Topic 2.3.2 - Asymmetric Encryption

Eric Casanovas

Universitat d'Andorra

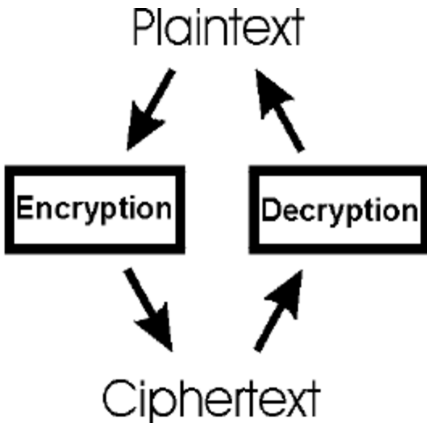
2nd March, 2022

Signatures - Recap

- A message is signed using the private key to generate the signature
- A signature is verified using the public key derived from the private key

Encryption - Recap

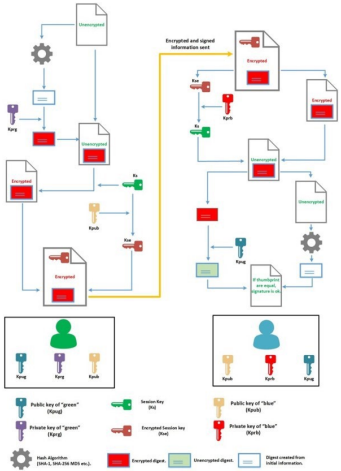
- A message is encrypted using the public key to generate the ciphertext
- The ciphertext is decrypted using the private key related to the public key



Signatures and Encryption I

- How to send a signed encrypted message (asymmetric key):

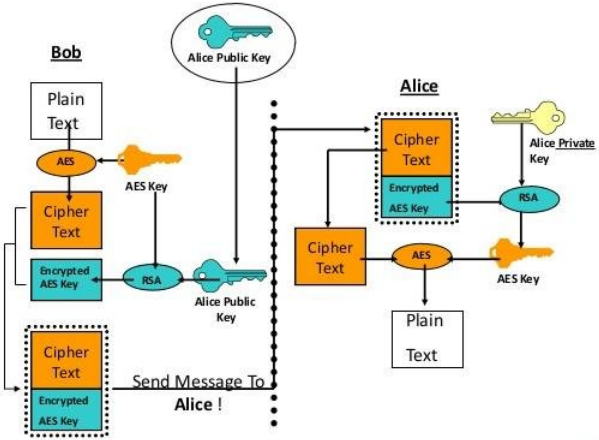
Signatures and Encryption II



Signatures and Encryption III

- How to send a signed encrypted message (hybrid key):

Signatures and Encryption IV



Signatures and Encryption V

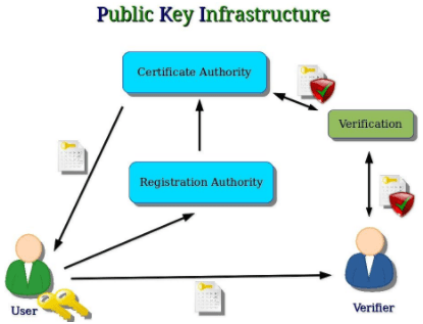
- But, How we know that it is Alice who sent us her public key?

Signatures and Encryption VI

- But, How we know that it is Alice who sent us her public key?
- We need PKIs
- We need certificates

What is a PKI?

- It is a system of resources, policies, and services that supports the use of public key encryption to authenticate parties involved in a transaction.



Why PKIs?

