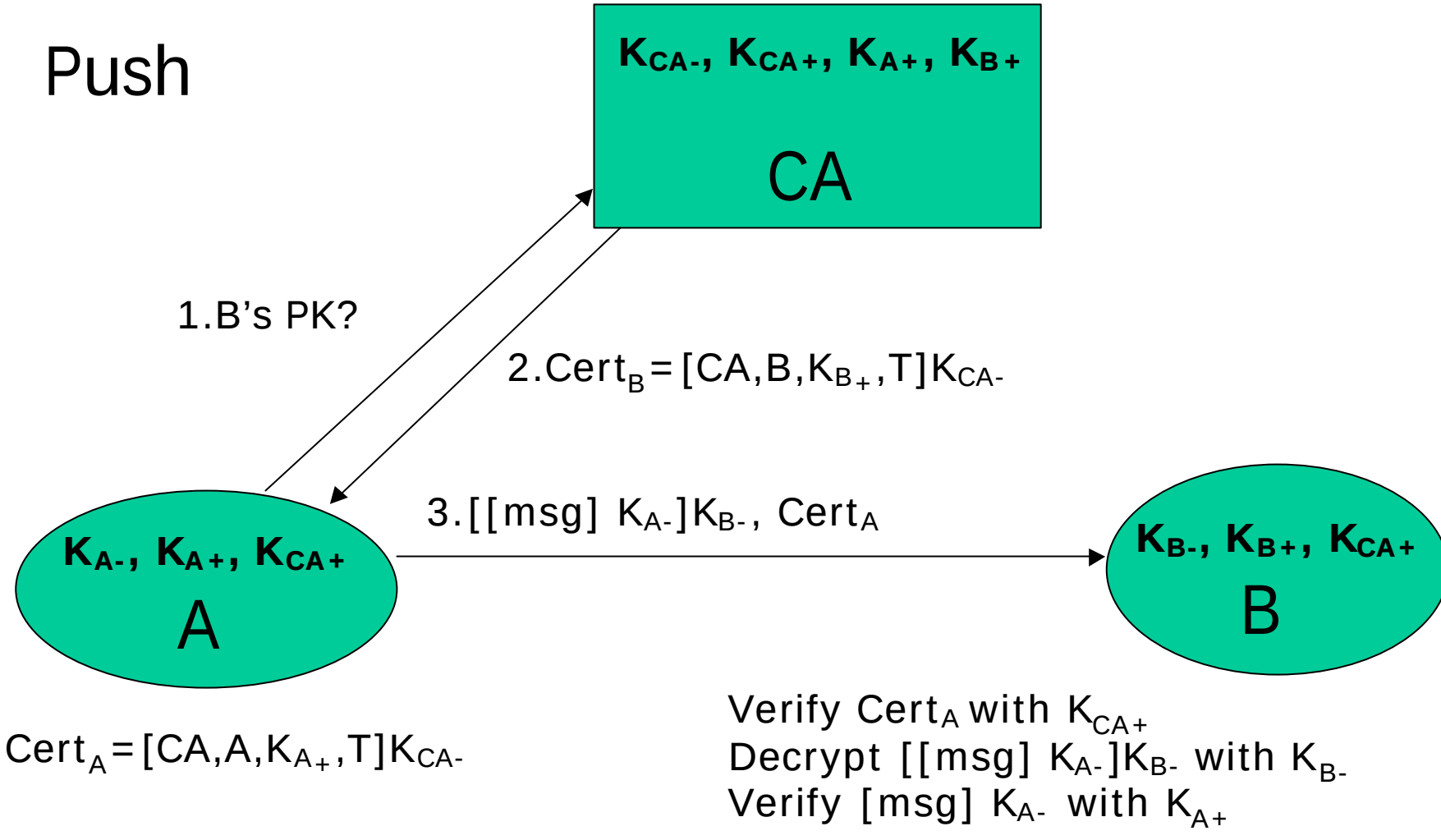
Estensioni

- KeyUsage: authentication, encryption, etc.
- Limits: e.g. only for transaction lower than 1000 Euro
- Revocation list locations
- others

