# Public-Key Infrastructure (PKI)



## Contents

**1 Aims & Objectives** **2**
1.1 Optional Additional Reading Resources . . . 2

**2 Lab Tasks** **2**
2.1 Task 1: RSA Key Cryptanalysis . . . 2
2.1.1 Task Requirements . . . 3
2.1.2 Guidelines . . . 3
2.2 Task 2: Creating a Certificate Authority (CA) . . . 3
2.2.1 Task Requirements . . . 3
2.2.2 Guidelines . . . 3
2.3 Task 3: Using your CA to Issue Certificates . . . 4
2.3.1 Task Requirements . . . 4
2.3.2 Guidelines . . . 5
2.4 Task 4: Deploying Certificates in OpenSSL HTTPS Server . . . 6
2.4.1 Task Requirements . . . 6
2.4.2 Guidelines . . . 6
2.5 Task 5: Deploying Certificates in Apache HTTPS Server . . . 8
2.5.1 Task Requirements . . . 8
2.5.2 Guidelines . . . 9
2.6 Task 6: Impersonating a Website (Man-in-the-Middle) . . . 10
2.6.1 Task Requirements . . . 10
2.6.2 Guidelines . . . 10
2.7 Task 7: Alternative Approaches to (Certificate-Based) PKI (Research Task) . . . 11
2.7.1 Task Requirements . . . 11

**3 What to Submit** **11**
3.1 Plagiarism . . . 12

**4 Marking Criteria** **13**

**5 Document Revision History** **14**

# 1 Aims & Objectives

The aim of this lab is to give students a hands-on experience of Public-Key Infrastructure (PKI) and help them understand its benefits, some of its inherent shortcomings, and the possible alternatives. This lab is *assessed* and consists of **7 tasks**. All tasks must be answered. By finishing the lab, students will learn how to issue and deploy digital certificates and how PKI can protect against some attacks. Additionally, the students will learn the shortcomings of (certificate-based) PKI and be able to discuss possible alternatives. Special attention should be paid to Section 3 and the What to Submit subsections of each task which list what exactly needs to be submitted for this lab.

## 1.1 Optional Additional Reading Resources

- Chapters 23 & 24 of the SEED Book, Computer & Internet Security: A Hands-on Approach, 2nd Edition, by Wenliang Du. See details at `https://www.handsonsecurity.net`.
- Chapters 8 & 13 of the Handbook of Applied Cryptography, 5th Edition, by Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone. See details at `http://cacr.uwaterloo.ca/hac`.
- Apache HTTP Server `https://httpd.apache.org/docs/2.4/ssl/ssl_howto.html`.

# 2 Lab Tasks

This section contains the specification of the required tasks.

**Important Note:** In some of the tasks you will be required to create a domain of the form `UWEFirstNameLastName.com`. You need to replace `FirstName` and `LastName` with your own first and last names, respectively. For instance, my domain would be `UWEEssamGhadafi.com`. No marks will be given for the tasks in question if your domain does not correspond to your own name.

## 2.1 Task 1: RSA Key Cryptanalysis

This task requires you to recover the private signing key belonging to a server from partially leaked information and the public key.

### 2.1.1 Task Requirements

In the file `Task1.txt` in Blackboard you will find an RSA public key $(e, N)$ (all in hexadecimal). Also, one of the two prime factors (p) of the modulus $N$ is in the same file. Your task is to recover the private exponent $d$ which would allow anyone to sign on behalf of (and hence masquerade as) the server to whom this key belongs.

**What to Submit:** You need to submit the value of the private exponent $d$ (in hexadecimal), the code snippet you used, and a screenshot of the steps you used to complete the task. Also, you are free to include any interesting observations you have made/learnt from doing the task.

### 2.1.2 Guidelines

The lecture slides and lab sheet concerned with Public-Key and Digital Signatures might come in handy when answering this task.

## 2.2 Task 2: Creating a Certificate Authority (CA)

In this task you will learn how to use OpenSSL to create a self-signed certificate for a certificate authority (CA).

### 2.2.1 Task Requirements

You need to create a key and self-signed certificate for a certificate authority. Some related guidelines can be found in Section 2.2.2.

**Important Note:** The CA key you generate must be 4096-bit RSA key and the used hash function to self-sign the CA certificate need to be SHA 512.

**What to Submit:** You need to include a screenshot of the steps and commands you used to complete the task. Also, you are free to include any interesting observations you have made/learnt from doing the task.

