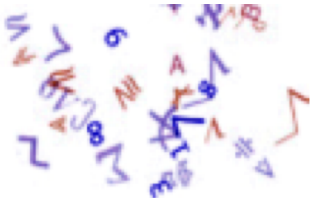
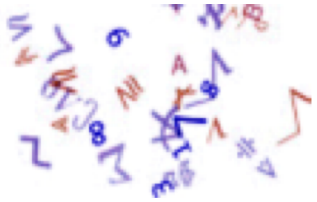


PKI

Public Key Infrastructure

Ph. Jaillon



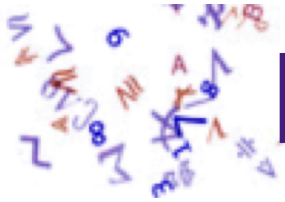
What is the problem

- We *need* to offer to a *large population* the opportunity to *securely* exchange data or access services.
- A solution is to provide at least these features with cryptographic technique:

authentication, confidentiality, integrity, signature

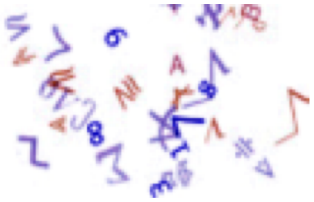
but also

legitimacy, *validity, usage*…



Problem: how to manage keys?

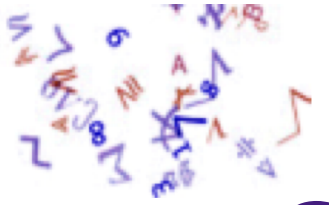
- Using secret keys (DES, AES…):
  - about n^2 keys to manage
- Using public/private keys (RSA):
  - n private keys
  - n public keys to share
- Confidence in public keys
  - Is this key really the one of the other peer of the transaction?
  - Could the owner of this key be trusted for the planned action?
  - Is this key still valid?



A solution : using certificate

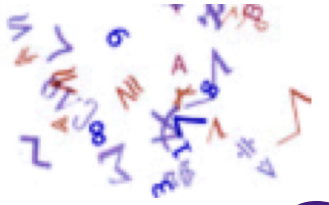
- Certificate establishes relations between
  - a public key,
  - the certificate owner (person, application, site),
  - Its defined usage.

- X509
  - ITU-T X.509 international standard V3 - 1996,
  - RFC 2459 (1999) obsoleted by RFC 3280 (2002).



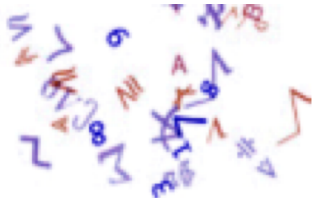
Certificate

- For people
  - Proof of identity, like an identity card and for the purpose fixed by the certification authority which signed it.
- For an application
  - It's really the application we want to use (the code wasn't modified, integrity check);
- For a site or a computer
  - It's really the site/host we want to access/connect to.



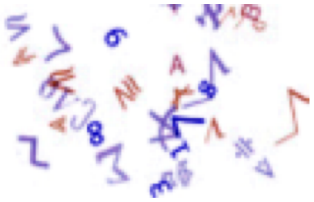
Certificate usage example

- User authentication
  - In France, each person must have a certificate to access its “online fiscal folder”. These certificates are provided by the “Ministère des Finances”.
- Downloading update or software
  - Microsoft, Apple, SUN and Linux updates are now signed by issuers (using their private key). Before installing, you can verify packages and issuer *legitimacy*;
- Electronic commerce
  - You can verify that you are connected to the *right website*: not a copy looking like the original, it can help to protect from phishing.



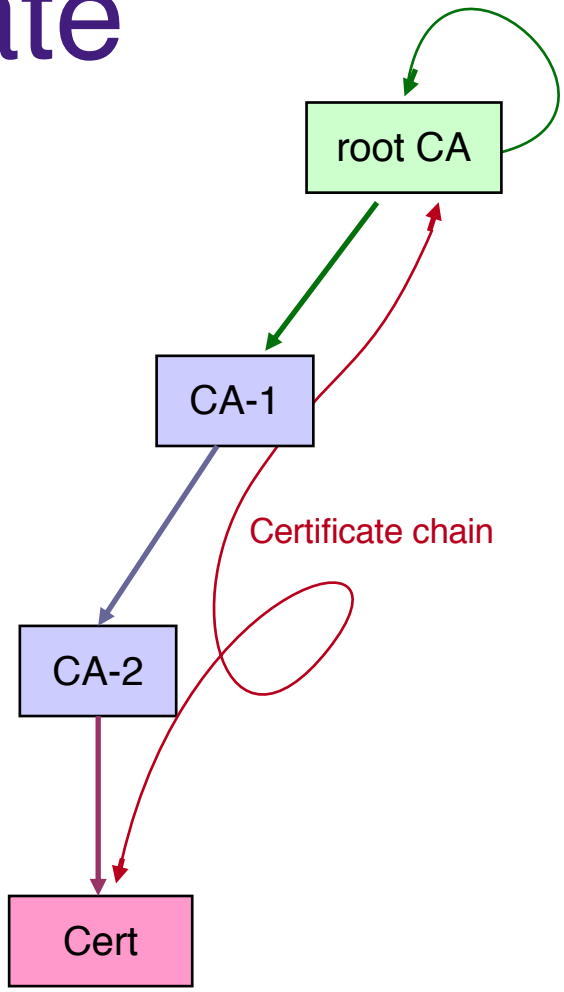
How it works

- Information stored in the certificate (identity, validity, usage and the public key) are signed by a third party: a certification authority (CA).
- This certification authority has its own certificate. This certificate can be signed by a higher level certification authority or by itself (in this case, it's a *root* certificate).



Verifying a certificate

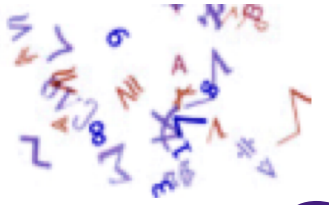
- You need to recursively verify that the certificate was signed by an authority you trust.
- To validate the certificate, you must trust the root certificate of the build chain.



Certificate content

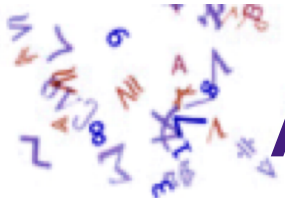
- The goal of the certificate is to link a public key to information about the related entity and to define its usage purposes.