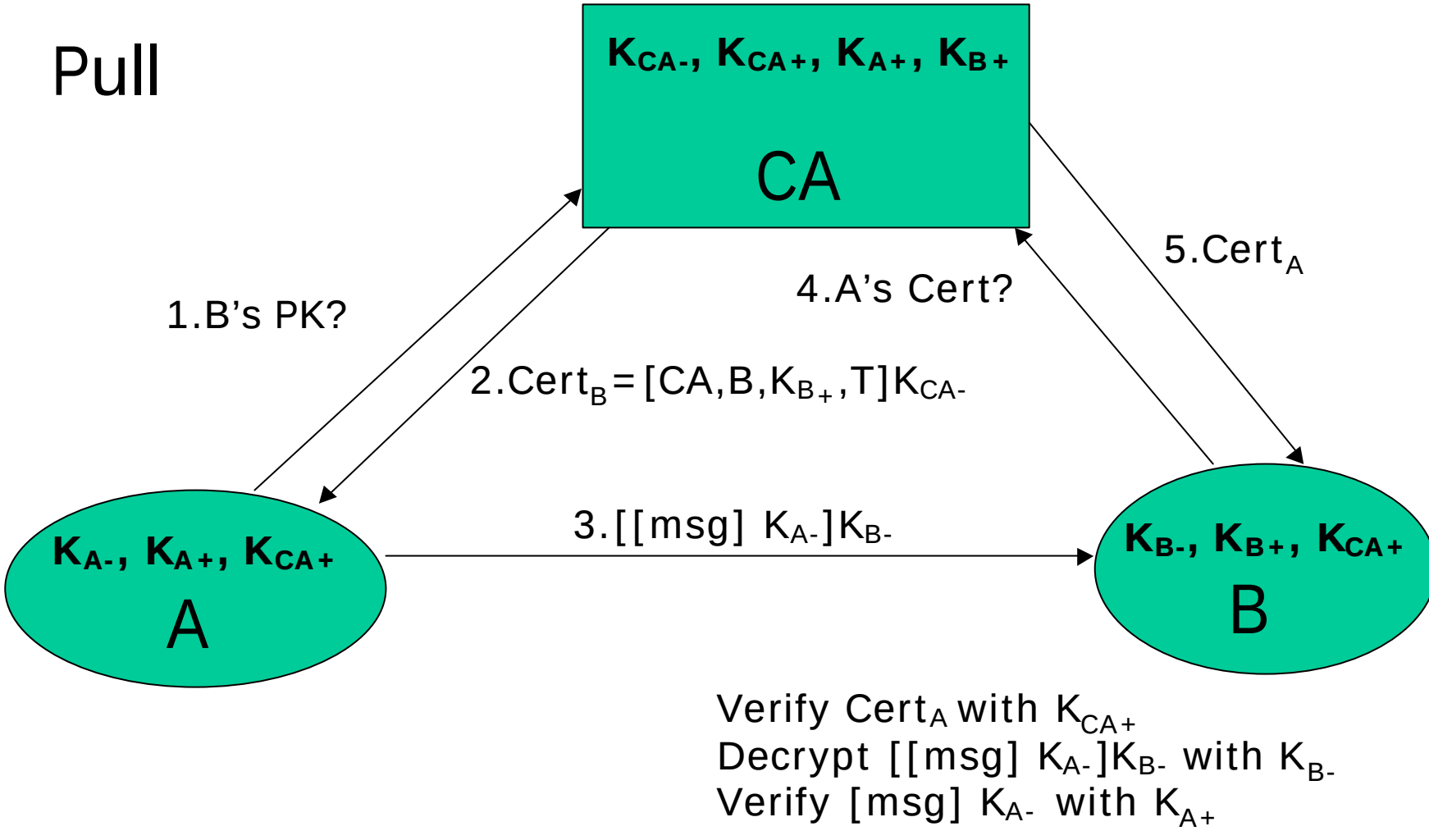
Digital Certificates

- Three types of certificates
  - End-user certificates
  - CA certificates
  - Cross-certificates

