Applied Cryptography

Symmetric Crypto

Asymmetric Crypto

Applications

Roadmap

- Public-Key Infrastructure (PKI)
  - X509 Certificates
- Communication Protocols
  - TLS
- Document Signing
  - PADES
- Cryptographic Tokens
  - National Citizen Card

Public-Key Infrastructure (PKI)

Certificate Standards

- The widespread use of public-key certificates requires a solid foundation for its establishment and systematisation:
    - Formats and technology;
    - Conventions and assumptions;
    - Practices and procedures;
    - Regulatory and legal framework;
    - Etc.
- In fact, a PKI bridges the gap between concepts and elements of a technological nature (cryptographic keys) and aspects that are ultimately social (identity; individuality; personality).

Public Key Infrastructure (PKI)

A **Public Key Infrastructure** (**PKI**) is a set of roles, policies, hardware, software and procedures required to create, manage, distribute, use, store and revoke **digital certificates**.

- It is an area that has largely developed in response to localised stimuli and successful solutions to specific problems that, once sufficiently mature, have been widely adopted.
  - The X509 was the first successful attempt to standardise public key certificates (ITU-1998) as the authentication mechanism for X500 standard (Directory Services).
  - These certificates were later adopted for the SSL protocol for securing HTTP(S) connections (*Netscape* SSLv2 — 1995).
  - The IETF-PKIX-WG was set up (1995-2013) to promote the development of technical standards (RFCs) to support PKI based on X509 certificates.

PKIX

- Different entities involved in a PKI:
  - **Certificate holders** (**subjects**): who requests the certification of his/her public key;
  - **Clients**: who need to use the public key contained in certificates;
  - **Certification Authorities** (**CA**s): entities responsible in issuing/renewing/revoking certificates.
  - **Registration Authorities**: (**RA**s) entities responsible in establishing the association between public keys and holders' identities (optional).
  - **Repositories**: store and make available certificates and other relevant information (such as revoked certificates, etc.).
- PKIX defined a set of specifications and protocols allowing
  - systems/application to effectively rely on X509 certificates (operational protocols)
  - PKI management — operations (initialisation; certificate lifecycle; distribution); policies; best practices; etc.

X509 certificates

- X509 certificates include:
  - Informative Data:
    - **subject** (i.e. who holds the private key associated with that certified public key);
    - The holder's **public key**;
    - The identification of the **issuer** (CA);
    - Other relevant information (serial number; validity period; extensions; etc.)
  - **Signature** by the CA.
- The data is represented as an attribute/value data structure (dictionary).
- The encoding of this data is standardised, guaranteeing interoperability.
  - **DER** - binary format defined by the standard (ASN.1 notation)
  - **PEM** - representation of the information contained in the DER format in printable characters.
- The certificate is signed by the Certification Authority (CA) on the DER encoding of the informative data it contains.

X.509v3 certificates

- Certificates currently in use (X509v3 — 1996) addressed shortcomings of previous versions in some application domains (lack of attributes).
- This version introduced an **Extensions** field, thus equipping certificates with the flexibility needed, by allowing generic attributes to be associated with the subject or its public key.
- Each extension is itself a data structure with an identifier and a value appropriate to the type of attribute it represents.
- Extensions are marked as **Critical** or **Non Critical**. An application that finds a critical extension that it doesn't recognise must reject the certificate.

Certificate structure

- Basic attributes
    - **version** - Version of the X509 standard (v3).
    - **serialNumber** - Unique number assigned by the CE to the certificate.
    - **subject** - Identification of the holder of the public key contained in the certificate.
    - **subjectPublicKeyInfo** - Structure containing the certificate holder's public key and identification of the corresponding algorithm.
    - **issuer** - Identification of the CE issuing the certificate.
    - **signature** - Structure that identifies the algorithm used to generate the signature of the CE that accompanies the certificate.
    - **validity** - Structure containing two dates that delimit the validity period of the certificate.
- Extensions (selected)
    - **basicConstraints** - signals if subject is a (sub-)CA (and optionally a limit on the path-length);
    - **keyUsage** - restricts the uses of the key pair associated with the certificate;
    - …

Certificate chains

- Certificates are signed objects, which is why they can be transmitted in open channels.
- Of course, proper use of certificates imply verifying its signature, which in turn implies knowing the public key of the issuing CA. There are two options:
    1. The public key is already known to the user (e.g. it has been securely pre-installed);
    2. or it is provided by another certificate (issued by some other CA).
- Naturally, in the second case, the validity of the new certificate needs to be checked, which results in a **recursive validation process.**
- The sequence of certificates involved in verifying a particular certificate is called its **certificate chain**.

