MTAT.07.017
Applied Cryptography

Certificate Revocation List (CRL)

University of Tartu

Spring 2016
Certificate Validity

It may be required to invalidate a certificate before its expiration.

Examples:

- Private key compromised or lost
- Misissued certificate
- Affiliation has changed

Solution – Certificate Revocation List (CRL):

List of unexpired certificates that have been revoked by CA

- Where can relying party obtain CRL?
- How can relying party trust the content of CRL?
- How frequently the CA should issue CRL?
- How frequently the relying party should refresh CRL?
- How does relying party know that CRL is fresh?
- Legal terms: suspension and revocation
Digital Signatures Act

Division 3
Period of Validity, and Suspension and Revocation of Certificates

§ 11. Period of validity of certificates
(1) A certificate is valid as of the beginning of the period of validity set out in the certificate but not before entry of the corresponding data in the database of certificates which is maintained by the issuer of the certificate.
(2) A certificate expires upon expiry of the period of validity set out in the certificate or upon revocation of the certificate.

§ 12. Suspension of certificates
(1) A certification service provider has the right to suspend a certificate if the certification service provider has a reasonable doubt that incorrect data have been entered in the certificate or that it is possible to use the private key corresponding to the public key contained in the certificate without the consent of the certificate holder.
(3) After *verification of the legality of the claim* for suspension of a certificate, the certification service provider is required to *promptly enter* the data concerning suspension in the database of certificates which is maintained by it.

(6) Digital signatures or digital seals given *during the period* when a certificate is suspended are *invalid*.

§ 13. Termination of suspension of certificates
(1) Suspension of a certificate is *terminated on the basis of an application* of the certificate holder or a person or agency requesting the suspension of the certificate by entry of the corresponding data in the database of certificates which is maintained by the certification service provider that issued the certificate.
§ 14. Revocation of certificates
(1) The following are the bases for revocation of a certificate:
1) an application of the certificate holder;
2) an opportunity for using the private key corresponding to the public key set out in the certificate without the consent of the certificate holder;
3) divestment of the certificate holder of active legal capacity;
4) declaration of the death of the certificate holder;
5) the death of the certificate holder;
5.1) deletion from the register of the certificate holder due to dissolution or release or removal from office of a certificate holder who is a holder of office in public law;
6) submission of false data to a certification service provider by the certificate holder in order to obtain the certificate;
7) termination of the activities of the certification service provider;
8) other cases provided by law.
Certificate Revocation List (CRL)

CertificateList ::= SEQUENCE {
    tbsCertList TBSCertList, 
    signatureAlgorithm AlgorithmIdentifier, 
    signatureValue BIT STRING }

TBSCertList ::= SEQUENCE {
    version Version OPTIONAL, -- if present, MUST be v2(1) 
    signature AlgorithmIdentifier, 
    issuer Name, 
    thisUpdate UTCTime, 
    nextUpdate UTCTime OPTIONAL, 
    revokedCertificates SEQUENCE OF SEQUENCE  
        { 
            userCertificate CertificateSerialNumber, 
            revocationDate UTCTime, 
            crlEntryExtensions Extensions OPTIONAL -- in v2 } OPTIONAL, 
    crlExtensions [0] EXPLICIT Extensions OPTIONAL -- in v2 }

Certificate Revocation List (CRL)

- tbsCertList – DER structure to be signed by CRL issuer
- version – for v1 absent, for v2 contains 1
  - v2 introduces CRL and CRL Entry extensions
- signature – AlgorithmIdentifier from tbsCertList sequence
- issuer – identity of issuer who issued (signed) the CRL
  - CRL issued not by CA itself – indirect CRL
- thisUpdate – date when this CRL was issued
- nextUpdate – date when next CRL will be issued
- revokedCertificates – list of revoked certificates
  - userCertificate – serial number of revoked certificate
  - revocationDate – time when CA processed revocation request
  - crlEntryExtensions – provides additional revocation information
- crlExtensions – provides more information about CRL
CRL Entry Extensions

Provide methods for associating additional attributes with CRL entries. May be designated as critical or non-critical.

- **Reason Code**
  - Identifies the reason for the certificate revocation:
    
    ```
    CRLReason ::= ENUMERATED {
      unspecified  (0),
      keyCompromise (1),
      cACompromise  (2),
      affiliationChanged (3),
      superseded    (4),
      cessationOfOperation (5),
      certificateHold  (6),
      -- value 7 is not used
      removeFromCRL  (8),
      privilegeWithdrawn  (9),
      aACompromise  (10) }
    ```

- **Invalidity Date**
  - Provides the date on which the private key was compromised
  - How is it different from the revocation date?

