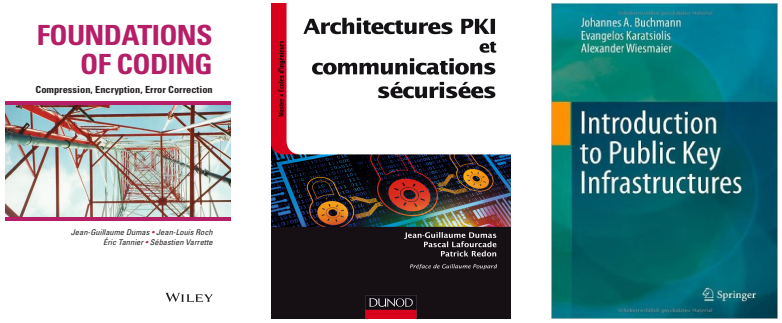
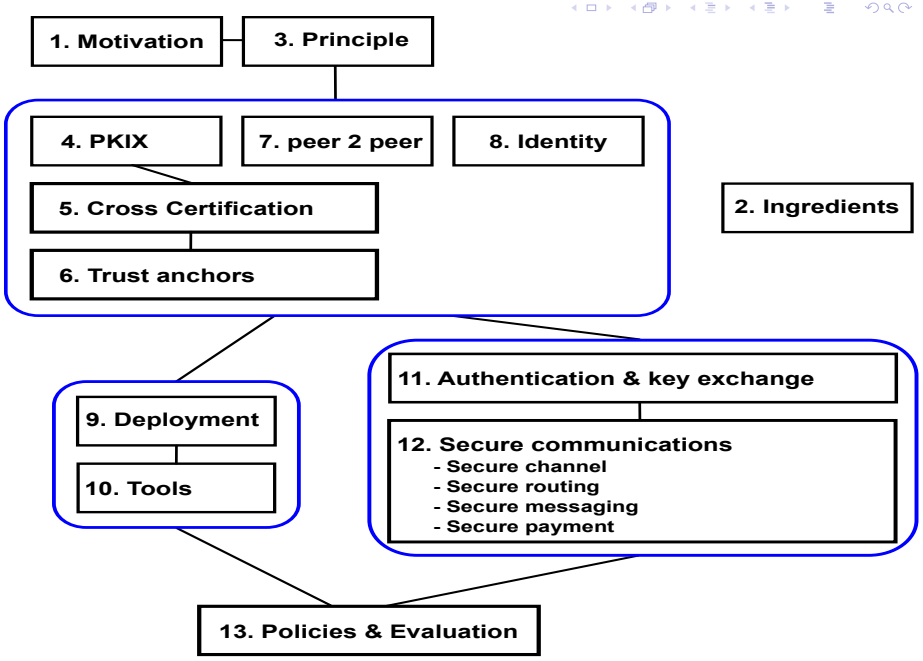
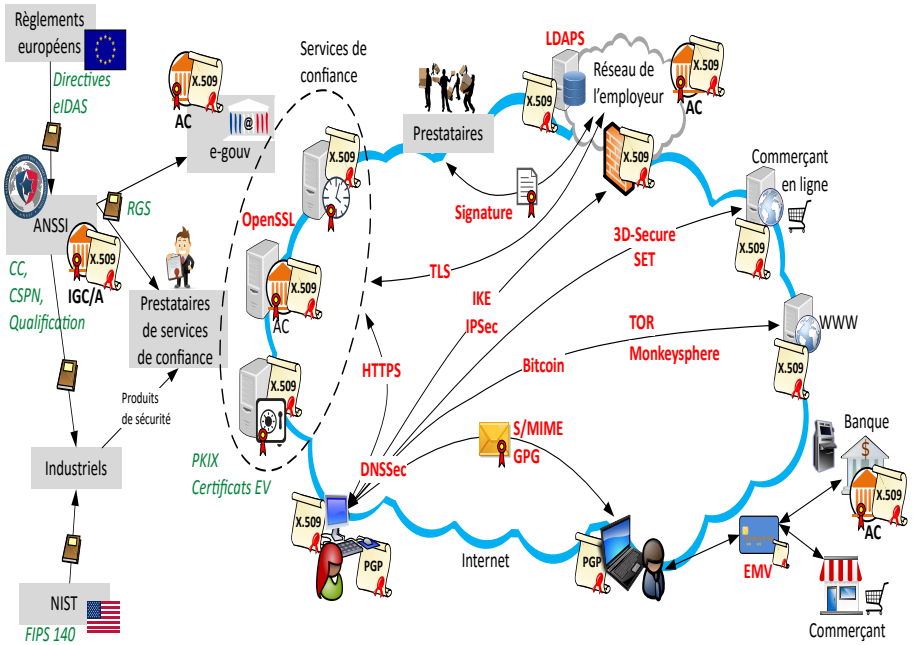
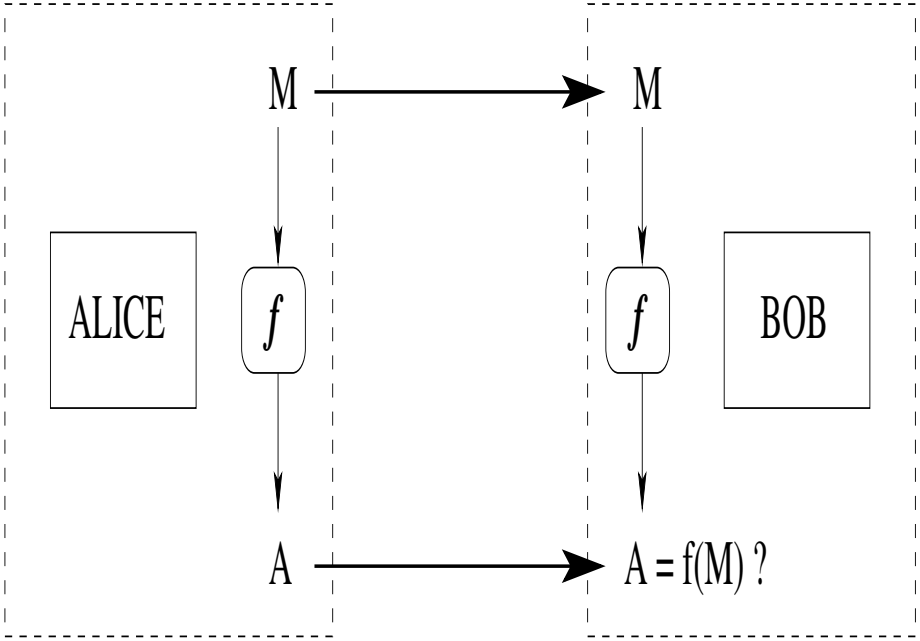
PKI architectures: references