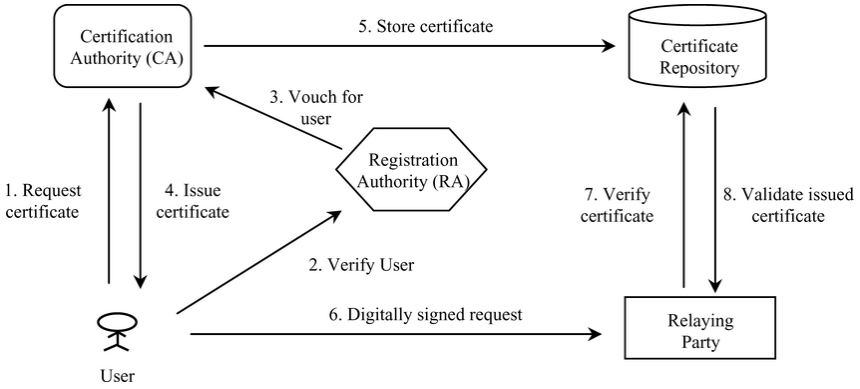
- Facilitate the secure electronic transfer of information
- Infrastructure that binds an identity to a public key
- 3rd party trust model

PKIs I

We can divide PKIs in:

- Certification Authorities (CAs): Stores, issues and signs the digital certificates. CAs can also generate new CAs called subCAs and the "mother" of all CAs is the root CA
- Registration Authorities (RA): Verifies the identity of entities requesting their digital certificates to be stored at the CA
- End Entities (EE): Entities that want certificates

PKIs II


Certification Authority (CA)
5. Store certificate
Certificate Repository
3. Vouch for user
Registration Authority (RA)
1. Request certificate
4. Issue certificate
7. Verify certificate
8. Validate issued certificate
2. Verify User
6. Digitally signed request
Relaying Party
User

What is certificate?

- Electronic document certified (signed) by an authority (CA)
- Binds an identity to a public key
- Contains:
  - Owner's PK
  - Owner's name
  - Expiration Date
  - Serial Number
  - Name of issuer
  - Digital signature of issuer
- Standards: **X.509**, PKIX

Certificates I

Types of certificates:

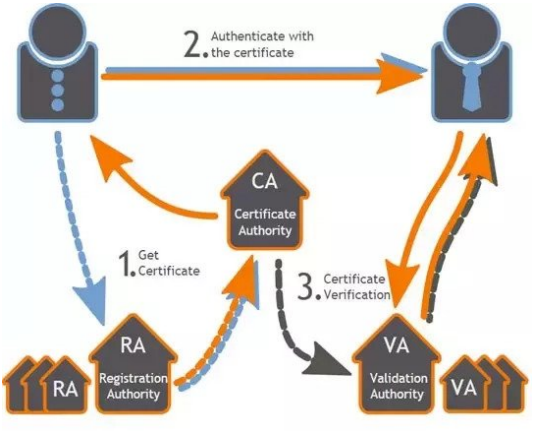
- Self-signed certificates
- Signed certificates

Certificates II

- PKIs can generate certificates
- PKIs can revoke certificates

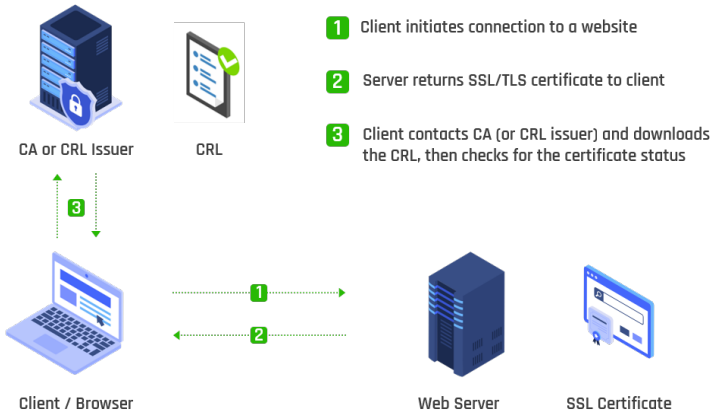
Certificates III

Certificate generation:



Certificates IV

Certificate revocation:



X-509 I

X.509 fields:

- Version
- Serial Number
- Algorithm ID
- Issuer
- Validity
- Not Before
- Not After
- Subject
- Subject Public Key Info
- Public Key Algorithm
- Subject Public Key
- Issuer Unique Identifier (Optional)
- Subject Unique Identifier (Optional)
- Certificate Signature Algorithm
- Certificate Signature

