

# Configuration Manual

MSc Research Project  
Cloud Computing

Sarang Shrikhande  
Student ID: x22202226

School of Computing  
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland  
Project Submission Sheet  
School of Computing



<b>Student Name:</b>	Sarang Shrikhande
<b>Student ID:</b>	x22202226
<b>Programme:</b>	MSc in Cloud Computing
<b>Year:</b>	2023-2024
<b>Module:</b>	MSc Research Project
<b>Supervisor:</b>	Diego Lugones
<b>Submission Due Date:</b>	20/12/2018
<b>Project Title:</b>	Configuration Manual
<b>Word Count:</b>	1350
<b>Page Count:</b>	7

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>	Sarang Shrikhande
<b>Date:</b>	16th September 2024

**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):	

# Configuration Manual

Sarang Shrikhande  
x22202226

## 1 Introduction

This is the configuration manual for my research project, titled as "Enhancing Cloud Security by Integrating Optimized Symmetric Key Encryption Algorithm in Ethereum Blockchain". It will provide a guided approach to implement the proposed system.

It covers the detailed process of establishing secure cloud storage by incorporating the algorithms of symmetric key encryption: Lightweight Bcrypt Symmetric Key Encryption(LBSK) and Blowfish on Ethereum blockchain. The system is based on segregation of data according to their level of sensitivity and implementing the right algorithm of encryption. This solution is able to provide a decentralized strategy for information sharing using AWS services and smart contracts based on Ethereum thus minimizing the reliance on central cloud service providers and enabling users to be in more control of their data.

## 2 AWS Cloud Setup

An AWS Account is needed to perform the steps mentioned below.

The AWS root account is used for demonstration purpose but it is recommended to have an AWS Identity and Access Management (IAM) account with appropriate policies following the principle of least privilege.

Following are the services used for implementation purpose:

- AWS EC2
- AWS S3 bucket
- AWS Lambda function
- AWS ECR (Elastic Container Registry)

After setting up the AWS services, the next step will be to setup the private Ethereum Blockchain on the EC2 instance which will be the decentralized system to store the path of the encryption keys and file path securely on the blockchain.

## 3 Setup Ethereum blockchain on EC2 instance

We install Ganache Truffle suite that will provide us with the Ethereum test net Verma et al. (2022). We download and install the following dependencies on AWS CLI on a EC2 instance and the commands are shown in Figure 1: We first downloaded the Node.js 16.x

setup and installed it using the following commands:

```
sudo curl -sL https://deb.nodesource.com/setup_16.x -o /tmp/nodesource_setup.sh
sudo bash /tmp/nodesource_setup.sh
```

To install Node.js(npm, the Node.js package manager) from the NodeSource repository set up earlier:

```
sudo apt install nodejs
```

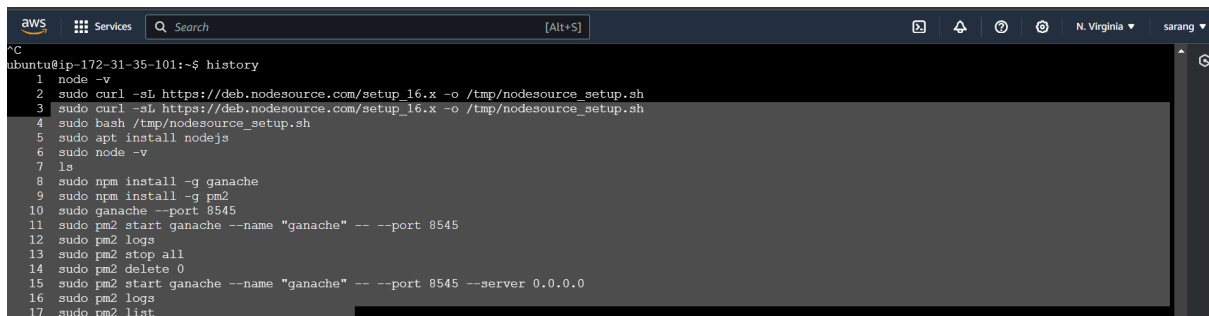
`sudo npm install -g ganache`: This installs Ganache which is a personal blockchain for Ethereum development.

`sudo npm install -g pm2`: It installs PM2 which is a process manager for Node.js apps.

`sudo pm2 start ganache --name "ganache" --port 8545`: It starts the Ganache blockchain using PM2 with the process named "ganache" on port 8545. This command is run directly in the terminal, and hence it blocks the terminal. `sudo pm2 start ganache --name "ganache" --port 8545 --server 0.0.0.0`: Starts Ganache using PM2, with the process named "ganache", running on port 8545 and binding to all network interfaces (0.0.0.0).

