MTAT.07.017
Applied Cryptography

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

University of Tartu

Spring 2018
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:

```bash
$ 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:
```

```bash
$ cat ~/.ssh/known_hosts
cs.ut.ee,193.40.36.81 ssh-rsa AAAAB3NzaC1yc2EAAAABIBIwAAAQEA2HotF0bR9U8MgTE67bGJr
math.ut.ee,193.40.36.2 ssh-rsa AAAAB3NzaC1yc2EAAAABIBIwAAAIEAzPcVb6OQ8QV0s3hdoFa0
```

- In the future:

```bash
$ ssh user@cs.ut.ee
user@cs.ut.ee’s password:
```
TOFU: Trust On First Use

• If the key has changed:

```
$ ssh user@cs.ut.ee
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
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 (subscriber, user, end-entity)
  - Trusted third party (certificate authority, issuer)
  - Relying party (verifier)

- The passport analogy
PKI Use Cases

Client authentication

- Digital signatures (eIDAS)
- Code signing
- E-mail (S/MIME)

Server authentication
Certificate Authorities

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

- Software vendors decided on your behalf (MS, Mozilla)
- EU Regulation 910/2014 (eIDAS)

- 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
- Certificate Authority – SK ID Solutions AS
- Registration Authority – Police and Border Guard Board
- Manufacturer – Trub Baltic AS (Gemalto company)
  - In addition generates RSA key pair for you
- 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 eIDAS)
    - 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 2019.01.01 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'
          :
          :
17 18:     SET {
19 16:       SEQUENCE {
21  3:         OBJECT IDENTIFIER organizationName (2 5 4 10)
26  9:           UTF8String 'Burgers Inc.'
             :
             :
37 20:     SET {
39 18:       SEQUENCE {
41  3:         OBJECT IDENTIFIER organizationalUnitName (2 5 4 11)
46 11:           UTF8String 'Helpdesk'
             :
             :
59 17:     SET {
61 15:       SEQUENCE {
63  3:         OBJECT IDENTIFIER commonName (2 5 4 3)
68  8:           UTF8String 'John Doe'
             :
             :
(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**
  - Limits the purpose of the key contained in the certificate

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

  - 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

  \[
  \text{ExtKeyUsageSyntax ::= SEQUENCE SIZE (1..MAX) OF KeyPurposeId} \\
  \text{KeyPurposeId ::= OBJECT IDENTIFIER} \\
  \text{id-kp-serverAuth \quad OBJECT IDENTIFIER ::= \{ 1 3 6 1 5 5 7 3 1 \}} \\
  \text{id-kp-clientAuth \quad OBJECT IDENTIFIER ::= \{ 1 3 6 1 5 5 7 3 2 \}} \\
  \text{id-kp-codeSigning \quad OBJECT IDENTIFIER ::= \{ 1 3 6 1 5 5 7 3 3 \}} \\
  \text{id-kp-emailProtection \quad OBJECT IDENTIFIER ::= \{ 1 3 6 1 5 5 7 3 4 \}}
  \]
Certificate Extensions (X.509v3 only)

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

