Sri Lanka Institute of Information Technology
Introduction to Cyber Security - IE2022
Lab Sheet 5
Year 2, Semester 1

Encryption & Decryption using OpenSSL

Cipher algorithms that can be used in OpenSSL

aes-128-cbc	aes-128-ecb	aes-192-cbc	aes-192-ecb	aes-256-cbc
aes-256-ecb	base64	bf	bf-cbc	bf-cfb
bf-ecb	bf-ofb	cast	cast-cbc	cast5-cbc
cast5-cfb	cast5-ecb	cast5-ofb	des	des-cbc
des-cfb	des-ecb	des-ede	des-ede-cbc	des-ede-cfb
des-ede-ofb	des-ede3	des-ede3-cbc	des-ede3-cfb	des-ede3-ofb
des-ofb	des3	desx	rc2	rc2-40-cbc
rc2-64-cbc	rc2-cbc	rc2-cfb	rc2-ecb	rc2-ofb
rc4	rc4-40			

Message digest algorithms that can be used in OpenSSL

md2	md4	md5	rmd160	sha
sha1				

How to encrypt a file using OpenSSL

Using a passphrase to generate the key to encrypt & decrypt
Step 1: Choose an appropriate algorithm that would suit your requirement
Step 2: Use the following format to encrypt or decrypt a file

```
$ openssl enc <cipher algorithm> e/d in <input_file_name> out
<output_file_name> md [md5/sha/sha1]
```

```
    <cipher algorithm> = one of the cypher algorithms above
    e/d              = e for encryption and d for decryption
    md               = massage digest which used to create a key from a
    pass phrase
    a                = encrypt && encode base64 / decode base64 &&
    decrypt
```

- OpenSSL result could be used to encrypt/decrypt files using private keys and public keys

- **Using the public key to encrypt & private key to decrypt**

Encryption

```
$ openssl rsautl encrypt pubin inkey <public_key>.<keyformat> keyform
<keyformat> in <file_to_encrypt> out <file_encrypted>
```

Decryption

```
$ openssl rsautl decrypt inkey <private_key>.<keyformat> keyform
<keyformat> in <file_encrypted> out <file_decrypted>
```

Signing RSA/DSA

```
$ openssl rsautl sign inkey <private_key>.<keyformat> keyform
<keyformat> in <file_to_sign> out <file_signed>
```

Verifying RSA/DSA

```
$ openssl rsautl verify pubin inkey <public_key>.<keyformat> keyform
<keyformat> in <file_signed>
```

Generating key pairs - [*public key / private key]*

Generating an RSA key pair

- This key can be used for signing and encryption

```
$ openssl genrsa out <private_key>.pem <key_length>
```

- If you would like to protect the private key with a password you should use an additional parameter

```
$ openssl genrsa <cipher algorithm> out <private_key>.pem<key_length>
```

- Now we have a private key. We derive the public key from the private key.

```
$ openssl rsa in <private_key>.pem pubout out <public_key>.pem
```

Generating a DSA key pair

- This key can only be used for signing

Step 1: generating parameters that intern used to generate the key

```
$ openssl dsaparam out <dsa_param_file>.pem <key_length>
```

Step 2-i: generating key without protected by a passphrase

```
$ openssl gendsa out <private_key>.pem <dsa_param_file>.pem
```

Step 2-ii: generating a key with password protected

```
$ openssl gendsa des3 out <private_key>.pem <dsa_param_file>.pem
```

- The whole thing could be done in a single step also. Then you will only be able to generate a single key from the parameters generated.

```
$ openssl dsaparam noout out <private_key>.pem genkey <key_length>
```

- Now we have a private key. We derive the public key from the private key.

```
$ openssl dsa in <private_key>.pem pubout out <public_key>.pem
```

Removing passphrase encryption of your private key

If you are running a daemon using your key, then you will have to enter the password whenever the daemon starts if the daemon configuration file does not give a directive to specify the crypt password contained file. This might be useful if you come up with a situation like the above.

```
$ openssl <algorithm> in <private_key_encrypted>.pem out
<private_key_plain>.pem <algorithm> = rsa or dsa
```