`sudo pm2 logs`: Shows the logs for all processes managed by PM2.

PM2 was selected to run "Node.js" processes since it keeps the services running – always necessary for a blockchain network like Ganache. Second, because AWS Lambda employs a serverless approach, it provided an efficient way to deploy the encryption algorithms in that it does not involve a lot of infrastructure.

A screenshot of an AWS Management Console terminal window. The terminal shows a series of commands being executed on an Ubuntu instance. The commands are: 1. node -v, 2. sudo curl -sL https://deb.nodesource.com/setup\_16.x -o /tmp/nodesource\_setup.sh, 3. sudo curl -sL https://deb.nodesource.com/setup\_16.x -o /tmp/nodesource\_setup.sh, 4. sudo bash /tmp/nodesource\_setup.sh, 5. sudo apt install nodejs, 6. sudo node -v, 7. ls, 8. sudo npm install -g ganache, 9. sudo npm install -g pm2, 10. sudo ganache --port 8545, 11. sudo pm2 start ganache --name "ganache" --port 8545, 12. sudo pm2 logs, 13. sudo pm2 stop all, 14. sudo pm2 delete 0, 15. sudo pm2 start ganache --name "ganache" --port 8545 --server 0.0.0.0, 16. sudo pm2 logs, 17. sudo pm2 list. The terminal output shows the version of Node.js and the successful installation of Ganache and PM2. The Ganache process is started and its logs are displayed, showing it is running on port 8545. The PM2 process is listed as running.

```
aws
Services
Search
[Alt+S]
N. Virginia sarang
ubuntu@ip-172-31-35-101:~$ history
1 node -v
2 sudo curl -sL https://deb.nodesource.com/setup_16.x -o /tmp/nodesource_setup.sh
3 sudo curl -sL https://deb.nodesource.com/setup_16.x -o /tmp/nodesource_setup.sh
4 sudo bash /tmp/nodesource_setup.sh
5 sudo apt install nodejs
6 sudo node -v
7 ls
8 sudo npm install -g ganache
9 sudo npm install -g pm2
10 sudo ganache --port 8545
11 sudo pm2 start ganache --name "ganache" --port 8545
12 sudo pm2 logs
13 sudo pm2 stop all
14 sudo pm2 delete 0
15 sudo pm2 start ganache --name "ganache" --port 8545 --server 0.0.0.0
16 sudo pm2 logs
17 sudo pm2 list
```

Figure 1: Setting up Ethereum Blockchain

Figure 1 shows the steps for setting up the Ethereum blockchain on the EC2 instance. The Node.js installation is necessary because Ganache requires Node.js to run the Ethereum test network, which we will use for storing the encrypted data.

Figure 2 shows the smart contract to store path of encrypted file and location of key deployed on the Blockchain. It is saved as a fileStorage.sol file.

## 4 Encrytion Algorithms on AWS Lambda setup

In AWS Lambda, create two Lambda functions called LBSK and Blowfish for the two encryption algorithms.

LBSK was chosen because of its efficiency in providing security to the data by hashing them and it is suitable for cloud environments with resource constraints. Blowfish is a



Figure 2: Smart contract deployed on Ethereum blockchain.png

strong symmetric key algorithm that is good for large amount of data encryption since it is fast enough to process the data though it is not as secure as other algorithms.

