1. ***Aims and Objectives***The major aims of this assignment are that you demonstrate that you can:

* Understand and make use of the Java security classes
* Develop a secure application using appropriate tools
* Analyse a typical application for security threats
* Communicate the nature and potential of those threats
* Appraise the usefulness of various security techniques for particular situations

1. ***Working Arrangements***

Work must be undertaken in pairs. Both members of the pair will receive the same mark unless evidence can be produced that demonstrates that the mark should be allocated proportionately. For this reason, it is strongly recommended that you keep a log of meetings and discussions. **You will demonstrate your work during the classroom sessions that follow the assignment submission date.**

1. ***Introduction***

This is principally a programming assignment in which you will amend the provided Encryption Server System to introduce authentication using certificates, key exchange for encryption and message transfer using SSL (Secure Socket Layer). You will be learning:

* Pass word-based encryption
* *keytool* (command line utility) to generate/manage keys and certificates
* Java JCE
* JCE APIs for key exchange and encryption
* JCE APIs to create and sign certificates programmatically
* JSSE (Java Secure Socket Extension) to carry out secure networking

In brief, the assignment consists of the following tasks:

1. Implement message/file encryption using a symmetric key
2. Implement secure key exchange
3. Build and use a public key infrastructure using X509 certificates
4. Secure all traffic using SSL
5. Use password authentication initially to procure the client certificate
6. Use SSL client certificates to successfully authenticate the client to the server
7. Report on the work done
8. ***The Encryption Server Set Up***

For the assignment, the Encryption Server system consists of three entities:

* Clients (CL),
* The Encryption Server (ES), and
* A Certificate Authority (CA).

A text file is also provided (authdb.txtin the assignment code’s *Files* folder), which contains a pre-registered username and password. The password in this file will be used for authentication to the encryption server and the CA. **You will not use the CA in all of tasks.**

The system exists in a hypothetical environment where a client (CL) can send a plain text file to a server (ES) for encryption and storage/archiving[[1]](#footnote-1). You have been provided with a basic version of this system with implementations of the relevant system entities (server, client, CA). The system is currently insecure and, through the tasks listed in this assignment, you will be securing the system incrementally to meet most of the security goals you have learned in the previous semester. The main functionality will remain constant; the system’s job is to allow clients to send files to the server for encryption and storage[[2]](#footnote-2).

You will be provided with a starter version of the code in a zipped NetBeans project titled BasicESS. You can extract this zip file on your disk and then import it in NetBeans. Further detailed instructions regarding BasicESS will be provided in the practical sessions following the announcement of the assignment.

The Encryption Server System code is organised into three main packages:

* uk.ac.uwe.cans.es *(es)*
* uk.ac.uwe.cans.cl *(cl)*
* uk.ac.uwe.cans.ca *(ca)*

It should be compiled with Java versions 6+. The es package contains the class files that implement the server functionality. The clpackage contains classes that implement the client’s functionality and the ca package contains the classes related to the CA’s functionality.

1. ***Task Details***

*NOTE: A* CODE *or a* REPORT *tag, which specifies where/how the solution of the task is to be provided, follows each task description.* CODE *means you have to implement the solution programmatically, and* REPORT *means you have to document the solution in your report.*

*Each task tagged with* CODE *will be saved as a NetBeans project with a specific name, note the instructions given against each* CODE *task below and follow to the letter!*

* 1. **Analysis of the basic version of the Encryption Server System**

Analyse and list the security flaws/vulnerabilities in the basic version of the Encryption Server System[[3]](#footnote-3) with respect to the goals of computer and network security. [REPORT]

**3 marks**

* 1. **Implement (trivial) password-based authentication**
     1. Extend the BasicESS project to implement password-based authentication between the CL and the ES. Before a CL can transfer a plaintext file to the ES, it must authenticate using its username and password combination and the ES should verify the credentials from the authdb.txt file. If authentication is successful, the CL can transfer the file to the ES. [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask2.*

**10 marks**

* + 1. Analyse and comment on any improvements that may have been introduced in the system through the extension you carried out in this task. Also reiterate the security vulnerabilities that still exist and ones that may have been introduced in the system through the extension carried out in this task. [REPORT]