Certificate validation

- For each certificate in a well-formed certification chain, verify:
    - certificate’s signature;
    - the applicability of the certificate (e.g. is not expired; usage is according its attributes; etc.);
    - (it has not been revoked)
- The root of the certification chain must be from a CA whose public key is already known - this is known as the **trust anchor** (root of the trust relationship).
- By convention, **root-certificates** are **self-signed** (subject is the issuer).

Trust anchors

- A user knows a limited number of public keys belonging to CAs (generally Root CAs), and which act as the roots of trust relationship.
- Management of the list of trusted anchors is normally carried out by the operating system.
- In particular, the compilation and update of Root certificates is normally ensured by the OS manufacturer.
- This makes the process of using certificates mostly transparent to the end user.
    - **Which is good!**, because both the concepts and the handling of certificates is complex.
    - **Which is bad!**, because there is no awareness of the trust relationships involved.

Certificate revocation

- Sometimes there is a need to revoke certificates that are still valid.
- Possible reasons for revocation:
    - Private key compromised;
    - Circumstance that justified the issuer's association with the public key no longer exists (e.g. issuer is the holder of a temporary position).
    - Data contained in the certificate is no longer correct (e.g. attribute no longer applies)
    - Etc.
- The original mechanism for revoking certificates was the **Certificate Revocation List** (**CRL**) — a signed list of certificates revoked by the CA that should be consulted when verifying certificates (the CRL distribution point, if applicable, is included in the certificate).
- But CRL's shortcomings recommend the use of a real-time verification mechanism — the **Online Certificate Status Protocol** (**OCSP**).

Communication Protocols

Secure Communication Protocols

- Various protocols allow the establishment of a **secure channel** — a communication channel ensuring:
    - Integrity of data;
    - Entity Authentication (Server side or mutual authentication);
    - Confidentiality (optional).
- These protocols act on different levels on the OSI model: e.g.
    - network layer (IPSec, VPNs);
    - transport layer (TLS);
    - application layer (SSH; S/MIME; PGP)

Typical structure

- Secure communication protocols typically include two phases:
    1. **Handshake** (or **key management**) — negotiate parameters; cryptographic algorithms and keys to setup a secure communication link;
    2. **Secure communication link** — realises the secure channel using symmetric cryptography.
- Asymmetric cryptography usually plays its role only at the first phase.
- Occasionally, the handshake phase may be replayed (e.g. rekeying).

TLS

- Originally conceived by *Netscape inc.* as **Secure Sockets Layer** (**SSL**).
- It is to TCP as IPsec is to IP. It is an upgrade of the transport layer to include security in communications.
- SSL v3 was adopted by the IETF under the name **Transport Layer Security** (**TLS**).
- SSL/TLS supports the following authentication modes:
    - **server authentication** - mode normally adopted in web scenarios (HTTPS) — server certifies its identity by means of a signature and using a X509 certificate with appropriate attributes.
    - **mutual authentication** - the server also requests that the client authenticate itself cryptographically, requiring it to display an appropriate certificate.

TLS security

- Even if the current version of the TLS protocol is considered secure, the truth is that the history of attacks that the SSL/TLS family of protocols has faced over the last years is embarrassing (see https://en.wikipedia.org/wiki/Transport_Layer_Security#Attacks_against_TLS.2FSSL) .
- In general, attacks aimed at the protocol itself have been overcome by successive revisions (e.g. "ciphersuit rollback", possible in the SSL.2 version, was overcome with the authentication of handshake messages required by SSL.3).
- The most recent attacks on SSL/TLS are mostly aimed at:
    - Less careful implementation aspects or obscure protocol features (e.g. as in the *Heartbleed attack)*;
    - interaction with specific cipher suites (e.g. use of RC4 or *padding oracle attack* in CBC mode).
- In practice, the sheer number of configuration options requires great care in deployment (see https://ssllabs.com).

