Understanding digital certificates

Mick O'Brien and George R S Weir

Department of Computer and Information Sciences,
University of Strathclyde
Glasgow G1 1XH
mickobrien137@hotmail.co.uk, george.weir@cis.strath.ac.uk

Abstract

Digital certificates are a core component in the provision of secure data communications. Gaining an understanding of the nature, creation and operation as well as the variety of these certificates is an essential step for students of computer, information or network security. In order to clarify the relationship between central technologies, including symmetric and asymmetric encryption, digital signatures, certificate key stores, certificate revocation lists, and the use of digital certificates in secure Web transactions, we have developed a software tool that allows users to explore these aspects of data security. This paper outlines some of the surrounding issues and describes the 'sandpit' application as a means of exploring and, thereby, gaining a better understanding of digital certificates.

1. Introduction

Some understanding of data security is becoming essential for the average computer user as much as for students of computing and other technical subjects [1]. Part of the difficulty faced by aspiring learners, is in grasping the nature and operation of the underlying security technologies. Irvine et.al. [2], discuss the requirements for adequate security education and note that 'the educational outcome for cultivating a focus on empirical reasoning skills includes the ability to construct experiments or prototypes to demonstrate some purpose or facilitate some meaningful exploration and the ability to observe, collect, analyze, and interpret data from experiments' (p.27). To this end, practical exposure to realistic security contexts and the associated software facilities is essential. First-hand operational use of security software and its associated concepts lends itself to honing the students' empirical reasoning skills.

Digital certificates provide a mechanism to authenticate and secure information on open networks. Applications using this mechanism include secure email, secure web communications, digital signing of software files, smart card authentication, and encrypting file systems. Certificates are a key building block for providing security services within an IT infrastructure, usually referred to as a public key infrastructure (PKI). Such contexts support:

- the binding of public keys to entities
- the distribution of public key certificates
- verification of entity public key certificates via a third party (the certificate authority)

This PKI infrastructure will then enforce user authentication, network encryption, data integrity and non-repudiation of origin for the data.

1.1 Certificate Components

The basic components within a digital certificate include:

- the name of the user/entity being certified
- the public key of the user/entity
- the name of the certification authority
- a digital signature

The certificate provides a binding link between a user/entity and a public key, so the certificates must use a well defined name space for the user/entity being identified. The International Telecommunication Union X.509 specification [3] provides a set of standards for the implementation of a public key infrastructure – one being for the structure of a digital public key certificate:

The X.509 certificate standard has evolved over a number of years. Version 1 was introduced in 1988 and assumed that by using the issuer distinguished name of a certificate, it would be possible to build a certificate chain going back to the root certificate. Version 2 was introduced in 1993 and introduced the concept of unique identifiers to allow for the re-use of issuer distinguished names. Version 3 was introduced in 1996 and allowed for anyone to define an extension and include it within their certificate. Version 1 certificates are mainly used as root or self-signed certificates, version 2 certificates have been superseded by version 3 certificates which are in use for most applications. Extended Validation (EV) certificates are sometimes referred to as Version 4 type certificates, however, as the EV profile entails a change to attributes within the version 3 definition rather than a change to the structure of a X.509 certificate this categorization can be ignored.

1.2 Learning about digital certificates

While many computer users and students of computing or information science are aware that digital certificates play a role in secure data exchange, the full nature and application of the associated technology is difficult to embrace. One source for this difficulty is the range of component technologies that combine in order to afford data security. In order to clarify the relationship between central technologies such as, symmetric and asymmetric encryption, digital signatures, certificate key stores, certificate revocation lists, and the use of digital certificates in secure Web transactions, we have developed a software tool that allows users to explore these aspects of data security. We describe this software as a ‘sandpit’, since it provides a convenient and safe context in which to ‘play’ with encryption, digital signatures and digital certificates. Our motivation is to offer this facility for general use in order to ease the challenges of understanding digital certificates.

