

# Attribute-Based Encryption and Key Agreement Protocol for Data Security in EHR Systems - Configuration Manual

MSc Research Project  
Cloud Computing

Murphy Elo  
Student ID: x21220263

School of Computing  
National College of Ireland

Supervisor: Rashid Mijumbi

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Murphy Elo
<b>Student ID:</b>	x21220263
<b>Programme:</b>	Cloud Computing
<b>Year:</b>	2023
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Rashid Mijumbi
<b>Submission Due Date:</b>	14/08/2023
<b>Project Title:</b>	Attribute-Based Encryption and Key Agreement Protocol for Data Security in EHR Systems - Configuration Manual
<b>Word Count:</b>	780
<b>Page Count:</b>	6

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

**ALL** internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

<b>Signature:</b>	
<b>Date:</b>	18th September 2023

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:**

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
<b>Attach a Moodle submission receipt of the online project submission</b> , to each project (including multiple copies).	<input type="checkbox"/>
<b>You must ensure that you retain a HARD COPY of the project</b> , both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	<input type="checkbox"/>

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

<b>Office Use Only</b>	
Signature:	
Date:	
Penalty Applied (if applicable):	

# Attribute-Based Encryption and Key Agreement Protocol for Data Security in EHR Systems - Configuration Manual

Murphy Elo  
x21220263

## 1 Introduction

This configuration manual will guide you in understanding the system software setup, and the necessary steps to implement this research project: Attribute-Based Encryption and Key Agreement Protocol for Data Security in EHR Systems.

## 2 System Configuration

These are the system software setup used for implementation:

- Docker 4.21.1 for running Django and Flask containers
- Python 3.3 to run Charm's framework on Flask for Ciphertext - Attribute-Based Encryption (CP-ABE) protocol.
- Python 3.10 for the EHR system on Django
- Kivy 2.2.1 for developing the mobile app.

## 3 Software Installation

### 3.1 Docker Installation

Docker has been used to provide isolated containers for the Django and Flask environments. The EHR system runs on Django, and the proposed algorithms run on the Flask framework. Below are the steps to install Docker and run the services using the given Docker Compose file in the downloaded code folder.

**Step 1: Install Docker:** To run the Docker Compose configuration, you need to have Docker installed on the system. To install Docker, open the terminal and use the following command in the Figure 1:

**Step 2: Create Directory Folder:** Create a directory folder and place the Django and Flask application code along with their respective Dockerfiles in the directory. The

```
sudo apt update
sudo apt install -y docker.io docker-compose
```

Figure 1: Install Docker

```
my-docker-project/
├── ehr-main/
│   └── Dockerfile.django
├── flask_abe/
│   └── Dockerfile.flask
└── docker-compose.yml
```

Figure 2: Directory Structure

directory structure should look like the Figure 2.

**Step 2: Configure Docker-compose.yml:** Now CD into the project directory created and run the following command to start the Docker service:

```
docker-compose up -d.
```

The `-d` flag runs the services in the background. The Django and Flask applications should now be running. You can access them using the following URLs:

```
Django: http://localhost:8000
Flask: http://localhost:5000
```