TLS 1.3

- The current version of the TLS protocol (v1.3) has recently completed its final standardisation phase (RFC-8446).
    - It consists of a substantial redefinition of the protocol:
    - It drops backward compatibility with previous versions;
    - It aims to achieve: greater efficiency and greater security (vs. TLS1.2).
- Some features:
    - Simplifies handshake (1 roundtrip).
    - Removes support for "obscure" or little-used features; obsolete cryptographic techniques (e.g. (3)DES, RC4, MD5, SHA-1, AES-CBC, ...); etc.
    - Chooses to support only a restricted set of possibilities (DHE groups; AEAD modes; …)

Message/Document Signing

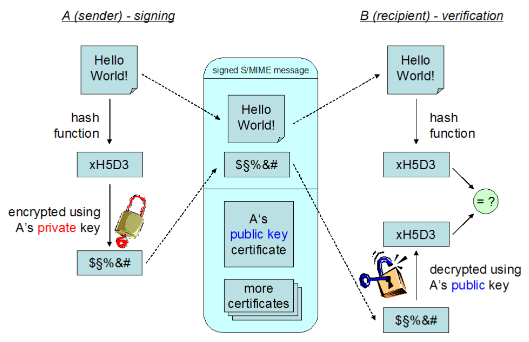
EMail security

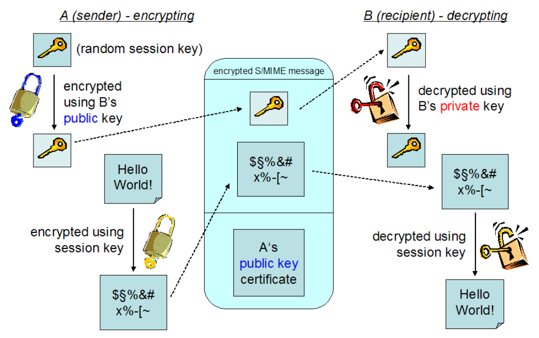
- The widespread use of email makes it a natural area to use cryptography to enhance its security.
- It provides combinations of the following security services:
  - **authentication**, **integrity** and **non-repudiation** of the origin of messages through the use of digital signatures;
  - data **confidentiality** through the use of a public key encryption (more precisely, a *digital envelope*).
- Early instances of secure email systems ran into considerable legal problems (specifically, PGP)…
- …making it a flagship in the promotion of online privacy.

S/MIME

Secure/Multipurpose Internet Mail Extensions

- S/MIME offers a consistent way of sending and receiving secure information in MIME format.
- It relies on X509 certificates and PKIX to authenticate public keys.
- Supported by most existing EMail applications (Mail User Agents — MUA).





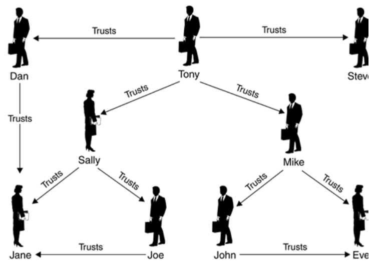
A (sender) - encrypting
B (recipient) - decrypting
(random session key)
encrypted using B's public key
Hello World!
encrypted using session key
$§%&# x%-[~
encrypted S/MIME message
$§%&# x%-[~
A's public key certificate
decrypted using B's private key
$§%&# x%-[~
decrypted using session key
Hello World!

PGP
Pretty Good Privacy

- PGP is a freeware application developed by *Phil Zimmermann* (1991) with the aim of making an infrastructure for protecting information (i.e. privacy) available to ordinary citizens.
- Controversy in the US due to laws restricting the widespread dissemination and use of so-called strong cryptography. The associated legal battle was an important milestone in the ever-shifting balance between individual privacy and national security.

"*if information protection is outlawed, only outlaws can have privacy!*"

- Technically, the most salient (and original) feature was its approach to public-key certification: the *web-of-trust*.
    - avoids dependency on a rigid trust hierarchy (such as in PKIX);
    - mimics the patterns of social interaction;
    - each user can certify other's public-keys;
    - …and choose to rely on others in that task.

