CRYPTOGRAPHY LECTURE, 2022

Computer Science and Algorithmics, PWr

Mirosław Kutyłowski

# PKI- Public Key Infrastructure

## Problem

how do you know that public key $\mathrm{PK}$ corresponds to Alice?

If you do not know this then electronic signatures, authentication, ...

– everything is useless

# Idea 1: identity based cryptography

- public key is derived from uniques identifier
- secret key derived by trusted KGC (Key Generation Center)
- solutions based for encryption, signatures, using bilinear mappings $e$
  - $e: G_1 \times G_2 \to G_T$
  - $e(g^a, g^b) = e(g, g)^{a \cdot b}$, $e(g, g) \neq 1$
  - typically notation additive (like for elliptic curves)
  - note: in presence of $e: G_1 \times G_1 \to G_T$ DDH is easy: $A, B, C$ is a DH triple iff

$$e(A, B) = e(g, C)$$

(bilinear mappings not much used so far in practice, but easy construction of schemes)

# Boneh-Franklin ID based encryption

**Setup:** private key: random $s$ public key: $K_{\text{pub}} := s \cdot P$

($P$ is group generator)

**User keys:** $Q_{\text{ID}} := \text{Hash}(\text{ID})$ is public key for ID

secret key: $s \cdot Q_{\text{ID}}$

**Encryption of $m$:**

i. choose $r$ at random

ii. $g_{\text{ID}} := e(K_{\text{pub}}, Q_{\text{ID}})$

iii. ciphertext $c = (r \cdot P, m \oplus \text{Hash}(g_{\text{ID}}^r))$

# Boneh-Franklin ID based encryption

**Encryption of $m$:**

i. choose $r$ at random

ii. $g_{\mathrm{ID}} := e(K_{\mathrm{pub}}, Q_{\mathrm{ID}})$

iii. ciphertext $c = (r \cdot P, m \oplus \mathrm{Hash}(g_{\mathrm{ID}}^r))$

## Decryption of $c = (u, v)$:

$$m := v \oplus \mathrm{Hash}(e(u, s \cdot Q_{\mathrm{ID}}))$$

Why it works?

$$g_{\mathrm{ID}}^r = e(K_{\mathrm{pub}}, Q_{\mathrm{ID}})^r = e(s \cdot P, Q_{\mathrm{ID}})^r = e(P, Q_{\mathrm{ID}})^{r \cdot s}$$

$$e(u, s \cdot Q_{\mathrm{ID}}) = e(r \cdot P, s \cdot Q_{\mathrm{ID}}) = e(P, Q_{\mathrm{ID}})^{r \cdot s}$$

## **Certificate** (X.509 standard)

essentially: digitally signed sentence "$K$ is the public key of Alice"

important fields:

- key usage (admitted usage of $K$)
- algorithms
- owner (Alice)
- issuer Certificate Authority
- issued at...
- valid until ...
- certification policy link

Problem: how to verify? Public key of CA needed

# Certificate chain

- root CA
- subordinate CA's
- users (their certificates have key usage$\neq$signing certificates)

Certificate validation: go to the root, root self-certificate must be trusted manually

# Validity

what if a card with a secret key gets stolen?

$\Rightarrow$ the certificate has to be invalidated

## Solution

- CRL -Certificate Revocation List (contains only revoked certificates that has not expired yet)
- OCSP - Online Certificate Status Protocol (lookup in CRL)

Unsolved problem: time between revocation request and CRL publication

# Short time certificates

do not care about revocation – they expire soon anyway

used for certificates of Document Verifiers (border control, . . . ) interacting with biometric passports

## Mediated signatures

no need to verify validity of certificates:

example for RSA

- split the private key $d$ to $d = d_1 \cdot d_2 \bmod (p-1)(q-1)$
- $d_1$ stored locally on a card, $d_2$ on the server
- computing $a^d \bmod n$ :
  - i card computes $c_1 := a^{d_1}$ and send to the server
  - ii server checks policy, revocation list ...
  - iii ... if ok then returns $c_2 := c^{d_2}$

instantenous revocation, monitoring , ...

used in Estonia after disaster with personal ID cards

## Forging certificates

"certificates, or certified lies"

from history of attacks:

i. not an attack on the signature but via weak hash function

ii. finding collision it changes key and key usage (from user to CA certifying software)

iii. then issue a signature for Operating System update – a malicious one containing malware

## Basic list of root CA'a

for servers: check how many of them are from EU

for digital signatures: in EU in each country a list of CA for qualified issuers

bridge certificates

# Alternative approach: SPKI

public key is the ID

extension of Access Control List