Data	Version
	Serial Number
	Signature Algorithm
	Isuer
	Validity
	Subject
	Subject Public Key Info
	X509v3 Extensions Extension1:[critical] Extension1 value Extension2:[critical] Extension2 value … Extensionn:[critical] Extensionn value
Signature	Signature Algorithm
	Signature



Certificate representation

- X.509 is part of the Open Systems Interconnection model (OSI).
- Historically, OSI uses ASN.1 for data representation.
- ASN.1: Abstract Syntax Notation One
  - INTERNATIONAL STANDARD ISO/IEC 8824-1
  - ITU-T RECOMMENDATION X.680



ASN.1
Abstract Syntax Notation One

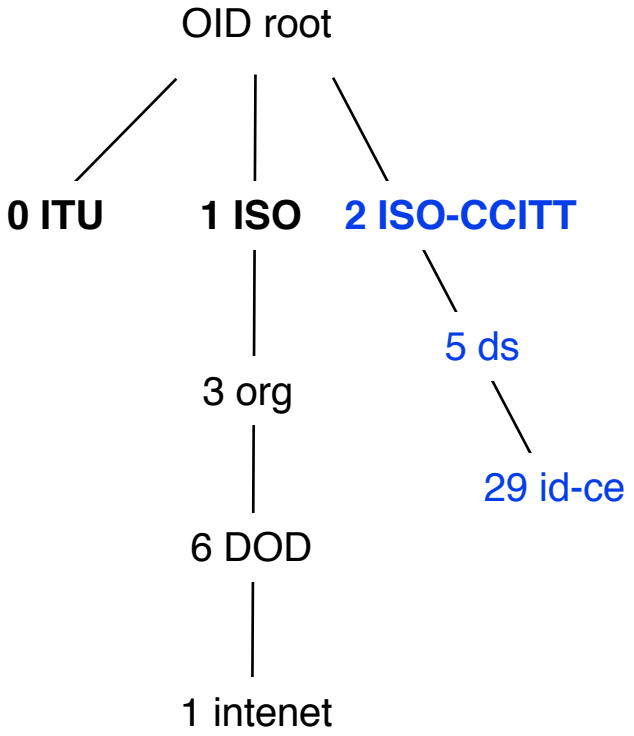
- ASN.1 defines a syntax

```
SubjectPublicKeyInfo ::= SEQUENCE  {
  algorithm              AlgorithmIdentifier,
  subjectPublicKey       BIT STRING
}
AlgorithmIdentifier  ::=  SEQUENCE  {
  algorithm              OBJECT IDENTIFIER,
  parameters             ANY DEFINED BY algorithm OPTIONAL
}
```

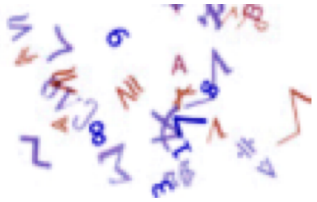
- ASN.1 defines basic data types
  - BOOLEAN, INTEGER, REAL, CHARACTER STRING, OCTET STRING, BIT STRING,
  - SEQUENCE, SET, SEQUENCE OF, SET OF, CHOICE, ENUMERATED,
  - OID, RELATIVE OID.

ASN.1: what is an OID ?

- An OID consists of a node in a hierarchically-assigned namespace.
- OIDs are uniq and assigned by IANA.
- A particular semantic is attached to each OID.




```
id-ce OBJECT IDENTIFIER ::= { joint-iso-ccitt(2) ds(5) 29 }
```



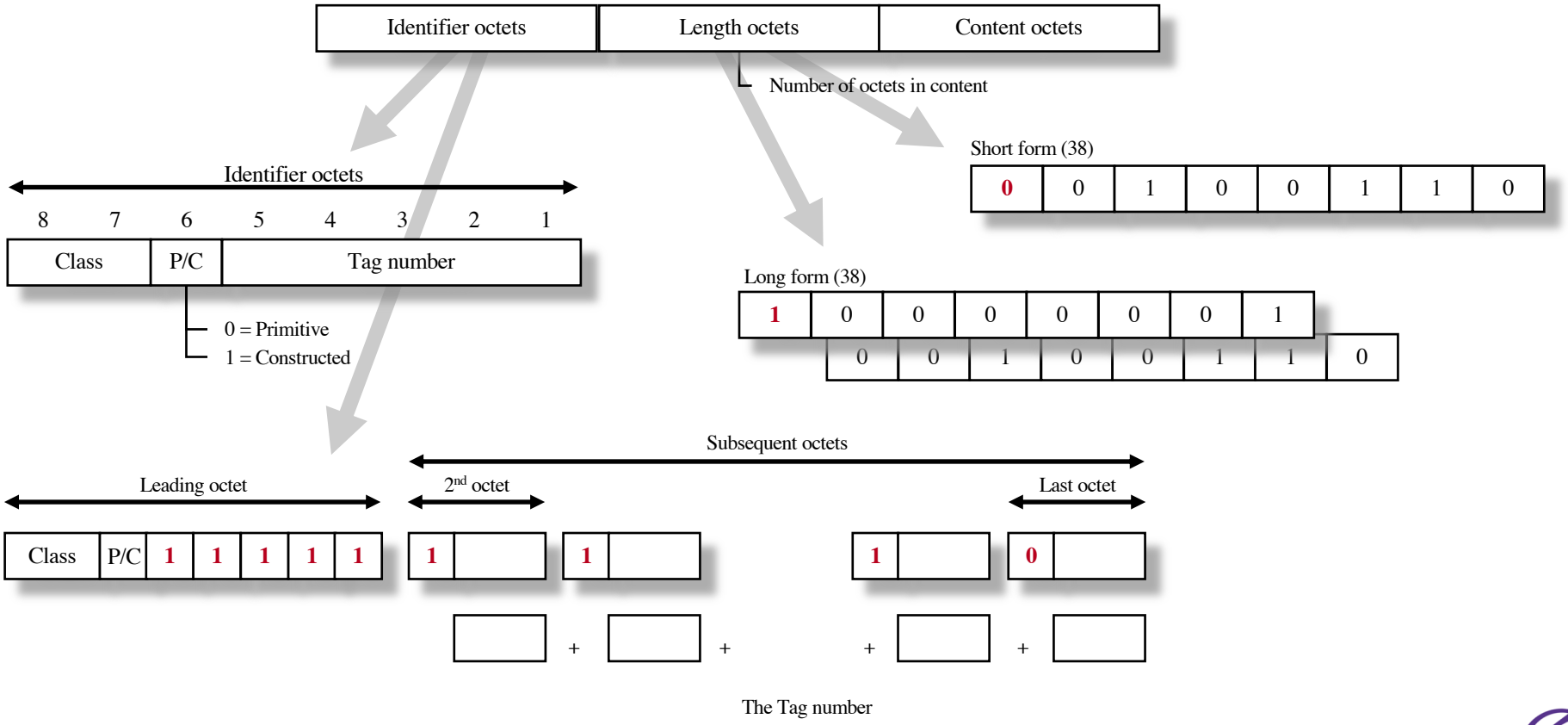
ASN.1 data encoding

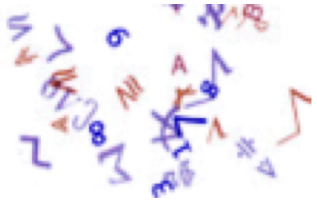
- BER : Basic Encoding Rules
- DER : Distinguished Encoding Rules
  - A BER subset.
  - Encoding is unique (required for signature).
- XER : XML Encoding Rules

  - ITU-T X.690, ISO 8825-1

Distinguished Encoding Rules

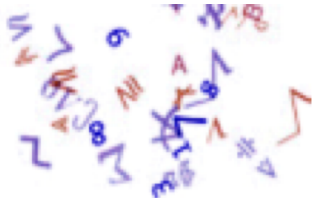
- *type-length-value encoding*





ASN.1 certificate definition

```
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,
   subjectUniqueID   [2]  IMPLICIT UniqueIdentifier OPTIONAL,
   extensions        [3]  EXPLICIT Extensions OPTIONAL }
```

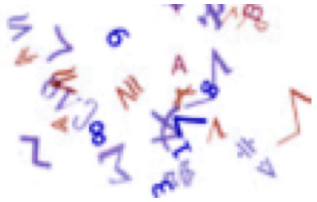



X509 v3 extensions

```
Extension  ::=  SEQUENCE  {
   extnID       OBJECT IDENTIFIER,
   critical     BOOLEAN DEFAULT FALSE,
   extnValue    OCTET STRING  }
```

- Each extension in a certificate is designated as either **critical** or **non-critical**.
  - A certificate MUST reject the certificate if it encounters a critical extension it does not recognize;
  - however, a non-critical extension MAY be ignored if it is not recognized.