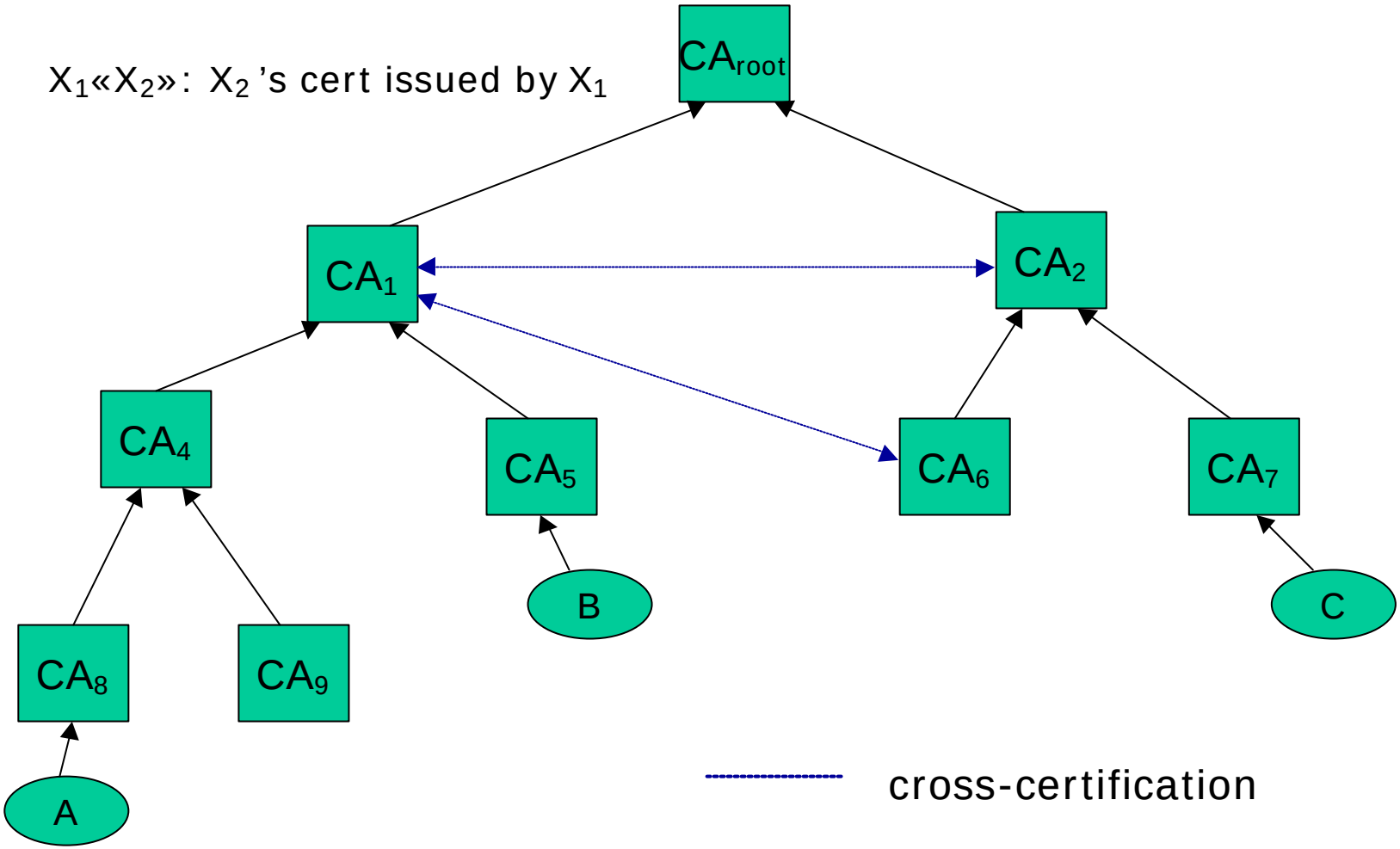
Use of digital certificates



Use of digital certificates



Trust Model



A→C: CA_8«A»,CA_4«CA_8»,CA_1«CA_4»,CA_2«CA_1»,CA_7«CA_2»
A→B: CA_8«A»,CA_4«CA_8», CA_1«CA_4», CA_1«CA_5»

Trust-paths

Trust navigation

- A certificate chain (*len³1)* is a linked list of certificates needed to verify a digital signature

CACertificate ::= {

.....

CertificatePair ::= SEQUENCE {

forward Certificate

reverse Certificate}

} K_{CA-}

Certificate expiration

- Certificates have a validity period for security reason (e.g. limit chosen-ciphertext attacks)
- By certifying a public key the certificate states that the correspondent private key can generates valid signatures
- When a certificate is expired the key in the certificate cannot be used anymore to verify signatures generated by the correspondent private key. Digital signature generated by the key are not valid

Key and certificate renewal

- Once expired a new certificate attesting the bound of a new key for that entity need to be issued
- Private keys correspondent to public keys of expired certificates need to be destroyed in order to avoid generation of invalid digital signatures with those keys

Key and certificate renewal

- Even if the certificate is invalid the public key cannot be destroyed if there are digital signatures that require that key to be verified