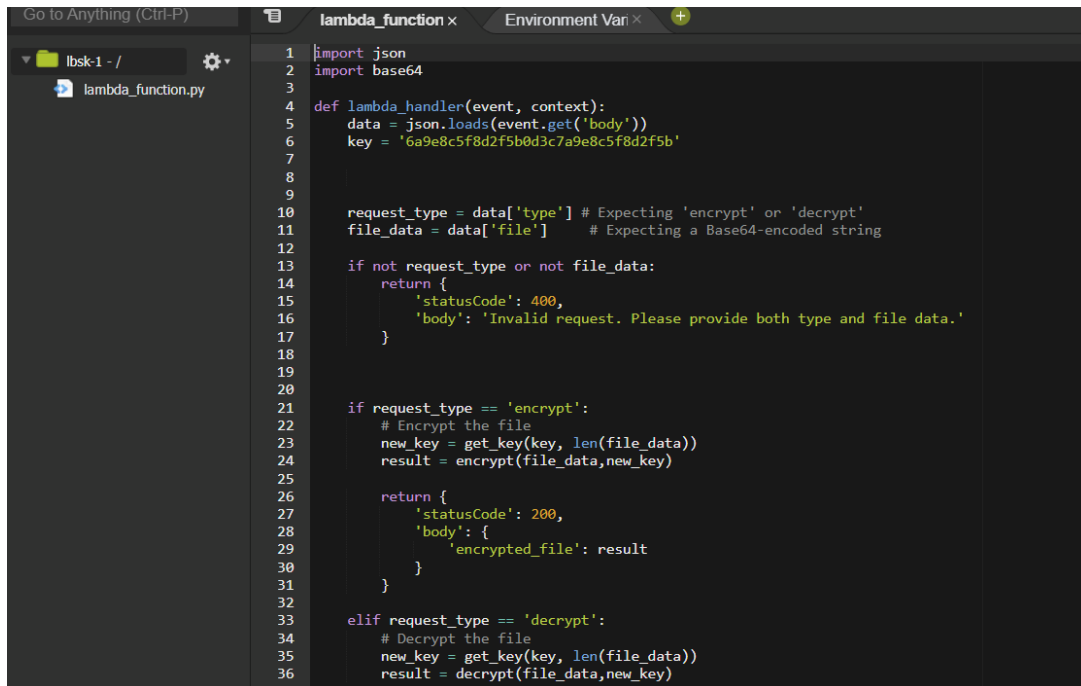


Figure 3: Lambda function for LBSK algorithm

Figure 3 shows the lambda function for the LBSK encryption and decryption algorithm.

Figure 4 shows the lambda function for the Blowfish encryption and decryption algorithm. Since the size of the Blowfish algorithm was greater than 250mb (uncompressed), we created a Docker image to package all the dependencies of the Blowfish algorithm.

Figure 5 shows script to package the Blowfish algorithm as a Docker image. We can simply run the command "docker build".

## 5 Dependencies and Libraries Required

- Pycryptodome: It is a python package offering low level cryptographic primitives. This is used by the Blowfish Encryption algorithm.
- lambda handler: This function is used as an entry point into AWS Lambda.
- Built-in Python libraries:
  - 'os': Packages that offer ways of facing the operating system; in this case they are used for the file operations. The only direct use of external modules is os which is used to handle file paths and write/read files to/from the /tmp directory the only writeable directory in AWS Lambda.
  - 'json': Used for parsing JSON input data from the event 'json.loads()' and to construct json responses 'json.dumps()'.
  - 'base64': used for encoding and decoding Base64 strings.
- truffle/contract (Version: 4.6.31): This is a Truffle library designed specifically to make it easy to work with smart contracts in JavaScript. It aids in the administration of the artifacts embodying the contracts, aids in retrieving instances deployed in the network, and aids in contract manipulation by partly concealing some of the features of Web3.
- web3 (Version: 4.11.1): Web3.js is a JavaScript library with functionality that enables one to deal with an Ethereum blockchain. This makes for an interface to communicate to Ethereum nodes using RPC (Remote Procedure Call). This library is used for tasks like sending the transaction, working with smart contracts, and querying the data of a blockchain.

```
1  from Crypto.Cipher import Blowfish
2  from Crypto import Random
3
4  def encrypt_file(input_file, output_file, key):
5      bs = Blowfish.block_size
6      iv = Random.new().read(bs)
7
8      cipher = Blowfish.new(key.encode(), Blowfish.MODE_CBC, iv)
9
10     with open(input_file, 'rb') as f:
11         plaintext = f.read()
12
13     padding_length = bs - len(plaintext) % bs
14     plaintext += bytes([padding_length]) * padding_length
15
16     ciphertext = iv + cipher.encrypt(plaintext)
17
18     with open(output_file, 'wb') as f:
19         f.write(ciphertext)
20
21 def decrypt_file(input_file, output_file, key):
22     with open(input_file, 'rb') as f:
23         ciphertext = f.read()
24
25     bs = Blowfish.block_size
26     iv = ciphertext[:bs]
27     cipher = Blowfish.new(key.encode(), Blowfish.MODE_CBC, iv)
28
29     plaintext = cipher.decrypt(ciphertext[bs:])
```