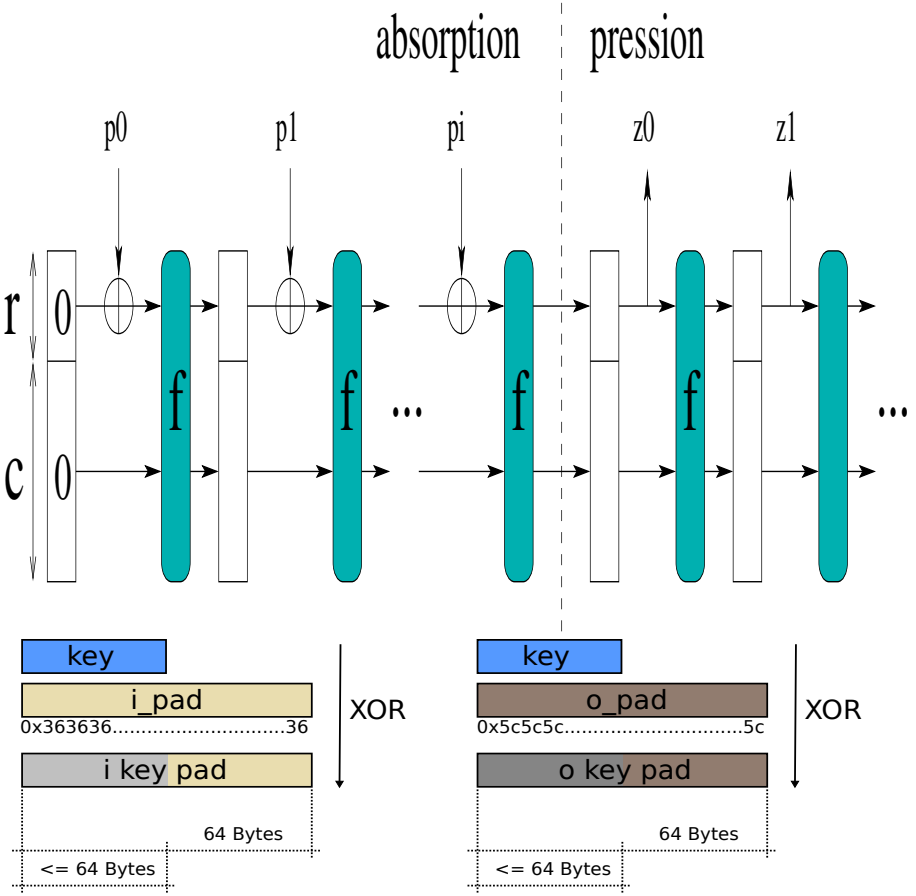


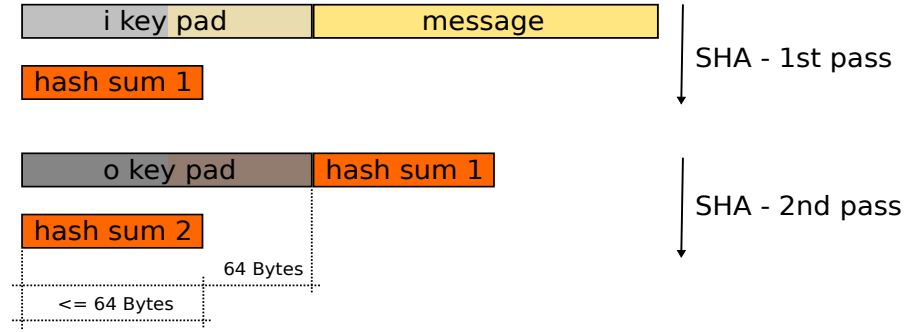

ljk.imag.fr/membres/Jean-Guillaume.Dumas/Enseignements/PKI

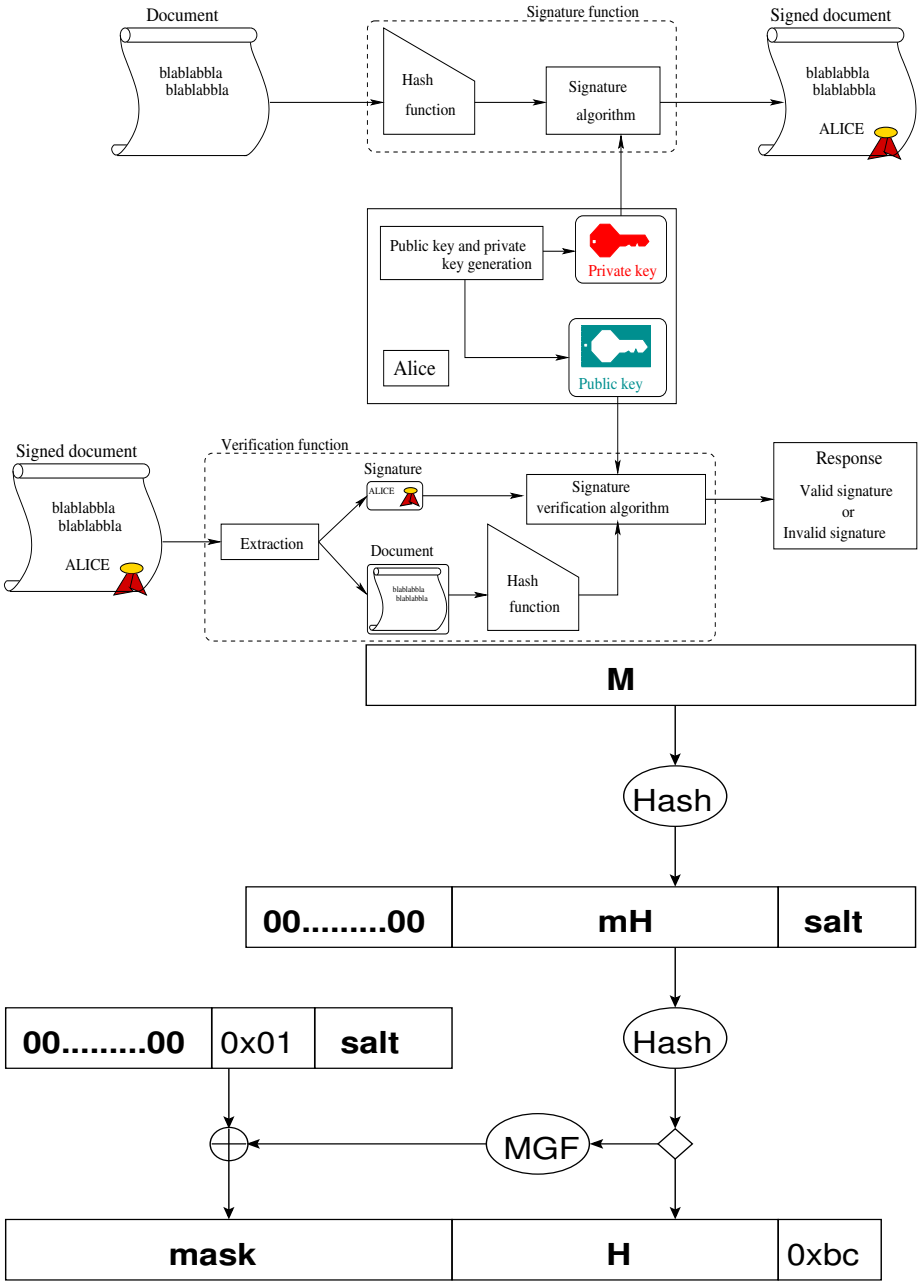


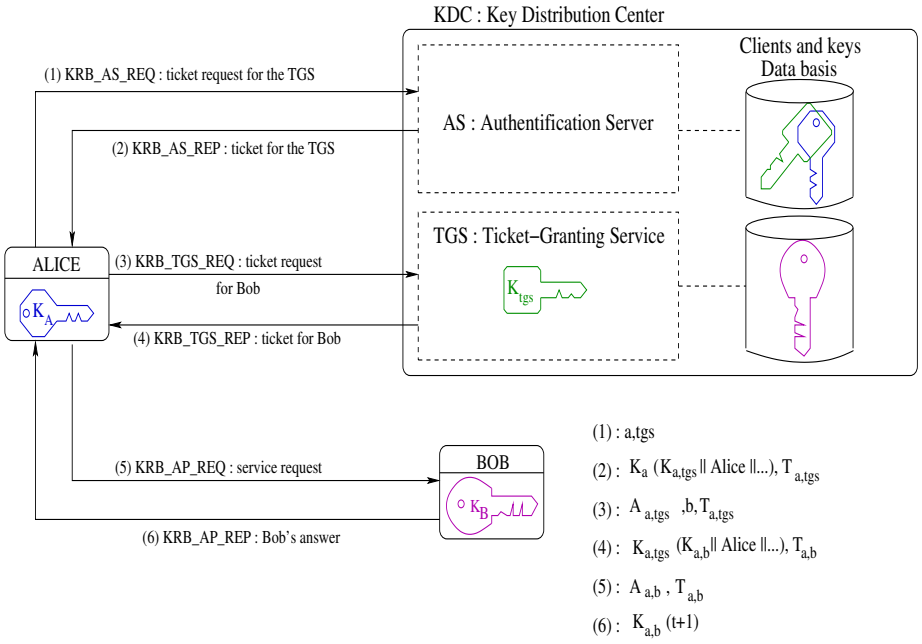

absorption
pression
p0
p1
pi
z0
z1
r
c
0
0
f
f
f
f
key
i_pad
0x363636.....36
XOR
i key pad
64 Bytes
<= 64 Bytes
key
o_pad
0x5c5c5c.....5c
XOR
o key pad
64 Bytes
<= 64 Bytes

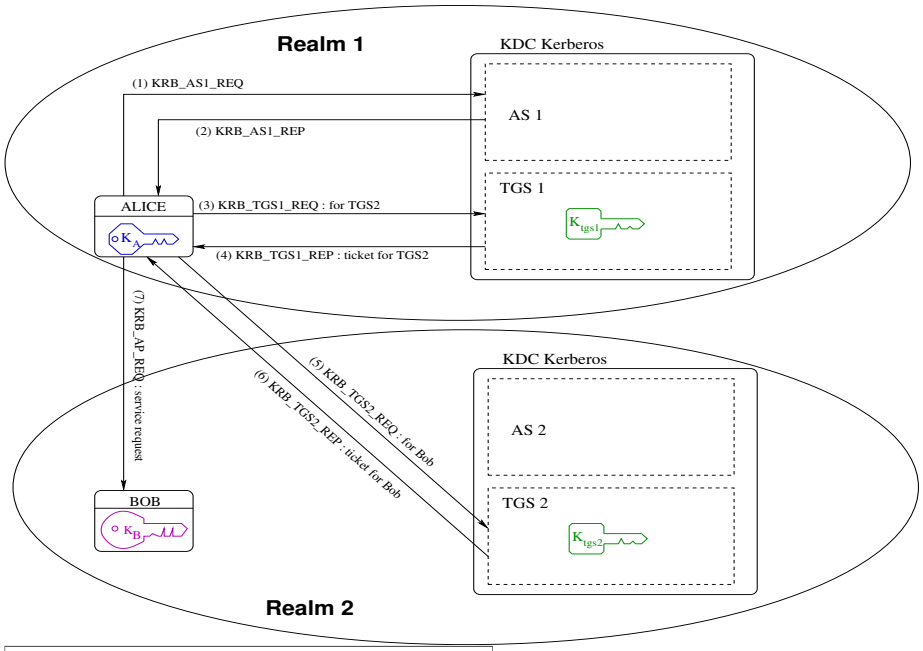

i key pad
message
SHA - 1st pass
hash sum 1
o key pad
hash sum 1
SHA - 2nd pass
hash sum 2
64 Bytes
<= 64 Bytes

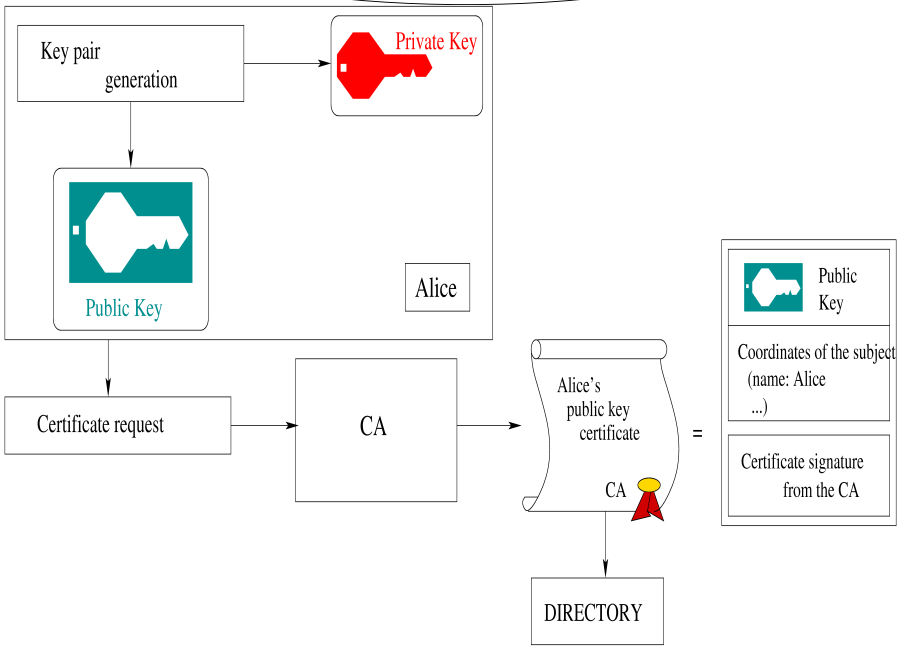

Document
blablabbla
blablabbla
Hash
function
Signature function
Signature
algorithm
Signed document
blablabbla
blablabbla
ALICE
Public key and private
key generation
Private key
Alice
Public key
Signed document
blablabbla
blablabbla
ALICE
Verification function
Signature
ALICE
Extraction
Document
Hash
function
Signature
verification algorithm
Response
Valid signature
or
Invalid signature
M
Hash
00.....00
mH
salt
00.....00
0x01
salt
Hash
MGF
mask
H
0xbc

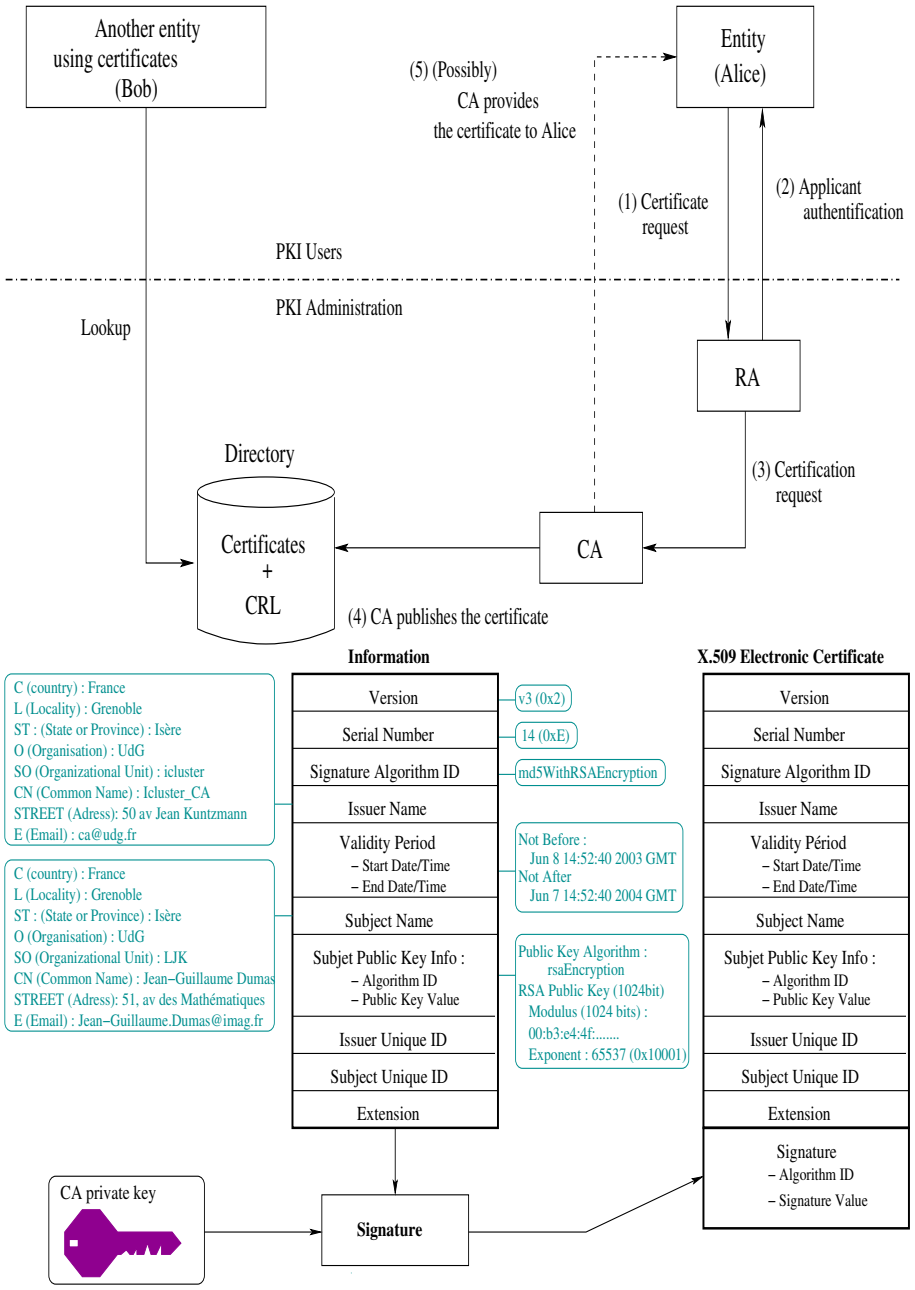
**PKCS Standards Summary, From Wikipedia**

| | Version | Name | Comments |
|---|---|---|---|
| **PKCS #1** | 2.1 | RSA Cryptography Standard[1] | See RFC 3447. Defines the mathematical properties and format of RSA public and private keys (ASN.1-encoded in clear-text), and the basic algorithms and encoding/padding schemes for performing RSA encryption, decryption, and producing and verifying signatures. |
| **PKCS #2** | - | *Withdrawn* | No longer active as of 2010. Covered RSA encryption of message digests; subsequently merged into PKCS #1. |
| **PKCS #3** | 1.4 | Diffie–Hellman Key Agreement Standard[2] | A cryptographic protocol that allows two parties that have no prior knowledge of each other to jointly establish a shared secret key over an insecure communications channel. |
| **PKCS #4** | - | *Withdrawn* | No longer active as of 2010. Covered RSA key syntax; subsequently merged into PKCS #1. |
| **PKCS #5** | 2.0 | Password-based Encryption Standard[3] | See RFC 2898 and PBKDF2. |
| **PKCS #6** | 1.5 | Extended-Certificate Syntax Standard[4] | Defines extensions to the old v1 X.509 certificate specification. Obsoleted by v3 of the same. |
| **PKCS #7** | 1.5 | Cryptographic Message Syntax Standard[5] | See RFC 2315. Used to sign and/or encrypt messages under a PKI. Used also for certificate dissemination (for instance as a response to a PKCS#10 message). Formed the basis for S/MIME, which is as of 2010 based on RFC 5652, an updated Cryptographic Message Syntax Standard (CMS). Often used for single sign-on. |
| **PKCS #8** | 1.2 | Private-Key Information Syntax Standard[6] | See RFC 5208. Used to carry private certificate keypairs (encrypted or unencrypted). |
| **PKCS #9** | 2.0 | Selected Attribute Types[7] | See RFC 2985. Defines selected attribute types for use in PKCS #6 extended certificates, PKCS #7 digitally signed messages, PKCS #8 private-key information, and PKCS #10 certificate-signing requests. |
| **PKCS #10** | 1.7 | Certification Request Standard[8] | See RFC 2986. Format of messages sent to a certification authority to request certification of a public key. See certificate signing request. |
| **PKCS #11** | 2.20 | Cryptographic Token Interface[9] | Also known as "Cryptoki". An API defining a generic interface to cryptographic tokens (see also Hardware Security Module). Often used in single sign-on, Public-key cryptography and disk encryption[10] systems. |
| **PKCS #12** | 1.0 | Personal Information Exchange Syntax Standard[11] | Defines a file format commonly used to store private keys with accompanying public key certificates, protected with a password-based symmetric key. PFX is a predecessor to PKCS#12.<br>This container format can contain multiple embedded objects, such as multiple certificates. Usually protected/encrypted with a password. Usable as a format for the Java key store and to establish client authentication certificates in Mozilla Firefox. Usable by Apache Tomcat. |
| **PKCS #13** | – | Elliptic Curve Cryptography Standard[12] | *(Under development as of 2012.)*[13] |
| **PKCS #14** | – | Pseudo-random Number Generation | *(Under development as of 2012.)*[13] |
| **PKCS #15** | 1.1 | Cryptographic Token Information Format Standard[14] | Defines a standard allowing users of cryptographic tokens to identify themselves to applications, independent of the application's Cryptoki implementation (PKCS #11) or other API. RSA has relinquished IC-card-related parts of this standard to ISO/IEC 7816-15.[15] |






| | |
|---|---|
| *Version* : | Version du format de certificat X.509 |
| *Numéro de série* : | Numéro de série du certificat(propre à chaque AC) |
| *Algorithme de signature (OID)* : | Identifiant des types d'algorithmes utilisés pour la signature du certificat |
| *Nom de l'émetteur* : | *Distinguished Name* (DN) de l'AC émettrice du certificat |
| *Période de validité* : | Période de validité du certificat |
| *Nom du sujet* : | *Distinguished Name* (DN) du détenteur de la clef publique |
| *Clef publique du sujet* : | Informations sur la clef publique du certificat |
| *Issuer Unique ID* : | Identifiant unique de l'émetteur du certificat |
| *Subject Unique ID* : | Identifiant unique du détenteur de la clef publique |
| *Extensions* : | Extensions génériques optionnelles |
| *Signature* : | Signature numérique par l'AC sur les champs précédents |

