MTAT.07.017
Applied Cryptography

Public Key Infrastructure (PKI)
Public Key Certificates (X.509)

University of Tartu

Spring 2016
Key Management

- The hardest problem
- How to obtain the key of the other party?
  - Symmetric encryption?
    - Vulnerable to passive and active attacks
    - Confidential and authentic channel needed
  - Asymmetric encryption?
    - Vulnerable to active attacks
    - Authentic channel needed
- Trust models:
  - Trust on first use (e.g., SSH)
  - Decentralized model - web of trust (e.g., PGP)
  - Centralized model - Trusted third party (e.g., TLS)
TOFU: Trust On First Use

- Used by SSH (encrypted telnet)

- For the first time:

  $ ssh user@cs.ut.ee
  The authenticity of host 'cs.ut.ee (193.40.36.81)' can't be established.
  Are you sure you want to continue connecting (yes/no)? yes
  Warning: Permanently added 'cs.ut.ee,193.40.36.81' (RSA) to the list of known hosts.

  user@cs.ut.ee’s password:

  $ cat ~/.ssh/known_hosts
  cs.ut.ee,193.40.36.81 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAIQA2HotF0bR9U8MgTE67bGJr
  math.ut.ee,193.40.36.2 ssh-rsa AAAAB3NzaC1yc2EAAAABIwAAAIEAzPcVb60Q8QV0s3hdoFa0

- In the future:

  $ ssh user@cs.ut.ee
  user@cs.ut.ee’s password:
If the key has changed:

```bash
$ ssh user@cs.ut.ee
```

IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
Please contact your system administrator.

- Threat model?
- How to improve ssh client to be even more secure?
WOT: Web of trust
PKI: Public key infrastructure

Public key certificate binds a public key with an identity

- Main actors:
  - Subject (user, end-entity)
  - Trusted third party (certificate authority, issuer)
  - Relying party (verifier)

- The passport analogy
PKI Use Cases

- Server authentication
- Digital Signatures
- Client authentication

Server authentication

Digital Signatures

Client authentication
Certificate Authorities

Certificate Authority – **Trusted** Third Party
Where the trust comes from?

- Software vendors decided on your behalf (MS, Mozilla)

- Root CA – self-signed certificate (trust anchor)
- CA can delegate trust to subordinate/intermediate CAs
- CA can constrain issued CA and end-entity certificates
Estonian ID card case

- Subject – Estonian residents
- Registration Authority – Police and Border Guard Board
- Manufacturer – Trüb Baltic AS
  - In addition generates RSA key pair for you
- Certificate Authority – AS Sertifitseerimiskeskus
- Relying party – anyone (e.g., your bank)
How to become a CA?

- The objective: profit (Mark Shuttleworth, Thawte (VeriSign), $575 million, Ubuntu)
- Get your root CA trusted:
  - Compliance audit (WebTrust, ETSI TS)
    - Ernst & Young or KPMG (15k EUR/year)
  - Liability insurance (required by EU directive)
    - Insurance industry reluctant (3k EUR/year)
  - Use of Hardware Security Modules (HSM)

http://bugzilla.mozilla.org/show_bug.cgi?id=414520
Hardware Security Module (HSM)

- Physically protected private key storage
- Smart card (miniHSM)

- Certifications:
  - Common Criteria (EAL1 through EAL7)
  - FIPS 140-2:
    - Level 1 – no protection
    - Level 2 – tamper evident
    - Level 3 – tamper resistant
    - Level 4 – tamper reactive
X.509 certificate

Subject: “John Smith”

Issuer: Verisign, Inc. Root CA

Public key: BC F7 C6 74 F5 32 D0 34 5D A9 ...

Serial #: 11:21:56:2D:2E

Valid: from 2015.01.01 15:00 to 2017.01.01 15:00

Other Data: ...

