SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

Established in collaboration with MIT

ISTD 50.005, Spring 2016
Signed certificate verification

1 Objective

This supplementary document will show you how to (i) verify a server's signed certificate using the CA's certificate and (ii) extract the public key from a signed certificate.

2 Background

You know from the lectures that public key will never be sent alone, the owner of this public key always sends his certificate, and the recipient will extract sender's public key from the received certificate. In addition, a certificate needs to be signed by trusted Certificate Authorities (CAs) before it can be used. Roughly speaking, to sign a certificate request, a CA just needs to encrypt the certificate request using his private key. Therefore, it's quite straightforward for you to verify a signed certificate: (1) extract CA's public key from CA's certificate, and (2) verify server's signed certificate using a CA's public key obtained from (1). To complete (1) and (2) you need to use X509Certificate class in Java.

3 X509Certificate Class

The X509Certificate class provides basic functions for (signed) certificate verification, checking (signed) certificate validity and extracting the public key from a signed certificate. For further information about the class, you can refer to the Java API documentation at http://docs.oracle.com/javase/7/docs/api/javax/security/cert/X509Certificate.html.

3.1 Create X509Certificate object

To use the X509Certificate class, you first need to create an X509Certificate object using CertificateFactory. Assuming that the certificate is stored in your computer with the file name *CA.crt*, the following example shows you how to create an X509Certificate object:

```
InputStream fis = new FileInputStream("CA.crt");

CertificateFactory cf = CertificateFactory.getInstance("X.509");

X509Certificate CAcert =(X509Certificate)cf.generateCertificate(fis);
```

3.2 Extract public key from X509Certificate object

To extract the public key from a certificate, you can use the `getPublicKey()` method of the X509Certificate class. Given the X509Certificate object (*CAcert*) created in section 3.1, here is how we extract public key from this object:

```
PublicKey key = CAcert.getPublicKey();
```

3.3 Check validity of signed certificate.

A certificate is valid if the current date and time are within the validity period specified in the certificate. To check if a certificate is currently valid or not, use the following method:

```
public abstract void checkValidity();
```

The key parameter is the public key of CA (Refer to Section 3.2 on extracting the public key from a certificate). Assuming that the public key of CA is *CAkey* and the X509Certificate object created from the server signed certificate is *ServerCert*, here is an example of how to check validity:

```
ServerCert.checkValidity();
```

4 What you need to do

You can check the validity of your signed certificate using the CA's public key (extracted from the CA's certificate).