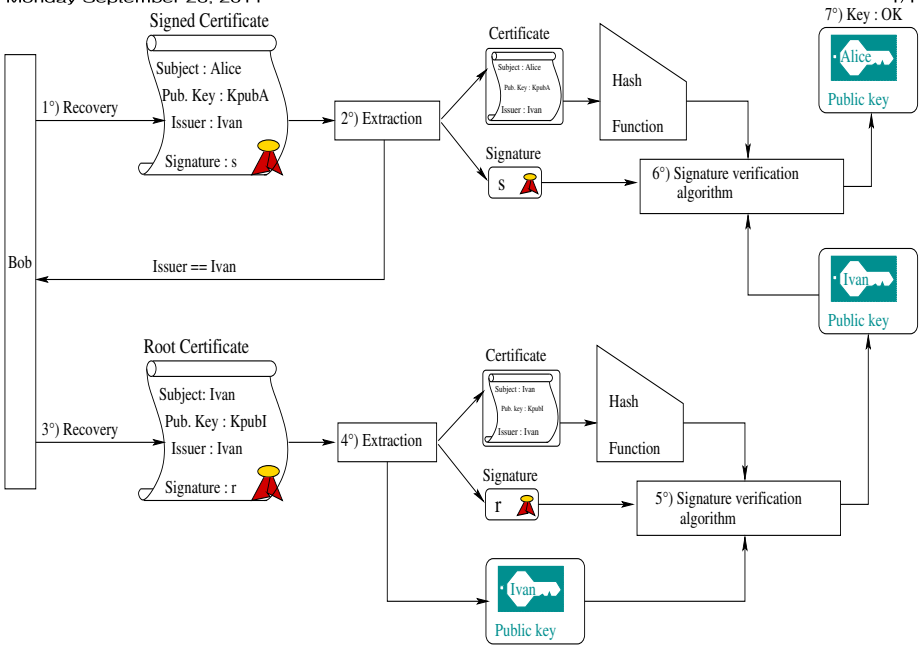
| | |
|---|---|
| *Version* : | X.509 format version number |
| *Serial number* : | certificate serial number (unique for each CA) |
| *Signature Algo. (OID)* : | Identifier of the used algorithm for the certificate signature |
| *Issuer name* : | *Distinguished Name* (DN) of the CA who created the certificate |
| *Validity period* : | The certificate lifetime |
| *Subject name* : | *Distinguished Name* (DN) of the owner of the certificate |
| *Subject pub. key info.* : | Data for the public key of the certificate |
| *Issuer Unique ID* : | Unique identifier for the CA issuing the certificate |
| *Subject Unique ID* : | unique identifier for the owner of the certificate |
| *Extensions* : | Optional generic extentions |
| *Signature* : | Electronic signature by the CA of all the previous fields |

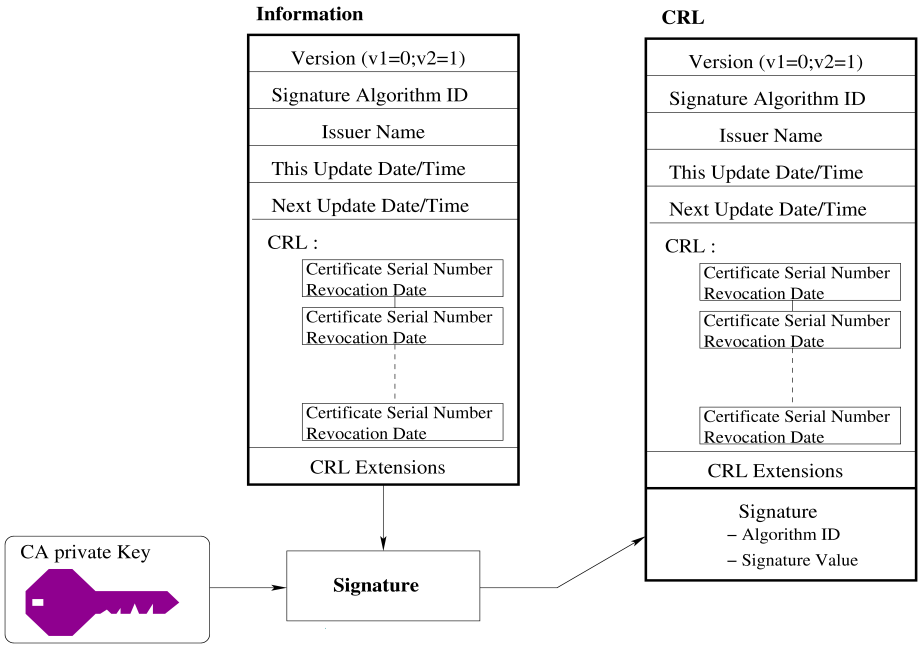


```
Certificate  ::=  SEQUENCE  {
        tbsCertificate       TBSCertificate,
        signatureAlgorithm   AlgorithmIdentifier,
        signatureValue       BIT STRING  }

TBSCertificate  ::=  SEQUENCE  {
        version         [0]  EXPLICIT Version DEFAULT v1,
        serialNumber         CertificateSerialNumber,
        signature            AlgorithmIdentifier,
        issuer               Name,
        validity             Validity,
        subject              Name,
        subjectPublicKeyInfo SubjectPublicKeyInfo,
        issuerUniqueID  [1]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        subjectUniqueID [2]  IMPLICIT UniqueIdentifier OPTIONAL,
                             -- If present, version MUST be v2 or v3
        extensions      [3]  EXPLICIT Extensions OPTIONAL
                             -- If present, version MUST be v3
        }
-----------------------------------------------------------------------
Version  ::=  INTEGER  {  v1(0), v2(1), v3(2)  }
-----------------------------------------------------------------------
CertificateSerialNumber  ::=  INTEGER
-----------------------------------------------------------------------
AlgorithmIdentifier  ::=  SEQUENCE{
        algorithm       OBJECT IDENTIFIER,
        parameters      ANY DEFINED BY algorithm OPTIONAL}
-----------------------------------------------------------------------
Validity ::= SEQUENCE {
        notBefore       Time,
        notAfter        Time }

Time  ::=  CHOICE {
        utcTime         UTCTime,
        generalTime     GeneralizedTime }
-----------------------------------------------------------------------
UniqueIdentifier  ::=  BIT STRING
-----------------------------------------------------------------------
SubjectPublicKeyInfo  ::=  SEQUENCE  {
        algorithm       AlgorithmIdentifier,
        subjectPublicKey     BIT STRING  }
-----------------------------------------------------------------------
Extensions  ::=  SEQUENCE SIZE (1..MAX) OF Extension

Extension  ::=  SEQUENCE  {
        extnID      OBJECT IDENTIFIER,
        critical    BOOLEAN DEFAULT FALSE,
        extnValue   OCTET STRING  }
-----------------------------------------------------------------------
Name ::= CHOICE {
        rdnSequence RDNSequence }

RDNSequence ::= SEQUENCE OF RelativeDistinguishedName

RelativeDistinguishedName ::= SET SIZE (1..MAX) OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
  type     AttributeType,
  value    AttributeValue }

AttributeType ::= OBJECT IDENTIFIER

AttributeValue ::= ANY DEFINED BY AttributeType
```

```
{ version v3,
  serialNumber 1234567890,
  signature
  {
    algorithm { 1 2 840 113549 1 1 5 }, -- SHA1RSA
    parameters RSAParams : NULL
  },
  issuer rdnSequence :
  { { {
        type { 2 5 4 6 },
        value PrintableString : "DE" } },
    { {
        type { 2 5 4 10 },
        value UTF8String :
          "GMD - Forschungszentrum Informationstechnik GmbH" }
    }
  },
  validity
  {
    notBefore utcTime : "000501100000Z",
    notAfter utcTime : "001101100000Z"
  },
  subject rdnSequence :
  { { {
        type { 2 5 4 6 },
        value PrintableString : "DE" } },
    { {
        type { 2 5 4 10 },
        value UTF8String : } },
    { {
        type { 2 5 4 4 },
        value UTF8String : "Barzin" },
      {
        type { 2 5 4 42 },
        value UTF8String : "Petra" } }
  },
  subjectPublicKeyInfo
  {
    algorithm
    {
      algorithm { 1 2 840 113549 1 1 1 }, -- RSA
      parameters RSAParams : NULL
    },
    subjectPublicKey '00110000 10000001 10000111 00000010 1000 ...'B
  },
  extensions
  {
    {
      extnId { 2 5 29 9 },   -- subjectDirectoryAttributes
      extnValue '305B301006082B0601050507090431041302445300F0 ...'H },
    {
      extnId { 2 5 29 15 }, -- keyUsage
      critical TRUE,
      extnValue '03020640'H },
    {
      extnId { 2 5 29 32 }, -- certificatePolicies
      extnValue '3009300706052B24080101'H },
    {
      extnId { 2 5 29 35 }, -- authorityKeyIdentifier
      extnValue '3016801400010203040506070809A0B0C0D0E0FFEDCBA98'H },
    {
      extnId { 1 3 6 1 5 5 7 1 3 }, -- qcStatements
      extnValue '302B302906082B06010505070B01301D301B81196D756 ...'H } }
}
```