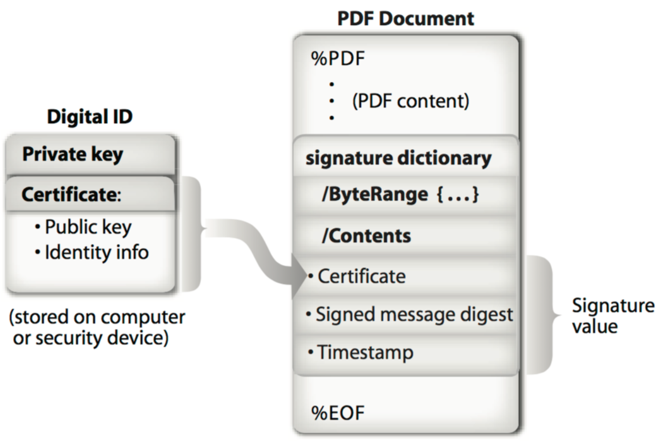


Document Signing

- An immediate application of digital signatures is to certify the origin of documents.
- Legislators have been paying attention to this application, giving it a legal framework.
- Regardless of the cryptographic details and the legal framework, we can affirm that the security of digital signatures depends on:
    1. The security of the cryptography algorithms and their implementation;
    2. Guarantees that the private key is truly secret;
    3. The legitimate holder of the public key can be verified;
    4. Users and Software correctly carry out the signature/verification protocol.

PAdES

- PDF is one of the most widely used formats for supporting digital documents.
- Originally from a private company (Adobe), it has since been standardised by the relevant bodies (ISO 32000-1, 32000-2).
- Includes support for digital signatures (since V1.7), allowing to take advantage of the feature using freely available applications (e.g. Acrobat Reader).
- More recent versions have updated support for digital signatures to comply with ETSI's **PAdES — PDF Advanced Electronic Signatures** standard.
- PAdES is a technical standard that aligns PDF digital signatures with EU regulation eIDAS.



Qualified Electronic Signature (QES)

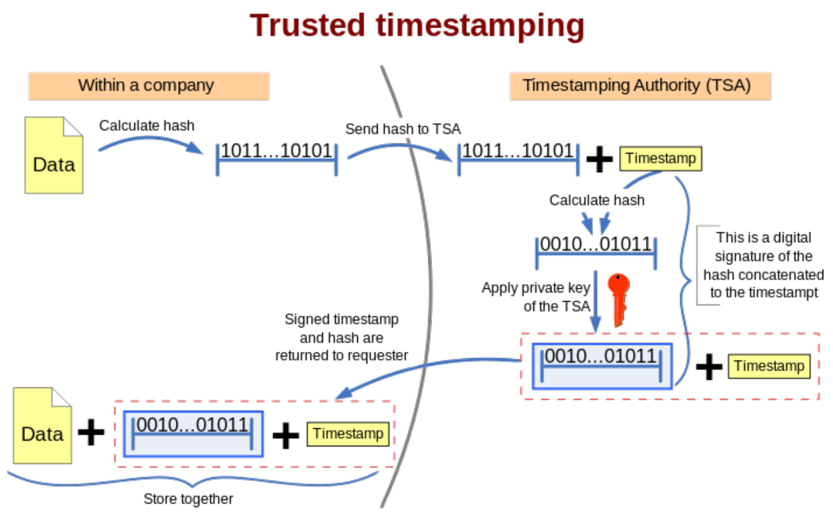
- When signing documents, a Qualified Signature is of particular relevance in the European context (eIDAS).
- A **Qualified Electronic Signature** (**QES**) is a digital signature in which:
    1. identity of the signatory is attested by a Qualified Certificate;
    2. Created by a secure electronic device.
- The notion of **Qualified Certificate** imposes strict requirements on CA (Certification Service Providers — CSP).
- In particular, each member state is responsible for accrediting CSPs with the capacity to issue qualified certificates certificates (valid in all member states).

Trusted Timestamping

- **Timestamping** is the process of keeping track of the creating and modification time of a document.
- A trusted timestamp is issued by a **Trusted Third Party** (**TTP**) acting as a **Time Stamping Authority** (**TSA**).
- Cryptographically, it consists of a signature linking the content of the document (its hash) and a timestamp obtained from a reliable source.
- When used in digital signatures, it establishes the time of the signature creation (important in, e.g., to check the validity of the certificate at that time).
- Timestamping is one of the **trust services** included in the eIDAS regulation.
- Thus, a **Qualified Timestamp** issued in the EU is valid and recognised in all other Member States.

Timestamp Service

Timestamping Authority (TSA)


Trusted timestamping
Within a company
Timestamping Authority (TSA)
Data
Calculate hash
1011...10101
Send hash to TSA
1011...10101
Timestamp
Calculate hash
0010...01011
This is a digital signature of the hash concatenated to the timestampt
Apply private key of the TSA
0010...01011
Timestamp
Signed timestamp and hash are returned to requester
Data
0010...01011
Timestamp
Store together