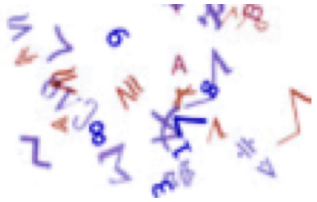
```
id-ce OBJECT IDENTIFIER
            ::= { joint-iso-ccitt(2) ds(5) 29 }
```

Some useful extensions

KeyUsage,
BasicConstraints,
SubjectAltName,
certificatePolicies,
extKeyUsage,
NameConstraints,
PolicyConstraints,
...

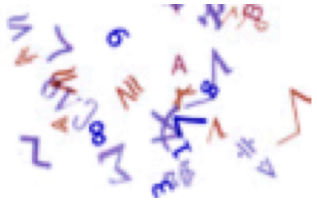
Old netscape specific extensions are deprecated.



KeyUsage

- Defines the purpose of the key contained in the certificate.

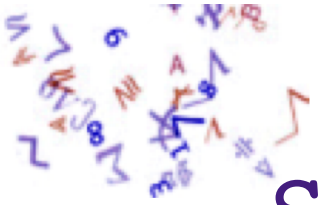
```
id-ce-keyUsage OBJECT IDENTIFIER ::= {id-ce 15}

KeyUsage ::= BIT STRING {
   digitalSignature        (0),
   nonRepudiation          (1),
   keyEncipherment         (2),
   dataEncipherment        (3),
   keyAgreement            (4),
   keyCertSign             (5),
   cRLSign                 (6),
   encipherOnly            (7),
   decipherOnly            (8)
}
```

BasicConstraints

- The basic constraints extension identifies whether the subject of the certificate is a CA and the maximum depth of valid certification paths.
  - MUST appear as critical if used to validate digital signatures
  - If the CA is not asserted, then the `keyCertSign` bit in the key usage MUST NOT be asserted.

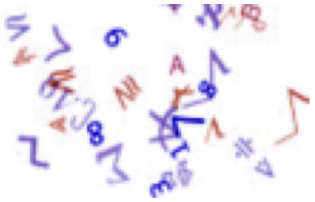
```
id-ce-basicConstraints OBJECT IDENTIFIER ::= {id-ce 19}

BasicConstraints ::= SEQUENCE {
   cA                      BOOLEAN DEFAULT FALSE,
   pathLenConstraint       INTEGER (0..MAX) OPTIONAL
}
```

SubjectAltName

- Allows additional identities to be bound to the subject of the certificate.

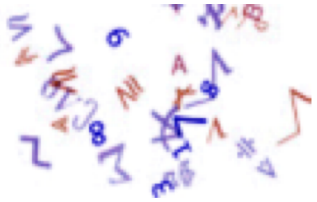
```
id-ce-subjectAltName OBJECT IDENTIFIER ::= { id-ce 17 }

SubjectAltName ::= GeneralNames

GeneralNames ::= SEQUENCE SIZE (1..MAX) OF GeneralName

GeneralName ::= CHOICE {
   otherName                       [0]  OtherName,
   rfc822Name                      [1]  IA5String,
   dNSName                         [2]  IA5String,
   x400Address                     [3]  ORAddress,
   directoryName                   [4]  Name,
   ediPartyName                    [5]  EDIPartyName,
   uniformResourceIdentifier       [6]  IA5String,
   iPAddress                       [7]  OCTET STRING,
   registeredID                    [8]  OBJECT IDENTIFIER
}
```

certificatePolicies

- The policy under which the certificate has been issued and its usage purposes.

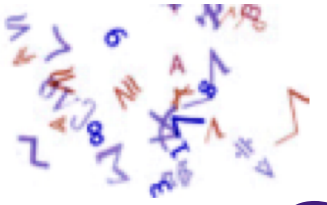
```
id-ce-certificatePolicies OBJECT IDENTIFIER ::=  { id-ce 32 }

anyPolicy OBJECT IDENTIFIER
        ::= { id-ce-certificate-policies 0 }

certificatePolicies :
        := SEQUENCE SIZE (1..MAX) OF PolicyInformation

PolicyInformation ::= SEQUENCE {
   policyIdentifier   CertPolicyId,
   policyQualifiers
          SEQUENCE SIZE (1..MAX) OF PolicyQualifierInfo OPTIONAL
}
CertPolicyId ::= OBJECT IDENTIFIER
```

extKeyUsage

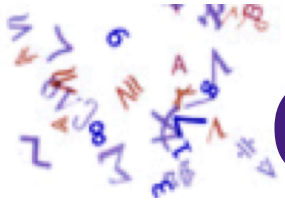
- one or more purposes for which the certified public key may be used, in addition to or in place of the key usage extension.

```
id-ce-extKeyUsage       OBJECT IDENTIFIER ::= { id-ce 37 }
ExtKeyUsageSyntax       ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId

KeyPurposeId            ::= OBJECT IDENTIFIER

     anyExtendedKeyUsage        OBJECT IDENTIFIER ::= { id-ce-extKeyUsage 0 }
     serverAuth                 OBJECT IDENTIFIER ::= { id-kp 1 }
     clientAuth                 OBJECT IDENTIFIER ::= { id-kp 2 }
     codeSigning                OBJECT IDENTIFIER ::= { id-kp 3 }
     emailProtection            OBJECT IDENTIFIER ::= { id-kp 4 }
     timeStamping               OBJECT IDENTIFIER ::= { id-kp 8 }
     OCSPSigning                OBJECT IDENTIFIER ::= { id-kp 9 }
```

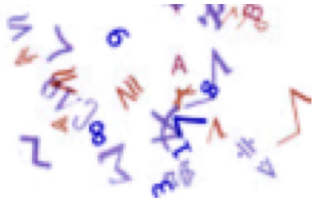
Certificate example

```
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 11 (0xb)
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C=FR,O=EMSE,OU=RIM,CN=RIM-CA
        Validity
            Not Before: Mar  4 17:28:49 2016 GMT
            Not After : Mar  4 17:28:49 2019 GMT
        Subject: C=FR,O=EMSE,OU=RIM,CN=Philippe Jaillon
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            RSA Public Key: (2048 bit)
                Modulus (2048 bit):
                    00:b8:ed:9b:2b:07:a6:d1:a8:64:d5:c5:7c:56:6c:
                    ...
                    ab:db:81:27:a8:2e:1e:c9:b4:04:c0:ab:89:3e:10:
                    1d:79
                Exponent: 65537 (0x10001)
```