Figure 4: Lambda function for Blowfish algorithm

```

1  # Use the official AWS Lambda Python base image
2  FROM public.ecr.aws/lambda/python:3.9
3
4  # Copy your function code and requirements.txt
5  COPY handler.py ${LAMBDA_TASK_ROOT}
6  COPY blowfish_utils.py ${LAMBDA_TASK_ROOT}
7  COPY requirements.txt ${LAMBDA_TASK_ROOT}
8
9  # Install dependencies
10 RUN pip install --no-cache-dir -r requirements.txt
11
12 # Set the CMD to your handler (could also be done as a parameter override outside of the Dockerfile)
13 CMD ["handler.lambda_handler"]
14

```

Figure 5: Script to create a Docker image of Blowfish algorithm

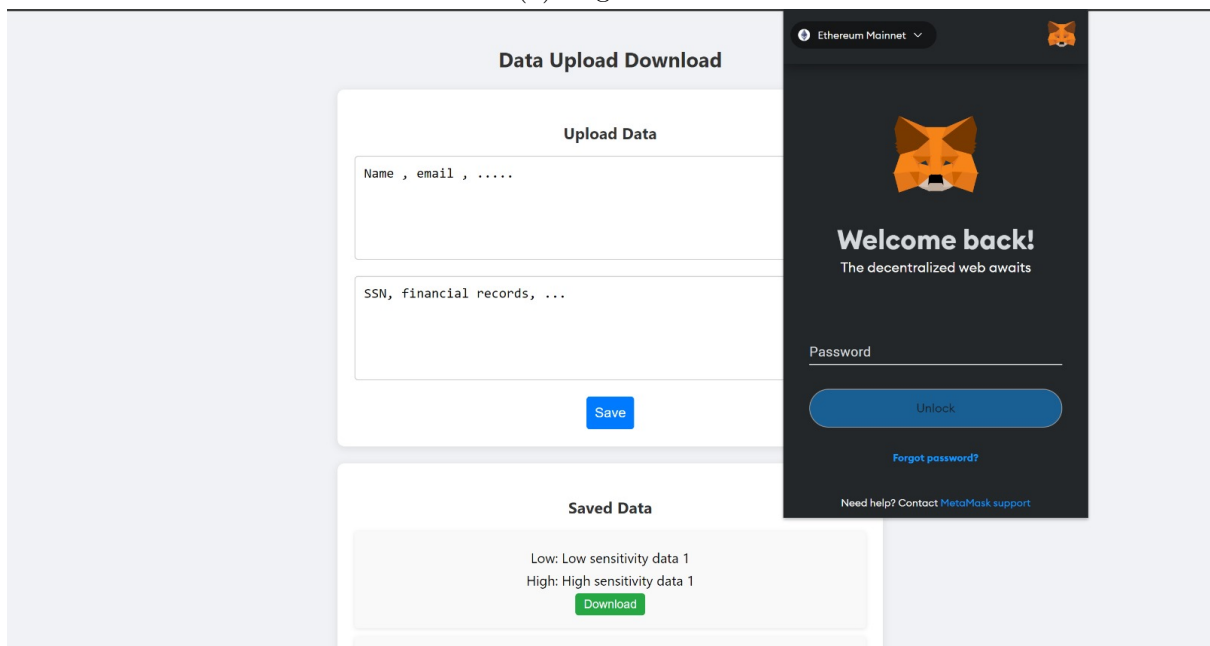
## 6 Front End website and AWS S3 setup

We used React framework to setup the application Gackenheim (2015). To setup our React website on the EC2 instance, on the same EC2 instance used for hosting the Ethereum blockchain, we first connect to the EC2 instance and then update and install the necessary packages: *sudo yum update*. We then install the dependencies using: *npm install*. Then we build the React app using: *npm run build*. Then we copy the static files to the AWS S3 bucket. The Figure 6 shows the Front End website where user can enter data to be encrypted.

We have also setup the Metamask wallet for interacting with the smart contract as a browser extension Lee and Lee (2019). The Ganache Truffle Suite provides us with 10 test accounts with 1000 Test Eth each for carrying out testing seamlessly.