- **Certificate Issuer**
  - Includes issuer name in case of indirect CRL
CRL Extensions

Provide methods for associating additional attributes with CRLs. May be designated as critical or non-critical.

- **CRL Number**
  - Monotonically increasing sequence number
  - Essential for delta CRL processing

- **Authority Key Identifier**
  - Identifies the certificate more precisely

- **Issuing Distribution Point**
  - Identifies the scope for a particular CRL

  ```
  IssuingDistributionPoint ::= SEQUENCE {
    distributionPoint [0] DistributionPointName OPTIONAL,
    onlyContainsUserCerts [1] BOOLEAN DEFAULT FALSE,
    onlyContainsCACerts [2] BOOLEAN DEFAULT FALSE,
    onlySomeReasons [3] ReasonFlags OPTIONAL,
    indirectCRL [4] BOOLEAN DEFAULT FALSE,
    onlyContainsAttributeCerts [5] BOOLEAN DEFAULT FALSE }
  ```

- If absent contains all revoked unexpired certificates
Delta CRLs

CRLs can contain huge list of revoked certificates:
http://www.sk.ee/crls/esteid/esteid2015.crl (64KB – 860)
http://www.sk.ee/repository/crls/esteid2011.crl (29MB – 396'197)

Delta CRL – CRL that provides changes to base CRL

- Delta CRL Indicator extension
  - Identifies a CRL as being a delta CRL
  - Contains BaseCRLNumber of complete CRL that is being updated

- Delta CRL Distribution Point extension
  - Identifies how delta CRL is obtained

- Complicated to process – not supported widely
Liability Analysis

Let’s assume subject’s private key has been compromised. Who is liable for actions made:

- in the time period before CA has been informed about the key compromise?
- in the time period while CA helpline not answering?
- in the time period before CA has marked the certificate revoked in their internal database?
- in the time period before next CRL has been issued?
- in the time period after revocation information has appeared in CRL?
- in the time period after CRL has been issued but not available to relying parties (e.g., CA server downtime)?
Digital Signatures Act

§ 3. Legal consequences of using digital signatures

(1) A digital signature has the same legal consequences as a hand-written signature (..)

(3) The giving of a digital signature does not have the consequences provided for in subsection (1) of this section if it is proved that the private key was used for giving the signature without the consent of the holder of the corresponding certificate.

(4) The giving of a digital signature without the consent of the holder of the corresponding certificate is deemed to be proved if the certificate holder proves the circumstances upon existence of which it may be presumed that the signature has been given without his or her consent.

(5) In the cases specified in subsection (3) of this section, the certificate holder shall compensate for damage caused to another person who erroneously presumed that the signature was given by the certificate holder, if the private key was used without the consent of the certificate holder due to the intent or gross negligence of the certificate holder.
Certificate Chain

- How to validate a certificate chain?
- Where to look whether the end entity certificate is revoked?
  - CRL issued by intermediate CA (usually every 12h)
  - Grace period
- Where to look whether the intermediate CA is revoked?
  - CRL issued by root CA (usually every 3 month)
  - Grace period?!
- Where to look whether the root CA is revoked?
  - CRL issued by root CA itself (vulnerable)

Who is liable for actions made after the root CA private key has been compromised?
CRL Distribution Points

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

Certificate Fields
- Subject's Public Key
- Extensions
  - Certificate Authority Key Identifier
  - Certificate Subject Key ID
  - Certificate Key Usage
  - Certificate Basic Constraints
  - Extended Key Usage
  - Certificate Policies
- CRL Distribution Points
- Authority Information Access
- Certificate Subject Alt Name

Field Value
- Not Critical
- URI: http://crl.tcs.terena.org/TERENASSLCA.crl
Hypertext Transfer Protocol (HTTP)

- Application layer client-server, request-response protocol
- Runs over TCP (Transmission Control Protocol) port 80