Certificate example

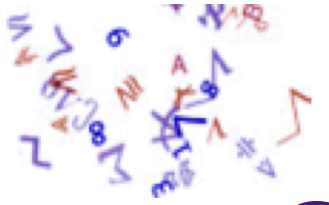
```
X509v3 extensions:
    X509v3 Basic Constraints:
            CA:FALSE
    X509v3 Key Usage:
            Digital Signature, Non Repudiation, Key Encipherment
    X509v3 Extended Key Usage:
            TLS Web Client Authentication, E-mail Protection
    X509v3 Subject Alternative Name:
            email:philippe.jaillon@emse.fr, email:jaillon@emse.fr
    X509v3 CRL Distribution Points:
            URI:http://www.emse.fr/crl.pem
    X509v3 Certificate Policies:
            Policy: 1.2.3.4.5.6.7
            CPS: https://www.emse.fr/cps.html
    X509v3 Subject Key Identifier:
            7D:DA:C1:D5:75:BE:00:2F:A9:9A:C4:75:4E:F4:57:D2:EA:E0:B0:68
    X509v3 Authority Key Identifier:
            keyid:AD:F5:2D:33:35:F5:3A:C1:3A:AC:1D:8F:EE:...
            DirName:/C=FR/O=EMSE/OU=RIM/CN=RIM-CA
            serial:02

Signature Algorithm: sha256WithRSAEncryption
    95:6c:2a:f5:06:68:a1:dd:ec:26:49:64:bf:19:1b:97:27:48:
    ...
```

How to deliver certificates

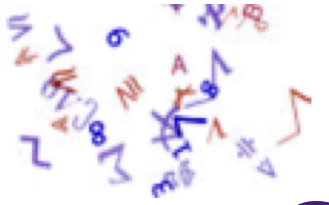
- To ensure confidence in the PKI, certificate issuers need to define tasks, procedures… policies to deliver certificates.

It’s the role of each Certification Authority.



Certification Authority (CA)

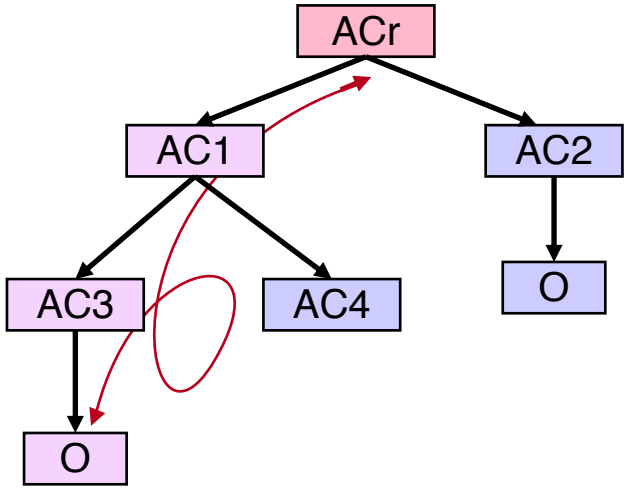
- It's an organisation which deliver certificates to a particular population.
- It has its own certificate.
- Where to use the certificate depends of your trust in the CA:
  - Inside an entity (at EMSE for example).
  - Just for exchange between two entities.
  - ...



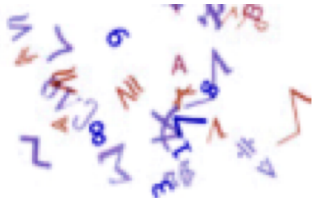
Certification authority

- A CA plays the role of a Third Trusted Party.
  - When a CA delivers a certificate, it is responsible for the identity and usage provided with this certificate.
- The trust level in a CA depends on:
  - How is verified the identity of the applicant,
  - How is protected the CA private key (security)

Hierarchical Certification



- At the top of the tree, we find the root CA. The root CA certificate is self signed.
- A root CA can deliver certificates to one or more other CA, these CA (*intermediary CA*) can deliver certificates to other CA.
- A certification chain is the set of certificates required to validate all the certificates necessary to reach a trusted root certificate.

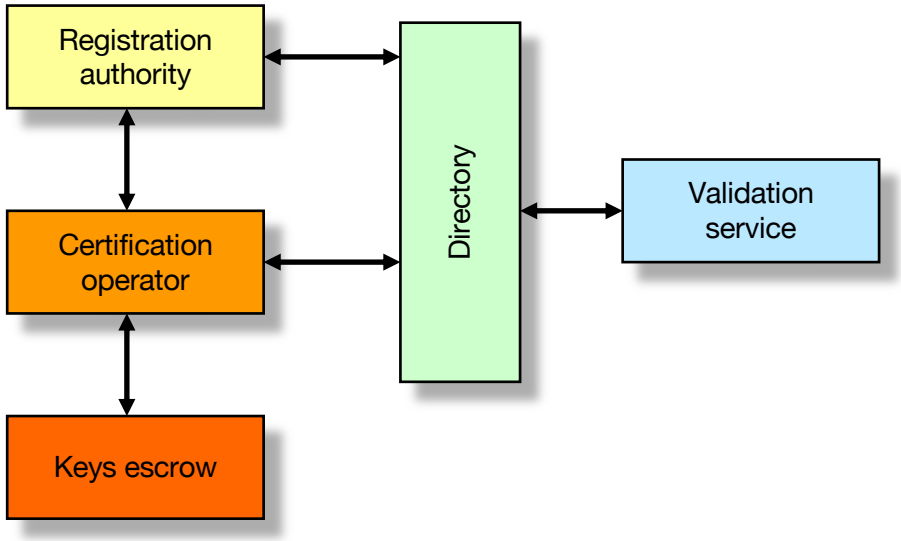


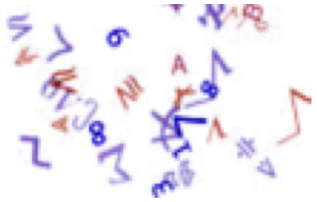
Public key infrastructure

- Purpose of a PKI
  - Certificate creation and revocation
  - Certificate directory management
  - Key escrow and private key recovery management
- A PKI is based on hardware, software, people, rules and procedure essential to a certification authority to create, manage and distribute X.509 certificates.
- It's the PKI which establishes the trust relation between entities involved.

Public Key Infrastructure

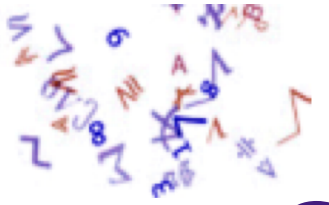
- Parts in a certification authority (CA)
  - Registration Autority (RA)
  - Certification Operator (CO)
  - Certificates directory
  - Validation service
  - Possibly, a key escrow service





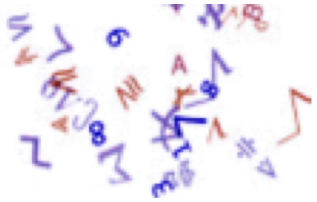
Registration Authority

- The RA receives and processes creation, renewal and revocation of certificates. For this purpose:
  - Must control subject information (identity) provided by the certificate applicant,
  - Must validate revocation requests,
  - Must manage certificate recovery to ensure continuity in signature and/or ciphering.
- The RA has two components:
  - inquiring interface for end users,
  - validation interface reserved to Registration operator.



