Koliokviumą siūlau surengti Lapkričio 7 d., 17:30.
Reikės išspręsti 2 uždavinius iš
https://imimsociety.net/en/14-cryptography
DH-KAP ir MiM Attack.
Prisiregistruokite savo pavardės pirmąją raide taškas Vardas, t.y. P.Vardas
ir gausite 10 Eur virtualių pinigų.
Pirkti reikia tik 1 uždavinį ir jį išsprendus pirkti kitą.

Public Key Infrastructure - PKI   Viešojo Rakto Infrastruktūra - VRI

$A: (PrK_A, PuK_A)$ $B: (PrK_B, PuK_B)$
PuK_A

$PuK_A = a = g^x \bmod p$

M - message to be signed

$|M| \sim 1\,GB$

Hash and sign paradigm:

$h = H(M)$; $|h| \sim 256$ bits ← SHA256

$Sign(PrK_A, h) = \sigma = (r, s)$

M, σ, PuK_A →
1) $h' = H(M)$
2) $Ver(PuK_A, \sigma, h') = \begin{cases} True \\ False \end{cases}$

1) If Ver = True, then signature σ is formed using A's private key PrK_A which corresponds (is mathematically related) with A's public key PuK_A.

ECDSA: $PrK_A = x$, $|x| \sim 256$ bits

$x \sim 2^{256}$ and $PuK_A = x \cdot G = A$ ←--→ $PuK_A = g^x \bmod p = a$

$Zo: (PrK_Z, PuK_Z)$ PuK_Z →

Dear Bob I am A and
I am sending you my
public key

Public Key Infrastructure - PKI

$CA = (PrK_{CA}, PuK_{CA})$ It is as notarius office

Certification Authority - CA ⇒ Registration Authorities - RA

Verisign → RA1, RA2, RA3 ....

https://verysign.com

Trusted Third Party - TTP ⇒ all users recognizes PuK_{CA} recognized by the users browsers: Chrome, Opera...

$A: PuK_A \longrightarrow RA \xrightarrow[\text{identity}, PuK_A]{\text{confirms } A} CA:$

(PrK_{CA}, PuK_{CA})

$M_A = PuK_A \| Data_A$

$h_A = H(PuK_A \| Data_A)$

$\sigma_A = Sign(PrK_{CA}, h_A)$

$Cert_A = \sigma_A \| PuK_A \| Data_A$

$A: PuK_{CA}.$ $\xleftarrow{Cert_A, PuK_{CA}}$

$h'_A = H(PuK_A \| Data_A)$

$Ver(PuK_{CA}, \sigma_A, h'_A) = \begin{cases} True \\ False \end{cases}$

$Sign(PrK_A, h) = \xrightarrow[Cert_A]{M, \sigma, PuK_A}$ $B: PuK_{CA}, PuK_A$

1) $Cert_A \longrightarrow \sigma_A$, $PuK_A \| Data_A$

2) $h''_A = H(PuK_A \| Data_A)$

3) $Ver(PuK_{CA}, \sigma_A, h''_A) = \begin{cases} True \\ False \end{cases}$

4) $h' = H(M)$

5) $Ver(PuK_A, \sigma, h') = \begin{cases} True \\ False \end{cases}$

- **X509 v3 Standard**
-
- **SerialNumber**
- **Issuer** ? Verisign

$Zo \leftarrow Cert_Z \leftarrow CA$

- **SerialNumber**
- **Issuer** } Verisign
- **notBefore** } 2021.11.10; 18:10:07
- **notAfter** } 2022.11.10; 18:10:07
- **Subject** } A
- **Algorithm** } ECDSA
- **SubjectPublicKey** } PuK_A
- **extensions**

Zo ← $Cert_Z$ ← CA

2021.10.10; 18:10:11
2022.11.10; 18:10:11
↓
2022.11.10; 18:10:12

Zo : (PrK_Z, PuK_Z); $Cert_Z$.

L - loan contract → $h = H(L)$

$Sign(PrK_Z, h) = \sigma_Z$

L, σ_Z, PuK_Z → B: 1) +
$Cert_Z$ 2) +
3) +
← Money transfer 4) +
5) +
← to pay % for the loan
→ my loan contract is invalid since at the time you've singned it my certificate validity term expired.

