# MTAT.07.017
# Applied Cryptography

Rakenduslik krüptograafia
Прикладная криптография

Juri Hudolejev
University of Tartu
Spring 2011

# Topics for This Week

Cryptographic key management

Public key certificate (as defined in X.509)

PKI: Public key infrastructure

# Unsecured Communication

`msg = “The Message”`

`→ → → → → → msg → → → → → →`

Problem: anyone can read this message

# Symmetric Encryption

```
enc = encrypt(msg, K)
```

→ → → → → → enc → → → → → →

```
msg = decrypt(enc, K)
```

Problem: how to exchange keys?

# Hybrid Encryption

```
enc1 = encrypt(msg, K)
enc2 = encrypt(K, K+)
 → → → → enc1, enc2  → → → →
          K = decrypt(enc2, K-)
         msg = decrypt(enc1, K)
```

Problem: how to get K+?

# Key Exchange

→ → → (requests K+) → → → →

← ← ← ← ← ← K+ ← ← ← ← ← ←

Computes enc1, enc2

→ → → → enc1, enc2 → → → →

Problem: man-in-the-middle attack

# Man-in-the-Middle Attack

Adam believes protocol works like this:

```
A → B : Message
B → A : Response
```

What is actually happening:

```
A → M : Message
M → B : AnotherMessage
B → M : Reponse
M → A : AnotherResponse
```

# Man-in-the-Middle Attack

```
A → M : request("PublicKey")
M → B : request("PublicKey")
B → M : K+(B)
M → A : K+(M)
```

Adam and Bob both believe that Adam has received Bob's public key

# Man-in-the-Middle Attack

```
A : enc1 = encrypt(msg, K)
A : enc2 = encrypt(K, K+(M))
A → M : enc1, enc2
M : K = decrypt(enc2, K-(M))
M : msg = decrypt(enc, K)
M : enc3 = encrypt(msg, K)
M : enc4 = encrypt(K, K+(B))
M → B : enc3, enc4
```

# Key Management

Most problematic part of cryptography?

Short-term keys

- Session keys – usually generated automatically

Long-term keys

- Signing and encryption – generated by user

# Key Management Problems

Key physical access control

Key backup/retrieval

- Secret sharing schemes

Key destruction

- Lifetime of public keys used to validate signatures

# Key Management Models

Offline (PGP, X.509)

Online (S/MIME, SIP, SSH, SSL/TLS)

Ad-hoc

# Offline Key Management

- Hierarchical
  - Relying on trusted authorities
  - Example: X.509
- Distributed
  - Relying on “web of trust”
  - Example: PGP