2. The Sandpit Application

A primary purpose of the application is to generate and examine public key digital certificates. Specific objectives for the application include:

- generation and use of symmetric keys for encrypting data (files);
- generation and use of asymmetric keys for encrypting data (files);
- generation and addition of digital signatures to data (files);
- generation of public key digital certificates;
- generation of digital certificates for specific purposes ;
- generation and use of certificate key stores;
- generation and use of certificate revocation lists;
- direct download of certificate revocation lists from a commercial website;
- indirect download (via CRL distribution point within a digital certificate) of a certificate revocation list from a commercial website;
- use of digital certificates within a secure socket layer communication;
- use of digital certificates within a simple HTTPS communication.

SandPit is written in Java, using cryptographic functions obtained from Bouncycastle (www.bouncycastle.com), and appears as window with a series of tabbed panes. These tabs give the user access to aspects of cryptography, digital certificates and their use. The initial screen is depicted in Figure 1, below.

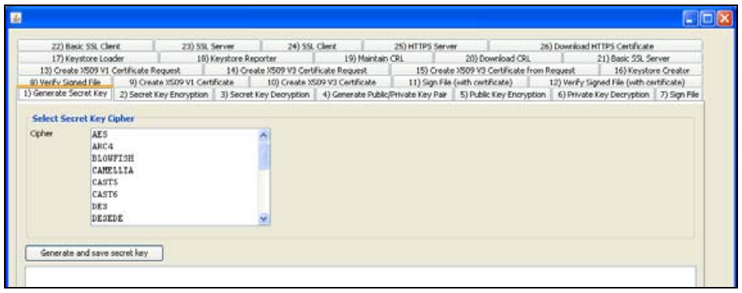


Figure 1: The Sandpit Application

The range of facilities afforded by SandPit is accessed through a series of tabs, each of which provides access to the creation or inspection of particular certificate-related features. A summary of the 26 application tabs is given in Table 1, below.

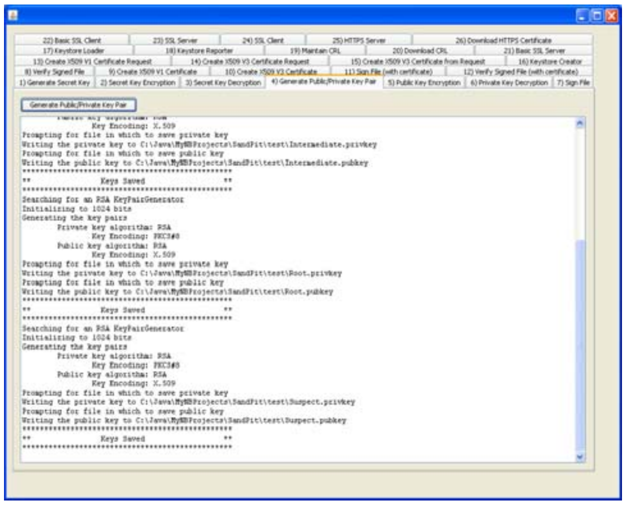
No.	Purpose	Description
1	Generate Secret Key	select an algorithm for secret key generation
2	Secret Key Encryption	use the secret key and algorithm to encrypt a plain text file
3	Secret Key Decryption	decrypt the file encrypted in the previous screen
4	Generate Public/Private Key Pair	generate an RSA public/private key pair
5	Public Key Encryption	encrypt a file using the RSA public key
6	Private Key Encryption	decrypt the file using the RSA private key
7	Sign File	sign a plain text file using the RSA public key
8	Verify Signed File	verify the signed file (using the RSA public key)
9	Create X.509 V1 Certificate	generate a root digital certificate using the RSA key pair
10	Create X.509 V3 Certificate	generate a version 3 X.509 certificate
11	Sign File (with certificate)	sign text file using public key from the digital certificate
12	Verify signed File (with certificate)	verify the signed file (using public key from the digital certificate)
13	Create X.509 V1 Certificate Request	generate certificate request for a X.509 version 1 certificate (root)
14	Create X.509 V3 Certificate Request	generate a certificate request for a X.509 version 3 certificate (end-entity)
15	Create X.509 Certificate from files	generate certificates from certificate request files
16	Key store Creator	create a key store
17	Key store Loader	load the key store with certificates and private key/certificate pairs
18	Key store Reporter	report on the contents of key stores
19	Maintain certificate revocation lists (CRLs)	create and amend CRLs
20	Download CRLs	download CRLs from the Internet
21	Basic SSL Server	simple SSL communication with a client
22	Basic SSL Client	simple SSL communication from a server
23	SSL Server	initiate a more complex SSL communication
24	SSL Client	used within a complex SSL communication
25	HTTPS Server	display a secure message within a web page
26	Download HTTPS Certificate	download a digital certificate via HTTPS