Signed: CA’s Signature
X.509 certificate

Certificate ::= SEQUENCE {
    tbsCertificate TBSCertificate,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue BIT STRING }

TBSCertificate ::= SEQUENCE {
    version [0] EXPLICIT Version DEFAULT v1(0),
    serialNumber INTEGER,
    signature AlgorithmIdentifier,
    issuer Name,
    validity Validity,
    subject Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    extensions [3] EXPLICIT Extensions OPTIONAL -- v3(2) only }

Validity ::= SEQUENCE {
    notBefore UTCTime,
    notAfter UTCTime }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical BOOLEAN DEFAULT FALSE,
    extnValue OCTET STRING }

X.509 certificate

- tbsCertificate – DER structure to be signed by CA
- version – X.509v1 or X.509v3 used
  - X.509 v3 introduces certificate extensions
- serialNumber – unique for every certificate issued by CA
- signature – AlgorithmIdentifier from outer Certificate sequence
- issuer – identity of CA who signed the certificate
- validity – period in which certificate should be assumed valid
- subject – identity of a subject whose public key in certificate
- subjectPublicKeyInfo – subject’s public key
- extensions – optional extensions providing more information
Distinguished Name (DN) in X.509 Certificate

"The issuer and subject field MUST contain a non-empty distinguished name (DN). The issuer field is defined as the X.501 type Name. Name is defined by the following ASN.1 structures:"

Name ::= RDNSequence
RDNSequence ::= SEQUENCE OF RelativeDistinguishedName
RelativeDistinguishedName ::= SET OF AttributeTypeAndValue

AttributeTypeAndValue ::= SEQUENCE {
  type OBJECT IDENTIFIER,
  value ANY -- DEFINED BY type }

• Yet another notation for unique identifiers
• Used in LDAP and related protocols
• Example:

CN=John Doe, OU=Helpdesk, O=Burgers Inc., C=US
Distinguished Name (DN) in X.509 Certificate

2 74: SEQUENCE {
4 11:   SET {
6 9:     SEQUENCE {
8 3:      OBJECT IDENTIFIER countryName (2 5 4 6)
13 2:      PrintableString 'US'
9:     }
17 18:   SET {
19 16:     SEQUENCE {
21 3:      OBJECT IDENTIFIER organizationName (2 5 4 10)
26 9:      UTF8String 'Burgers Inc.'
24:     }
37 20:   SET {
39 18:     SEQUENCE {
41 3:      OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
46 11:      UTF8String 'Helpdesk'
44:     }
59 17:   SET {
61 15:     SEQUENCE {
63 3:      OBJECT IDENTIFIER commonName (2 5 4 3)
68 8:      UTF8String 'John Doe'
66:     }
:
(2 5 4 4) : surname (SN)
(2 5 4 42) : givenName (GN)
(2 5 4 5) : serialNumber
(2 5 4 7) : localityName (L)
(2 5 4 8) : stateOrProvinceName (ST)
(1 2 840 113549 1 9 1) : emailAddress
Certificate Extensions (X.509v3 only)

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension
Extension ::= SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical BOOLEAN DEFAULT FALSE,
    extnValue OCTET STRING }

• Every extension has its OID
• RFC 5280 defines several standard extensions
• Certificate (path) validation algorithm must handle those

“Each extension in a certificate is designated as either critical or non-critical. A certificate-using system MUST reject the certificate if it encounters a critical extension it does not recognize or a critical extension that contains information that it cannot process. A non-critical extension MAY be ignored if it is not recognized, but MUST be processed if it is recognized.”

Certificate Extensions (X.509v3 only)

- **Key Usage**
  - Defines the purpose of the key contained in the certificate

  KeyUsage ::= BIT STRING {
    digitalSignature (0),
    nonRepudiation (1), -- contentCommitment
    keyEncipherment (2),
    dataEncipherment (3),
    keyAgreement (4),
    keyCertSign (5),
    cRLSign (6),
    encipherOnly (7),
    decipherOnly (8) }

  - key may be used for all purposes if extension absent

- **Extended Key Usage**
  - Indicates more specific purpose of the key
  - Usage must be consistent with Key Usage extension

  ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId

  KeyPurposeId ::= OBJECT IDENTIFIER

  id-kp-serverAuth OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 3 1 }
  id-kp-clientAuth OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 3 2 }
  id-kp-codeSigning OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 3 3 }
  id-kp-emailProtection OBJECT IDENTIFIER ::= { 1 3 6 1 5 5 7 3 4 }
Certificate Extensions (X.509v3 only)

- **Basic Constraints**
  - Identifies whether subject is CA – may sign certificates
  - For CA identifies maximum subordinate CAs it may have

  \[
  \text{id-ce-basicConstraints \ object identifier ::= \{ id-ce 19 \}} \\
  \text{BasicConstraints ::= sequence \{ \\
  \text{cA boolean default false,} \\
  \text{pathLenConstraint INTEGER (0..MAX) optional} \}} \\
  \]