Client request (http://kodu.ut.ee/~arnis/):

```
GET /~arnis/ HTTP/1.1
Host: kodu.ut.ee
Connection: close
```

```
POST /~arnis/ HTTP/1.1
Host: kodu.ut.ee
Content-Length: 24
Connection: close
```

sending_this_binary_blob

Server response:

```
HTTP/1.1 200 OK
Date: Fri, 20 Mar 2016 09:33:00 GMT
Server: Apache/2.0
Content-Length: 48
Connection: close
Content-Type: text/html; charset=ISO-8859-15

<html><body><img src="c.png"></body></html>
```

- Request lines must all end with \r\n
- Header lines are separated from the body by empty line
- POST requests have non-empty request body

Sockets in Python

```python
>>> import socket
>>> s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
>>> s.connect(('kodu.ut.ee', 80))
>>> s.send("GET /~arnis/ HTTP/1.1\r\nHost: kodu.ut.ee\r\n\r\n")
43
>>> print s.recv(20)
HTTP/1.1 200 OK
Dat
```

- `recv()` returns bytes that are available in the read buffer
- `recv()` will wait if the read buffer empty (blocking by default)
- `recv()` will return 0 bytes if the connection is closed
- You must know how many bytes you must get
- Correct way to read HTTP response:
  - Read byte-by-byte until full response header received
  - Extract body size from “Connection-Length:” header
  - Read byte-by-byte until full response body received

http://docs.python.org/2/howto/sockets.html
Task 1: Check certificates against CRL

Implement utility that checks if certificates are revoked.

$ ./crlcheck.py
usage: crlcheck.py --issuer issuer --certificates cert [cert ...] url

$ wget https://sk.ee/upload/files/Juur-SK.pem.crt
$ ./crlcheck.py http://www.sk.ee/crls/juur/crl.crl --issuer Juur-SK.pem.crt
--certificates revoked.pem valid.pem nonissued.pem
[+] Downloading http://www.sk.ee/crls/juur/crl.crl
[+] CRL signature check successful!
[+] Serial 1084184741 (revoked.pem) loaded
[+] Serial 1167992746 (valid.pem) loaded
[-] Serial 9412 (nonissued.pem) not loaded: not issued by CA
[-] Certificate 1084184741 revoked: 2009-12-01 10:10:36 (cessationOfOperation)

$ ./crlcheck.py http://kodu.ut.ee/~arnis/appcrypto2016/outdated.crl
--issuer Juur-SK.pem.crt --certificates revoked.pem valid.pem nonissued.pem
[+] Downloading http://kodu.ut.ee/~arnis/appcrypto2016/outdated.crl
[+] CRL signature check successful!

$ ./crlcheck.py http://kodu.ut.ee/~arnis/appcrypto2016/badsign.crl
--issuer Juur-SK.pem.crt --certificates revoked.pem valid.pem nonissued.pem
[-] CRL signature verification failed!
Hints

• Download CRL using Python sockets (in the correct way)

• Convert UTCTime to datetime object:
  >>> import datetime
  >>> dateobj = datetime.datetime.strptime(utctimestr, '%y%m%d%H%M%SZ')
  >>> print datetime.datetime.utcnow() > dateobj
  True

• Use urlparse for easy URL parsing:
  >>> import urlparse
  >>> urlparse.urlparse("http://kodu.ut.ee/~arnis/some.file")
  ParseResult(scheme='http', netloc='kodu.ut.ee', path='/~arnis/some.file',
               params='', query='', fragment='')
  >>> urlparse.urlparse("http://kodu.ut.ee/~arnis/some.file").netloc
  'kodu.ut.ee'

• Use regular expression operation to extract body size:
  >>> import re
  >>> re.search('content-length: \s*(\d+)\s', header, re.S+re.I).group(1)

• Use pyasn1 encoder.encode() to get decoded substructure as DER:
  >>> from pyasn1.codec.der import decoder, encoder
  >>> tbsCertList = encoder.encode(decoder.decode(crl)[0][0])
Comments

- **Wrong** ways to download HTTP response body:
  - Reading response in one go (**wrong**!):
    
    ```python
    body = s.recv(content_length)
    ```

    "The receive calls normally return any data available, up to the requested amount, rather than waiting for receipt of the full amount requested." (man page recv section 2)

  - Reading until socket closed (**wrong**!):
    ```python
    body = ""
    buf = s.recv(1024)
    while len(buf):
        buf = s.recv(1024)
        body+= buf
    ```

    After sending a response, an HTTP/1.1 server will wait for more request/response exchanges, unless header "Connection: close" was specified by the client. Therefore `s.recv()` will hang until the timeout configured by the server is reached.