Table 1: Summary of SandPit Facilities

In what follows, we describe some of these certificate-related interactions in further detail. We begin with the creation of RSA keys.

2.1 Creating RSA Keys

Here, the SandPit user is required to create four key pairs: EndEntity key pair, Intermediate key pair, Root key pair and Suspect key pair. The result of generating these key pairs is shown in Figure 2, below. As can be seen from the text area within the public/private key generation panel, a 1024-bit public/private key size is generated. The security of the RSA algorithm is based on the problem of factoring large numbers; hence the public key is generated by multiplying two large prime numbers.


22) Basic SSL Client
23) SSL Server
24) SSL Client
25) HTTPS Server
26) Download HTTPS Certificate
17) Keystore Loader
18) Keystore Reporter
19) Maintain CRL
20) Download CRL
21) Basic SSL Server
13) Create X509 V1 Certificate Request
14) Create X509 V3 Certificate Request
15) Create X509 V3 Certificate from Request
16) Keystore Creator
8) Verify Signed File
9) Create X509 V1 Certificate
10) Create X509 V3 Certificate
11) Sign File (with certificate)
12) Verify Signed File (with certificate)
1) Generate Secret Key
2) Secret Key Encryption
3) Secret Key Decryption
4) Generate Public/Private Key Pair
5) Public Key Encryption
6) Private Key Decryption
7) Sign File
Generate Public/Private Key Pair

```
                Key Encoding: X.509
Prompting for file in which to save private key
Writing the private key to C:\Java\MyNBProjects\SandPit\test\Intermediate.privkey
Prompting for file in which to save public key
Writing the public key to C:\Java\MyNBProjects\SandPit\test\Intermediate.pubkey
************************************************
**              Keys Saved                    **
************************************************
Searching for an RSA KeyPairGenerator
Initializing to 1024 bits
Generating the key pairs
        Private key algorithm: RSA
                Key Encoding: PKCS#8
        Public key algorithm: RSA
                Key Encoding: X.509
Prompting for file in which to save private key
Writing the private key to C:\Java\MyNBProjects\SandPit\test\Root.privkey
Prompting for file in which to save public key
Writing the public key to C:\Java\MyNBProjects\SandPit\test\Root.pubkey
************************************************
**              Keys Saved                    **
************************************************
Searching for an RSA KeyPairGenerator
Initializing to 1024 bits
Generating the key pairs
        Private key algorithm: RSA
                Key Encoding: PKCS#8
        Public key algorithm: RSA
                Key Encoding: X.509
Prompting for file in which to save private key
Writing the private key to C:\Java\MyNBProjects\SandPit\test\Suspect.privkey
Prompting for file in which to save public key
Writing the public key to C:\Java\MyNBProjects\SandPit\test\Suspect.pubkey
************************************************
**              Keys Saved                    **
************************************************
```

Figure 2: Generating Key Pairs

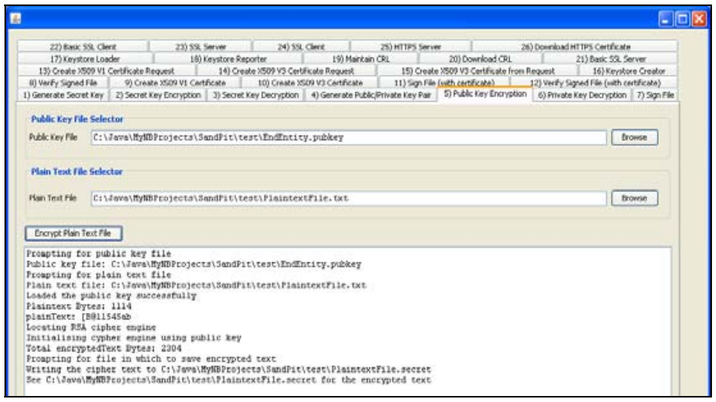
2.2 Public Key Encryption and Decryption