  - If not present in v3 certificate then not a CA
  - If cA boolean is TRUE then keyUsage must be absent or must have keyCertSign bit set

- **Name Constraints**
  - In CA certificate indicates a name space constraint in all subsequent certificates
  - ".example.com" matches both host.example.com and my.host.example.com
  - Must be marked critical
  - Pain to process – not used in practice
Certificate Extensions (X.509v3 only)

- Certificate Policies
  - Contains policy information terms under which the certificate has been issued and the purposes for which the certificate may be used
    - URL to certificate practice statement (CPS)
    - OID of the CPS document version
    - Explicit notice text

- Policy Mappings
  - Maps equivalent policy OIDs
  - Pain to process – not used in practice

- Policy Constraints
  - Constrain policies that may be included in CA issued certificates
  - Pain to process – not used in practice
Certificate Extensions (X.509v3 only)

- **Subject Alternative Name**
  - Identifies subject alternatively to the subject name
  - Include email, DNS name, IP addresses, URI, etc.
  - New standards promote use of this extension

- **Authority Key Identifier and Subject Key Identifier**
  - Uniquely identifies subject and issuer
  - KeyIdentifier, GeneralNames, CertificateSerialNumber

- **CRL Distribution Points**
  - Includes URI where CRL is available (HTTP or LDAP)

- **Authority Information Access**
  - Indicates how to access information about CA services

- **Subject Information Access**
  - Indicates how to access information about subject

Extensions may include a picture of the subject, attributes, roles etc.
Use in HTTPS (TLS)

- TLS server certificates – the most popular use case
- What does the browser verify before the connection is considered secure?
  - Certificate signed by a trusted CA
  - Host name in the address bar matches the CN in the certificate
  - Validity date, extensions, etc.
## Server Certificate

### General

**This certificate has been verified for the following uses:**

**SSL Server Certificate**

<table>
<thead>
<tr>
<th><strong>Issued To</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name (CN)</td>
<td>auth.ut.ee</td>
</tr>
<tr>
<td>Organization (O)</td>
<td>&lt;Not Part Of Certificate&gt;</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
<td>Domain Control Validated</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Issued By</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name (CN)</td>
<td>TERENA SSL CA</td>
</tr>
<tr>
<td>Organization (O)</td>
<td>TERENA</td>
</tr>
<tr>
<td>Organizational Unit (OU)</td>
<td>&lt;Not Part Of Certificate&gt;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Period of Validity</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Begins On</td>
<td>10/10/2013</td>
</tr>
<tr>
<td>Expires On</td>
<td>10/10/2016</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Fingerprints</strong></th>
<th><strong>Value</strong></th>
</tr>
</thead>
</table>
## Server Certificate

### Certificate Hierarchy
- UTN-USERFirst-Hardware
  - TERENA SSL CA
    - auth.ut.ee

### Certificate Fields
- auth.ut.ee
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
  - Issuer
    - Validity
      - Not Before
      - Not After
    - Subject
    - Subject Public Key Info

### Field Value
- CN = TERENA SSL CA
- O = TERENA
- C = NL
Server Certificate

$ openssl x509 -in auth.ut.ee.pem -text
Version: 3 (0x2)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=NL, O=TERENA, CN=TERENA SSL CA
Validity
   Not Before: Oct 10 00:00:00 2013 GMT
Subject: OU=Domain Control Validated, CN=auth.ut.ee
Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   Public-Key: (2048 bit)
       Modulus:
           Exponent: 65537 (0x10001)
X509v3 extensions:
   X509v3 Key Usage: critical
       Digital Signature, Key Encipherment
   X509v3 Basic Constraints: critical
       CA:FALSE
   X509v3 Extended Key Usage:
       TLS Web Server Authentication, TLS Web Client Authentication
X509v3 Certificate Policies:
   Policy: 1.3.6.1.4.1.6449.1.2.2.29
   Policy: 2.23.140.1.2.1
X509v3 CRL Distribution Points:
   Full Name: URI:http://crl.tcs.terena.org/TERENASSLCA.crl
   Authority Information Access:
       CA Issuers - URI:http://crt.tcs.terena.org/TERENASSLCA.crt
       OCSP - URI:http://ocsp.tcs.terena.org
X509v3 Subject Alternative Name:
   DNS:auth.ut.ee, DNS:parool.ut.ee, DNS:passwd.ut.ee
Signature Algorithm: sha1WithRSAEncryption
Use in HTTPS (TLS)