```plaintext
id-ce-basicConstraints OBJECT IDENTIFIER ::= { id-ce 19 }
BasicConstraints ::= SEQUENCE {
  cA BOOLEAN DEFAULT FALSE,
  pathLenConstraint INTEGER (0..MAX) OPTIONAL }
```

- 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
CA documentation

- Certificate Policy (CP)
- Certification Practice Statement (CPS)

“A CP may be used by a relying party to help in deciding whether a certificate, and the binding therein, are sufficiently trustworthy and otherwise appropriate for a particular application.”

“A more detailed description of the practices followed by a CA in issuing and otherwise managing certificates may be contained in a certification practice statement (CPS) published by or referenced by the CA.”

“X.509 certificate may contain a field declaring that one or more specific certificate policies apply to that certificate.”

https://www.ietf.org/rfc/rfc3647.txt
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

This certificate has been verified for the following uses:

- SSL Client Certificate
- SSL Server Certificate

Issued To
- Common Name (CN): auth.ut.ee
- Organization (O): Tartu Ülikool
- Organizational Unit (OU): <Not Part Of Certificate>

Issued By
- Common Name (CN): TERENA SSL CA 3
- Organization (O): TERENA
- Organizational Unit (OU): <Not Part Of Certificate>

Period of Validity
- Begins On: September 27, 2016
- Expires On: October 2, 2019

Fingerprints
Server Certificate

Certificate Hierarchy
- DigiCert Assured ID Root CA
  - TERENA SSL CA 3
    - auth.ut.ee

Certificate Fields
- auth.ut.ee
  - Certificate
    - Version
    - Serial Number
    - Certificate Signature Algorithm
  - Issuer
    - Validity
      - Not Before
      - Not After

Field Value
- CN = TERENA SSL CA 3
- O = TERENA
- L = Amsterdam
- ST = Noord-Holland
- C = NL

Export...
Server Certificate

$ openssl x509 -in authutee.crt -text
Version: 3 (0x2)
Signature Algorithm: sha256WithRSAEncryption
Issuer: C=NL, ST=Noord-Holland, L=Amsterdam, O=TERENA, CN=TERENA SSL CA 3
Validity
   Not Before: Sep 27 00:00:00 2016 GMT
   Not After : Oct 2 12:00:00 2019 GMT
Subject: C=EE, ST=Tartumaa, L=Tartu, O=Tartu ulikool, CN=auth.ut.ee
Subject Public Key Info:
   Public Key Algorithm: rsaEncryption
   Public-Key: (2048 bit)
      Modulus:
      Exponent: 65537 (0x10001)
X509v3 extensions:
   X509v3 Basic Constraints: critical
      CA:FALSE
   X509v3 Key Usage: critical
      Digital Signature, Key Encipherment
   X509v3 Extended Key Usage:
      TLS Web Server Authentication, TLS Web Client Authentication
X509v3 Subject Alternative Name:
   DNS:auth.ut.ee, DNS:parool.ut.ee, DNS:passwd.ut.ee
X509v3 CRL Distribution Points:
   Full Name: URI:http://crl3.digicert.com/TERENASSLCA3.crl
Signature Algorithm: sha256WithRSAEncryption
Use in HTTPS (TLS)

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

<table>
<thead>
<tr>
<th>Certificate Name</th>
<th>Security Device</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon</td>
<td></td>
</tr>
<tr>
<td>Amazon Root CA 1</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Amazon Root CA 2</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Amazon Root CA 3</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Amazon Root CA 4</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Amazon</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>AS Sertifitseerimiskeskus</td>
<td></td>
</tr>
<tr>
<td>EE Certification Centre Root CA</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>KLAS53-SK 2016</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>KLAS53-SK 2010</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>KLAS53-SK 2010</td>
<td>Software Security Device</td>
</tr>
<tr>
<td>Atos</td>
<td></td>
</tr>
<tr>
<td>Atos TrustedRoot 2011</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesional CIF A62634068</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Autoridad de Certificacion Firmaprofesional CIF A62634068</td>
<td>Builtin Object Token</td>
</tr>
<tr>
<td>Baltimore</td>
<td></td>
</tr>
</tbody>
</table>

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

- Domain Validation (DV): $20/year  $0/year
  - Checks whether you control the domain

  [Image: https://www.pilet.ee/cgi-bin/splususer/splususer.cgi]

- Organization Validation (OV): $200/year
  - Checks whether you operate the organization

  [Image: https://www.eesti.ee/et/index.html]

- Extended Validation (EV): $500/year
  - Checks whether you operate the organization 2x

  [Image: Swedbank AS (EE) https://www.swedbank.ee/private]
### Domain Validated vs Organization Validated

<table>
<thead>
<tr>
<th>General</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>This certificate has been verified for the following uses:</strong></td>
<td><strong>This certificate has been verified for the following uses:</strong></td>
</tr>
<tr>
<td>SSL Client Certificate</td>
<td>SSL Client Certificate</td>
</tr>
<tr>
<td>SSL Server Certificate</td>
<td>SSL Server Certificate</td>
</tr>
</tbody>
</table>

**Issued To**
- **Common Name (CN):** pilet.ee
- **Organization (O):** <Not Part Of Certificate>
- **Organizational Unit (OU):** Domain Control Validated
- **Serial Number:** 40:D3:7F:5E:19:23:E3:CF

**Issued By**
- **Common Name (CN):** Go Daddy Secure Certificate Authority - G2
- **Organization (O):** GoDaddy.com, Inc.
- **Organizational Unit (OU):** http://certs.godaddy.com/repository/

**Period of Validity**
- **Begins On:** January 19, 2018
- **Expires On:** January 20, 2019

**Fingerprints**

<table>
<thead>
<tr>
<th>General</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Issued To</strong></td>
<td><strong>Issued By</strong></td>
</tr>
</tbody>
</table>
| **Common Name (CN):** pile.ee | **Common Name (CN):** eesti.ee
- **Organization (O):** Estonian Information System Authority |
- **Organizational Unit (OU):** <Not Part Of Certificate>

**Issued By**
- **Common Name (CN):** DigiCert SHA2 High Assurance Server CA
- **Organization (O):** DigiCert Inc
- **Organizational Unit (OU):** www.digicert.com

**Period of Validity**
- **Begins On:** June 29, 2016
- **Expires On:** August 29, 2019

**Fingerprints**
Certificate Signing Request (CSR)

$ openssl genrsa -out priv.pem 1024
$ openssl req -new -key priv.pem -out auth.ut.ee.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) []:
auth.ut.ee
Email Address [] :

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

$ cat req.csr
-----BEGIN CERTIFICATE REQUEST-----
MIIBZzCB0QIBADAoMQswCQYDVQQGEwJFRTETZMBcGA1UEAwQd3d3LmFwcGNyeXB0 I2Vmz+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."

\[
\text{CertificationRequest ::= SEQUENCE } \{ \\
\quad \text{certificationRequestInfo CertificationRequestInfo,} \\
\quad \text{signatureAlgorithm AlgorithmIdentifier,} \\
\quad \text{signature BIT STRING} \\
\} \\
\text{CertificationRequestInfo ::= SEQUENCE } \{ \\
\quad \text{version INTEGER v1(0),} \\
\quad \text{subject Name,} \\
\quad \text{subjectPKInfo SubjectPublicKeyInfo,} \\
\quad \text{attributes [0] IMPLICIT Attributes} \\
\} \\
\]

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

- Why does the subject have to prove the possession of the corresponding private key?
Certificate Signing Request (CSR)

$ openssl req -in auth.ut.ee.csr -text

Certificate Request:

Data:

  Version: 0 (0x0)
  Subject: C=EE, CN=auth.ut.ee
  Subject Public Key Info:
    Public Key Algorithm: rsaEncryption
    Public-Key: (1024 bit)
    Modulus:
      2f:63:bc:8b:3d:bc:29:e7:3d
    Exponent: 65537 (0x10001)

Attributes:

  challengePassword :unable to print attribute

Signature Algorithm: sha256WithRSAEncryption

  6c:f7:be:7f:02:42:8f:4b:53:0d:52:4e:9f:e8:5c:dc:b0:7d:
  00:cd
Certificate Signing Request (CSR)