Tab 5 in the SandPit application supports public key encryption using the keys generated previously (using Tab 4). A specified plaintext file is encrypted in blocks of 64 bytes (as a 1024-bit key is generated) and this will mimic the process whereby a secret message needs to be sent from the public domain. The application will prompt for a filename to which the cipher text will be saved and the final screen will appear as indicated in Figure 3, below.

The corresponding decryption process can be applied from Tab 6. This requires the user to specify the private key file and the file to be decrypted. The application will save the decrypted text to a user-specified file.

2.3 Digital Signatures

Asymmetric key pairs are used to sign and verify a plaintext file (Tab 7). For this purpose, a hash of the plaintext file is created using MD5. This hash is then encrypted via RSA using the private key to generate a digital signature. The signed file contains two distinct objects – the original content of the plaintext file and the associated signature.


22) Basic SSL Client
23) SSL Server
24) SSL Client
25) HTTPS Server
26) Download HTTPS Certificate
17) Keystore Loader
18) Keystore Reporter
19) Maintain CRL
20) Download CRL
21) Basic SSL Server
13) Create X509 V1 Certificate Request
14) Create X509 V3 Certificate Request
15) Create X509 V3 Certificate from Request
16) Keystore Creator
8) Verify Signed File
9) Create X509 V1 Certificate
10) Create X509 V3 Certificate
11) Sign File (with certificate)
12) Verify Signed File (with certificate)
1) Generate Secret Key
2) Secret Key Encryption
3) Secret Key Decryption
4) Generate Public/Private Key Pair
5) Public Key Encryption
6) Private Key Decryption
7) Sign File
Browse
Browse
Encrypt Plain Text File

Public Key File Selector

Public Key File: C:\Java\MyNBProjects\SandPit\test\EndEntity.pubkey

Plain Text File Selector

Plain Text File: C:\Java\MyNBProjects\SandPit\test\PlaintextFile.txt

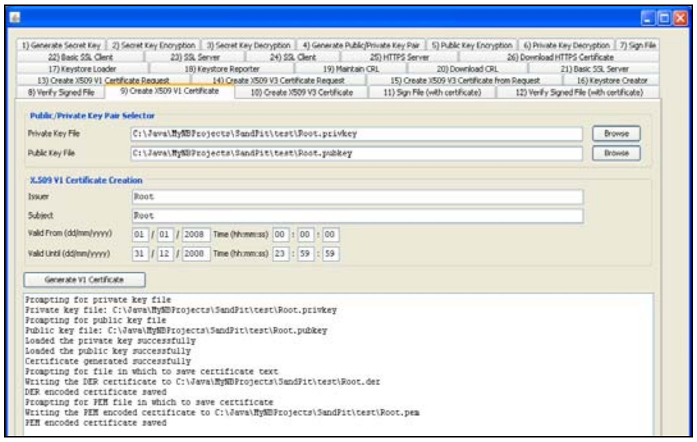
```
Prompting for public key file
Public key file: C:\Java\MyNBProjects\SandPit\test\EndEntity.pubkey
Prompting for plain text file
Plain text file: C:\Java\MyNBProjects\SandPit\test\PlaintextFile.txt
Loaded the public key successfully
Plaintext Bytes: 1114
plainText: [B@11545ab
Locating RSA cipher engine
Initialising cypher engine using public key
Total encryptedText Bytes: 2304
Prompting for file in which to save encrypted text
Writing the cipher text to C:\Java\MyNBProjects\SandPit\test\PlaintextFile.secret
See C:\Java\MyNBProjects\SandPit\test\PlaintextFile.secret for the encrypted text
```

Figure 3: Public Key Encryption

Verifying a signed file is supported through Tab 8. The signed file must initially have the encrypted hash value decrypted using the signer's public key. The retrieved hash value must be then compared against a hash value calculated using the same hashing algorithm as was used by the signer. If the two hash values agree, then the content of the document has not been modified in transit from the signer and so the file is verified.