CA services: CRL - Certificates Revocation List
OCSP - On-line Certificates Status Protocol

6) Verify if $Cert_Z$ is not in certification revocation list (CRL).

7) If validity of $Cert_Z$ is not expired.

- **Certificates Revocation List - CRL:**

Is a list of digital certificates that have been revoked by the issuing certificate authority (CA) before their scheduled expiration date and should no longer be trusted.

There are two different states of revocation defined in RFC 5280:

Revoked

A certificate is irreversibly revoked if, for example, it is discovered that the certificate authority (CA) had improperly issued a certificate, or if a private-key is thought to have been compromised. Certificates may also be revoked for failure of the identified entity to adhere to policy requirements, such as publication of false documents, misrepresentation of software behaviour, or violation of any other policy specified by the CA operator or its customer. The most common reason for revocation is the user no longer being in sole possession of the private key (e.g., the token containing the private key has been lost or stolen).

Hold

This reversible status can be used to note the temporary invalidity of the certificate (e.g., if the user is unsure if the private key has been lost). If, in this example, the private key was found and nobody had access to it, the status could be reinstated, and the certificate is valid again, thus removing the certificate from future CRLs.

A CRL is generated and published periodically, often at a defined interval. A CRL can also be published immediately after a certificate has been revoked. A CRL is issued by a CRL issuer, which is typically the CA which also issued the corresponding certificates, but could alternatively be some other trusted authority. All CRLs have a lifetime during which they are valid; this timeframe is often 24 hours or less. During a CRL's validity period, it may be consulted by a PKI-enabled application to verify a certificate prior to use.

To prevent spoofing or denial-of-service attacks, CRLs usually carry a digital signature associated with the CA by which they are published. To validate a specific CRL prior to relying on it, the certificate of its corresponding CA is needed.

The certificates for which a CRL should be maintained are often X.509/public key certificates, as this format is commonly used by PKI schemes.

From <https://en.wikipedia.org/wiki/Certificate_revocation_list>

- **On-line Certificates Status Protocol - OCSP:**

is an Internet protocol used for obtaining the revocation status of an X.509 digital certificate.[1] It is described in RFC 6960 and is on the Internet standards track. It was created as an alternative to certificate revocation lists (CRL), specifically addressing certain problems associated with using CRLs in a public key infrastructure (PKI).[2] Messages communicated via OCSP are encoded in ASN.1 and are usually communicated over HTTP. The "request/response" nature of these messages leads to OCSP servers being termed *OCSP responders*.

Some web browsers use OCSP to validate HTTPS certificates.

- Since an OCSP response contains less data than a typical certificate revocation list (CRL), it puts less burden on network and client resources.[3]
- Since an OCSP response has less data to parse, the client-side libraries that handle it can be less complex than those that handle CRLs.[4]
- OCSP discloses to the responder that a particular network host used a particular certificate at a particular time. OCSP does not mandate encryption, so other parties may intercept this information.[

From <https://en.wikipedia.org/wiki/Online_Certificate_Status_Protocol>

Qualified and **Non-qualified** certificates

mathes with e-signature law

Is valid according to contract between parties

Eureka

EU e-document system

2008 m. – 2009 m.

Gemalto Sagem

800 000 €   1 200 000 € ...

$\Sigma \sim 2\,400\,000$

Time Stamping Authority – TSA – Trusted Third Party (TTP)

A: L – loan contract → $h = H(L)$

$Sign(PrK_A, h) = \sigma$ $\xrightarrow[Cert_A]{L,\ \sigma,\ PuK_A}$ TSA: (PrK_{TS}, PuK_{TS}), $Cert_{TS}$.

PuK_{CA}, PuK_A

1. $Ver(PuK_{CA}, Cert_A) = True$
2. $Ver(PuK_A, \sigma, h) = True$
3. $DT = YYYY.MM.DD:hh:mm:ss:...$
4. $h_{TS} = H(h, \sigma, DT, PuK_{TS}, Cert_{TS})$
5. $Sign(PrK_{TS}, h_{TS}) = \sigma_{TS}$

A: PuK_{CA} $\xleftarrow[PuK_{TS},\ Cert_{TS}]{DT,\ \sigma_{TS}}$

1. Verifies DT
2. Verifies validity of $Cert_{TS}$
3. $h'_{TS} = H(h, \sigma, DT, PuK_{TS}, Cert_{TS})$
4. $Ver(PuK_{TS}, \sigma_{TS}, h'_{TS}) = True$ ⇒

