

# Certificates and Public Key Infrastructure (PKI)

## ***Abstract***

*Basic concepts of certificates and Public Key Infrastructure (PKI) are presented. The chain of trust of certificates and Certification Authorities is explained. The model of X.509 certificates is presented.*

# Outline

---

- Public key cryptography and Public Key Infrastructure (PKI)
- Certificates and chain of trust
- X.509 certificates

# Public Key Cryptography

---

- The encryption key is public, but the decryption key is known only by the intended recipient
  - ◆ everyone knows what has to be done to find the decryption key, based on public data
  - ◆ security is based on the fact that finding the decryption key from public data is theoretically possible, but computationally unfeasible
- How can I trust that a public key is really posted by the legitimate entity?



- Public Key Infrastructure (PKI)

# Public Key Infrastructure (PKI)

---

- Set of policies and procedures
  - ◆ to generate and publish keys
  - ◆ to generate, publish, validate and revoke digital certificates
  - ◆ to manage public-key encryption
- PKIs define certification and validation operations
  - ◆ a *certificate* binds a public key to the respective entity (user or piece of information)
  - ◆ *validation* guarantees that a certificate is valid
  - ◆ a certificate is signed by its publisher (*Certification Authority, CA*)
  - ◆ two types of certificates
    - *identity certificates* provide and guarantee some entity's identity information (e.g., address, public keys)
    - *credential certificates* provide and guarantee information describing access rights to a resource

# Trusting a Certificate and Public Keys

- A Certificate of Alice published by a CA
  - ◆ provides and guarantees the public key  $K_A$  of Alice
  - ◆ is signed by the Certification Authority (or *Trusted Authority*, TA) using its private key  $K_{TA}^{-1}$

$$C_A = \{ A, K_A, \{h(A, K_A)\}_{K_{TA}^{-1}} \}$$

- ◆ can be validated by anyone using the public key  $K_{TA}$  of TA
- Who trusts what?
  - ◆ if I trust CA → I trust the information in the certificate ( $K_A$ )  
→ I do not necessarily trust A
  - ◆ I have to trust  $K_{CA}$  to trust the certificate
  - ◆ a CA can certify a lower-level CA (also *Registration Authority*, RA) building a *chain of trust* (multiple-level hierarchy)
  - ◆ a first-level CA signs its own certificate with its private key, while its public key is supposed trustable (provided with Internet browsers)

# X.509 Certificates

---

- The ITU-T X.500 series of Recommendations defines a directory service
  - ◆ directory: database of user information, incl. public keys
  - ◆ v1: 1988; v7: 2012
- ITU-T Rec. X.509 define authentication services by the X-500 directory to its users
- X.509 is based on public-key cryptography and digital signatures
  - ◆ does not dictate specific digital signature and hash algorithms
- The X.509 certificate format is used in S/MIME, IPsec, SSL/TLS

# Basic Information in an X.509 Certificate

---

- *Version*
- *Serial number* of the certificate (unique for each CA)
- *Issuer name* (the CA who created and signed the certificate)
- *Period of validity* (from/to)
- *Subject name* (the certificate certifies the public key of the subject who holds the corresponding private key)
- *Public Key of the Subject*
- *Signature* (a field specifies the signature algorithm)

# Revocation of Certificates

---

- Each certificate includes a period of validity
- In addition, a certificate could be *revoked* before it expires, because
  - ◆ the user's private key has been compromised
  - ◆ the user is no longer certified by that CA
  - ◆ the CA's certificate has been compromised
- Each CA must maintain a list of all *revoked but not expired certificates*, which it previously issued
  - ◆ the list (signed by the CA) is posted in the directory
  - ◆ certificates are identified by their SNs
  - ◆ users should maintain a local cache to avoid checking the revocation list each time a certificate is received

# Example of X.509 Certificate

- Subject's Public Key (RSA):
  - ♦ control bits
  - ♦  $n$  ( $p \cdot q$ , 1024 bits)
  - ♦  $e = 2^{16} + 1$

## Certificate Hierarchy

► Verisign Class 1 Public Primary Certification Authority - G2

## Certificate Fields

Verisign Class 1 Public Primary Certification Authority - G2

### Certificate

Version: Version 1

Serial Number: 39:CA:54:89:FE:50:22:32:FE:32:D9:DB:FB:1B:84:19

Certificate Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Issuer: OU = VeriSign Trust Network

OU = (c) 1998 VeriSign, Inc. - For authorized use only

OU = Class 1 Public Primary Certification Authority - G2

O = VeriSign, Inc.

C = US

### Validity

Not Before: 05/17/98 20:00:00 (05/18/98 00:00:00 GMT)

Not After: 05/18/18 19:59:59 (05/18/18 23:59:59 GMT)

Subject: OU = VeriSign Trust Network

OU = (c) 1998 VeriSign, Inc. - For authorized use only

OU = Class 1 Public Primary Certification Authority - G2

O = VeriSign, Inc.

C = US

Subject Public Key Info: PKCS #1 RSA Encryption

Subject's Public Key:

30 81 89 02 81 81 00 aa d0 ba be 16 2d b8 83 d4  
ca d2 0f bc 76 31 ca 94 d8 1d 93 8c 56 02 bc d9  
6f 1a 6f 52 36 6a 75 56 0a 55 d3 df 43 87 21 11  
65 8a 7e 8f bd 21 de 6b 32 3f 1b 84 34 96 05 9d  
41 36 ab 92 eb 96 dd aa 59 3f 01 53 6d 99 4f ed  
e5 e2 2a 5a 90 c1 b9 c4 a6 15 cf c8 46 eb a6 5d  
8e 9c 3e f0 64 24 76 a5 cd ab 1a 6f b6 d8 7b 51  
61 6a a8 7f 87 c8 c2 b7 e5 34 dc 41 68 ea 09 40  
be 73 92 3d 6b e7 75 02 03 01 00 01

Certificate Signature Algorithm: PKCS #1 SHA-1 With RSA Encryption

Certificate Signature Value:

8b f7 1a 10 ca 76 5c 07 ab 83 99 dc 17 80 6f 34  
39 5d 98 3e 6b 72 2c e1 c7 a2 7b 40 29 b9 78 88  
ba 4c c5 a3 6a 5e 9e 8e 7b e3 f2 02 41 0c 66 be  
ed fb ae a2 14 ce 92 f3 a2 34 8b b4 b2 b6 24 f2  
e5 d5 e0 c8 e5 62 6d 84 7b cb be bb 03 8b 7c 57  
ca f0 37 a9 90 af 8a ee 03 be 1d 28 9c d9 26 76  
a0 cd c4 9d 4e f0 ae 07 16 d5 be af 57 08 8a d0  
a0 42 42 42 1e f4 20 cc a5 78 82 95 26 38 8a 47