2.4 Digital Certificates

Through Tab 9, digital certificates can be created directly (to mimic v1 or self-signed certificates) and via certificate requests (to mimic the generation of certificates by a certification authority). With a root certificate, we can begin to build certificate chains and act as a certificate authority. The generated certificate is saved in two formats – a DER (Distinguished Encoding Rules) encoding and a PEM (Privacy Enhanced Mail) encoding (see Figure 4, below).


1) Generate Secret Key
2) Secret Key Encryption
3) Secret Key Decryption
4) Generate Public/Private Key Pair
5) Public Key Encryption
6) Private Key Decryption
7) Sign File
22) Basic SSL Client
23) SSL Server
24) SSL Client
25) HTTPS Server
26) Download HTTPS Certificate
17) Keystore Loader
18) Keystore Reporter
19) Maintain CRL
20) Download CRL
21) Basic SSL Server
13) Create X509 V1 Certificate Request
14) Create X509 V3 Certificate Request
15) Create X509 V3 Certificate from Request
16) Keystore Creator
8) Verify Signed File
9) Create X509 V1 Certificate
10) Create X509 V3 Certificate
11) Sign File (with certificate)
12) Verify Signed File (with certificate)
Browse
Browse
Generate V1 Certificate

Public/Private Key Pair Selector

Private Key File: C:\Java\MyNBProjects\SandPit\test\Root.privkey

Public Key File: C:\Java\MyNBProjects\SandPit\test\Root.pubkey

X.509 V1 Certificate Creation

Issuer: Root

Subject: Root

Valid From (dd/mm/yyyy): 01 / 01 / 2008 Time (hh:mm:ss) 00 : 00 : 00

Valid Until (dd/mm/yyyy): 31 / 12 / 2008 Time (hh:mm:ss) 23 : 59 : 59

```
Prompting for private key file
Private key file: C:\Java\MyNBProjects\SandPit\test\Root.privkey
Prompting for public key file
Public key file: C:\Java\MyNBProjects\SandPit\test\Root.pubkey
Loaded the private key successfully
Loaded the public key successfully
Certificate generated successfully
Prompting for file in which to save certificate text
Writing the DER certificate to C:\Java\MyNBProjects\SandPit\test\Root.der
DER encoded certificate saved
Prompting for PEM file in which to save certificate
Writing the PEM encoded certificate to C:\Java\MyNBProjects\SandPit\test\Root.pem
PEM encoded certificate saved
```

Figure 4: Creating X.509 Version 1 Certificate

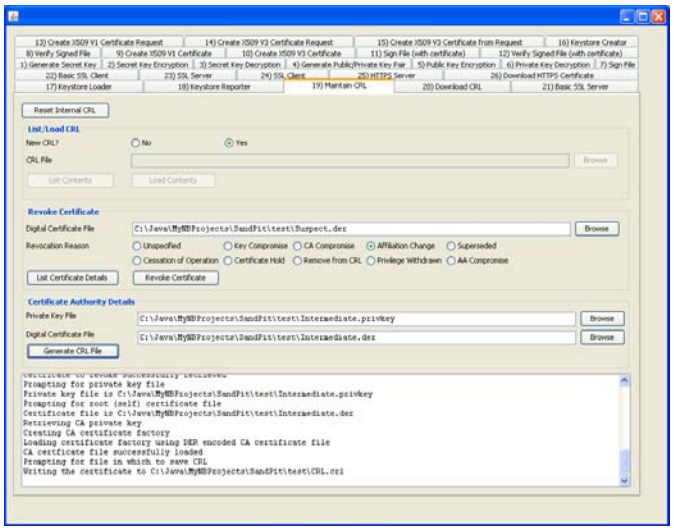
A self signed version 3 certificate, similar to this version 1, but with the addition of certificate type, alternative name, key usage and extended key usage fields may be created from Tab 10. Signing a file with a digital certificate is similar to the earlier digital signature example. This is applied via Tab 11 and the resultant certificate-signed file can be verified from the options provided under Tab 12. Tabs 13, through 15 support actions using requested certificates, i.e., a certificate provided by a recognised Certificate Authority.