- Requesting party (website owner):
  - Key generation
  - Certificate request submission
- Certificate Authority (CA):
  - Distribution of root certificates
  - Requesting party identity verification
  - Certificate signing (issuance)
- Relying party (website visitor):
  - Certificate verification
Certificate Authorities

You have certificates on file that identify these certificate authorities:

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>AffirmTrust Networking</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>America Online Inc.</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>America Online Root Certification Authority</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>America Online Root Certification Authority</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>AS Sertifitseerimiskeskus</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Juur-SK</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>KLASS3-SK</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>KLASS3-SK 2010</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesio...</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesio...</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesio...</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Baltimore</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Baltimore CyberTrust Root</td>
<td>Builtin Object Token</td>
</tr>
</tbody>
</table>

[View...  Edit Trust...  Import...  Export...  Delete or Distrust...  OK]
Identity Verification

- Domain Validation (DV): $20/year $0/year
  - Checks whether you control the domain
    - [pilet.ee](https://www.pilet.ee/cgi-bin/splususer/splus)

- Organization Validation (OV): $200/year
  - Checks whether you operate the organization
    - [eesti.ee](https://www.eesti.ee/est)

- Extended Validation (EV): $500/year
  - Checks whether you operate the organization 2x
    - [Swedbank AS (EE)](https://www.swedbank.ee/private)
Domain Validated vs Organization Validated

This certificate has been verified for the following uses:
SSL Server Certificate

Issued To
Common Name (CN)       * .pilet.ee
Organization (O)        * .pilet.ee
Organizational Unit (OU) Domain Control Validated

Issued By
Common Name (CN)        Go Daddy Secure Certification Authority
Organization (O)        GoDaddy.com, Inc.
Organizational Unit (OU) http://certificates.godaddy.com/repository

Validity
Issued On               01/20/2012
Expires On              01/20/2014

Fingerprints

This certificate has been verified for the following uses:
SSL Server Certificate

Issued To
Common Name (CN)       * .eesti.ee
Organization (O)        Estonian Informatics Centre
Organizational Unit (OU) <Not Part Of Certificate>

Issued By
Common Name (CN)        Thawte Premium Server CA
Organization (O)        Thawte Consulting cc
Organizational Unit (OU) Certification Services Division

Validity
Issued On               02/08/2010
Expires On              03/10/2012

Fingerprints
Certificate Signing Request (CSR)

$ openssl genrsa -out priv.pem 1024
$ openssl req -new -key priv.pem -out sso.ut.csr

You are about to be asked to enter information that will be incorporated into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN. There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter ’.’, the field will be left blank.

Country Name (2 letter code) [AU]: EE
State or Province Name (full name) [Some-State]:.
Locality Name (eg, city) []:.
Organization Name (eg, company) [Internet Widgits Pty Ltd]:.
Organizational Unit Name (eg, section) []:.
Common Name (e.g. server FQDN or YOUR name) []: sso.ut.ee
Email Address []:.

Please enter the following 'extra' attributes to be sent with your certificate request
A challenge password []: asdadasd
An optional company name []:.

$ cat req.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIBZzCB0QIBADAoMQswCQYDVQQGEwJFRTEZMBcGA1UEAwwQd3d3LmFwcGNyeXBO
I2Vmj+8IpKax5en8M29CGwuL4e10ua6LejVE
-----END CERTIFICATE REQUEST-----
Certificate Signing Request (CSR)

"A certification request consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. Certification requests are sent to a certification authority, which transforms the request into an X.509 public-key certificate."

\[
\text{CertificationRequest ::= SEQUENCE } \\
\text{ certificationRequestInfo CertificationRequestInfo,} \\
\text{ signatureAlgorithm AlgorithmIdentifier,} \\
\text{ signature BIT STRING } \\
\]

\[
\text{CertificationRequestInfo ::= SEQUENCE } \\
\text{ version INTEGER v1(0),} \\
\text{ subject Name,} \\
\text{ subjectPKInfo SubjectPublicKeyInfo,} \\
\text{ attributes [0] IMPLICIT Attributes } \\
\]