```plaintext
$ openssl req -in auth.ut.ee.csr -outform der -out auth.ut.ee.csr.der
$ dumpasn1 auth.ut.ee.csr.der

0 376: SEQUENCE {
4 226:    SEQUENCE {
7  1:      INTEGER 0
10  34:    SEQUENCE {
12  11:      SET {
14  9:        SEQUENCE {
16  3:          OBJECT IDENTIFIER countryName (2 5 4 6)
21  2:            PrintableString 'EE'

25  19:        SET {
27  17:          SEQUENCE {
29  3:            OBJECT IDENTIFIER commonName (2 5 4 3)
34  10:              UTF8String 'auth.ut.ee'

46 159:    SEQUENCE {
49  13:      SEQUENCE {
51  9:        OBJECT IDENTIFIER rsaEncryption (1 2 840 113549 1 1 1)
62  0:          NULL

64 141:      BIT STRING, encapsulates {
68 137:        SEQUENCE {
71 129:          INTEGER

203 3:            INTEGER 65537

208 23:          [0] {
210 21:            SEQUENCE {
212  9:              OBJECT IDENTIFIER challengePassword (1 2 840 113549 1 9 7)
223  8:                SET {
225  6:                  UTF8String 'asdasd'

233 13:            SEQUENCE {
235  9:              OBJECT IDENTIFIER sha256WithRSAEncryption (1 2 840 113549 1 1 11)
246  0:                NULL