Monday September 26, 2011 1/1



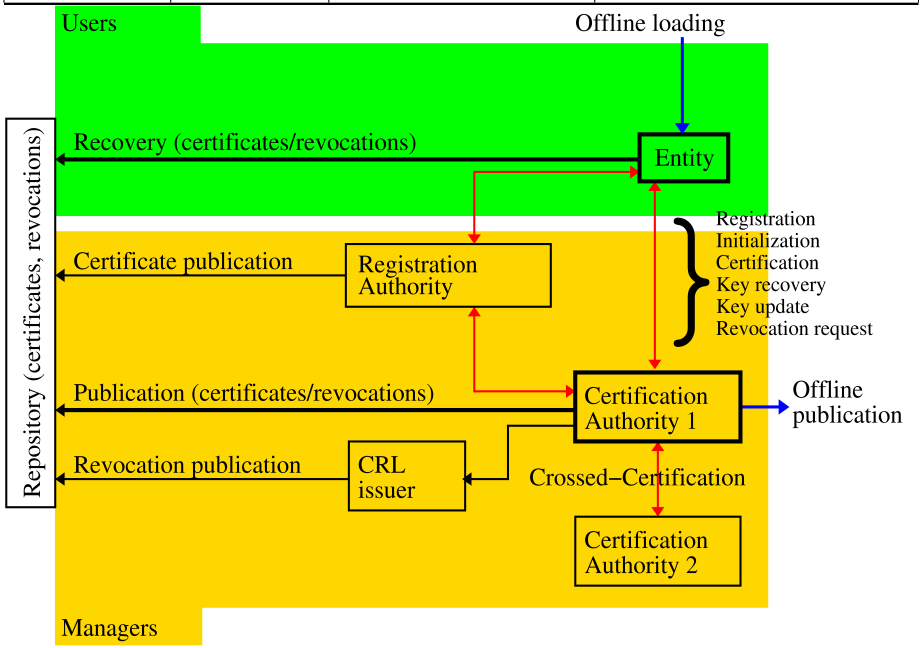



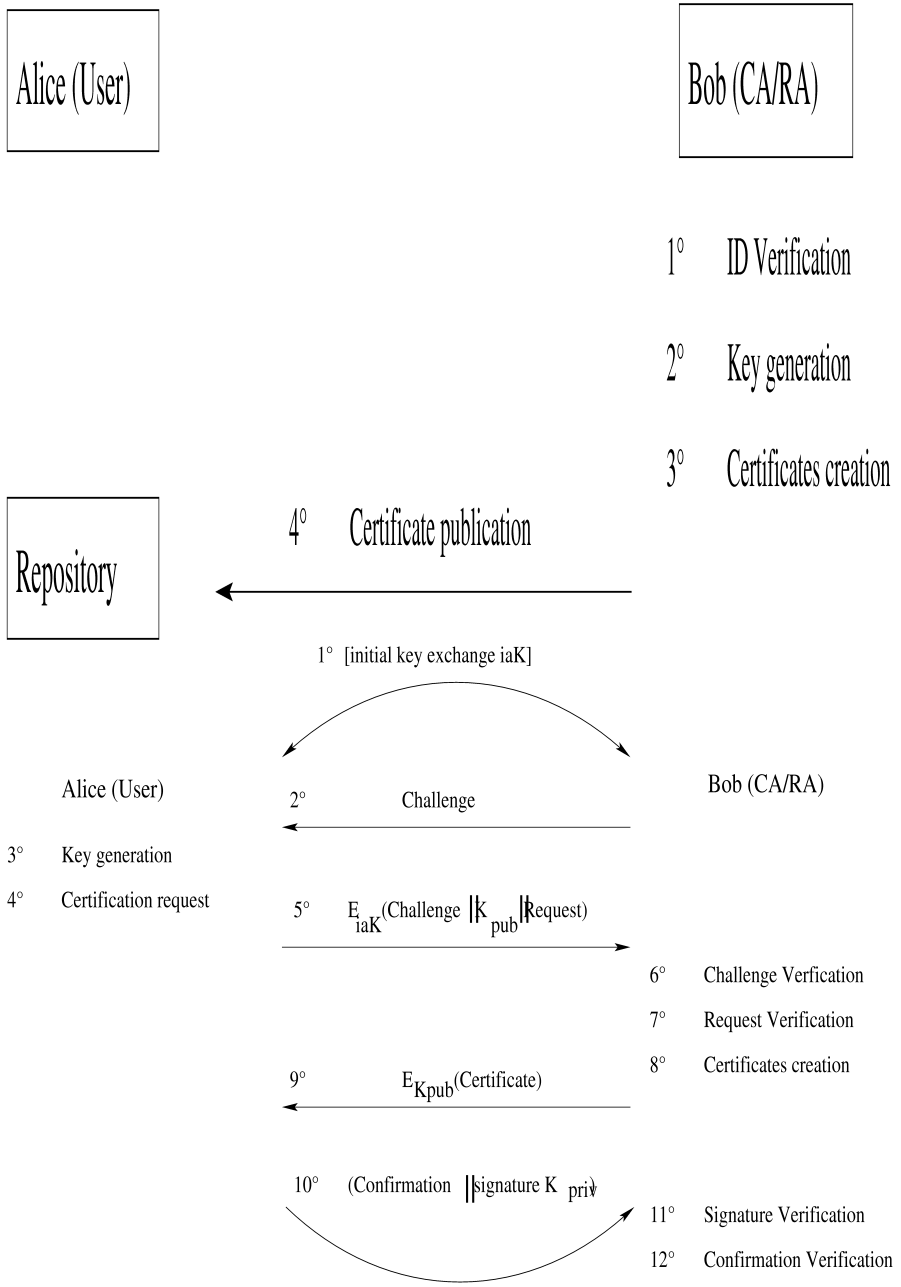

# Reasons to revoke a certificate [RFC 5280]

- unspecified (0)
- keyCompromise (1)
- CACompromise (2)
- affiliationChanged (3)
- superseded (4)
- cessationOfOperation (5)
- certificateHold (6)
- removeFromCRL (8)
- privilegeWithdrawn (9)
- AACompromise (10)

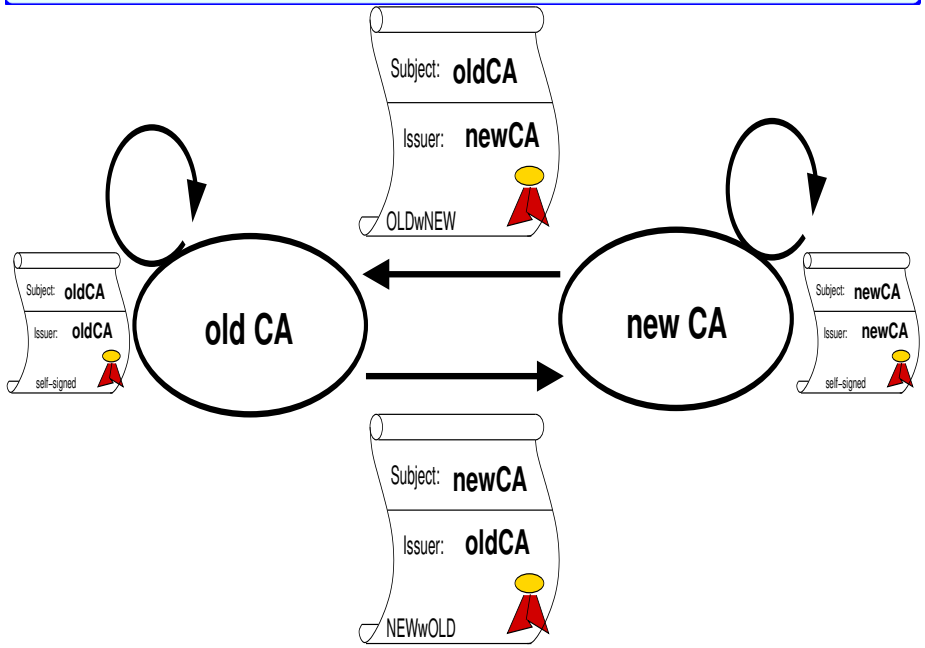
Obs.: Value 7 is not used.

| Alg. | Hash. | OID | Identificator |
|---|---|---|---|
| 3DES-CBC | | 1.2.840.113549.3.7 | DES-EDE3-CBC |
| RSA | | 1.2.840.113549.1.1.1 | RSAEncryption |
| RSA | MD5 | 1.2.840.113549.1.1.4 | md5withRSAEncryption |
| RSA | SHA-1 | 1.2.840.113549.1.1.5 | sha1withRSAEncryption |
| DSA | | 1.2.840.10040.4.1 | id-dsa |
| DSA | SHA-1 | 1.2.840.10040.4.3 | id-dsawithSha1 |
| DSA | SHA-1.320 | 1.3.14.3.2 | id-dsawithSha1.320 |
| ECDSA | SHA-1 | 1.2.840.10045.1 | ecdsawithSha1 |