PKCS#10: https://tools.ietf.org/html/rfc2986

- Why does the subject has to prove the possession of the corresponding private key?
$ openssl req -in sso.ut.ee.csr -text
Certificate Request:
  Data:
    Version: 0 (0x0)
    Subject: C=EE, CN=sso.ut.ee
    Subject Public Key Info:
      Public Key Algorithm: rsaEncryption
        Public-Key: (1024 bit)
          Modulus:
          Exponent: 65537 (0x10001)
      Attributes:
        challengePassword :unable to print attribute
    Signature Algorithm: sha1WithRSAEncryption
      35:44
Certificate Signing Request (CSR)

```bash
$ openssl req -in sso.ut.csr -outform der -out sso.ut.csr.der
$ dumpasn1 sso.ut.csr.der
```
Enrollment: Step 1

Billing Contact Information

The billing contact will receive the receipt for the purchase when a credit card is used.

First Name * Foo
Last Name * Bar
Title
E-mail * [redacted]t.ee
Phone Number * +3722934312
Company Name
Address * [redacted]
City * Tartu
State/Province * Armed Forces Africa, Eu
Postal Code * EE1000
Country * ESTONIA

Back  Continue
Enrollment: Step 2
RapidSSL Enrollment

Verify Server URL

The CSR you generated is designed to work with the following URL:

https://sso.ut.ee

If this is not the correct URL (computed from the Common name in the CSR), or if any of the CSR Information below is incorrect, then please generate a new CSR and click the Replace CSR button.

CSR Information

**Common Name:** sso.ut.ee  
**Organization:**  
**Org. Unit:**  
**Locality:**  
**State:**  
**Country:** EE

**Note:** The value for the Common Name must exactly match the name of the server you plan to secure.

[Replace CSR]

[Continue]
Approval of Your Certificate Request

The RapidSSL.com RapidSSL® service relies upon the Subscriber or the Subscriber's authorized administrator to approve all certificate requests for all hosts in the domain. It is important that you select the correct authorized administrator below. By selecting an authorized administrator, you warrant that the individual is authorized to approve the request. Your request for a RapidSSL® server certificate will not be processed beyond this point if you select an incorrect email address.

Registered Domain Contacts

Unfortunately, we were not able to obtain the domain contacts from the registrar. It is possible that the registration information has not yet been published for this domain, or that the registrar is not currently available to RapidSSL.

Alternate Approval Email Addresses

The following approval email addresses can be used. You must make sure that the email account has been set up and is available before you submit this order, or the approval email will not be delivered.

**Level 2 Domain Addresses**
- admin@ut.ee
- administrator@ut.ee
- hostmaster@ut.ee
- webmaster@ut.ee
- postmaster@ut.ee

**Level 3 Domain Addresses**
- admin@sso.ut.ee
- administrator@sso.ut.ee
- hostmaster@sso.ut.ee
- webmaster@sso.ut.ee
- postmaster@sso.ut.ee
What if CA goes bad?

- Comodo hack (2011 March)
  - Google, Skype, Mozilla - MITM in Iran
  - Fraudulent certificates revoked

- DigiNotar hack (2011 September)
  - Google wildcard - MITM in Iran
  - Root certificates revoked
  - Went bankrupt

- TrustWave spycerts (2012 February)
  - Subordinate root certificate for client’s DLP
  - Subordinate root certificate revoked
  - Mozilla CA policy changed

- Turktrus trust mistake (2013 January)
  - Left Basic Constraints cA:TRUE for 2 end-entity certificates
  - Certificates in question revoked
  - Punished by Mozilla/MS/Google by not including new CA

Certificate Authorities too big to fail?
Task 1: Self-signed CA Root Certificate

Implement utility that creates self-signed CA root certificate.