## 3.2 Kivy Installation

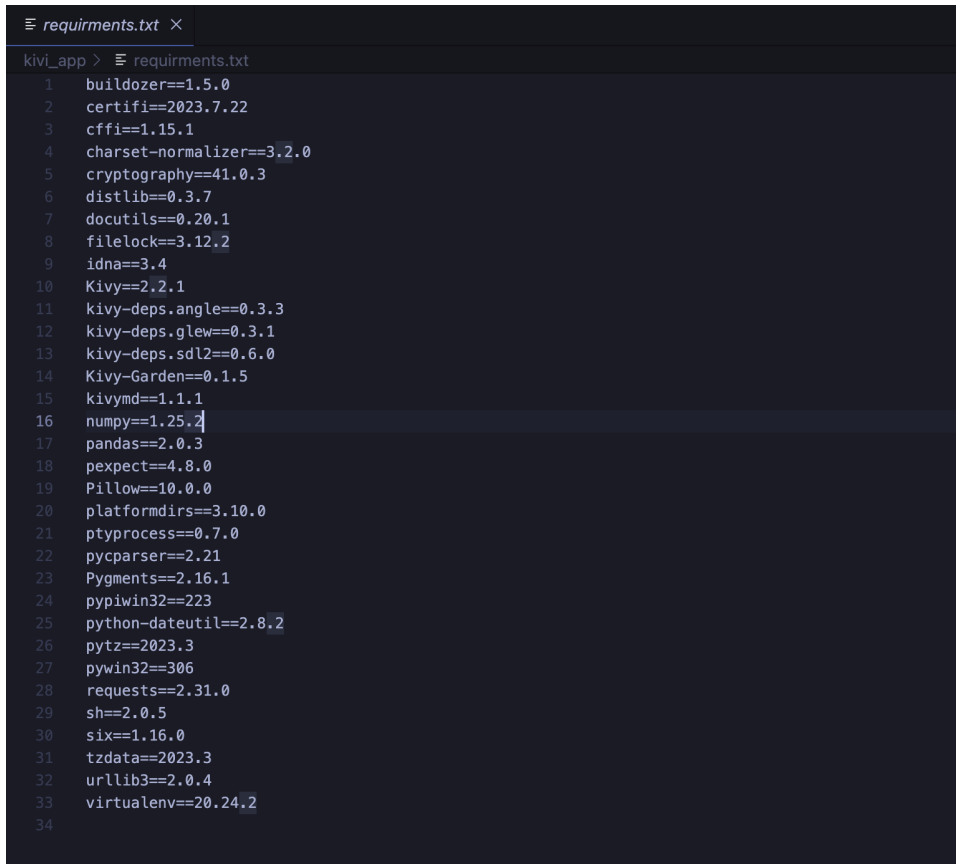
Kivy was used in this project to build the mobile app as the third party system. It is an open-source Python App Development Framework. You will need to install Kivy in the development environment. Use the following command to install Kivy:

```
pip3 install kivy
```

### 3.2.1 Required Kivy Dependencies

After installation, run the following command to install the requirements.txt file to have the mobile app running:

pip install requirements.txt



```
requirements.txt
kivi_app > requirements.txt
1  buildozer==1.5.0
2  certifi==2023.7.22
3  cffi==1.15.1
4  charset-normalizer==3.2.0
5  cryptography==41.0.3
6  distlib==0.3.7
7  docutils==0.20.1
8  filelock==3.12.2
9  idna==3.4
10 Kivy==2.2.1
11 kivy-deps.angle==0.3.3
12 kivy-deps.glew==0.3.1
13 kivy-deps.sdl2==0.6.0
14 Kivy-Garden==0.1.5
15 kivymd==1.1.1
16 numpy==1.25.2
17 pandas==2.0.3
18 pexpect==4.8.0
19 Pillow==10.0.0
20 platformdirs==3.10.0
21 ptyprocess==0.7.0
22 pycparser==2.21
23 Pygments==2.16.1
24 pywin32==223
25 python-dateutil==2.8.2
26 pytz==2023.3
27 pywin32==306
28 requests==2.31.0
29 sh==2.0.5
30 six==1.16.0
31 tzdata==2023.3
32 urllib3==2.0.4
33 virtualenv==20.24.2
34
```

Figure 3: Requirements File for Mobile App

## 4 Implementation

The data modification algorithm scheme can be seen implemented in the `abe.py` file that can be found in the `flask` folder of the code:

To provide secure communication between the Django-built EHR system and Kivy-built mobile app, ECDH (Elliptic Curve Diffie-Hellman) has been implemented in the `ecdh.py` file that can be found in the `ehr-main` folder of the code, and the `main.py` file in the `kivy` app folder.