### 2.2.2 Guidelines

To create a self-signed certificate for a CA, we can use the OpenSSL `req -x509` to generate a key and self-signed certificate for the CA. The syntax of the command is as follows:

```
openssl req -new -newkey rsa:KeySize -x509 -keyout KeyFile -out CertFile \
          -config ConfigFile
```

Where:

`KeySize`: is the desired size in bits of the key.

`KeyFile`: is the name of the file to which the key will be stored.

`CertFile`: is the name of the file to which the certificate will be stored.

`ConfigFile`: is the name of the file containing the configuration. In Blackboard you can find an example configuration file `openssl.cnf`.

Note that there are other options of the command than the above. For instance, one can choose the hashing algorithm to be used in the signing by adding, e.g.`-sha256`, `-sha512`, `-md5`, etc.

Using the configuration file (`openssl.cnf`) requires creating some directories and files. After copying the configuration file into your current directory, you need to create several sub-directories as specified in the configuration file (which can be found under the [CA default] section in the file):

```
dir             = ./demodir           # Where everything is kept
certs           = $dir/cert           # Where the issued certs are kept
crl_dir         = $dir/crl            # Where the issued crl are kept
database        = $dir/index.txt      # database index file
new_certs_dir = $dir/newcerts         # default place for new certs
serial          = $dir/serial         # The current serial number
```

Towards that end, under your current directory create a directory with the name specified in `dir`. Then under the newly created directory create the above 3 directories (i.e. `cert`, `crl` and `newcerts`). Also, along with those 3 subdirectories, you need to create 2 files: `index.txt` and `serial`. The file `index.txt` can be left empty, whereas in the file `serial` put a single number in string format (e.g. 1000).

You can also specify the default hash function to be used by modifying the following line in the `openssl.cnf` file:

```
default_md      = md5             # which md to use
```

Where you can replace `md5` with any other supported hash function, e.g. `sha256`.

When you run the above command, you will be prompted for some information (e.g. Country, Organisation Name, etc.) and a password. You need to remember the chosen password as this will be needed every time you need this CA to issue a certificate. For the other requested details, you can fill them however you want.

To view the content of a certificate, you can use the following command:

```
openssl x509 -in CertFile -noout -text
```

Where:

`CertFile`: is the name of the file containing the certificate.

## 2.3 Task 3: Using your CA to Issue Certificates

In this task you will learn how to use a CA to issue certificates to servers.

### 2.3.1 Task Requirements

Using the CA you created in the previous task, issue a certificate to a sever. Some related guidelines can be found in Section 2.3.2.

**Important Note:** The server's key you generate here must be 2048-bit RSA key and the used hash function to sign the server's certificate need to be SHA 256. Also, when generating your CSR, you must use the Common Name `UWEFirstNameLastName.com` where you replace `FirstName` and `LastName` with your own first and last names, respectively.

**What to Submit:** You need to include screenshots of the commands/steps you used. Also, you are free to include any interesting observations you have made/learnt from doing the task.

### 2.3.2 Guidelines

The steps one needs to follow when a server would like to obtain a certificate from a CA are as follows:

1. The server generates its key pair.
2. The server submits a Certificate Signing Request (CSR) to the CA to request the certificate.
3. The CA issues the certificate which binds the server's public-key to its identity.

Step 1 was covered in the Public-Key and Digital Signature lab sheet.

To obtain a CSR, the server uses the following command:

```
openssl req -new -key KeyFile -out CSRFile -config ConfigFile
```

Where:

`KeyFile`: is the name of the file containing the (private) server's key.

`CSRFile`: is the name of the file to which the CSR will be stored.

`ConfigFile`: is the name of the file containing the configuration. You can use the file `openssl.cnf` used earlier.

When you run the above command, you will be prompted for some information (e.g. Country, Organisation Name, etc.) and the password associated with your server's key file. Please remember that for this task the Common Name in the CSR must be in the form `UWEFirstNameLastName.com` as detailed above. The rest of the requested information in the CSR can be filled however you want.

When the CA receives the server's certificate request (CSR), after verifying its identity, the CA will generate the certificate for the server. This can be achieved by the following command:

```
openssl ca -in CSRFile -out CertFile -cert CACertFile -keyfile CAKeyFile \
          -config ConfigFile
```

Where:

`CSRFile`: is the name of the file containing the server's CSR.

`CertFile`: is the name of the file where the server's certificate will be stored.