$ ./selfsigned.py
usage: selfsigned.py private_key_file output_cert_file

$ openssl genrsa -out priv.pem 1024
$ ./selfsigned.py priv.pem rootCA.pem
$ openssl verify -check_ss_sig -CAfile rootCA.pem rootCA.pem: OK

- Must support PEM/DER inputs, PEM output
- Signature has to verify successfully
- Use sha1WithRSAEncryption (1.2.840.113549.1.1.5)
- Put subject name that identifies you
- Put whatever serial you want
- Critical extension: basic constraints CA:TRUE
- Critical extension: key usage: keyCertSign, cRLSign
- Certificate must be valid at least ± 3 months
- Use your own DER encoder and pyasn1
Task 1: Self-signed CA Root Certificate

$ openssl x509 -in rootCA.pem -text
Certificate:
Data:
    Version: 3 (0x2)
    Serial Number: 1 (0x1)
Signature Algorithm: sha1WithRSAEncryption
Issuer: C=EE, O=University of Tartu, OU=IT dep, CN=Arnis Root CA
Validity
    Not After : Aug 28 12:59:31 2016 GMT
Subject: C=EE, O=University of Tartu, OU=IT dep, CN=Arnis Root CA
Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (1024 bit)
    Modulus:
    Exponent: 65537 (0x10001)
X509v3 extensions:
    X509v3 Basic Constraints: critical
    CA:TRUE
    X509v3 Key Usage: critical
    Certificate Sign, CRL Sign
Signature Algorithm: sha1WithRSAEncryption
f4:c6
Task 2: Certificate Signer (Bonus +4 points)

Implement utility that issues TLS server certificate based on certificate signing request.

$ ./signcert.py
usage: signcert.py private_key_file CA_cert_file csr_file output_cert_file

$ ./signcert.py priv.pem rootCA.pem req.csr issued.pem
[+] Issuing certificate for "www.appcrypto.ee"
$ openssl verify -CAfile rootCA.pem -purpose sslserver issued.pem
issued.pem: OK
$ openssl verify -CAfile rootCA.pem -purpose smimesign issued.pem
issued.pem: C = EE, CN = www.appcrypto.ee
error 26 at 0 depth lookup: unsupported certificate purpose
OK

- Fetch subject’s CN from CSR (other fields may be arbitrary)
- Fetch subject’s public key from CSR (subjectPublicKeyInfo)
- Fetch issuer’s distinguished name from CA certificate
- Sign subject’s certificate using CA private key
- Critical extensions:
  - basic constraints CA:FALSE
  - key usage: digitalSignature
  - extended key usage: id-kp-serverAuth
- Use your own DER encoder and pyasn1
Task 2: Certificate Signer (Bonus +4 points)

$ openssl x509 -in issued.pem -text
Certificate:
Data:
  Version: 3 (0x2)
  Serial Number: 1 (0x1)
Signature Algorithm: sha1WithRSAEncryption
  Issuer: C=EE, O=University of Tartu, OU=IT dep, CN=Arnis Root CA
Validity
  Not After : Aug 28 12:59:31 2016 GMT
Subject: C=EE, CN=www.appcrypto.ee
Subject Public Key Info:
  Public Key Algorithm: rsaEncryption
  Public-Key: (1024 bit)
    Modulus:
    Exponent: 65537 (0x10001)
X509v3 extensions:
  X509v3 Basic Constraints: critical
    CA:FALSE
  X509v3 Key Usage: critical
    Digital Signature
  X509v3 Extended Key Usage: critical
    TLS Web Server Authentication
Signature Algorithm: sha1WithRSAEncryption
  34:ca
Hints

- pyasn1 will fail to decode CSR if it contains no attributes (challenge password) since it expects implicit tagging:
  - Make sure your CSR contains challenge password

- pyasn1 can easily encode decoded substructures:
  ```python
  encoder.encode(decoder.decode(der)[0][0][5])
  ```

- You might want to implement `asn1_bitstring_der()` which takes byte string (instead of bitstring) as input

- Read ASN.1 definitions or dumpasn1 example certificates to find out DER encoding of certificate and its extensions
  ```bash
  openssl x509 -inform pem -in cert.pem -outform der -out cert.der
  ```

- For debugging use two windows to compare your dumpasn1 output with reference output
Questions

• What does PKI and X.509 certificates solve?
• Which are two most important fields in X.509 certificate?
• Who defines trusted CAs for digital signature certificates?
• What is Hardware Security Module useful for?
• What browser checks in a certificate received from the server?
• Who defines trusted CAs for web server certificates?
• How are DV certificates different from OV certificates?
• How does CA verify whether the buyer owns the domain?
Note

No lecture and practice session next weeks (March 25 and April 1)!

Deadline for the homework is in three weeks from now (April 8 12:15).