Certification Operator

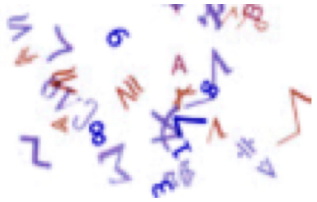
- Certification Authority delegates to the Certification Operator operation requiring the CA private key:
  - certificate creation and distribution,
  - certificate revocation
- The CO manages the certificate life cycle with the registration authority.



CO security consideration

The major risk for a PKI is the CA Private key compromise.

- Certification Operator must not be connected to the network.
- Certificate request transfer between Registration and Certification Authority must not be automatic.
- Physical security of the CO is critical and must be studied with care.
- …

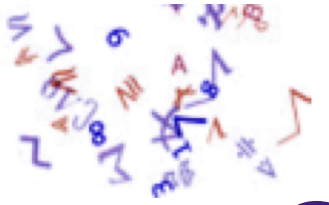


Key escrow

- If a user loses the private key associated to its certificate, he loses the possibility to decipher any data ciphered with its certificate (the public key).
- A key escrow service can archive all the private/public key pairs associated to certificates provided by the PKI.
- PKI can offer a recovery service.

◆ Problems

- ➢ Private key is shared by the user and the PKI.
- ➢ It is impossible to archive private keys if certificates are used for signature (legal aspect of signature).



Certification Policy

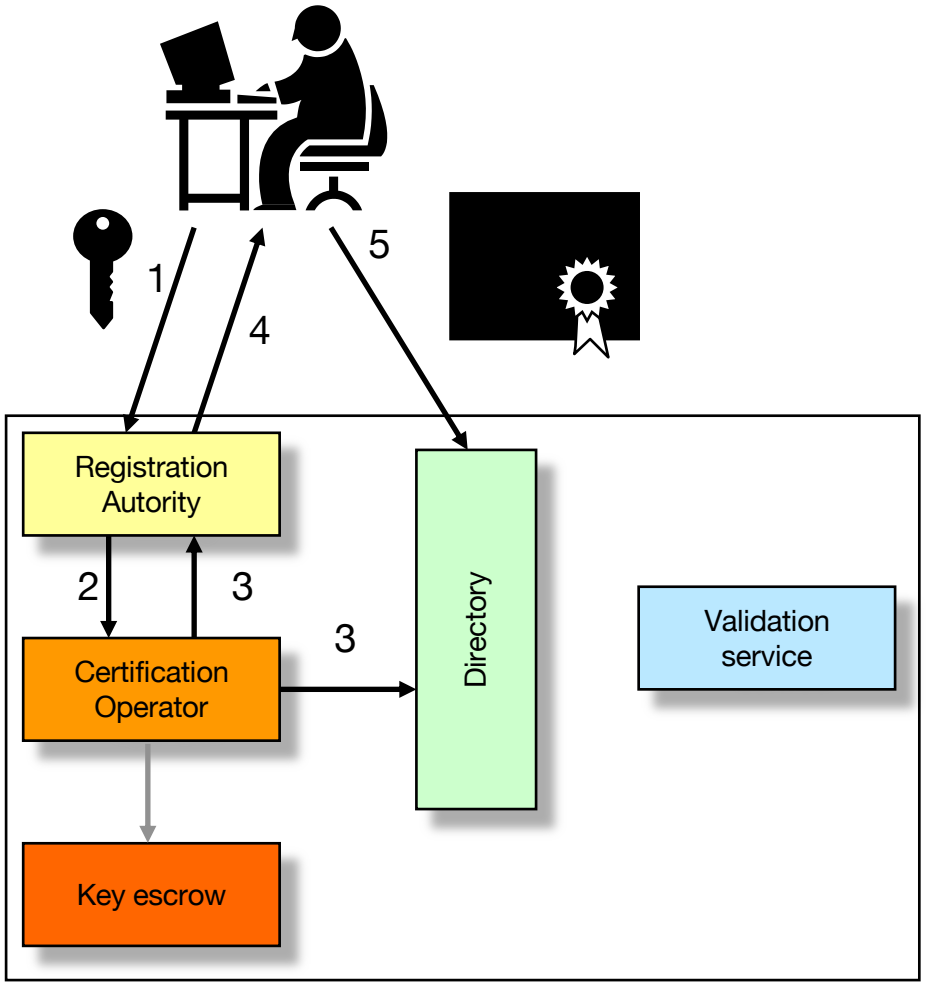
A set of rules describing why certificates are made for, by who and what are the technical, administrative and legal usage conditions.

- Certification Policy (CP)
  - It expresses the security objectives of the CA.
  - The CP defines the security level associated to the certificate.
- Certification Practice Definition (CPD)
  - Describes technical process used by the different parts of the PKI (CA, RA,...), conforming to the CP.

Certification policies are used to establish common standard and insurance criteria for interoperability between entities.

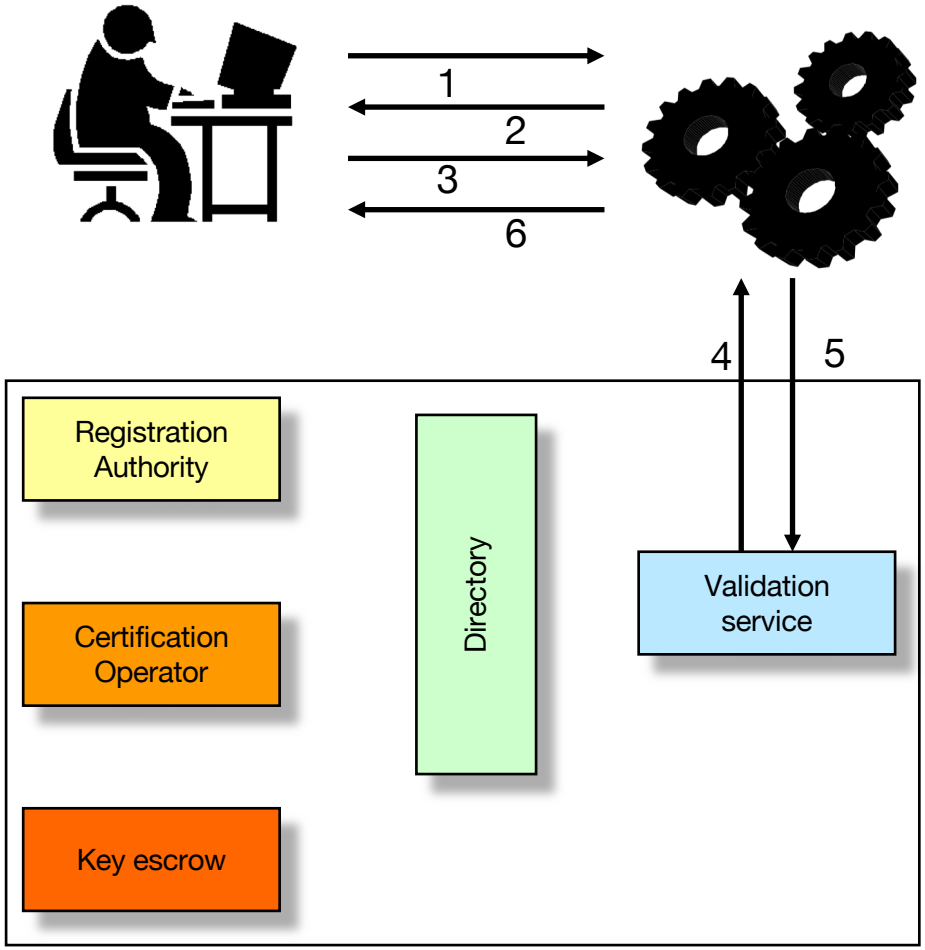
Certificate request

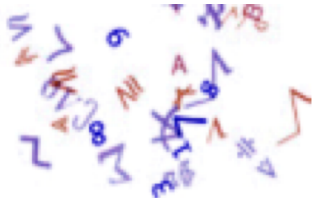
1. User send its request to the RA
2. The RA verifies identification data and private key ownership. The RA validates the request and transfers it to the CO
3. The CO verifies the request validity and creates the certificate. The certificate is published into the directory and send to the RA.
4. The RA warns the user about the certificate availability.
5. The user retrieves the certificate