`CACertFile`: is the name of the file containing the CA's own certificate.

`CAKeyFile`: is the name of the file containing the CA's key which will be used to sign the server's certificate.

`ConfigFile`: is the name of the file containing the configuration. You can use the `openssl.cnf` file for this.

To avoid OpenSSL rejecting to sign the server's certificate if the CA details do not align with those of the server, e.g. they belong to different countries, change the following line in the configuration file:

```
policy          = policy_match
```

to

```
policy          = policy_anything
```

## 2.4 Task 4: Deploying Certificates in OpenSSL HTTPS Server

In this task you will explore how PKI and digital certificates can be used to secure the web. In particular, you will learn how to deploy digital certificates in OpenSSL web server (`https://www.openssl.org/docs/man1.0.2/man1/openssl-s_server.html`).

### 2.4.1 Task Requirements

You are required to link the server's certificate you created in the previous task to the server's domain and report the behaviour of the web browser before and after trusting the issuing CA. Follow the guidance in Section 2.4.2 and report your findings.

**Important Note:** As stated in Section 2.4.2, you must update the file `index.html` to replace `FirstName` and `LastName` with your own first and last names, respectively. No marks will be given for this task if you do not do that.

**What to Submit:** You need to submit a screenshot of the commands/steps you used, and your observations and findings.

### 2.4.2 Guidelines

To resolve the IP address of the server (i.e. `UWEFirstNameLastName.com`) for which you have issued your server's certificate, you need to add the following line to the file (`/etc/hosts`):

```
127.0.0.1       UWEFirstNameLastName.com
```

Note that to edit `/etc/hosts`, you need to be a super user, e.g. to edit the file, you can use the following command from a terminal :

```
sudo gedit /etc/hosts
```

Of course, you can use any other preferred editor than `gedit`, e.g. `nano`, to edit the file if you wish.

The next step is to launch the OpenSSL web server using your server's certificate. This can be achieved by performing the following steps:

1. **Combining your Server's Key & Certificate:**
   The aim here is to combine the server's certificate (you created in the previous task) and the corresponding server's key into one file. This can be achieved by the following command:

   ```
   cat ServerKeyFile ServerCertFile > NewFileName
   ```

   Where:

   `ServerKeyFile`: is the name of the file containing your server's key.

   `ServerCertFile`: is the name of the file containing your server's certificate.

   `NewFileName`: is the name of the file where the combination of key and certificate of the server will be stored. You can choose whatever name you wish but it is a good idea to make the extension of the file `.pem`.

   Note that if you are running the above command from a directory different from that where the first two files are stored, you need to provide the full path to those two files.

2. **Launching the OpenSSL Web Server:**
   The aim here is to launch the OpenSSL web server using the combined key and certificate file from the previous step. This can be achieved by the following command:

   ```
   openssl s_server -cert NewFileName -WWW
   ```

   Where `NewFileName` is the same file name as that you used in the previous step. Note that the default port on which the server will listen is 4433. This can be overridden by adding the option `-accept PortNo` to the above command, where `PortNo` is the port number you wish the server to listen on instead of port 4433. Also, note that you need to the leave the window from which you ran this command open so that the server is still running.

3. **Accessing the Server's Web Page:**
   Assuming you have finished the first 2 steps, download the the simple web page `index.html` from Blackboard and save it to the same directory from within which you have executed the command in the previous step. Then using an editor of your choice, e.g. `nano`, `gedit`, etc., edit `index.html` and replace

FirstName and LastName with your own first and last names, respectively, and save the file.

Now using a web browser of your choice, e.g. Firefox, browse the url:

```
https://UWEFirstNameLastName.com:PortNo/index.html
```

where you replace PortNo with the actual port NO you used to launch the web server in the previous step. Also, FirstName and LastName are your first and last names, respectively.

If you followed the above steps correctly, the browser should display an error message along the lines `Potential Security Risk Ahead` or `The certificate is not trusted because the issuer certificate is unknown`. This is due to the fact that the CA your created is not among the authorities your browser trusts.

The next step is for you to add your CA to the list of the trusted authorities by the browser. For instance, in FireFox, you can add your CA as a trusted certification authority by choosing `Preferences` → `Privacy & Security` → `View Certificates` → `Authorities` → `Import` from the FireFox menu and then navigating to your CA's certificate file that you have created in Task 2. Note that unless your CA certificate file has the extension `.pem`, you need to select `All Files` rather than `Certificate Files` when navigating to the file. Tick `This certificate can identify web sites` when adding the authority as a trusted one. Note that these steps assume you are using the version of FireFox on the UWE VM. The latter step might differ if you are using another browser or a different version of FireFox.