If: $\left\{ \begin{array}{l} h'_{TS} = h_{TS} \\ PuK_{TS} = g^{x_{TS}} \bmod p \end{array} \right\} \rightarrow True$

B: (PrK_B, PuK_B)

PuK_{CA}

A: $\xrightarrow{L,\ \sigma,\ PuK_A,\ Cert_A}$

$DT, \sigma_{TS}, PuK_{TS}, Cert_{TS}$

1. $Ver(PuK_{CA}, Cert_{TS}) = True$
2. $Ver(PuK_{CA}, Cert_A) = True$

3. $h' = H(L)$; $h''_{TS} = H(h, s, DT, PuK_{TS}, cert_{TS})$

4. $Ver(PuK_{TS}, \sigma_{TS}, h''_{TS}) = True$

5. $Ver(PuK_A, \sigma, h') = True$

6. OCSP : to verify that notAfter > (DT) → Yes

7. CRL : do the $Cert_A$ is not revoked → Not

A: ← money transfer

AKAP

A: $PrK = x$; $PuK = a$.

$PuK_B = b$; PuK_{CA}; $cert_A$.

$u \leftarrow randi(\mathbb{Z}_{p-1})$

$t_A = g^u \bmod p$

$t \leftarrow randi(\mathbb{Z}_{p-1})$

$r = g^t \bmod p$

$h = H(t_A \| r)$

$s = t + xh \bmod (p-1)$

$t_A, \sigma = (r, s)$ → PuK_A, $cert_A$

$PuK_A = a$.

B: $PrK_B = y$; $PuK_B = b$.

$Ver(a, \sigma, t_A) = True$

$Ver(cert_A) = True$

- - - - - - - - -

Executes AKAP.

Till this place

① A browser verifies TTP signature on PuK_B.

② A verifies B signature $\sigma_B = (R, S)$ on t_B

← $t_B, \sigma_B = (R, S)$, $cert_B$

$v \leftarrow randi(\mathbb{Z}_{p-1})$

$t_B = g^v \bmod p$

$l \leftarrow randi(\mathbb{Z}_{p-1})$

$R = g^l \bmod p$

(2) A verifies TS signature $\sigma_B = (R, s')$ on t_B

(3) A computes common secret key

$k_{AB} = (t_B)^u \bmod p$

$R = g^l \bmod p$

$H = H(t_B \| R)$

$s' = l + yH \bmod (p-1)$

$Cert(PuK_B) = Cert_B$ ↑ signed

Trusted Third Party – TTP

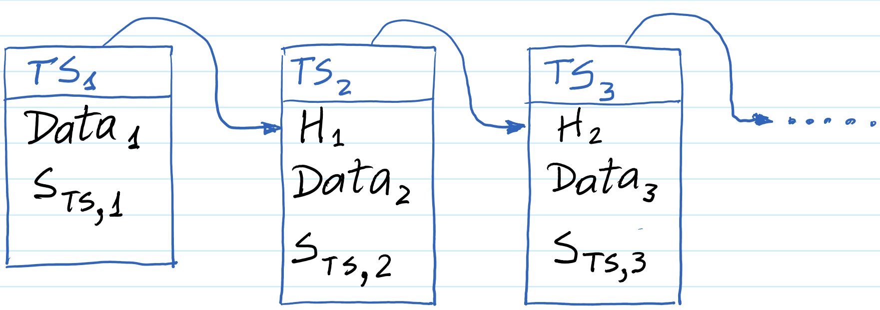
(PrK_{TTP}, PuK_{TTP})

Veri Sign

$k_{BA} = (t_A)^v \bmod p$

$k_{AB} = k = k_{BA}$

TSA fraud --> Prevention using Blockchain



TS_{n+1}

Data_{n+1}:

$H(TS_n) = H_n$

DT_{n+1}

$H(H_n \| DT_{n+1}) = h_{n+1}$ → A: $S_A = Sign(PrK_A, h_{n+1})$

$Vert(PuK_A, S_A, h_{n+1})$ ← PuK_A

$H(h_{n+1} \| S_A) = H_{n+1}$

S_A

$Sign(PrK_{TS}, H_{n+1}) = S_{TS}$

Business operational control system - BOCS

IoT → smart meters ; Blockchain → Smart contracts