Accessing resource

1. User requests access to the application
2. Application asks for user certificate
3. User provides its certificate
4. Application verifies certificate validity and revocation state with PKI validation service
5. Answer of the validation service
6. Application grants access to the user according to the verification results

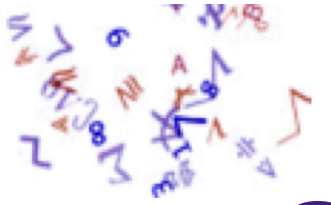




IETF Standards:

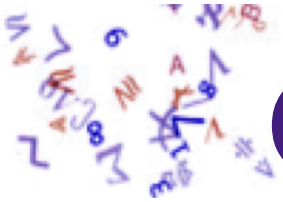
PKIX group-1995. Working group on usage of X.509 certificate in the Internet:

- RFC 5280 Certificate and Certificate Revocation List Profile.
- RFC 4210 Certificate Management Protocols: create, revoke certificates, cross certification...
- RFC 3647 Certificate Policy and Certification Practices Framework: how to identify subject, physical security, revocation rules...
- RFC 2559, 2560, 2585 : protocols for certificates and revocation lists diffusion (LDAP/ X.500, OCSP, HTTP/FTP).
- RFC3029 Data Validation and Certification Server Protocols.
- RFC3161 Time-Stamp Protocol (TSP).
- …



Open problems

- Certificate lost
  - It is not possible to verify signature or certificate validity (CA)
- Keys validity
  - Large revocation lists, OCSP protocol
  - Was the document signed during validity period of the certificate (time-stamping problem)
- Outdated keys
  - Keys, even when outdated, must be kept, they are essential to verify signature, decipher documents…
  - Who is in charge of preserving and archiving these keys (billions or more)

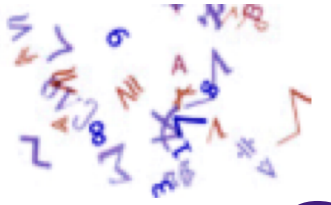


OCSP

Online Certificate Status Protocol

- Validation is a complex task:
  - It is initiated by client,
  - It is critical task,
  - validation is often incomplete,
  - CRL update/download isn't automatic.
- OCSP partially solve the problem:
  - Certificate state is up to date,
  - client doesn't need to get and process CRL (less network and CPU usage),
  - Reduce information leak - a CRL can be seen as a certificate bad holders list.
  - But there are privacy issues… OCSP server knows in real time where you are connecting to.

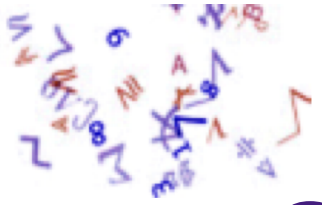
OCSP : rfc 2560



OCSP

Alice and **B**ob are **I**van's clients (CA). **A**lice want to complete transaction with **B**ob.

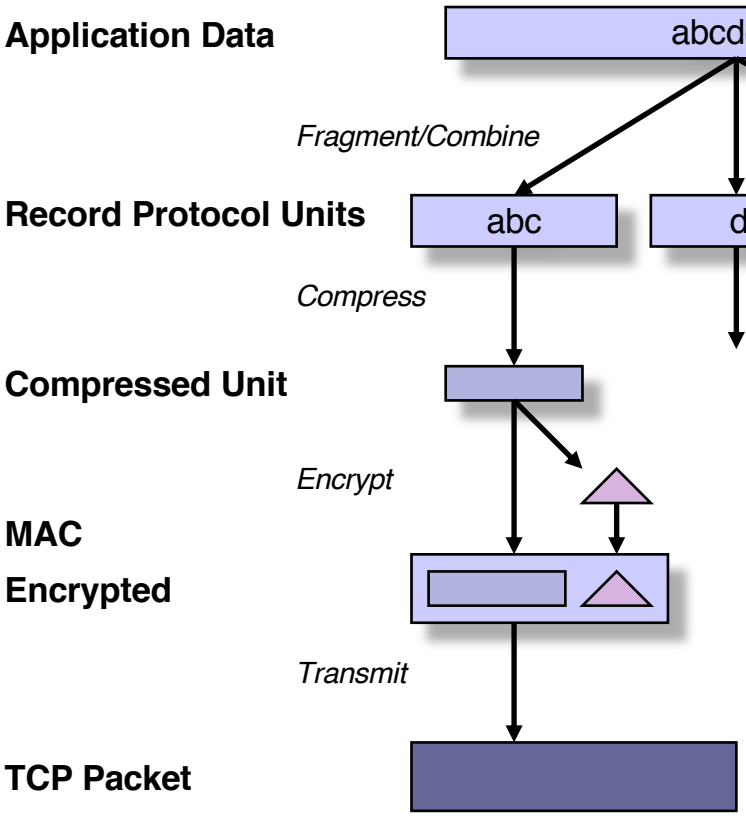
1. **A**lice send his certificate to **B**ob,
2. To verify that **A**lice certificate wasn't revoked, **B**ob builds an OCSP request using **A**lice certificate and send it to the OCSP service managed by **I**van,
3. OCSP service verifies the **A**lice's certificate status,
4. OCSP service signs the current certificate's validity and sends it back to **B**ob,
5. **B**ob verifies the answer signature.