248 129:            BIT STRING

34 / 44
```
Enrollment: Step 1

Symantec Corporation (US)  https://website-security.geotrust.com/5938?toc=w88710382

Note: * = required.
Business Email*
Country *

Do you agree to receive communications from DigiCert about available offers and services?
☑ Yes please, I'd like to hear about offers and services.

First Name*
Last Name*
Company*
Province
Phone Number*
What are you primarily interested in?*
Securing transactions & protecting sensitive info online
How many domains are you looking to secure?*
1-10
Do you have a shopping cart on your website?*
No
What is your timeframe for implementation?*
Immediate
Need a quote?*
No
Comments

By clicking Continue you agree to DigiCert, Inc. or its affiliates processing your data in accordance with DigiCert's Privacy Policy.
Enrollment: Step 2

GeoTrust SSL Trial Enrollment

Enter CSR

After generating your server’s Certificate Signing Request as described in Generate CSR, paste the CSR in the form below. Please make sure that it contains the complete header and footer "BEGIN" and "END" lines exactly as in the example below.

Certificate Signing Request *

-----BEGIN CERTIFICATE REQUEST-----
MIIBBeDCB4gIBADAgIKoQsMwCQYDVQQGEwJFRTEUMBIGA1UEAwwKYXV0aC51dcC5lZTCB
nzANBgkqhkiG9w0BAQEEFAAOBjQAwgYkCgYEAlJTaMFCN7U5SHxO0M08HDLNghFV
ww+vCaeyk6CVynh6A21z7xUSF38Zn9pjZrfe+Egl6w9WV8CVRJgfoypcsHGD
stcY6nKHGdykotKVgsGM348hPvno00xf9718N0kdPqC7B1n3E4MQeKYeHD3TRFop+p
nRsvY7yLPbpw5z0CAwAAAAMBGUCsG5Ib3DQEJBzEIDAZhc2Rhc2QwDQYJKoZI
hvqNAQELBQADgYEAbPefwJCj9tTDVJ0n+hc3LB9urgBfQCL/2h4KW5VXRh/g6n
6GBmZ8d9eSpACLCenLBszT5D3jA8bWP/J3vTUR938sdgmu3y/KJ+amGo9az+
VSLcMI+B5qeX0Bu2YLFafQEpou80hNe/arjX+MjvdhKnWAYbVtxMKcibAM0=
-----END CERTIFICATE REQUEST-----

Back  Continue
Enrollment: Step 3

SSL Trial

GeoTrust SSL Trial Enrollment

Verify Server URL

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

https://auth.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.

Replace CSR

CSR Information

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

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

Continue
GeoTrust SSL Trial Enrollment

Approval of your certificate request
The GeoTrust SSL Trial service relies upon the Subscriber or the subscriber authorized administrator to approve all certificate requests for all hosts in the domain. We'll send an email to the domain contact in the domain's WHOIS record of each domain listed in your certificate information. To validate control of the domain, the owner of domain or an authorized representative must approve the request.

Select your authorized approvers
We send the approval email to the WHOIS contacts for each domain. In case the domain doesn't have any WHOIS contacts, we send the approval email to the pre-approved email addresses. You can customize the email recipients for each of your base domains if required.

No WHOIS contacts found for this domain. View subdomains in your order

List of authorized approvers
You can select a different email address for this base domain, if required.

Pre-approved email addresses
- admin@ut.ee
- administrator@ut.ee
- hostmaster@ut.ee
- webmaster@ut.ee
- postmaster@ut.ee

Back  Continue
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 sha256WithRSAEncryption (1.2.840.113549.1.1.11)
- 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 (may hardcode)
- 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: sha256WithRSAEncryption
    Issuer: C=EE, O=University of Tartu, OU=IT dep, CN=Arnis Root CA
    Validity
      Not After : Aug 28 12:59:31 2018 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: sha256WithRSAEncryption
      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

- Use CN from subject’s CSR DN (other fields must be ignored)
- 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: sha256WithRSAEncryption
  Issuer: C=EE, O=University of Tartu, OU=IT dep, CN=Arnis Root CA
  Validity
    Not After : Aug 28 12:59:31 2018 GMT
  Subject: 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: sha256WithRSAEncryption
    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 as input (padding always 0x00)
- 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?