- Public keys need to be available until there uis even a single digital signature generated with the correspondent private key

Validation

- Once received an encrypted and signed message the receiver has to validate the message
  - Verify $Cert_A$ with K_{CA+}
  - Decrypt $[[msg]\ K_{A-}]K_{B-}$ with K_{B-}
  - Verify $[msg]\ K_{A-}$ with K_{A+}
  - If all checks are successfull the receiver can infer that the message is secret, authentic, generated by A, and has not being tampered with

- *Many things can go wrong though!!*

Validation

$A \rightarrow B$: $[msg]_{K_{A-}}$, $[CA, A, K_{A+}]_{K_{CA-}}$

if

- Verifies the digital signature ($[msg]_{K_{A-}}$) with K_{A+}

then if

- Verify that $[msg]K_{A-}$ existed at time T *timestamp*

then if

- Verifies the validity of K_{A+} *revocation*

ok

- *Being able to repeat the checks at any time or convince an independent arbitrator about the results of the checks*

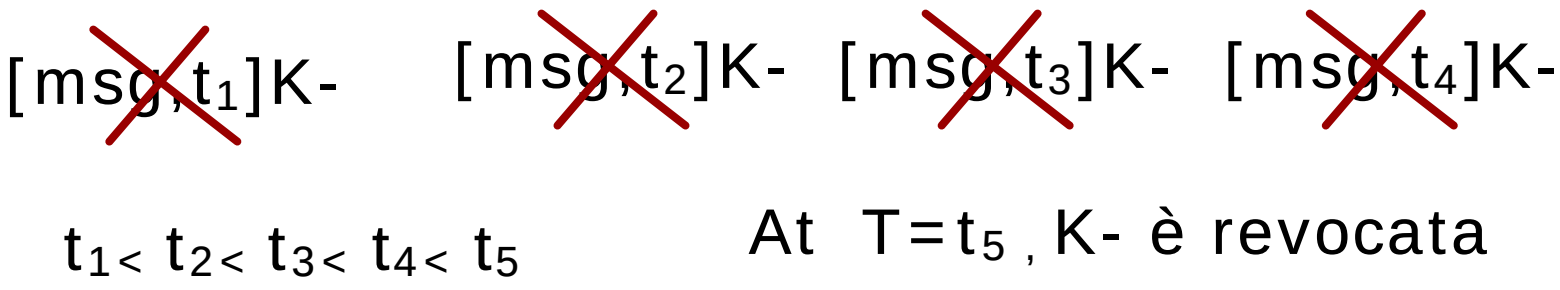
Revocation

Motivations

- Private key is stolen
- Private key is damage
- Other information change (i.e. name)

PKI: Revocation

- As consequence of revocation of a public key all the digital signatures generated by the correspondent private keys are invalid because there is no more guarantee that the key has been used by the legitimate owner or by the attacker.