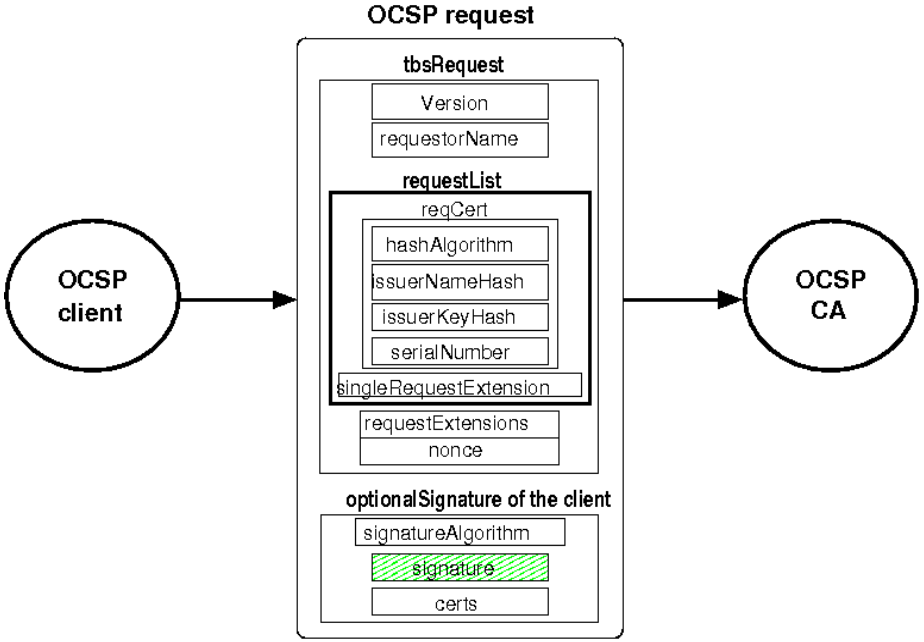


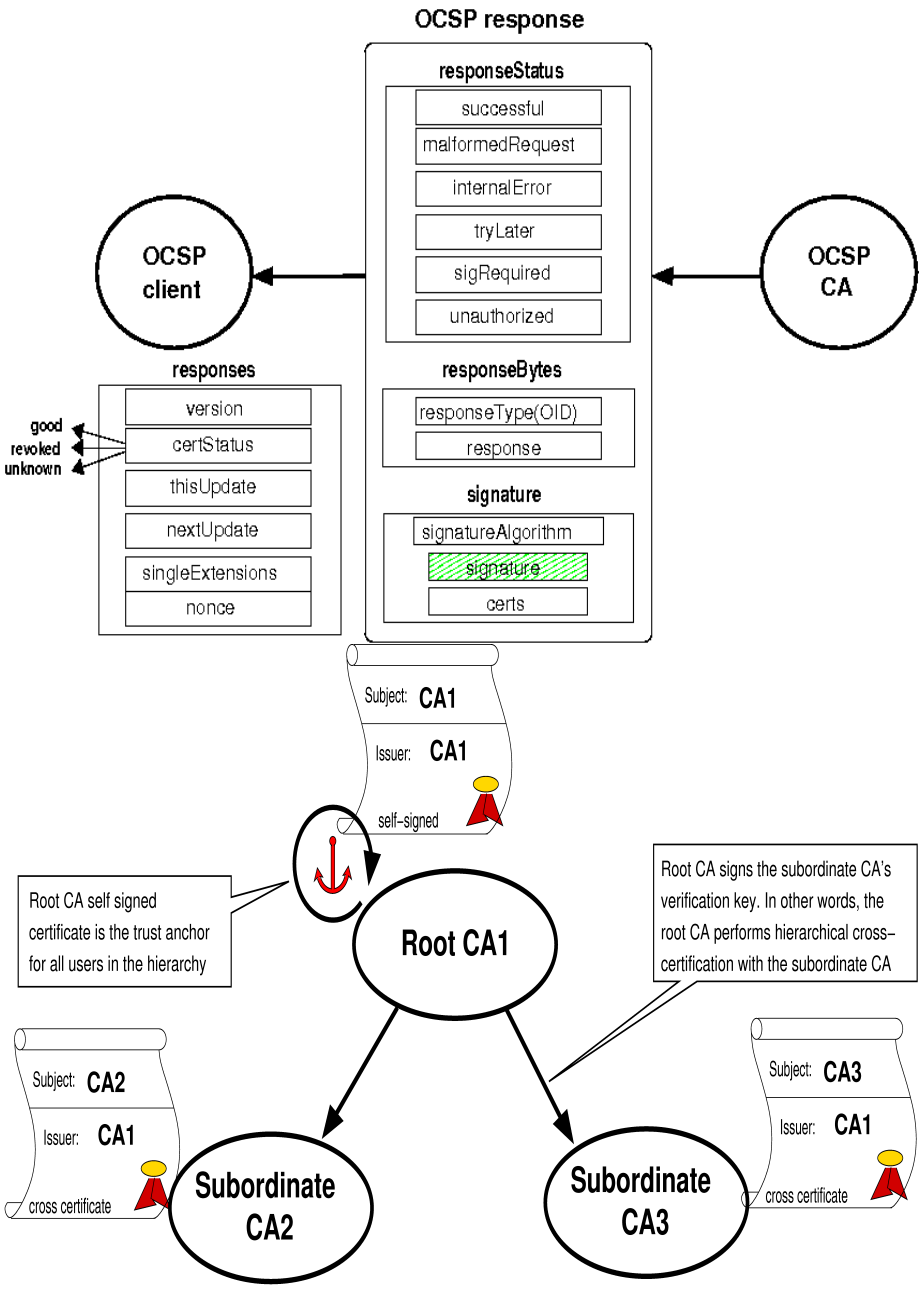



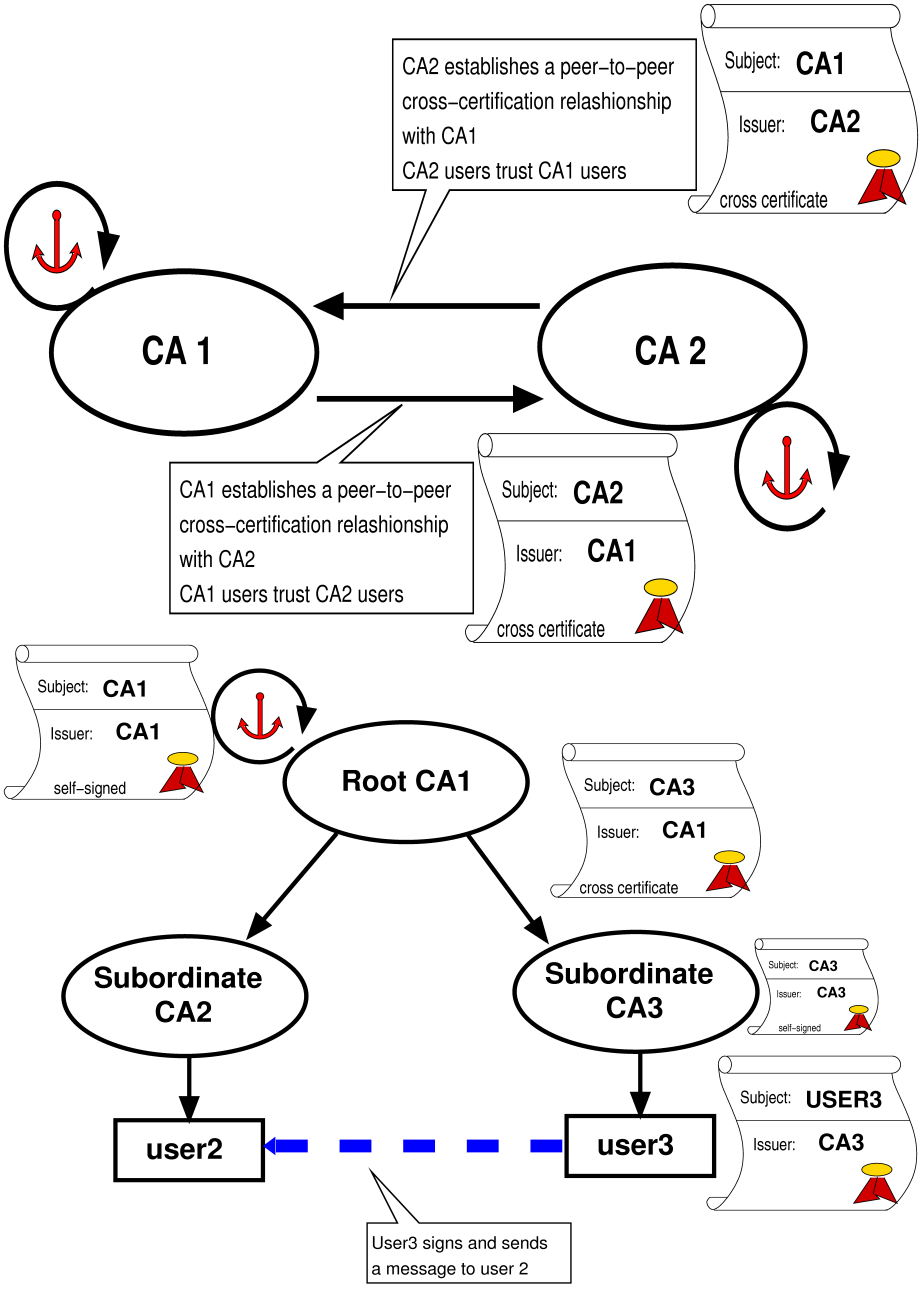
```
CertReqMsg ::= SEQUENCE {
  certReq   CertRequest,
  signature SIGNED{ certReq }  OPTIONAL,
}

CertRequest ::= SEQUENCE {
  certReqId    INTEGER,      -- ID for matching request and reply
  certTemplate CertTemplate, -- Selected fields of cert to be issued
}

CertResponse ::= SEQUENCE {
  certReqId           INTEGER,
  status              PKIStatusInfo,
  certificate         Certificate    OPTIONAL,
  rspInfo             OCTET STRING   OPTIONAL
}
```

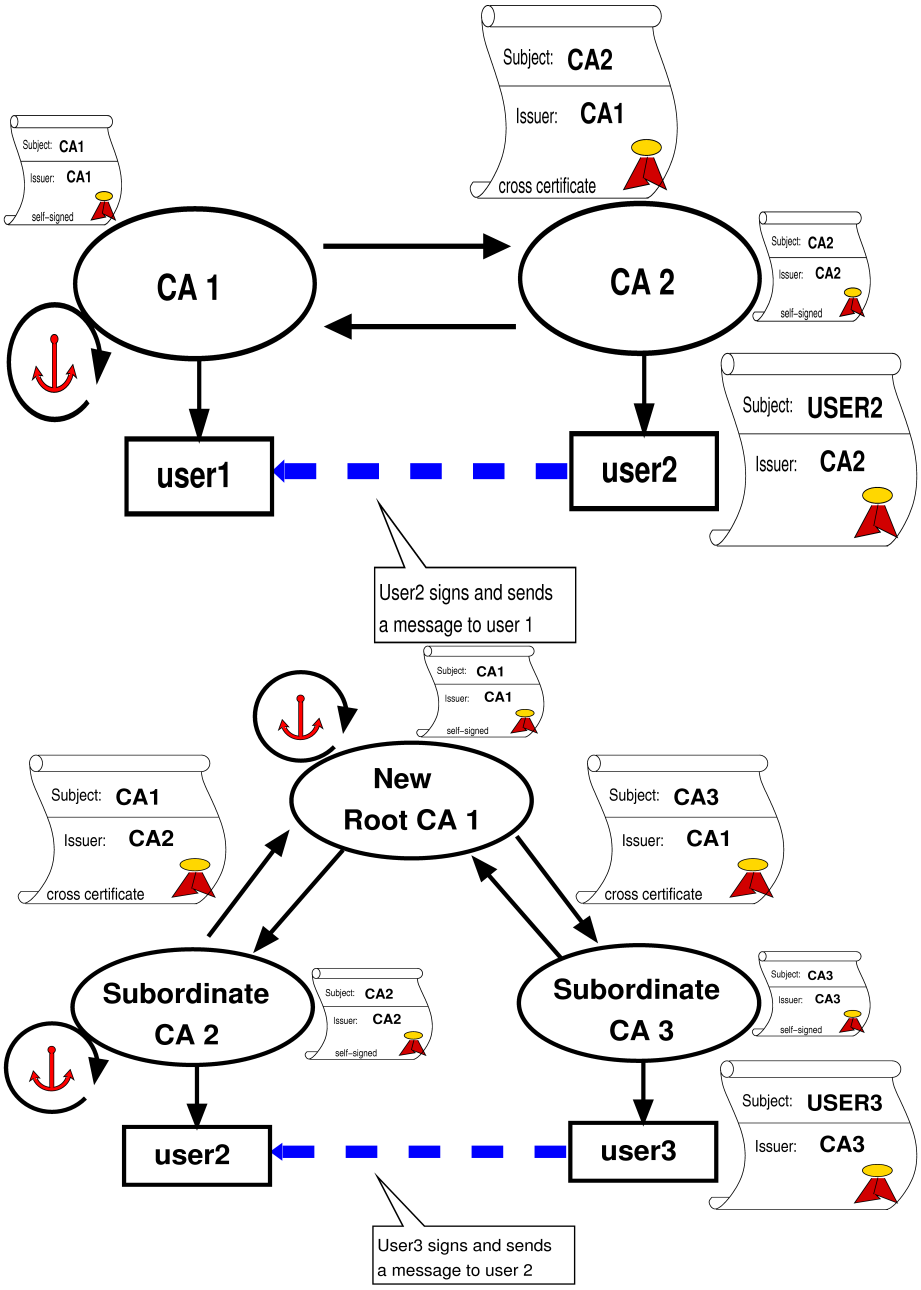



| | Certificate signed with $NEW_{priv}$ | Certificate signed with $OLD_{priv}$ |
|---|---|---|
| Repository contains OLD and NEW | | |
| $NEW_{pub}$ within PSE | DIRECT verification | Recover OLDwNEW<br>Verify OLDwNEW with $NEW_{pub}$<br>Verify certificate with $OLD_{pub}$ |
| $OLD_{pub}$ within PSE | Recover NEWwOLD<br>Verify NEWwOLD with $OLD_{pub}$<br>Verify certificate with $NEW_{pub}$ | DIRECT verification |
| Repository contains only OLD | | |
| $NEW_{pub}$ within PSE | DIRECT verification (without the Repository) | FAILURE (Repository's fault) |
| $OLD_{pub}$ within PSE | FAILURE (Repository's fault) | DIRECT verification |




OCSP response
responseStatus
successful
malformedRequest
internalError
tryLater
sigRequired
unauthorized
OCSP client
OCSP CA
responses
version
certStatus
thisUpdate
nextUpdate
singleExtensions
nonce
good
revoked
unknown
responseBytes
responseType(OID)
response
signature
signatureAlgorithm
signature
certs
Subject: CA1
Issuer: CA1
self-signed
Root CA self signed certificate is the trust anchor for all users in the hierarchy
Root CA1
Root CA signs the subordinate CA's verification key. In other words, the root CA performs hierarchical cross-certification with the subordinate CA
Subject: CA2
Issuer: CA1
cross certificate
Subordinate CA2
Subordinate CA3
Subject: CA3
Issuer: CA1
cross certificate

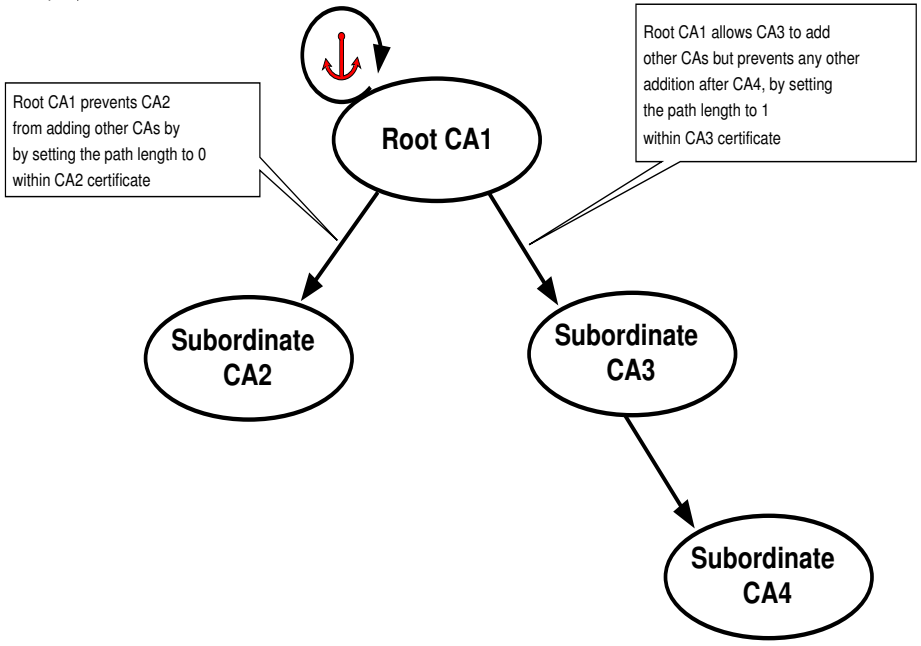

CA2 establishes a peer-to-peer cross-certification relashionship with CA1
CA2 users trust CA1 users
Subject: CA1
Issuer: CA2
cross certificate
CA 1
CA 2
CA1 establishes a peer-to-peer cross-certification relashionship with CA2
CA1 users trust CA2 users
Subject: CA2
Issuer: CA1
cross certificate
Subject: CA1
Issuer: CA1
self-signed
Root CA1
Subject: CA3
Issuer: CA1
cross certificate
Subordinate CA2
Subordinate CA3
Subject: CA3
Issuer: CA3
self-signed
Subject: USER3
Issuer: CA3
user2
user3
User3 signs and sends a message to user 2