Now if you navigate to the same url, you should not see the error message any more and instead you should see the simple web page which contains your name.

## 2.5 Task 5: Deploying Certificates in Apache HTTPS Server

In this task you will explore how PKI and digital certificates can be used to secure the web. In particular, you will learn how to deploy digital certificates in the Apache web server.

### 2.5.1 Task Requirements

Similarly to the previous task, you are required to deploy your server's certificate you created in Task 3 in the Apache web server (installed on the UWE VM) and report the behaviour of the web browser. Follow the guidance in Section 2.5.2 and report your findings.

**What to Submit:** You need to submit a screenshot of the commands/steps you used, and your observations.

### 2.5.2 Guidelines

Rather than using the simple OpenSSL HTTPS server, here we will use the Apache web server which is already installed on the UWE VM. We need to configure the Apache server so that it can link the server's private key and certificate to its domain. Also, we need to inform the Apache server where the associated HTML files are stored.

In the folder `/etc/apache2/sites-available`, you will find the 2 files: `000-default.conf` and `default-ssl.conf`
Carefully edit the file `default-ssl.conf` (using an editor of your choice) to add the following entries for your server's website:

```
<VirtualHost *:443>
ServerName UWEFirstNameLastName.com
DocumentRoot Folder
DirectoryIndex index.html

SSLEngine On
SSLCertificateFile ServerCertFile
SSLCertificateKeyFile ServerKeyFile
</VirtualHost>


<VirtualHost *:80>
ServerName UWEFirstNameLastName.com
DocumentRoot Folder
DirectoryIndex index.html
</VirtualHost>
```

Where `Folder` is the path of the folder which contains the HTML files for the server's website. Usually such folders are under `/var/www/`, so create a folder there and copy the `index.html` you used in the previous task to the new folder. `ServerCertFile` and `ServerKeyFile` are the files (including the full path) containing the server's certificate and key files, respectively. Note that to edit the above file your need to be a super user. Also, it might be a good idea to backup the file before you edit it just in case things go wrong and you need to revert to the original version.

In order for these entries to be recognised by the Apache server, you need to execute the following steps from the command-line:

1. Testing the Apache configuration file:
   This can be done by executing the following command:

   ```
   sudo apachectl configtest
   ```

2. Enabling the SSL module:
   This can be done by executing the following command:

   ```
   sudo a2enmod ssl
   ```

3. Configuring Apache for HTTPS:
   This can be done by executing the following command:

```
sudo a2ensite default-ssl
```

4. Restarting the Apache Server:
   This can be done by executing the following command:

```
sudo service apache2 restart
```

If you followed all the above steps correctly, now launch the web browser of your choice and browse the following 2 urls and report your findings:
`http://UWEFirstNameLastName.com`
`https://UWEFirstNameLastName.com`

Again, FirstName and LastName are replaced with those of your own.

## 2.6 Task 6: Impersonating a Website (Man-in-the-Middle)

In this task you will explore how PKI could help prevent man-in-the-middle attacks.

### 2.6.1 Task Requirements

Follow the steps in the guidelines in Section 2.6.2 and report your findings.

**What to Submit:** You need to submit a screenshot of the commands/steps you used and your observations/findings.

### 2.6.2 Guidelines

Here you will attempt to impersonate the UWE website `www.uwe.ac.uk` so that instead of the visitor being redirected to the genuine IP address, they will be redirected to our fake server. In particular, we will redirect the visitor to the server's web page you used in the previous task. Please follow the below steps and report your findings:

1. Add the following line to the file (`/etc/hosts`):

```
127.0.0.1       www.uwe.ac.uk
```

   Remember that to edit the above file you need to be a super user.

2. As in the previous task, add the 2 entries for `www.uwe.ac.uk` to `default-ssl.conf`. The entries you need to add are identical to those you added for your server in the previous task. The only difference is that instead of the server name being `UWEFirstNameLastName.com` as it was for your sever, it will be `www.uwe.ac.uk`. The rest of the details will remain the same as they were for your server in the previous task. More precisely, add the following entries to the file:

```
<VirtualHost *:443>
ServerName www.uwe.ac.uk
DocumentRoot Folder
DirectoryIndex index.html
```

```
SSLEngine On
SSLCertificateFile ServerCertFile
SSLCertificateKeyFile ServerKeyFile
</VirtualHost>


<VirtualHost *:80>
ServerName www.uwe.ac.uk
DocumentRoot Folder
DirectoryIndex index.html
</VirtualHost>
```

Remember that to edit the file you need to be a super user.