2.5 Key stores, private keys and certificates

Although there are numerous types and formats of key stores available, our application only includes Sun's Java Key store (JKS) and Public Key Cryptography Standard 12 (PKCS#12). The Sun JKS is a key store format created by Sun that defines a collection of keys and certificates. This key store can only store private keys and trusted certificates. In contrast, PKCS#12 defines a portable format for storing and transporting public/private keys and certificates. PKCS#12 key store contents are password protected and need to be transformed to another key store format (such as JKS) in order to be used. The PKCS#12 key store can be used in several ways that are based around the combination of two privacy modes and two integrity modes. The privacy modes use encryption to protect personal information from exposure and the integrity modes protect personal information from tampering. Tabs 16 through 18 in the SandPit program allow for the creation and use of such key stores.

2.6 Certificate Revocation Lists

Certificate Revocation Lists can also be exercised using this application. From Tab 19 it is possible to create or load CRLs (Figure 5) and then add certificates with reasons for revocation. CRLs from the Internet may be downloaded using Tab 20.



List/Load CRL
New CRL? No [✓ Yes]
CRL File

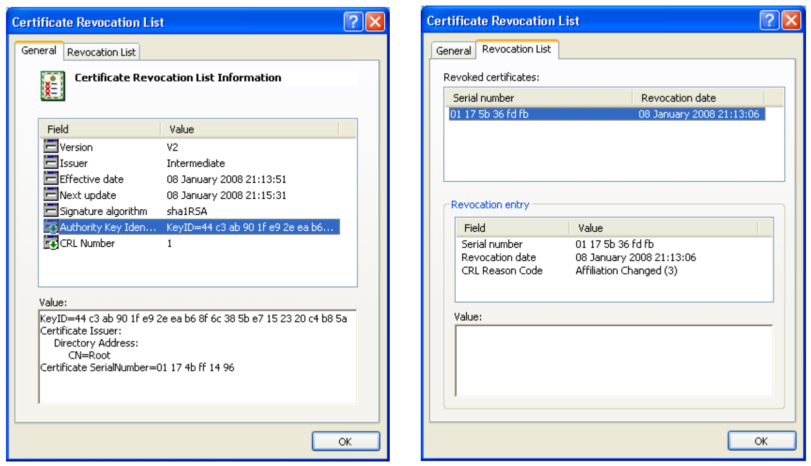
Revoke Certificate
Digital Certificate File: C:\Java\MyNBProjects\SandPit\test\Suspect.der
Revocation Reason: Unspecified, Key Compromise, CA Compromise, [✓ Affiliation Change], Superseded, Cessation of Operation, Certificate Hold, Remove from CRL, Privilege Withdrawn, AA Compromise

Certificate Authority Details
Private Key File: C:\Java\MyNBProjects\SandPit\test\Intermediate.privkey
Digital Certificate File: C:\Java\MyNBProjects\SandPit\test\Intermediate.der

```
Certificate to revoke successfully retrieved
Prompting for private key file
Private key file is C:\Java\MyNBProjects\SandPit\test\Intermediate.privkey
Prompting for root (self) certificate file
Certificate file is C:\Java\MyNBProjects\SandPit\test\Intermediate.der
Retrieving CA private key
Creating CA certificate factory
Loading certificate factory using DER encoded CA certificate file
CA certficate file successfully loaded
Prompting for file in which to save CRL
Writing the certificate to C:\Java\MyNBProjects\SandPit\test\CRL.crl
```

Figure 5: Creating a Certificate Revocation List

The results of certificate revocation can be viewed in a local Web browser. Figure 6, illustrates the CRL displays from Internet Explorer.