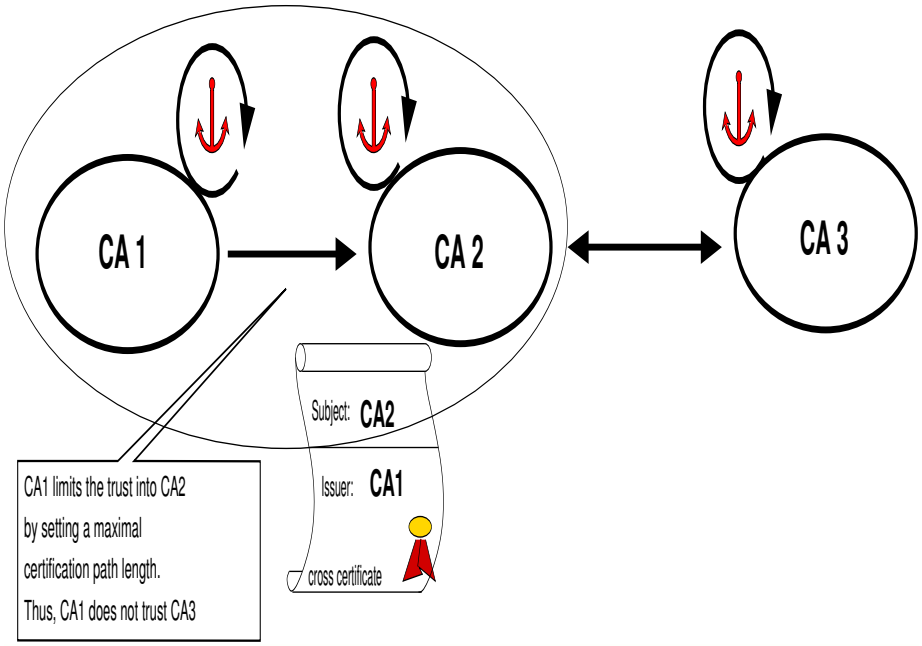


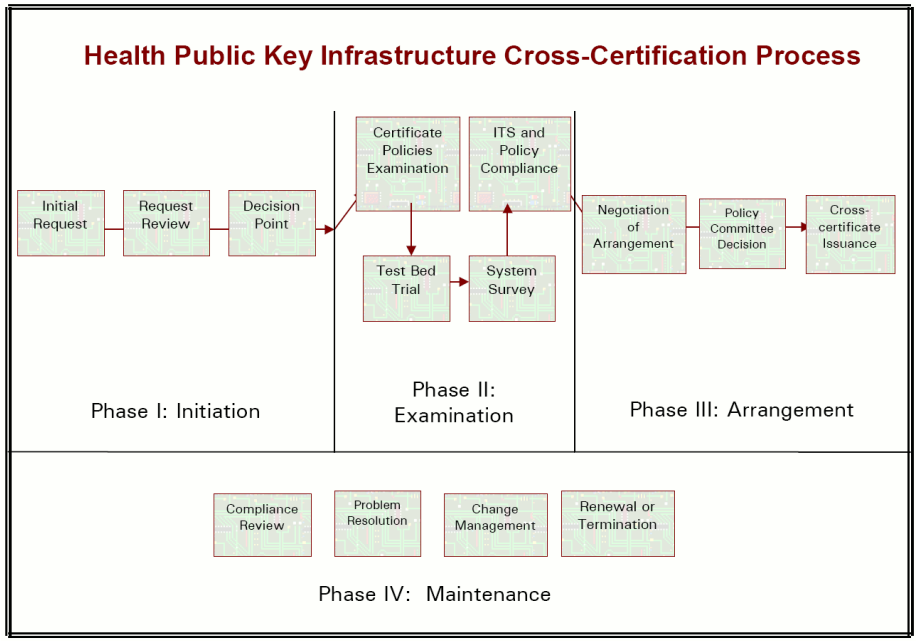


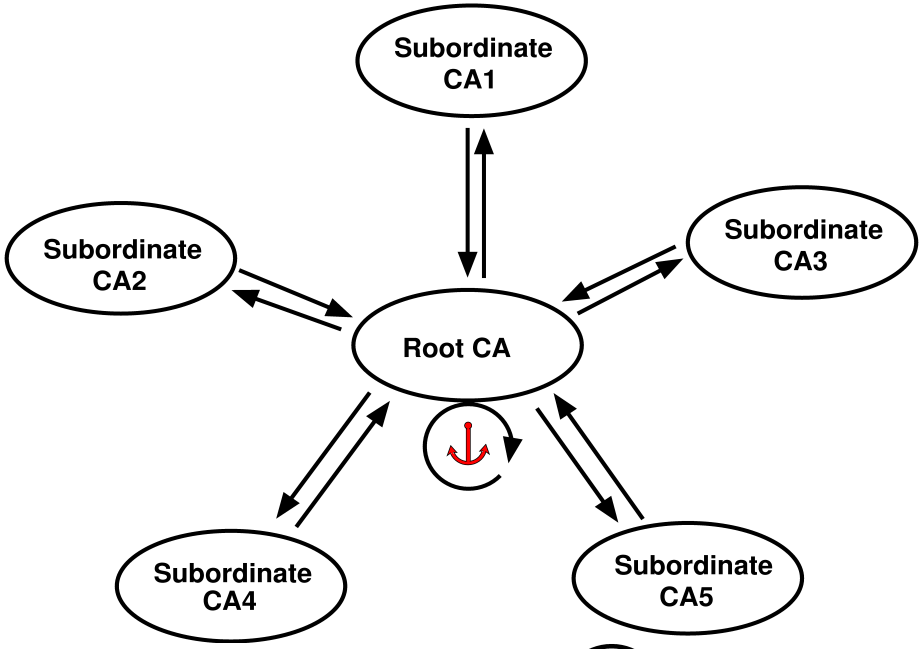

```
pkiCA   OBJECT-CLASS   ::= {
   SUBCLASS OF    { top}
   KIND           auxiliary
   MAY CONTAIN    {cACertificate |
                   certificateRevocationList |
                   authorityRevocationList |
                   crossCertificatePair }
   ID     joint-iso-ccitt(2) ds(5) objectClass(6) pkiCA(22)}

cACertificate     ATTRIBUTE  ::=  {
     WITH SYNTAX   Certificate
     EQUALITY MATCHING RULE   certificateExactMatch
     ID  joint-iso-ccitt(2) ds(5) attributeType(4) cACertificate(37) }

crossCertificatePairATTRIBUTE::={
   WITH SYNTAX   CertificatePair
   EQUALITY MATCHING RULE certificatePairExactMatch
 ID joint-iso-ccitt(2) ds(5) attributeType(4) crossCertificatePair(40)}

-----------------------------------------------------
CertificatePair ::= SEQUENCE {
  forward     [0]  Certificate OPTIONAL,
  reverse     [1]  Certificate OPTIONAL,
  -- at least one of the pair shall be present -- }
```

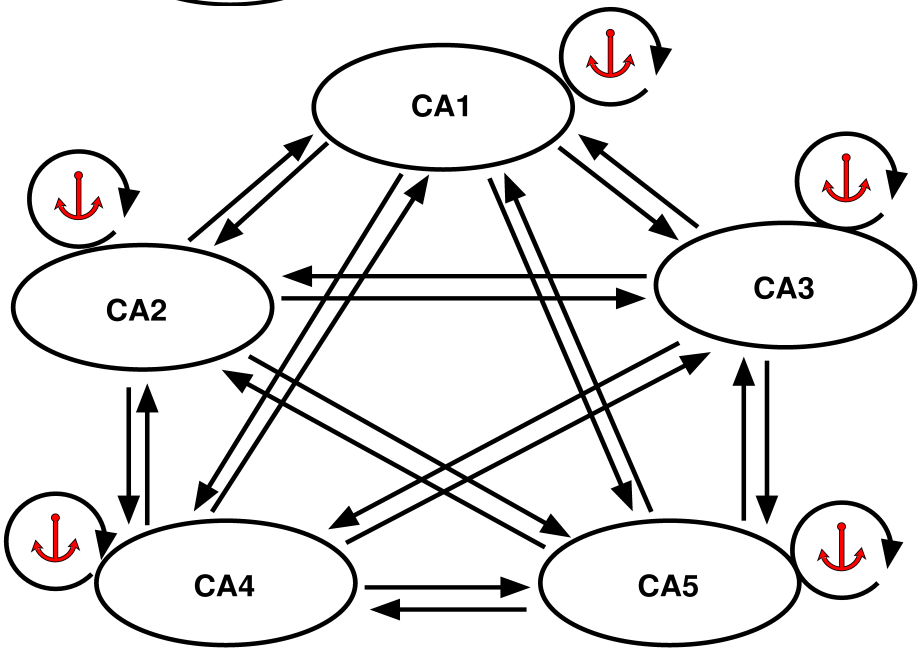


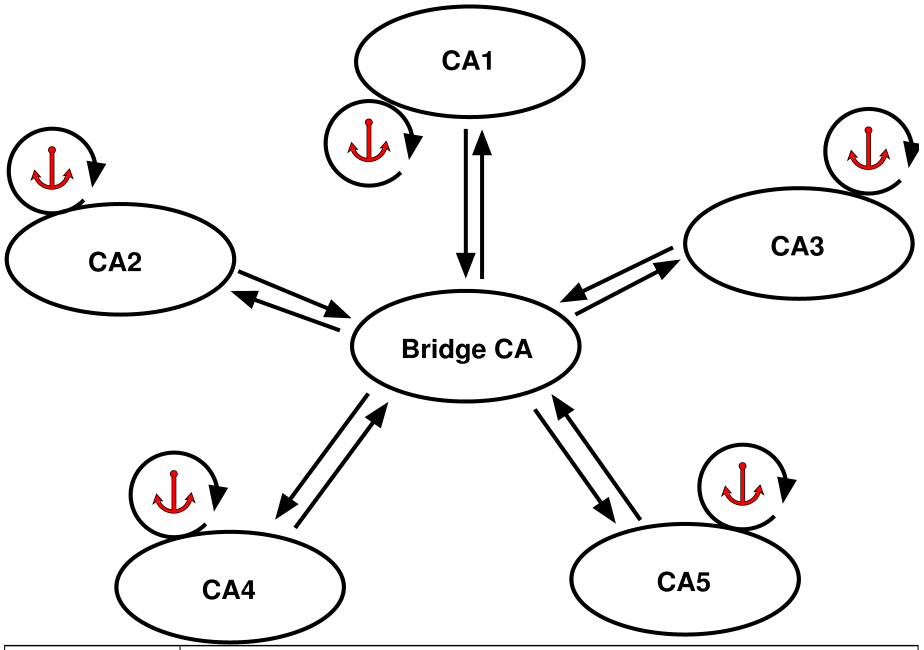



CA 1
CA 2
CA 3
Subject: CA2
Issuer: CA1
cross certificate
CA1 limits the trust into CA2
by setting a maximal
certification path length.
Thus, CA1 does not trust CA3

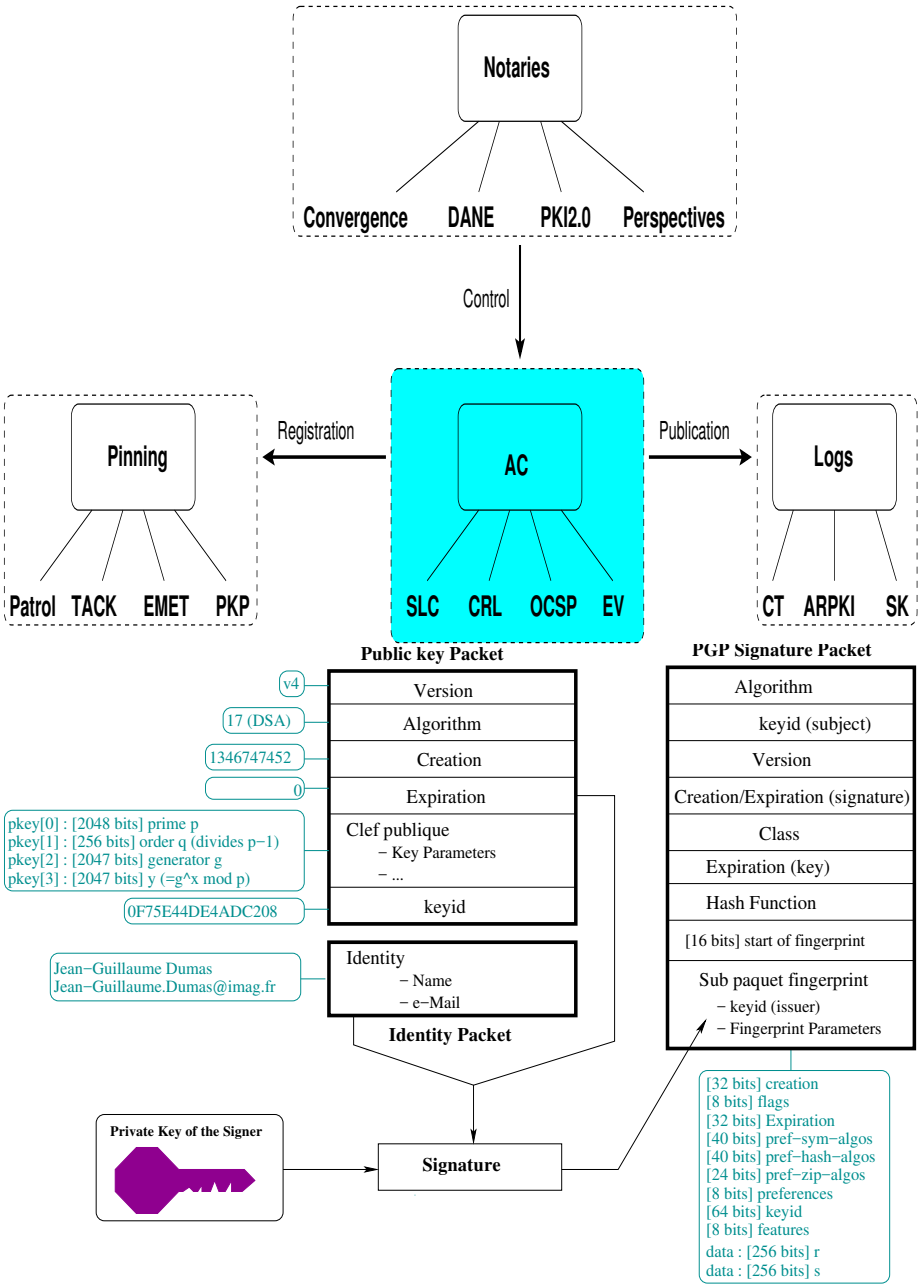

Health Public Key Infrastructure Cross-Certification Process
Initial Request
Request Review
Decision Point
Certificate Policies Examination
ITS and Policy Compliance
Test Bed Trial
System Survey
Negotiation of Arrangement
Policy Committee Decision
Cross-certificate Issuance
Phase I: Initiation
Phase II: Examination
Phase III: Arrangement
Compliance Review
Problem Resolution
Change Management
Renewal or Termination
Phase IV: Maintenance


Subordinate CA1
Subordinate CA2
Subordinate CA3
Root CA
Subordinate CA4
Subordinate CA5