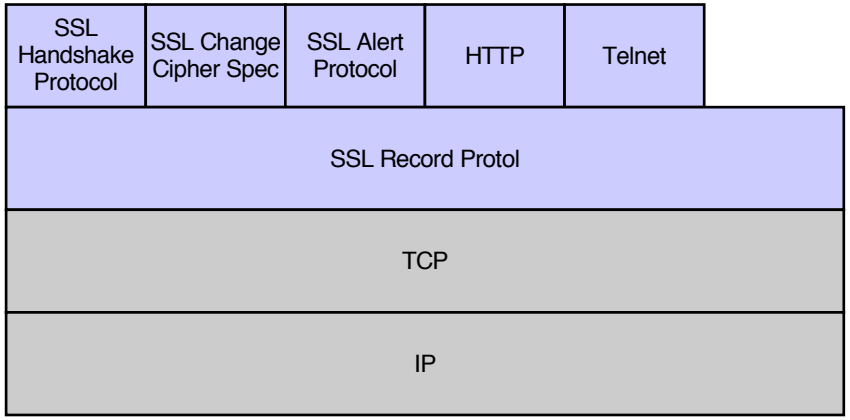


Secure Socket Layer (SSL)

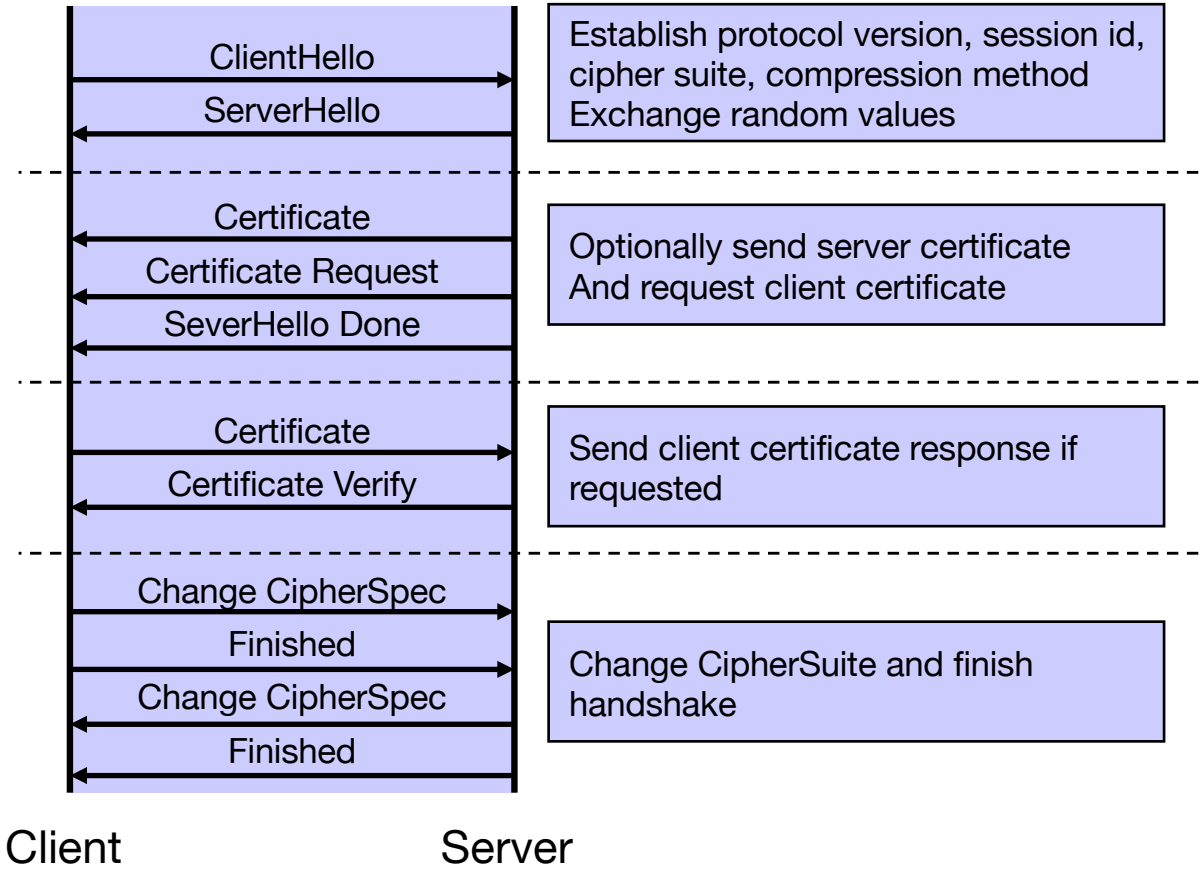
- Protocol at application layer using TCP/UDP. SSLv3 (Netscape), TLS V1 (RFC 2246)
  - Integrity,
  - Confidentiality,
  - identification of peers.
- Large usage for authentication and ciphering in email and web applications
- X509 certificates (RSA based) are used for peers identification

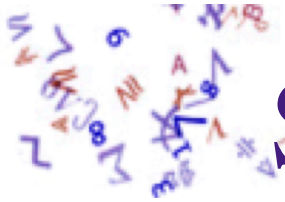
SSL : communications





SSL connexion negotiation





Security of SSL/TLS
Connexion negotiation

Phase 1 : Hello messages

1 C -> S : {v, NC ,id?,cipher[],compression[],extension[]}
2 S -> C : {v, NS ,id,cipher,compression,extension[]}

Phase 2 : server send its certificate and the certification chain

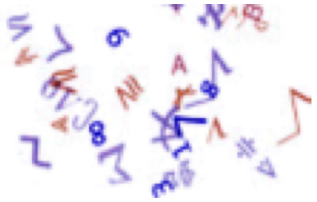
3 S -> C : {**S**, Ca[]}
4 S -> C : {}
5 S -> C : {certType[],signAlgo[],certAuth[]}
6 S -> C : {}

Phase 3 : client send certificate certification chain (*optional*)

7 C -> S : {**C**, Ca[]}
8 C -> S : $\{v,pmkey\}_S$
9 C -> S : $\{msg[1..8]\}_{C\ -1}$

Phase 4 : finish handshake

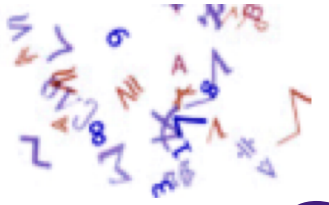
10 C -> S : PRF(mkey,"client finish",hash(msg[1..9]))
11 S -> C : PRF(mkey,"server finish",hash(msg[1..10]))



Security of SSL/TLS

- If server doesn't request client certificate, SSL/TLS is sensible to *man-in-the-middle* attack if server certificate is accepted by the client.
  - Its goal of On-Shelf products (SSL Inspector, Mitm…)
  - Remember the ANSSI problem in 2013.
  - To minimize this threat, Google hard-code its own certificates in Chrome, Android... (certificate pinning)
- In any case, identities used by clients and servers are accessible to an observer.

Be careful with SSL/TLS in insecure places.



Commercial aspects

- Certificate cost examples (2015/01):
  - Verisign Secure Site SSL certificate :
    - Class 1 : 399$, class 2 : 995$, class 3 (EV) :1499$/year
      wilcard server : 1999$/year
  - Certinomis
    - Persons (3 year) : 255€ (pro **) / 200€ (perso *)
    - Servers (1 year **) : 350€ pièce, wilcard server : 1150€/year
      - Scellement de données envoyées par un serveur
      - Authentification de Web Service
      - Authentification d'un serveur web
    - RGS-A requires server certificates.

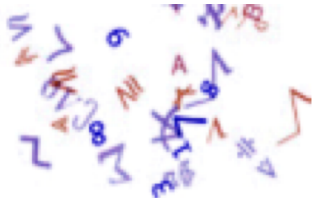
  For economical reasons, certificates need to be renewed frequently.