X-509 II

```
~/Downloads > openssl x509 -in 2048b-rsa-example-cert.pem -text -noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number: 3580 (0xdfc)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C = JP, ST = Tokyo, L = Chuo-ku, O = Frank4DD, OU = WebCert Support, CN = Frank4DD Web CA, emailAddress = support@frank4dd.com
        Validity
            Not Before: Aug 22 05:27:41 2012 GMT
            Not After : Aug 21 05:27:41 2017 GMT
        Subject: C = JP, ST = Tokyo, O = Frank4DD, CN = www.example.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                RSA Public-Key: (2048 bit)
                Modulus:
                    00:b4:cf:d1:5e:33:29:ec:0b:cf:ae:76:f5:fe:2d:
                    c8:99:c6:78:79:b9:18:f8:0b:d4:ba:b4:d7:9e:82:
                    52:86:80:f4:18:03:4c:d4:70:d1:42:a0:29:13:92:
                    73:50:77:f6:84:89:ac:03:2c:d6:f1:06:ab:ad:6c:
                    c0:d9:d5:a6:ab:ca:cd:5a:d2:56:26:51:e5:4b:08:
                    8a:af:cc:19:0f:25:34:90:b0:2a:29:41:0f:55:f1:
                    6b:93:db:9d:b3:cc:dc:ec:eb:c7:55:18:d7:42:25:
                    de:40:35:14:32:92:8c:1e:c6:69:e3:3c:f8:f4:9a:
                    f8:fb:8b:c5:e0:1b:7e:fd:4f:25:ba:3f:e5:96:57:
                    9a:24:79:49:17:27:d7:89:4b:6a:2e:0d:87:51:d9:
                    23:3d:06:85:56:f8:58:31:8e:ee:01:99:70:68:cd:
                    6e:44:7e:c9:da:8c:5a:7b:1c:bf:24:40:29:4b:d1:
                    03:9c:ef:dc:ae:2a:5d:f8:f7:6a:c7:e9:bc:c5:b0:
                    59:f6:95:fc:16:cb:d8:9c:ed:c3:fc:12:90:93:78:
                    5a:75:b4:56:83:fa:fc:41:84:f6:64:79:34:35:1c:
                    ac:7a:85:0e:73:78:72:01:e7:24:89:25:9e:da:7f:
                    65:bc:af:87:93:19:8c:db:75:15:b6:e0:30:c7:08:
                    f8:59
                Exponent: 65537 (0x10001)
    Signature Algorithm: sha1WithRSAEncryption
         40:cb:fe:04:5b:c6:74:c5:73:91:06:90:df:ff:b6:9e:85:73:
         fe:e0:8a:6f:3a:44:2f:cc:53:73:16:32:3f:79:64:39:e8:78:
         16:8c:62:49:6a:b2:e6:91:85:08:b7:4f:38:da:03:b9:81:69:
         2e:18:c9:49:96:84:c2:eb:e3:23:f4:eb:ac:68:4b:57:5a:51:
         1b:d7:eb:c0:31:6c:86:a0:f6:55:a8:f8:10:d0:42:86:1e:94:
         a5:e0:68:a7:9f:b6:f3:9c:d0:e1:22:3b:ab:85:3d:a1:27:9b:
         50:32:62:b8:ec:7a:fa:d6:7d:2b:29:e6:ad:b2:69:4d:28:b4:
         f8:13
~/Downloads
```

Use cases

- SSL/TLS (we can see it on any website)
- User/Client in local networks (for instance active directory)

What is a Mac? I

- Using all techniques known as far we can:
  - Send secure messages
  - Prove the identity
  - Check data integrity?
- Hashing only is not 100% secure (Rainbow Tables)
- MAC is a piece of data to authenticate the message

What is a Mac? II

- Many Schemes:
  - HMAC (Hash MAC)
  - CMAC (Cipher Block MAC)
  - Checksum

Questions

- Encrypt-then-sign or sign-then-encrypt
- Encrypted signature or not

The END!