Revocation

Motivations

- User signs and then change his mind so he revokes his certificate falsely claiming his key has been stolen and by doing so repudiates all his signatures

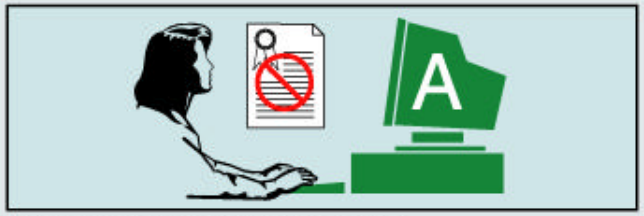
Timestamp

- Revocation solve the problem of checking the current validity of the public key but does not solve the problem of protect old signatures.
- This problem can be fixed by the use of timestamps.
- Timestamps protect already validated signatures against subsequent revocation of the key used to generate such signatures

Timestamp: problem

December 1st: A sends a digitally signed letter to B admitting that she owes him € 1000. B will receive the money at Christmas.

December 2nd: B verifies the signature and looks forward to Christmas.



December 13: A revokes her certificate.

Christmas: B asks A for the money.

Christmas: A refuses to pay claiming that the signature is forged.

After Christmas: A and B meet in court.

Timestamp: solution

- December 1st: A digitally signed a letter that admits she owes him 1000 Euro. B will receive the money at Christmas. A gets a timestamp on the signed letter. A send the signed and time-stamped letter to B
- December 2nd: B receives the letter and verifies signature and timestamp
- December 13th: A revoke her certificate
- December 25th: B asks A for his money

  A refuses to pay the money claiming her signature is forged
- A and B meet in court. The judge can see that the signature was present before A's certificate was revoked and rule that B should get the money since the signature is valid. Only signatures generated after 13th of December are invalid

Logs & Attributability

- Sender needs to log the fact that he revoked the key
- Receiver needs to log the fact that he performed revocation check
- Transaction itself need to be logged
- CA need to log revoked keys
- Timestamp request need to be logged

...all these logs need to be protect from forgeries.