# Online Key Management

## On-demand distribution

- Public key is sent on request
- Example: SSL/TLS, S/MIME

## Key continuity

- Regular key exchange initiated
- Example: SSH

# Public Key Certificate

Binds public key with key owner's identity

Defines public key validity period

Issued and signed by trusted third party

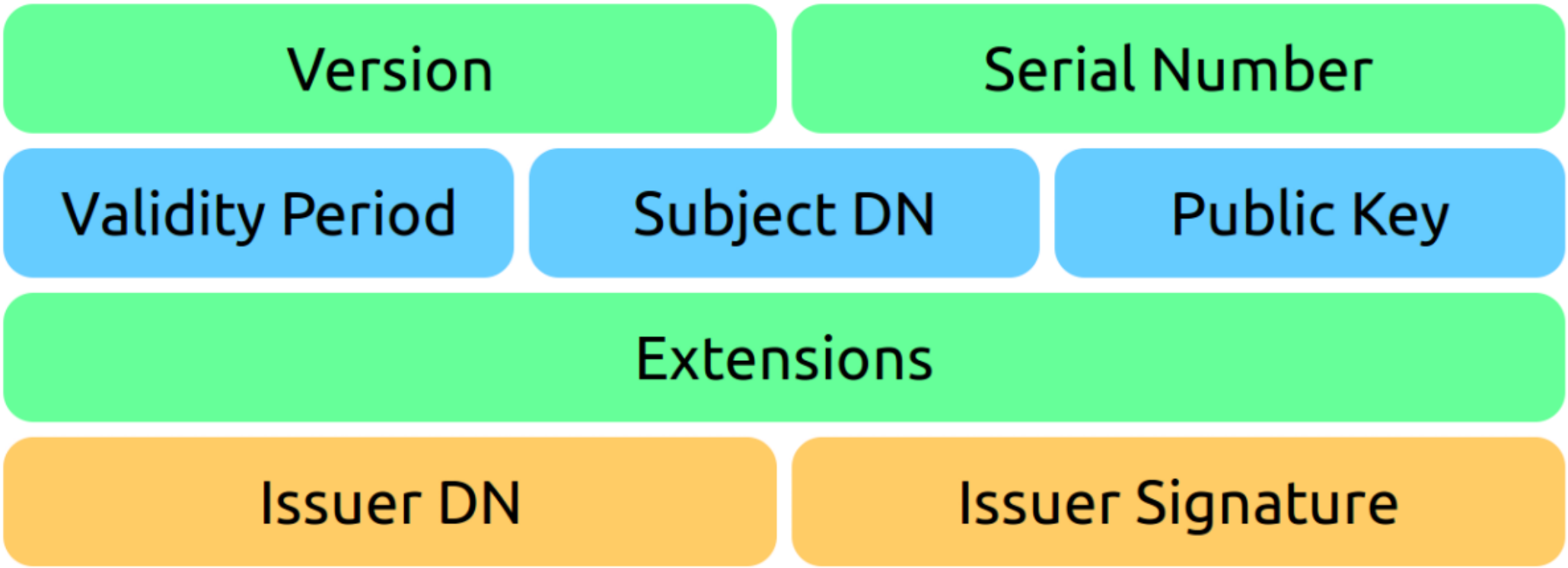
# Public Key Certificate

```
-- Key-to-identity binding
-- See RFC 2459 for details
TBSCertificate ::= SEQUENCE {
     -- Some fields skipped
     validity          Validity,
     subject           Name,
     subjectPKInfo     SubjectPublicKeyInfo
     -- Some fields skipped
}
```

# Public Key Certificate

```
-- TBSCertificate and authority signature
-- See RFC 2459 for details
-- http://tools.ietf.org/html/rfc2459
Certificate ::= SEQUENCE {
    tbsCertificate  TBSCertificate,
    signatureAlg    AlgorithmIdentifier,
    signatureValue  BIT STRING
}
```

# X.509 Certificate



# DN: Distinguished Name

Yet another notation for unique identifiers

Used in LDAP and related protocols

Example:

```
CN=John Doe, OU=Infernal IT, O=Evil Inc., C=US
```

See also: RFC 4514 (http://tools.ietf.org/html/rfc4514)

# Requesting a Certificate

```
subj = “CN=John Doe, OU=...”
{ K+, K- } = generateKeys()
tbsReq = { subj, K+ }
reqSig = sign(tbsReq, K-)
certReq = { tbsReq, reqSig }
```

Certification request **certReq** is sent to CA

# Generating a Certificate

```
result = verifySignature(certReq)
validity = { notBefore, notAfter }
tbsCert = { validity, subj, K+ }
certSig = sign(tbsCert, KCA-)
cert = { tbsCert, certSig }
```

Certificate **cert** is sent back to client

# Problems Obtaining a Certificate

How client locates CA?

How client gets CA's public key?

How client sends own public key to CA?

How CA verifies client's data?

How CA delivers certificate to client?

EUGridPMA: http://www.eugridpma.org/

# Questions?

# See You Next Week

No homework for this week (:

Next lab session:

Friday 2011-04-01 **08:30** EET @ Liivi 2 - 205