## 5 Evaluation

As an evaluation of the result, a mobile app is implemented as a third-party system performing patient health data retrieval from the EHR system.

```

abe.py 3 X
flask_abe > abe.py > ...
1 from flask import Flask, request, jsonify
2 from charm_toolbox.pairinggroup import PairingGroup,ZR,G1,G2,GT,pair
3 from charm_schemes.abenc.abenc_vct14 import EKPabe
4 from charm_core.engine.util import objectToBytes, bytesToObject
5 import logging
6 import base64
7 import json
8 import os
9 import ast
10
11 os.environ['PYTHONUTF8'] = '1'
12
13 logging.basicConfig(level=logging.DEBUG)
14
15 logger = logging.getLogger(__name__)
16 app = Flask(__name__)
17
18
19 group = PairingGroup('MNT224')
20 kpabe = EKPabe(group)
21
22 with open('data.json', 'r') as file:
23     existing_data = json.load(file)
24
25 @app.route("/check")
26 def hello():
27     try:
28         print('test api')
29         group = PairingGroup('MNT224')
30         kpabe = EKPabe(group)
31         attributes = ['ONE', 'two', 'THREE']
32         (master_public_key, master_key) = kpabe.setup(attributes)
33         policy = '(ONE or THREE) and (THREE or two)'
34         secret_key = kpabe.keygen(master_public_key, master_key, policy)
35         print(policy, attributes)
36         print('secret type', type(secret_key))
37         msg = "Some Random Message"
38         cipher_text = kpabe.encrypt(master_public_key, msg.encode("utf-8"), attributes)
39         print('cipher type', type(cipher_text))
40         decrypted_msg = kpabe.decrypt(cipher_text, secret_key)
41         if msg==decrypted_msg:
42             print('msg is same', decrypted_msg)
43         else:
44             print('not same msg')
45

```

Figure 4: Code Snippet for Data Modification

```

edch.py X
ehr-main > ehrapp > edch.py > ...
1 from cryptography.hazmat.primitives import serialization, hashes
2 from cryptography.hazmat.primitives.asymmetric import ec
3 from cryptography.hazmat.primitives.kdf.x963kdf import X963KDF
4 from cryptography.hazmat.primitives import padding
5 from cryptography.hazmat.primitives.ciphers import Cipher, algorithms, modes
6 from cryptography.hazmat.primitives import serialization
7
8 def generate_server_keys_for_server():
9     private_key = ec.generate_private_key(ec.SECP256R1())
10    return private_key
11
12 def generate_shared_secret_on_server(private_key, peer_public_key):
13    peer_public_key = serialization.load_pem_public_key(peer_public_key)
14    shared_key = private_key.exchange(ec.ECDH(), peer_public_key)
15    # shared_key_string = base64.b64encode(shared_key).decode('utf-8')
16    return shared_key
17
18 def encrypt_message_on_server(shared_secret, message):
19    kdf = X963KDF(algorithm=hashes.SHA256(), length=32, sharedinfo=None)
20    derived_key = kdf.derive(shared_secret)
21
22    iv = b'0123456789abcdef'
23    cipher = Cipher(algorithms.AES(derived_key), modes.CFB(iv))
24    encryptor = cipher.encryptor()
25    padder = padding.PKCS7(128).padder()
26    padded_data = padder.update(message.encode()) + padder.finalize()
27
28    ciphertext = encryptor.update(padded_data) + encryptor.finalize()
29    return ciphertext
30
31 def decrypt_message_on_server(shared_secret, ciphertext):
32    kdf = X963KDF(algorithm=hashes.SHA256(), length=32, sharedinfo=None)
33    derived_key = kdf.derive(shared_secret)
34
35    iv = b'0123456789abcdef'
36    cipher = Cipher(algorithms.AES(derived_key), modes.CFB(iv))
37    decryptor = cipher.decryptor()
38
39    decrypted_padded_data = decryptor.update(ciphertext) + decryptor.finalize()
40    unpadder = padding.PKCS7(128).unpadder()
41    decrypted_message = unpadder.update(decrypted_padded_data) + unpadder.finalize()
42
43    return decrypted_message.decode()
44
45