Non Repudiation

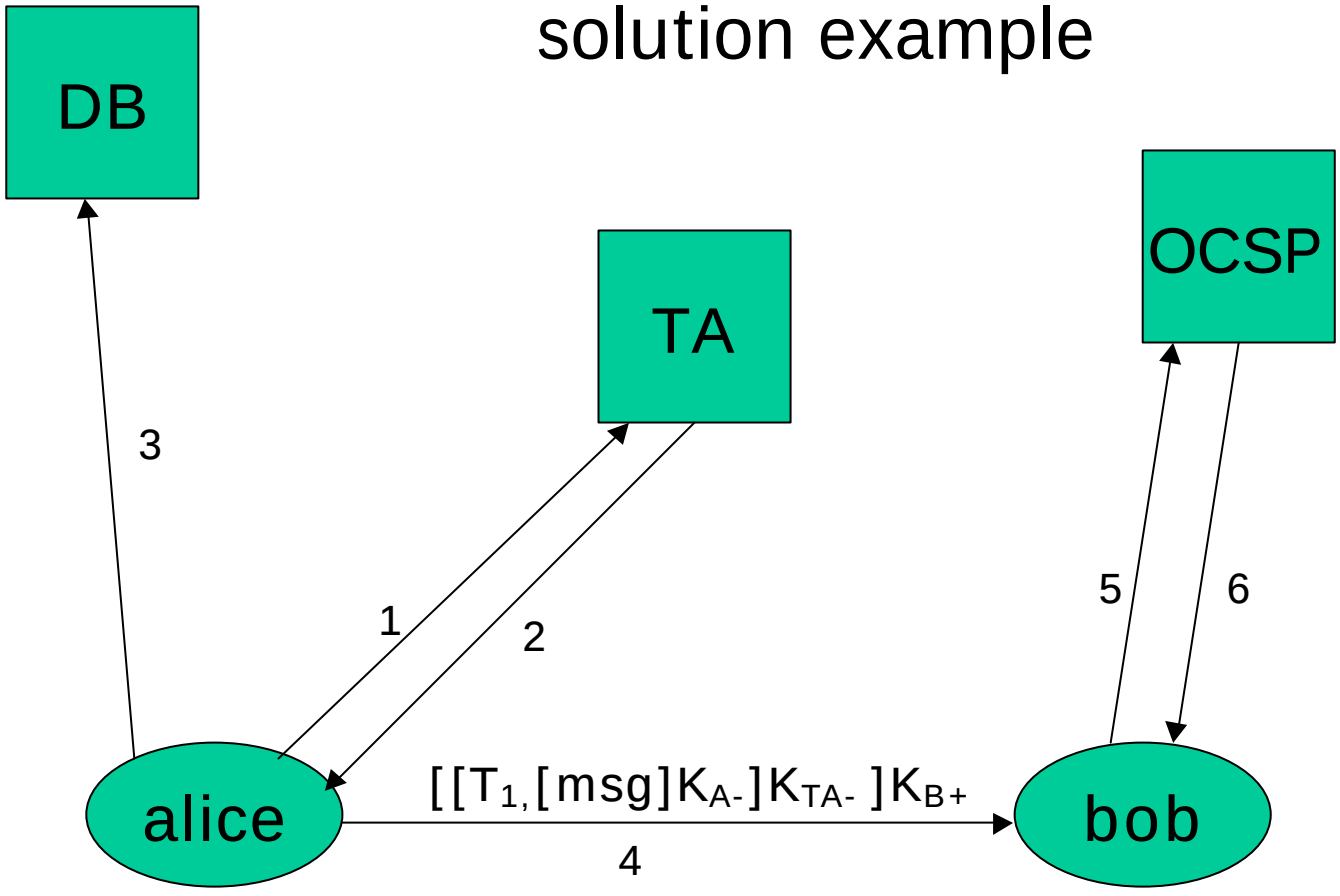
- The property for which an entity cannot falsely deny to have perfomed an action

DigSig+Revocation+Timestamp+Logs

Non repudiation

Non-Repudiation

solution example



CA Administration

- The Certification Authority is the most critical component of the infrastructure
  - His key pair initialise and distribute *trust relations* (by mean of certificates)
  - Revocation and distribution of lists
  - Compromising (loss or steal) CA's private key affect all the issued certificates and revocation

CA Administration: Dual Control

- Often distributed among several roles
  - *SSO*: System Security Offficer. Generates CA keys, creates roles, registers LRAs
  - *CAA*: CA Adminsitrator. Issuing user certificate, revoking them, publishing CRL
  - *LRAA*: LRA Administrator: registers users.
  - *SA*: System Administrator: shutting down the server

Secret sharing

- Because of its importance, CA's private key can be distributed and protected by using a secret sharing scheme
- A single admin cannot access the private key. There must be the collaboration of at least $N \geq 2$ administrators

Secret sharing

(3,n) scheme (with three shadows)