CA1
CA2
CA3
CA4
CA5



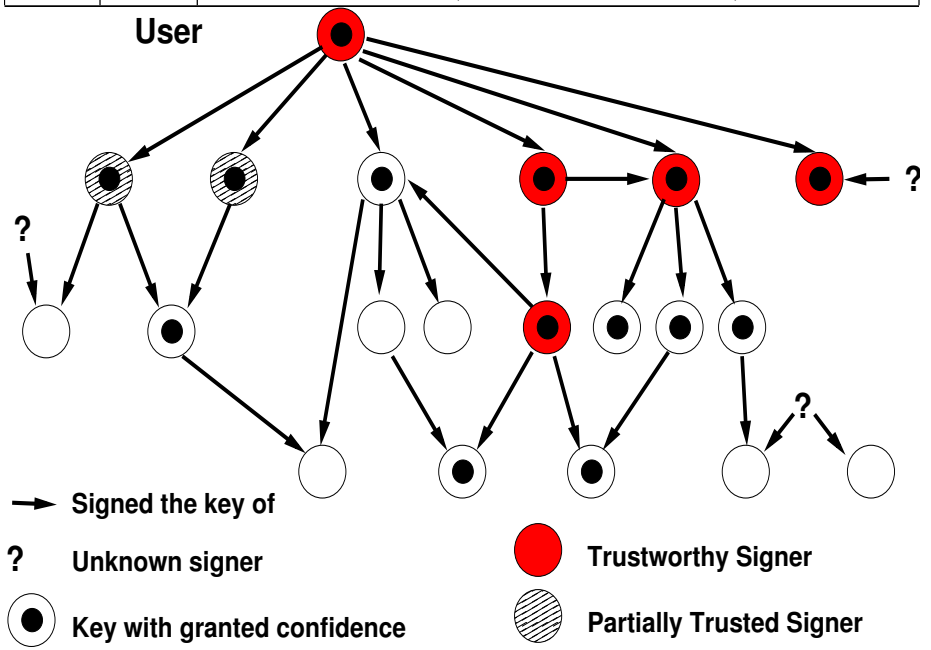

| System | Trust Anchor Stores |
|---|---|
| Debian 7 | `/etc/ssl/certs/ca-certificates.crt` |
| Windows 10 | `C:\Windows\System32\certmgr.msc` |
| Mac OS X | `Applications>Utilities>Keychain Access` |
| Android 4 | `Settings>Security>Credential Storage>Trusted Credentials>Display Trusted CA certificates` |
| Iceweasel 31 | `/usr/share/ca-certificates/mozilla/*.crt` |
| Firefox 36 | `Preferences>Privacy&Security>Certificates>Manage Certificates` |
| Chrome 40 | `Settings>Advanced settings>Manage certificates` (using `certmgr.msc` or `Keychain`) |
| Safari 5 | using `Keychain` |
| IE 11 | `Internet Options>Content>Certificates` (using `certmgr.msc`) |



| | |
|---|---|
| 0 | – Reserved - a packet tag MUST NOT have this value |
| 1 | – Public-Key Encrypted Session Key Packet |
| 2 | – Signature Packet |
| 3 | – Symmetric-Key Encrypted Session Key Packet |
| 4 | – One-Pass Signature Packet |
| 5 | – Secret-Key Packet |
| 6 | – Public-Key Packet |
| 7 | – Secret-Subkey Packet |
| 8 | – Compressed Data Packet |
| 9 | – Symmetrically Encrypted Data Packet |
| 10 | – Marker Packet |
| 11 | – Literal Data Packet |
| 12 | – Trust Packet |
| 13 | – User ID Packet |
| 14 | – Public-Subkey Packet |
| 17 | – User Attribute Packet |
| 18 | – Sym. Encrypted and Integrity Protected Data Packet |
| 19 | – Modification Detection Code Packet |
| 60 to 63 | – Private or Experimental Values |

| | |
|---|---|
| 0x00 | Signature of a binary document |
| 0x01 | Signature of a canonical text document |
| 0x02 | Standalone signature |
| 0x10 | Generic certification of a User ID and Public-Key packet |
| 0x11 | Persona certification of a User ID and Public-Key packet |
| 0x12 | Casual certification of a User ID and Public-Key packet |
| 0x13 | Positive certification of a User ID and Public-Key packet |
| 0x18 | Subkey Binding Signature |
| 0x19 | Primary Key Binding Signature |
| 0x1F | Signature directly on a key |
| 0x20 | Key revocation signature |
| 0x28 | Subkey revocation signature |
| 0x30 | Certification revocation signature |
| 0x40 | Timestamp signature |
| 0x50 | Third-Party Confirmation signature |

| field | byte | content |
|---|---|---|
| a.1 | 1 | 0x99 |
| a.2 | 1 | most-significant bits of the size of fields (b) to (e) |
| a.3 | 1 | least-significant bits of the size of fields (b) to (e) |
| b | 1 | version number = 4 |
| c | 4 | key creation date |
| d | 1 | signature algorithm, (e.g., 17 for DSA) |
| e | specific to the algorithm, e.g., for DSA : | |
| e.1 | 256 | prime number (e.g., $p$ on 2048 bits) |
| e.2 | 32 | group order (e.g., $q$ on 256 bits) |
| e.3 | 256 | group generator (e.g., $g$ on 2048 bits) |
| e.4 | 256 | public key value (e.g., $y$ on 2048 bits) |



| keyid | public key | *ciphered* private key | timestamp | user |
|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $0x70096AD1$ | $KU_i$ | $E_{H(P_i)}(KR_i)$ | $Time_i$ | $User_i$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

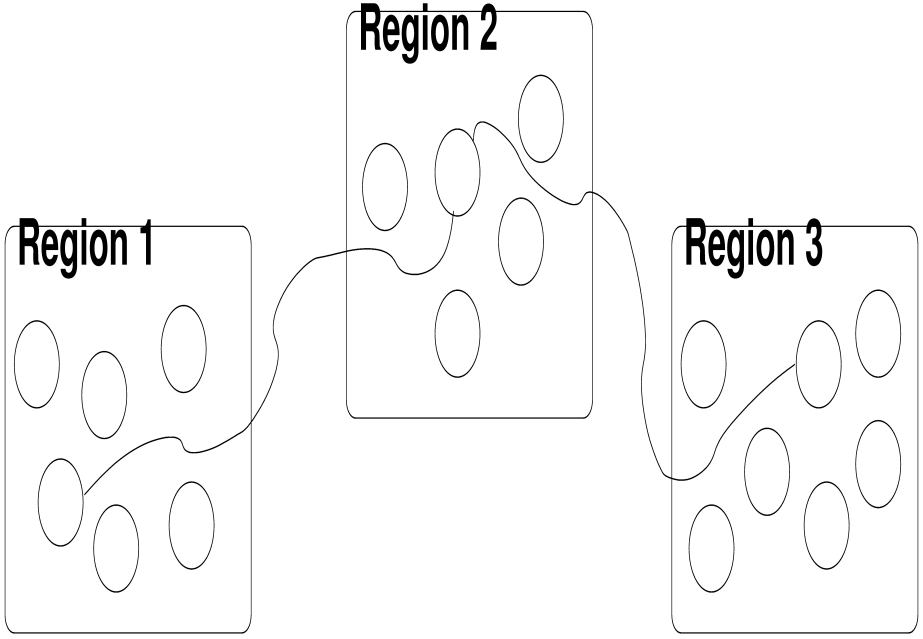
| keyid | public key | *ciphered* private key | timestamp | user | trust | signature |
|---|---|---|---|---|---|---|
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| $0x70096AD1$ | $KU_i$ | $E_{H(P_i)}(KR_i)$ | $Time_i$ | $User_i$ | $trust_i$ | $sign_i$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |

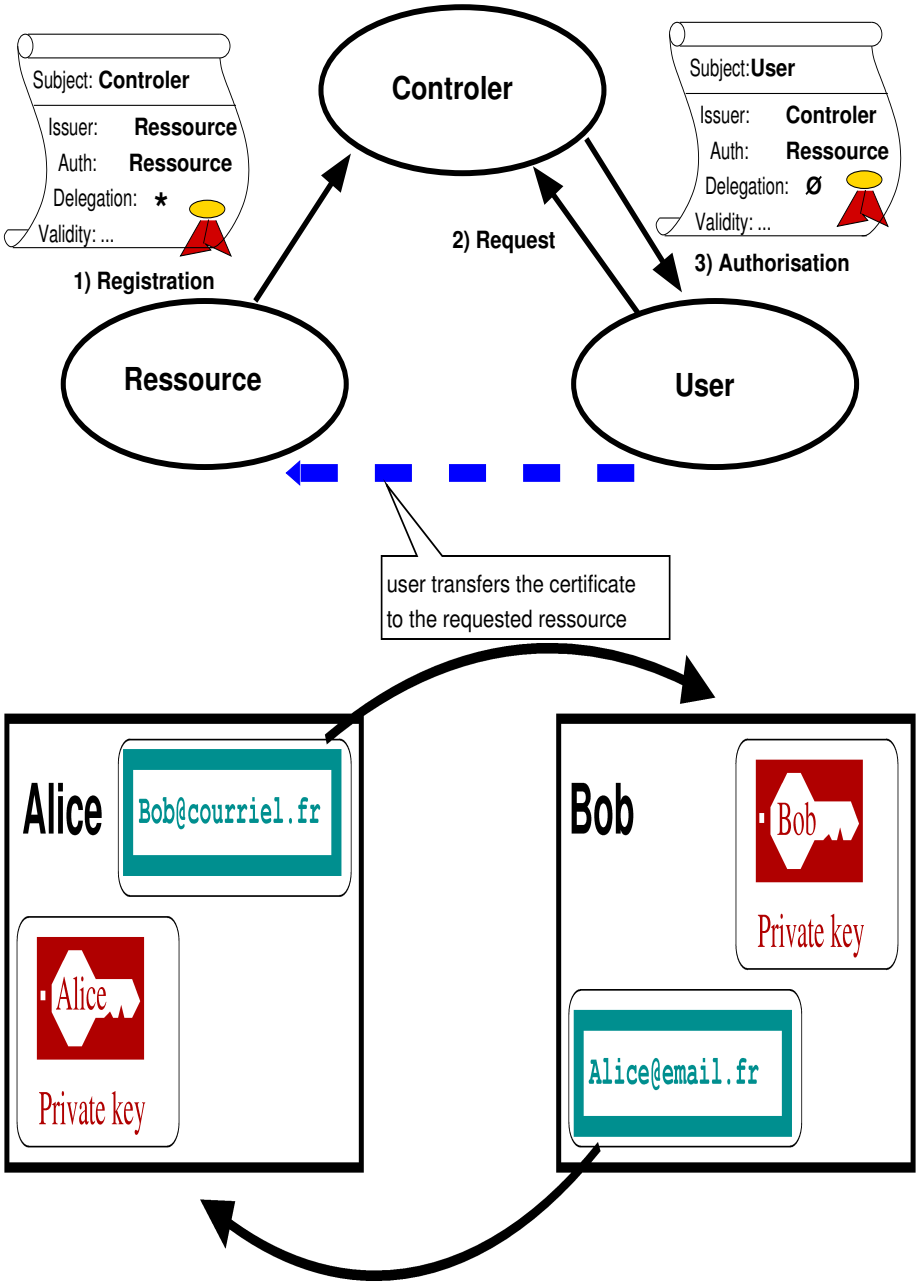
# Search results for '0x0f75e44de4adc208'

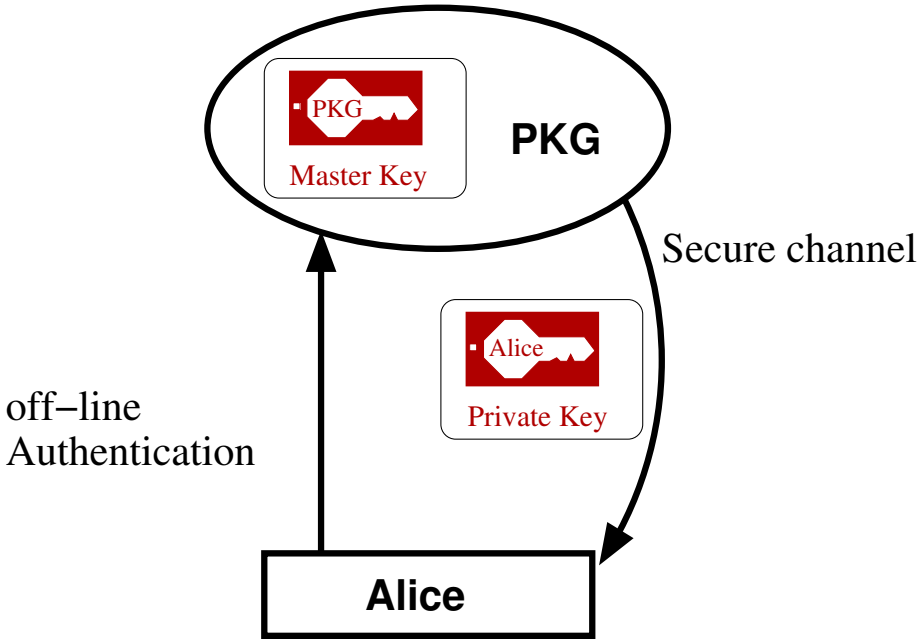
```
Type bits/keyID    cr. time   exp time   key expir
```

---

```
pub  2048D/E4ADC208 2012-09-04

uid Jean-Guillaume Dumas <jgdumas@imag.fr>
sig  sig3  E4ADC208 2012-09-04 _______ 2016-09-03 [selfsig]
sig  sig   1F829E58 2012-12-07 _______ _______ []
sig  sig   7A6574F0 2012-12-07 _______ _______ Dominique Duval (pour safe pki) <Dominique.Duval@imag.fr>

sub  2048g/322DC83F 2012-09-04
sig sbind  E4ADC208 2012-09-04 _______ 2016-09-03 []
```