```

Figure 5: Code Snippet for Secure Communication

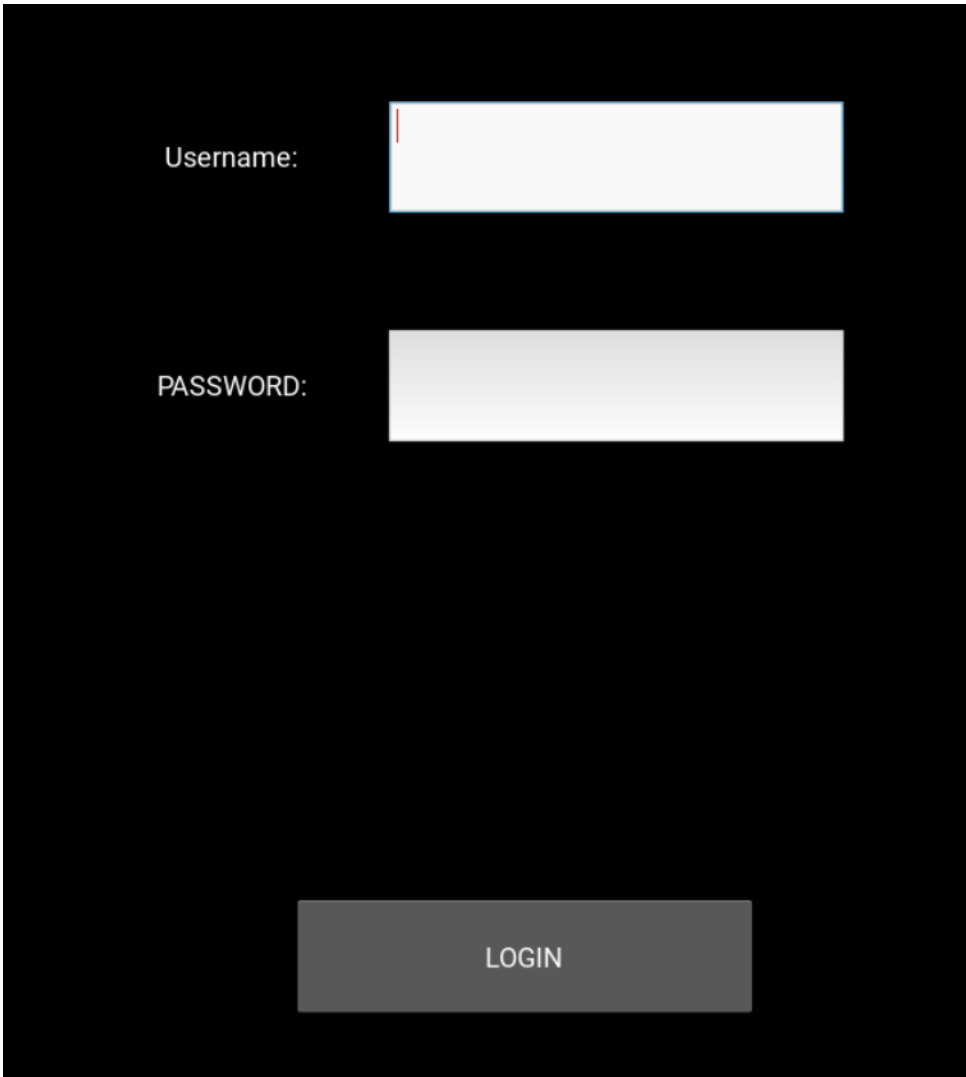


Figure 6: Kivy Mobile App Login



Figure 7: Patient Retrieval Interface

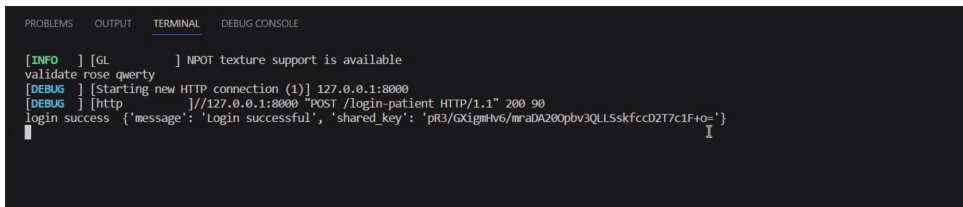


Figure 8: Shared Key for Secure Communication