$F(x)=(ax^2 + bx + c) \bmod p$ p prime

a,b random

c=key

shadows=$k_i=F(x_i)$ $i=1,2,3,....$

3 shadows can rebuilt the secret, 2 can’t

Secret sharing: example

(3,5) scheme

$$F(x)=(7x^2 + 8x + 11) \bmod 13$$

shadows

$k_1=F(1)=7+8+11 \equiv 0 \pmod{13}$

$k_2=F(2)=28+16+11 \equiv 3 \pmod{13}$

$k_3=F(3)=63+24+11 \equiv 7 \pmod{13}$

$k_4=F(4)=112+32+11 \equiv 12 \pmod{13}$

$k_5=F(5)=175+40+11 \equiv 5 \pmod{13}$

3 shadows (i.e. K_1, k_3, k_5)

$a*2^2+b*2+c \equiv 0 \pmod{13}$

$a*2^2+b*2+c \equiv 7 \pmod{13}$

$a*2^2+b*2+c \equiv 5 \pmod{13}$

$a=7,\ b=8 \rightarrow c=11$

CA: Auxiliary services

- Key generation on behalf of users.
  - pros: competence, centralised admin
  - cons: privacy, non-repudiation
- Key Recovery
  - Many times key back-up is needed to prevent the loss/damage of
    - Signature keys
    - Encryption keys
- Key Escrow
  - Government agency impose deposit of private key (only encryption keys can be justified)

Directory Service

- Service to publish users profiles with certificate and revocation information
- Can be used also to archive log of transactions
- Can be distributed and replicated because information is public
- Anybody can query it. Only CA can write information
- LDAP: *Lightweight Directory Access Protocol*, standard used for CA$\leftrightarrow$Dir $\leftrightarrow$ users communications

Revocation

Revocation Mechanisms

- CRL: Certificate Revocation List
- deltaCRL
- OCSP: Online Certificate Status Protocol
- SCVP: Simple Certificate Verification Protocol

....

CRL

CertificateList ::= {
thisUpdate Time,
nextUpdate Time
{CertSerialNumber1, RevocationTime},
{CertSerialNumber2, RevocationTime},
{CertSerialNumber3, RevocationTime},
....
} K_{CA-}

- CRL need to be signed also if empty or has not being changed from the previous time of issue

CRL: limitations

- Updates at prefixed time only allow window of uncertainty
- Size of CRL. The entire list is issued each time even if no certificates are added
- The distribution point of the CRL is indicated in the certificates. Not very flexible

deltaCRL

- To reduce size and frequency problems
- New updates contain only fresh revoked certificates since the last deltaCRL. Only the new updates are sent. However the receiver needs to verify against the entire list.

OCSP/SCVP

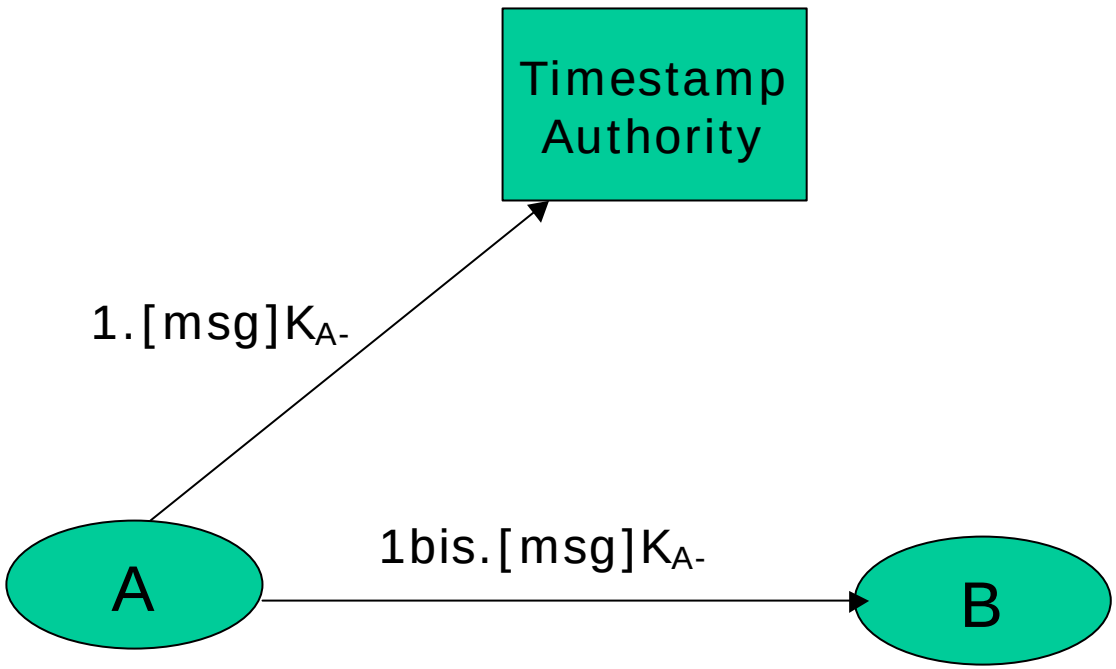
- Often users are interested to check if a particular certificate has been revoked and not all the revoked certificates.
- OCSP allows to query the current state of a single certificate
- OCSP's database of revoked certificates is updated by the CA with a back-office protocol