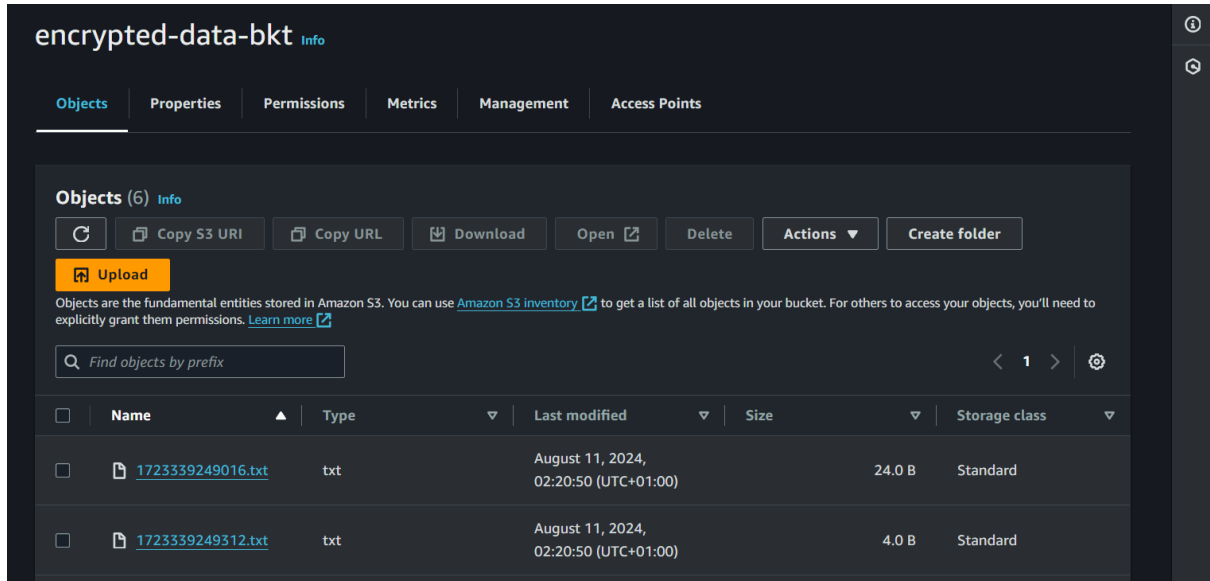
(a) Login view



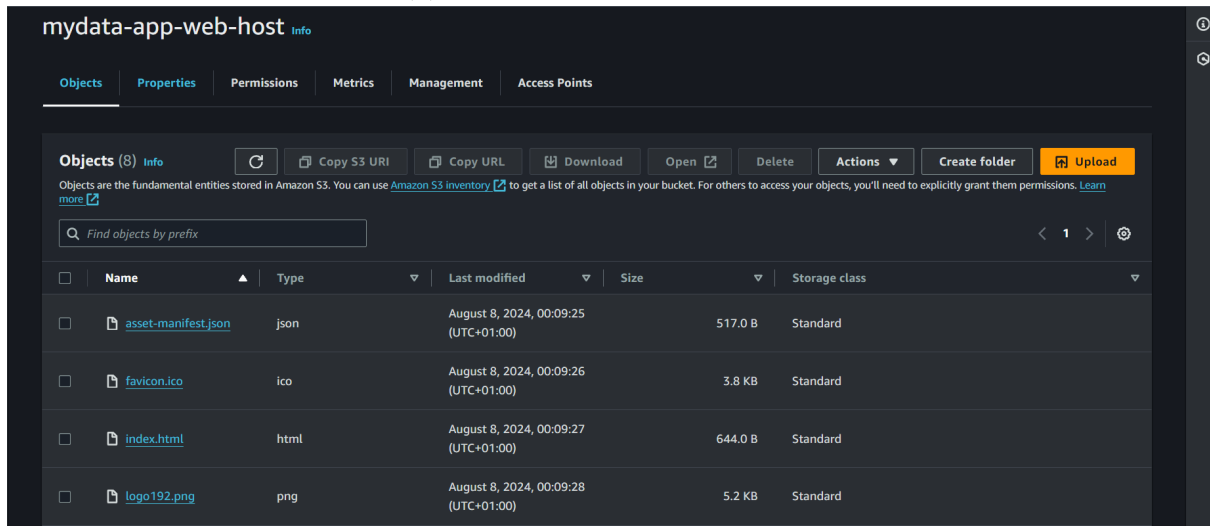
(b) Data to be encrypted

Figure 6: Front end view of the React Application





(a) S3 bucket of Encrypted data



(b) Website files stored in the S3 bucket

Figure 7: S3 buckets having the Encrypted data and hosted website files

The encrypted files as well as the Front-end and Back-end code is stored on the S3 bucket shown in Figure 7

## References

- Gackenheim, C. (2015). *Introduction to React*, Apress.
- Lee, W.-M. and Lee, W.-M. (2019). Using the metamask chrome extension, *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript* pp. 93–126.
- Verma, R., Dhanda, N. and Nagar, V. (2022). Application of truffle suite in a blockchain environment, *Proceedings of Third International Conference on Computing, Communications, and Cyber-Security: IC4S 2021*, Springer, pp. 693–702.