Certificate Revocation List
General
Revocation List
Certificate Revocation List Information

Field	Value
Version	V2
Issuer	Intermediate
Effective date	08 January 2008 21:13:51
Next update	08 January 2008 21:15:31
Signature algorithm	sha1RSA
Authority Key Iden...	KeyID=44 c3 ab 90 1f e9 2e ea b6...
CRL Number	1

Value:

```
KeyID=44 c3 ab 90 1f e9 2e ea b6 8f 6c 38 5b e7 15 23 20 c4 b8 5a
Certificate Issuer:
    Directory Address:
        CN=Root
Certificate SerialNumber=01 17 4b ff 14 96
```

OK
Certificate Revocation List
General
Revocation List

Revoked certificates:

Serial number	Revocation date
01 17 5b 36 fd fb	08 January 2008 21:13:06

Revocation entry

Field	Value
Serial number	01 17 5b 36 fd fb
Revocation date	08 January 2008 21:13:06
CRL Reason Code	Affiliation Changed (3)

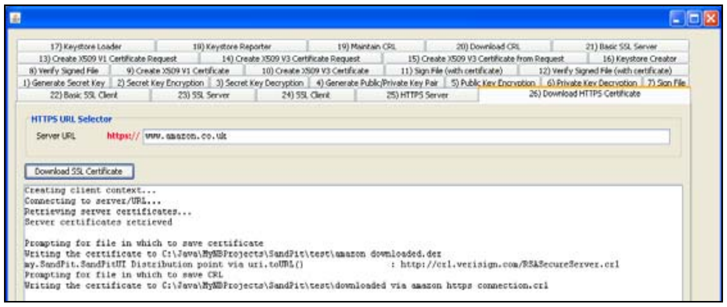
Value:

Figure 6: Viewing certificate revocation list

2.7 Download HTTPS Certificate

The final digital certificate aspect of the application allows for the download of digital certificates from websites via HTTPS. These certificates are not validated within SandPit, rather HTTPS has been configured within the application to trust all digital certificates. This is not a security risk, since we are simply examining the certificates rather than using them to establish a context for e-commerce.

Once the certificate has been downloaded, it is then checked to see if it contains a URL that is a CRL distribution point – if one is found then the CRL is also downloaded. In Tab 26 the user may enter a URL for a website that uses HTTPS – www.amazon.co.uk is used in this example – and the application screen will appears as indicated in Figure 7.



Server URL https:// www.amazon.co.uk

```
Creating client context...
Connecting to server/URL...
Retrieving server certificates...
Server certificates retrieved

Prompting for file in which to save certificate
Writing the certificate to C:\Java\MyNBProjects\SandPit\test\amazon downloaded.der
my.SandPit.SandPitUI Distribution point via uri.toURL()                : http://crl.verisign.com/RSASecureServer.crl
Prompting for file in which to save CRL
Writing the certificate to C:\Java\MyNBProjects\SandPit\test\downloaded via amazon https connection.crl
```

Figure 7: Downloading digital certificates via HTTPS

3. Conclusion

The SandPit application allows users to explore a variety of techniques specific to data security. These include the use of popular algorithms in symmetric cryptography, the use of RSA keys within asymmetric cryptography, the generation of self-signed X.509 certificates, the generation of X.509 digital certificates from certificate request files, the storage of private keys and digital certificates within a local key store, and the generation, maintenance and download of certificate revocation lists. Additional facilities, not detailed in the present paper, include the use of digital certificates within a secure socket layer (SSL) communication and the use of digital certificates within a HTTPS communication.

Our purpose in developing this system is to afford easy interaction with the actual algorithms, file formats and operations that are commonly employed, often behind the scenes, in secure data transactions. Such first-hand experience is considered the best means of understanding digital certificates and related techniques.

References

1. Anttila, J., Savola, R,, Kajava, J, and Lindfors, J., Fulfilling the Needs for Information Security Awareness and Learning in Information Society, in The 6th Annual Security Conference, Las Vegas, 2007.
2. Irvine, C.E., Chin, S.K. and Fryncke, D. Integrating Security into the Curriculum, Computer, 31 (12), 25-30, 1998.
3. ITU-T. Rec. X.509 (revised) the Directory Authentication Framework, International Telecommunication Union, Geneva, 1993