Timestamp

Timestamps

- Independent timestamp service that certifies that a signature existed at a certain point in time t_x
- The timestamp service does not need to know the content of the signed document. Its hash is enough.
- Several mechanisms has been proposed

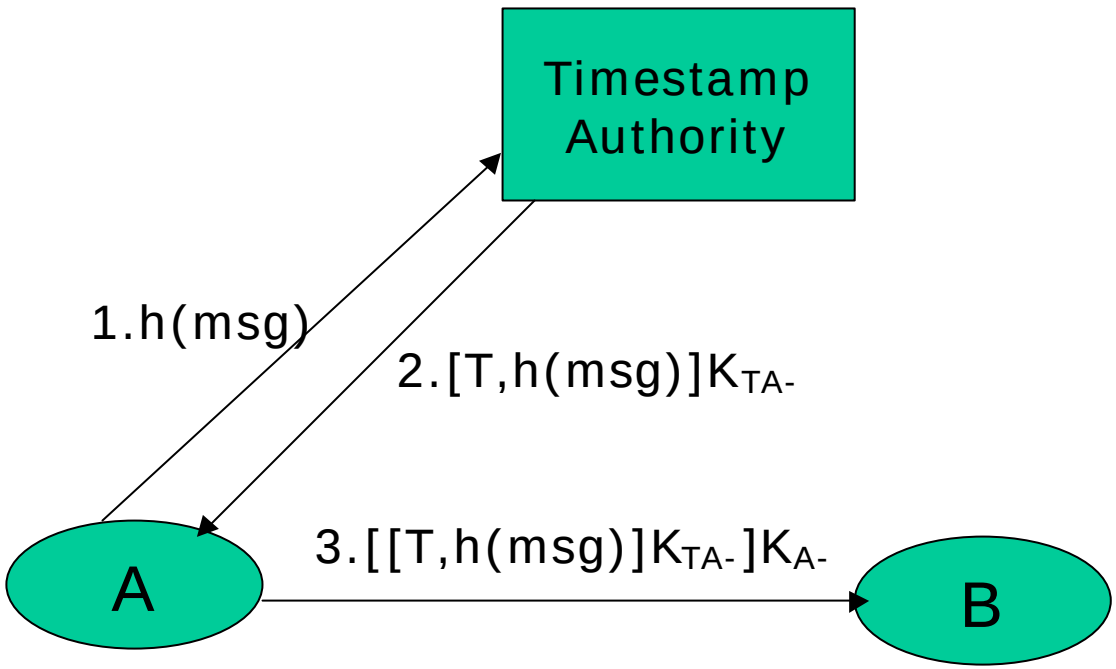
Timestamp: solution 1



Limits

- No privacy
- Timestamp must be trusted
- Subject to passive attack
- Not reliable
- Massive storage

Timestamp solution 2: push



Adopted by IETF and de facto standard

Timestamp solution 2: pull