**2 marks**

* 1. **Improving the password-based authentication**
     1. Extend the BasicESS project to implement password-based authentication between the CL and the ES (as in the previous task), such that passwords are not sent in their original form. Use the concept of hashing. [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask3.*

**10 marks**

* + 1. Analyse and comment on any improvements that may have been introduced in the system through the extension you carried out in this task. Also reiterate the security vulnerabilities that still exist and ones that may have been introduced in the system through the extension carried out in this task. [REPORT]

**3 marks**

* 1. **Symmetric Encryption and Key Exchange**
     1. Implement a Diffie-Hellman based key-exchange between the CL and the ES and use the symmetric key for encrypting the file being transferred from the CL to the ES[[4]](#footnote-4). [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask4.*

**15 marks**

* + 1. Analyse and comment on the additional security features that have been incorporated into the system after successful implementation of this task. Also reiterate the security vulnerabilities that still exist in the system. [REPORT]

**3 marks**

* 1. **Setting up Public Key Infrastructure**

The CA should have a public/private key pair, which can be generated offline (no coding required) using keytool. The keytool is used to generate a *keystore* for each entity in the system (CA, ES, CL)[[5]](#footnote-5). Here is the sequence of actions that need to be performed for this task (and before the subsequent tasks can be carried out):

1. Generate a public/private key pair for the CA. (The public key of the CA is self-signed). Generate public/private key pairs for the ES and the CL. Document the steps (commands, input, output, screenshots, etc.). [REPORT]

**3 marks**

1. Export the CA’s self-signed certificate to a file and import it in all the other keystores (i.e. ES and CL keystores).[REPORT]

**3 marks**

1. Write a separate program that takes in ES keystore and its associated password and signs the ES public key with the CA private key. You should also store the signed certificate in an appropriate location. [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask5.*

**10 marks**

At the end of the above steps you will have the ES with a public key signed by the CA. All keystores will have the CA self-signed certificate, which indicates that ES and CL trusts the CA. *Note that the CL public keys are not yet signed.*

* 1. **SSL based communication and signed certificates**

Extend the system to incorporate SSL based communication between the ES and CL. When a CL starts up, it should first connect to the CA[[6]](#footnote-6). The CL transmits its username, password and public key to the CA[[7]](#footnote-7). The CA verifies the username and password against the authdb.txt file. If this verification is successful, the CA generates a certificate, by signing the public key of the client with the CA’s private key, and sends it back to the CL. The CL should save the certificate at an appropriate location (describe the location you use clearly in your report). [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask6.*

**15 marks**

* 1. **Authentication via Certificates**

1. In this task, the CL should use the newly issued certificate from the CA, to connect to the ES over SSL and carry out the file transfer. Note that this time around the SSL connection will provide authentication in both directions and hence you will not need any password-based authentication between the client and the server. [CODE]

*Note: Once you have completed this task, duplicate the current NetBeans project and rename it to ESSTask7.*

**15 marks**

1. Analyse and comment on the level of security that has now been introduced into the Encryption Server System i.e. in comparison to previous tasks and an overall assessment with respect to security goals. [REPORT]

**3 marks**

1. A sample text file is provided in the assignment code’s *Files* folder (Plain.txt). [↑](#footnote-ref-1)
2. The file encryption at the server is already implemented through a Password Based Encryption technique that uses a hardcoded server password. Storage is not - and does not - need to be implemented. [↑](#footnote-ref-2)
3. The basic version of the Encryption Server Systems is provided to you as a NetBeans project named BasicESS. [↑](#footnote-ref-3)
4. You do not need to authenticate i.e. you can start from the BasicESS project. If you want, you are free to extend ESSTask2 or ESSTask3 to implement this task. [↑](#footnote-ref-4)
5. For your convenience, name the keystores: ca-keystore, es-keystore and cl-keystore. [↑](#footnote-ref-5)
6. At this point the CL does not have a signed certificate for making the fully authenticated SSL connection. [↑](#footnote-ref-6)
7. It is up to you if you want to send the password in clear or hashed. [↑](#footnote-ref-7)