In order for these entries to be recognised by the Apache server, you need to run the four required commands as we did in the previous task.

3. If you finished the above steps, visit `http://www.uwe.ac.uk` and `https://www.uwe.ac.uk` (from your VM web browser) and report your findings.

If you have followed the steps correctly, the browser should complain that for the HTTPS version the url does not match the domain for which the certificate was issued and hence will foil the impersonation attempt.

## 2.7 Task 7: Alternative Approaches to (Certificate-Based) PKI (Research Task)

It is well-known that (certificate-based) public-key infrastructure, which relies on using certificates form trusted authorities to ensure authenticity of entities' public keys, has some inherent limitations, such as the high trust placed in the certificate authorities and the overhead associated with revoking certificates. This task requires you to undertake some research regarding alternative approaches to (certificate-based) PKI for managing and distributing public keys.

### 2.7.1 Task Requirements

Write a short report (1000 words max) discussing alternative approaches to (certificate-based) PKI and how they might overcome some of the inherent PKI limitations. Your report should include relevant references from the literature and your discussion should cover advantages and disadvantages of the alternative approaches.

# 3 What to Submit

You need to submit a detailed report, with screenshots, to describe what you have done, what you have observed, and how you reached your conclusion/answer for each task. The format of the report is up to you but you must adhere to the requirements mentioned in What to Submit at the end of the tasks. The report should be of a

professional standard. You need to provide explanation to the observations that are interesting or surprising. Please also list any important code snippets you have written followed by explanation. Simply attaching code or screenshots without any explanation will not receive credits. The report must demonstrate your understanding of the subject and material and not just be a log of your actions. *All screenshots in the report must have your student number and date stamp in the user prompt. Failure to include these details in the screenshots will invalidate the report and receive a mark of zero.*

## 3.1 Plagiarism

This is an assessed lab sheet and while it is acceptable to discuss your assignment with your peers as per the university rules, this assignment is intended as an individual assignment. Submissions that are substantially similar will be subject to investigation according to university regulations and any proven cases will be dealt with according to the regulations. More details can be found here `http://www1.uwe.ac.uk/students/academicadvice/assessments/assessmentoffences.aspx`

# 4 Marking Criteria

| | 0-29% | 30-39% | 40-49% | 50-59% | 60-69% | 70-84% | 85-100% |
|---|---|---|---|---|---|---|---|
| **Tasks 1-6 (65%)** | Little to no attempt/Serious gaps or errors in understanding the topic | Some tasks completed with major omission/Some evidence of understanding the topic with major errors or gaps | Most tasks completed but with minor omissions/Evidence of understanding the topic but with minor errors or gaps | All tasks completed in full. Evidence incomplete or unclear in places/Adequate understanding of the topic | All tasks completed in full. Evidence of a good standard to detail tasks/Clear understanding of topic | All tasks completed in full. Excellent use of evidence to detail tasks/Thorough and comprehensive understanding of topic | All tasks completed in full. Highly reflective use of evidence to develop argument /Impressive and original depth of understanding of topic |
| **Task 7 (25%)** | Little to no discussion | Poor analysis. Demonstrates little or no insight into the problem | Below-average analysis. Analysis is lacking in most aspects | Adequate analysis of relevant works but lacks depth; demonstrates some insight into the problem | Good analysis of relevant works but could be more critical; demonstrates good insight into the problem. Good use of sources | Very good analysis of relevant works; demonstrates excellent insight into the problem. Good use of sources and all sources are appropriately referenced | Outstanding analysis of relevant works; demonstrates outstanding insight into the problem and fully covers all aspects. Excellent use of sources and all sources are appropriately referenced |
| **Report Presentation (10%)** | Very poor presentation | Weak presentation | Has not followed required conventions; poor proofreading | Partially follows required practices; some issues to be addressed e.g., typos, punctuation | Follows required presentational practices; a few typos/errors in punctuation or grammar | Excellent presentation: typos/errors in punctuation etc. are rare | Excellent presentation |

# 5 Document Revision History

| Version | Date | Changes |
|---|---|---|
| 1.0 | 03rd April 2021 | Initial Release |