Alice

verifier

2°) d

Bob

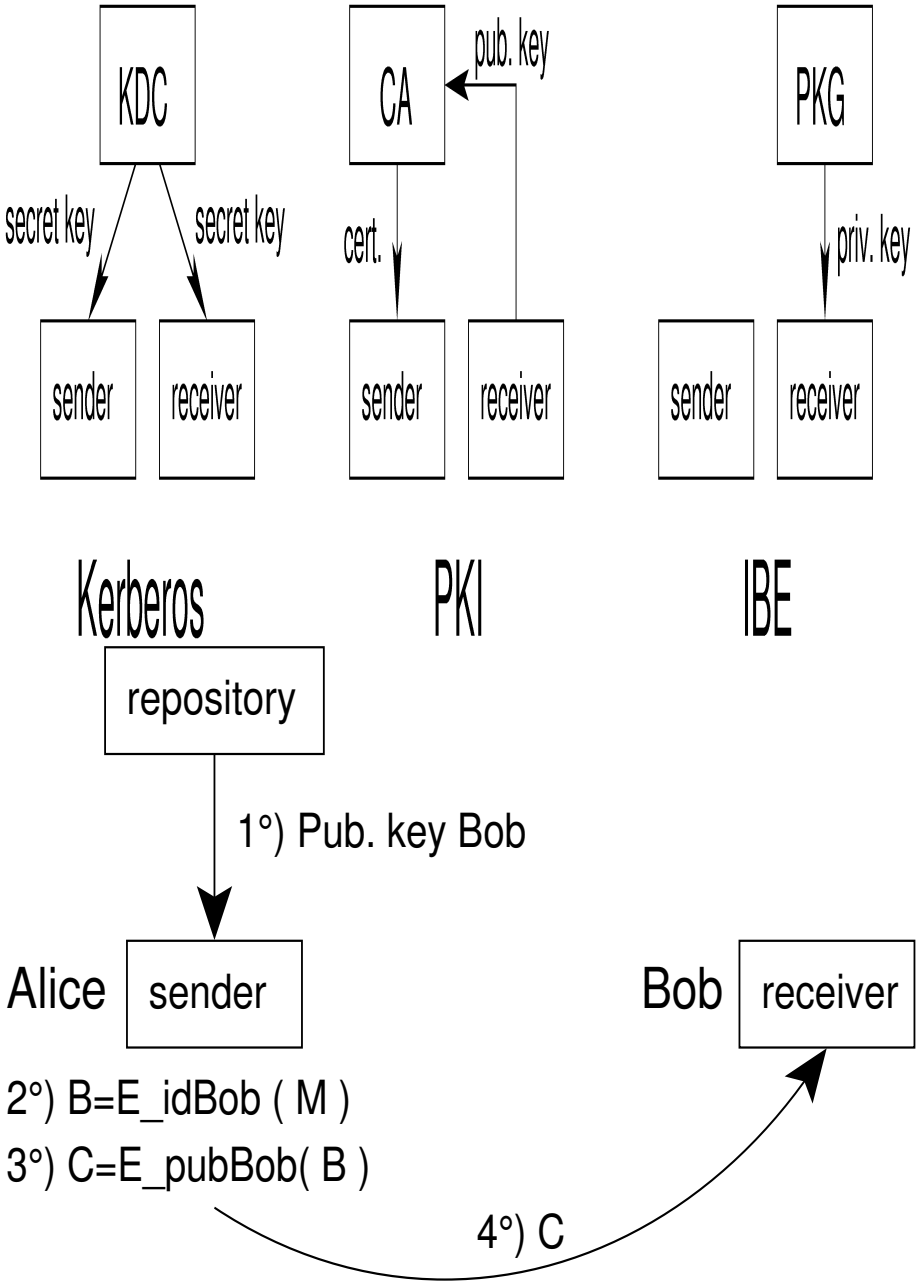
signer

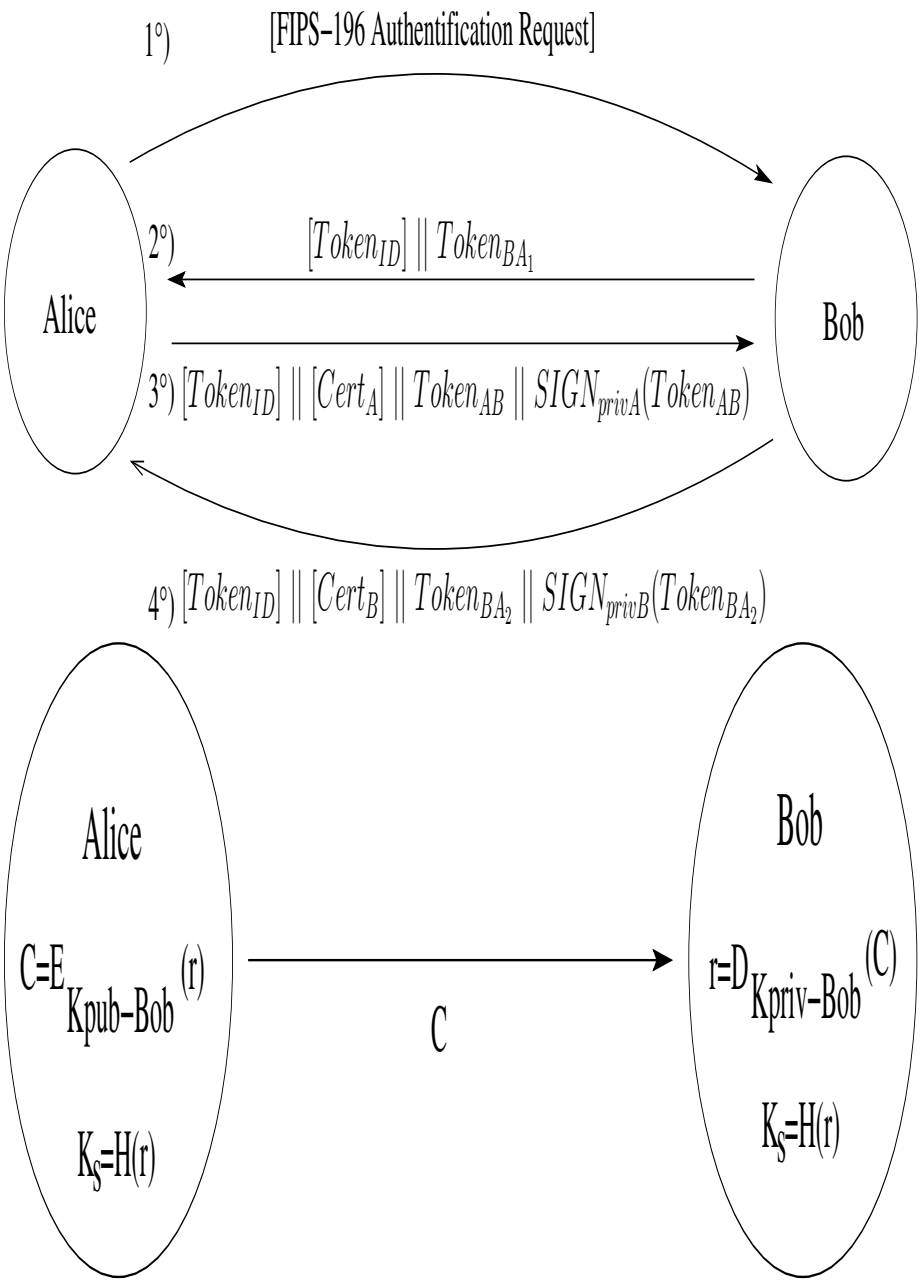
3°) Random R

4°) C=ID_M( R )

5°) D_d( C ) =?= R

1°) Generate d
associated to ID_M


KDC
CA
pub. key
PKG
secret key
secret key
cert.
priv. key
sender
receiver
sender
receiver
sender
receiver
Kerberos
PKI
IBE
repository
1°) Pub. key Bob
Alice
sender
Bob
receiver
2°) B=E_idBob ( M )
3°) C=E_pubBob( B )
4°) C

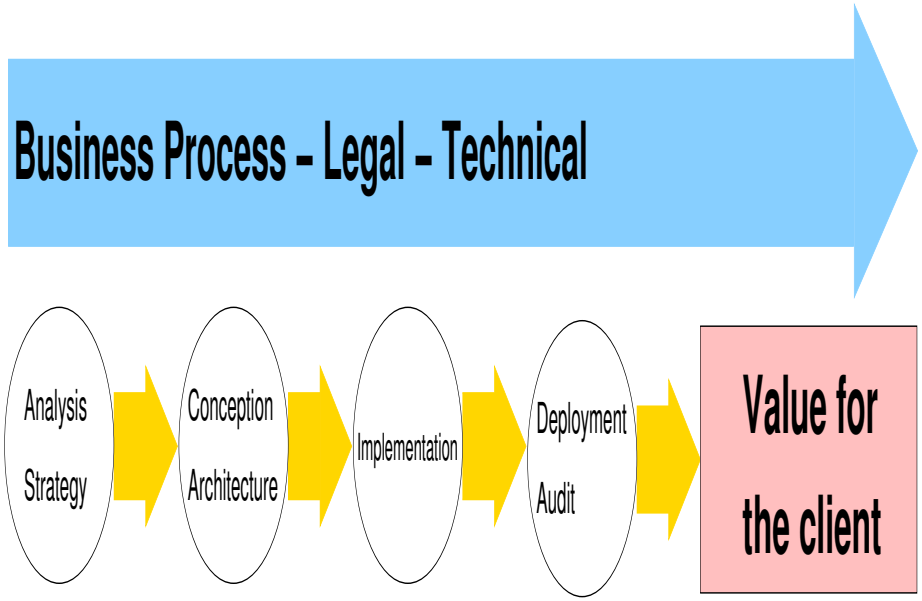

1°)
[FIPS-196 Authentification Request]
2°)
$[Token_{ID}] \,||\, Token_{BA_1}$
Alice
Bob
3°) $[Token_{ID}] \,||\, [Cert_A] \,||\, Token_{AB} \,||\, SIGN_{privA}(Token_{AB})$
4°) $[Token_{ID}] \,||\, [Cert_B] \,||\, Token_{BA_2} \,||\, SIGN_{privB}(Token_{BA_2})$
Alice
$C=E_{Kpub-Bob}(r)$
$K_S=H(r)$
C
Bob
$r=D_{Kpriv-Bob}(C)$
$K_S=H(r)$

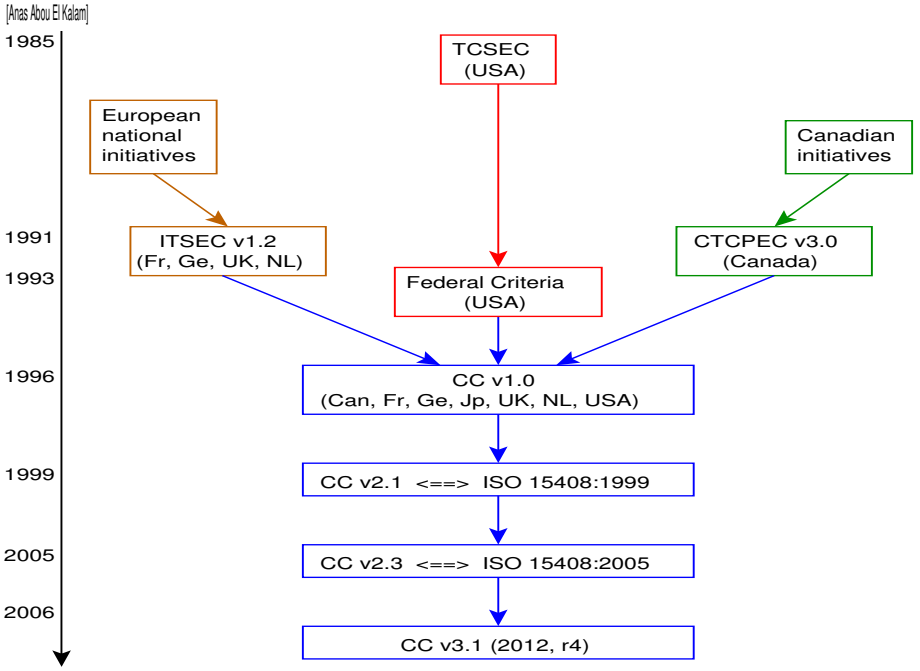
Co-signature

Countersignature

| | PDF Signature | Binary Signature | XML Signature |
|---|---|---|---|
| « simple » Signature | PDF | PKCS#7 CMS | XML/DSig |
| « advanced » Signature | PAdES | CAdES | XAdES |

| Certificate policy | Certification practice statement | Terms and conditions |
|---|---|---|
| Certificates will be validated at each use to ensure that the certificate remains current. Validations returns will be made in nn minutes...<br><br>The registration of new users will be conducted by xyz registration authority and will require the presentation of two sets of IDs...<br><br>Certificates will be valid for nn years | Certificates will be validated using an OCSP-based validation service held in a secure location and managed...<br><br>The RA will communicate with the CA via a secure link utilizing abc encryption techniques and each request will be signed...<br><br>The CA will revoke and reissue certificates every nn years using the existing keys to roll over the certificate and automated notification to users of their new certificate... | Each time you use your certificate we will check to make sure it is valid...<br><br>On requesting a certificate you will be required to provide two items of proof of your name and address; this can be a utility bill, bank statement, driving licence...<br><br>We will issue you with an updated certificate every nn years... |


Business Process - Legal - Technical
Analysis
Strategy
Conception
Architecture
Implementation
Deployment
Audit
Value for
the client


[Anas Abou El Kalam]
1985
1991
1993
1996
1999
2005
2006
TCSEC
(USA)
European
national
initiatives
Canadian
initiatives
ITSEC v1.2
(Fr, Ge, UK, NL)
CTCPEC v3.0
(Canada)
Federal Criteria
(USA)
CC v1.0
(Can, Fr, Ge, Jp, UK, NL, USA)
CC v2.1 <==> ISO 15408:1999
CC v2.3 <==> ISO 15408:2005
CC v3.1 (2012, r4)