Commercial aspects

- Within a browser, any root certificates are equal.
- The Extended validation
  - Must respect the Guidelines for Extended Validation Certificates (CA/Browser Forum).
  - Use the *certificate policies* extension

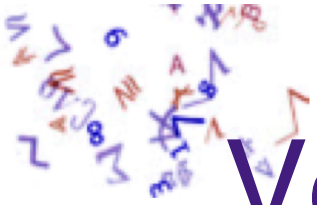
```
X509v3 Certificate Policies:
        Policy: 2.16.840.1.113733.1.7.23.6
        CPS: https://www.verisign.com/rpa
```

  - Link between the policies OID and the corresponding CA are hard-coded in browsers
  - If the browser can verify that the OID is signed by the right CA, it sets the *green bar*



Implementation problems

- All the browsers are provided with a large collection of root certificates
- How to make the difference between certificate policies
  - Example : Verisign has 4 certificate policies.
  - What can we think about certificates provided in foreign countries (without justice or commercial relations with us).
- Every certificates signed by these CA are valid for your browser.



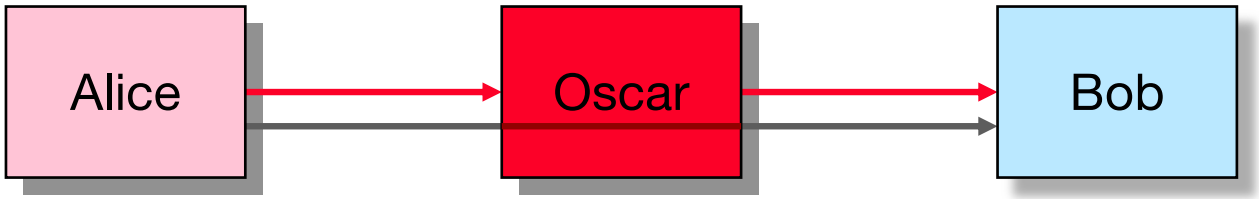
Verisign

Certificate type, validation procedures and usages:

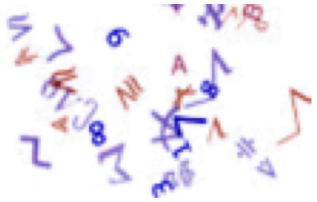
- Class 1 Certificates.
  - Based on assurances that the Subscriber's DN is unique and unambiguous within the domain of a particular CA and that a certain e-mail address is associated with a public key. It isn't a proof of identity.
- Class 2 Certificates.
  - Authentication includes procedures based on a comparison of information submitted by the certificate applicant against information in business records or databases or the database of a VeriSign-approved identity proofing service.
- Class 3 Certificates.
  - Provide assurances of the identity of the Subscriber based on the personal (physical) presence of the Subscriber before a person that confirms the identity of the Subscriber using, at a minimum, a well-recognized form of government-issued identification and one other identification credential.

Implementation problems

- What can we think about an unverifiable certificate?
- If we can't verify certificate authenticity, certificate authentication is sensitive to *"man in the middle"* attack.



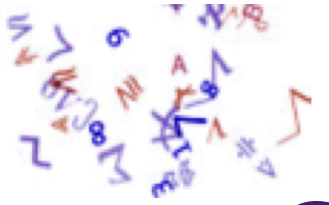
We provide our password
as clear text to unknown people…



What can we think about a certificate?

A certificate is valid if we trust the issuing CA

- My OS trusts 170 CA (OSX 10.12.1)????
- How to know that legitimates Google certificates are signed by Thawte?
- What can we said if a Google server provide a certificate signed by one of the 170 other trusted CA?



COMODO

- March 15, 2011
  - a user account with an affiliate registration authority had been compromised.
  - Nine certificates were issued: mail.google.com, login.live.com, www.google.com, login.yahoo.com, login.skype.com, addons.mozilla.org, and global trustee.
  - The attack was traced to IP address 212.95.136.18, which originates in Tehran, Iran.
  - All of the certificates have been revoked ???
- March 26, 2011
  - "ComodoHacker" made several posts, claiming to be an Iranian responsible for the attacks.

DigiNotar drama

- A well known Dutch CA
- Provide certificates for the Dutch government

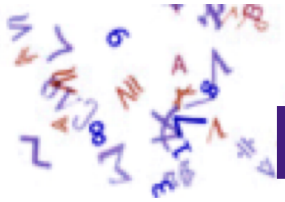
July 10, 2011
- wildcard certificate was issued for Google by an attacker who access to DigiNotar systems

July 19, 2011
- DigiNotar detect an intrusion into its certificate authority infrastructure

August 28, 2011
- certificate problems were observed on multiple Internet service providers in Iran.
- 531 misissued certificates for domains including Google, Yahoo!, Mozilla, WordPress, The Tor Project…
- DigiNotar could not guarantee all such certificates had been revoked
- OS and browsers updates remove CA certificate from trusted roots (Windows, OSX, Firefox, Chrome…)

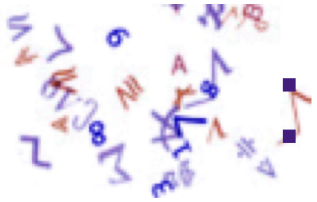
September 3, 2011
- The Dutch government switch to a different firm as certificate authority.

September 20, 2011
- DigiNotar was declared bankrupt



How to know Google certificates are issued from Thawte CA ?

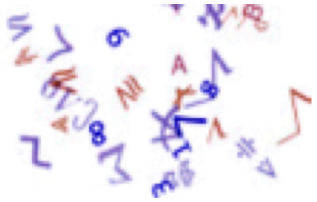
- Using DNSSEC and DANE proposal ?
  - DNSSEC provides a chain of trust like X509 but with only **one** root of trust
  - DNSSEC records are signed
  - TSLA: new DNS record with information on legitimate certificate and CA
- But
  - Requires world wide deployment of DNSSEC
  - Validation by end user application



Certificate Transparency

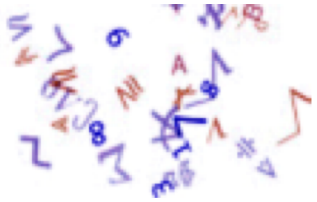
- Any CA can publish a certificate for any domain name.
- RFC 6962
  - Encourage or require AC to publish openly the certificates they sign.
  - Certifcate holder which fears a CA sign a certificate in his name without his permission has to monitor the public certificate file...

https://www.certificate-transparency.org/



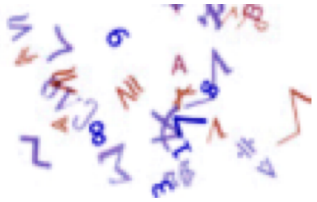
How to use it

- Only using certificates signed by trusted CA (excluding all the other one) within a contractual framework.
- Using its own CA inside a community (enterprise, project …)
- A well managed CA is a good and secure alternative to login/password
  - ciphering, authentication …



Tools

- openssl : not a tool, but a toolbox
  - Cryptography
  - Certificate creation
  - Simple CA management
  - Tools for client and server
- certutil (NSS Mozilla project)
  - Tool to manage certificates and key databases
- stunnel
  - Add ssl to any network services
- mod_ssl
  - Using ssl within your apache web server



vpn ssl

Transport IP packet in a ssl UDP session,
or worst using TCP ???

- IPSec is difficult to use within the Internet
  - No end to end communications : firewalls, proxy, address translation…
  - Now we use only some UDP/TCP ports,
  - Firewalls work are switching from network level to application level,
  - Internet became an HTTP network.
- Such mechanism isn't at its right place in the OSI architecture:

  transport level is provided in application level.