National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Junior Khan Azamah
Student ID: x23110970

School of Computing
National College of Ireland

Supervisor: Prof. Sean Heeney

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Junior Khan Azamah
Student ID:	x23110970
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Prof. Sean Heeney
Submission Due Date:	12/12/2024
Project Title:	Configuration Manual
Word Count:	761
Page Count:	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.

Signature:	
Date:	12th December 2024

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

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

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

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Junior Khan Azamah
x23110970

1 Cloud tier

1.1 Prerequisites

Before beginning the installation process, ensure you have:

- macOS operating system
- Administrator access to your machine
- AWS account with appropriate permissions

1.2 Tools needed

1.2.1 Install Node.js

Install it using Homebrew:

```
brew install node
```

1.2.2 Install Typescript

```
npm install -g typescript
```

1.3 AWS account

- Create IAM user in AWS: [1]
- Install AWS CLI: [2]

1.3.1 Configure AWS Credentials

Set up your AWS credentials:

```
aws configure
```

You'll be prompted to enter:

- AWS Access Key ID

[1] AWS:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users_create.html
[2] AWS:https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

- AWS Secret Access Key
- Default region name (e.g., eu-west-2)
- Default output format (json)

1.3.2 Install AWS CDK

Install the AWS CDK Toolkit globally using npm:

```
npm install -g aws-cdk
```

1.3.3 Bootstrap AWS Environment

Bootstrap AWS environment:

```
cdk bootstrap aws://ACCOUNT-NUMBER/REGION
```

Replace `ACCOUNT-NUMBER` and `REGION` with your AWS account number and region.

1.4 Clone and Set Up Codebase

Clone the repository and navigate to the project directory:

```
git clone https://github.com/Kaid00/Cloud-Tier-Cvs.git
cd Cloud-Tier-Cvs
```

1.4.1 Install Dependencies

Install project dependencies:

```
npm install
```

1.4.2 Synthesizing CloudFormation Templates

Generate CloudFormation templates:

```
cdk synth
```

1.4.3 Deploying Stacks

Deploy all stacks

```
cdk deploy --all
```

2 Blockchain tier

2.1 Tools needed

1. Metamask: [3]
2. Remix IDE: [4]
3. Brave Browser: [5]

2.2 Metamask Initial Configuration

- Open MetaMask
- Click the network dropdown at the top
- Select "Add network"
- Choose "Add network manually"

2.3 Test Networks Configuration

2.3.1 Sepolia (Ethereum Testnet)

- Network Name: Sepolia Test Network
- New RPC URL: [6]
- Chain ID: 11155111
- Currency Symbol: SepoliaETH
- Block Explorer URL: [7]

2.3.2 Binance Smart Chain Testnet

- Network Name: BSC Testnet
- New RPC URL: [8]
- Chain ID: 97
- Currency Symbol: tBNB
- Block Explorer URL: [9]

[3]Metamask: `https://metamask.io`
[4]Remix: `https://remix.ethereum.org/`
[5]Brave: `https://brave.com/download/`
[6]Sepolia rpc: `https://rpc.sepolia.org`
[7]Sepolia etherscan: `https://sepolia.etherscan.io`
[8]Binance: `https://data-seed-prebsc-1-s1.binance.org:8545/`
[9]BSCscan: `https://testnet.bscscan.com`

2.3.3 Polygon zkEVM Testnet

- Network Name: Polygon zkEVM Testnet
- New RPC URL: [10]
- Chain ID: 1442
- Currency Symbol: ETH
- Block Explorer URL: [11]

2.4 Obtaining Testnet Tokens

- Sepolia ETH: [12]
- BSC Testnet BNB: [13]
- Polygon zkEVM Testnet: [14]

3 Experiments

3.1 Experiment / Comparing Civil Record Encryption Times for Various Symmetric Encryption Algorithms

This experiment evaluates the performance of different encryption algorithms (AES, DES, 3DES, RC4) with varying key lengths on a CSV dataset. It measures encryption time and outputs the results to a CSV file for further analysis.

3.1.1 Experiment setup

- Node.js
- VsCode: `https://code.visualstudio.com/download`

3.1.2 Clone and Set Up Codebase

Clone the repository and navigate to the project directory:

```
1 git clone https://github.com/Kaid00/Experiment-1-Encryption-Times.
     git
2 cd Experiment-1-Encryption-Times
```

3.1.3 Install Dependencies

```
1 npm install fs csv-parser crypto-js perf_hooks csv-writer
```

[10] Polygon zkEVM RPC: `https://rpc.public.zkevm-test.net`
[11] Polygonscan: `https://cardona-zkevm.polygonscan.com/`
[12] Sepolia faucet : `https://www.alchemy.com/faucets/ethereum-sepolia`
[13] Bnbchain faucet : `https://www.bnbchain.org/en/testnet-faucet`
[14] Polygon faucet: `https://portal.polygon.technology`

3.1.4 Running the Experiment

In the terminal, navigate to the directory containing the Javascript file, **stressTestEncryptionAlgo.js**

3.1.5 Understanding the Output

The script will print the encryption performance results for each algorithm, key size, and record batch size to the console. It will then export the results to a CSV file named **encryption_benchmark_results.csv**. This file can be opened in any spreadsheet application for further analysis.

3.2 Experiment / Cost Analysis of Deploying and Interacting with the Smart Contract Across Different Blockchains: Sepolia, Polygon zkEVM and Binance Smart Chain

3.2.1 Experiment setup

- Metamask: [15]
- Remix IDE: [16]
- Brave Browser: [17]

3.2.2 Clone and Set Up Codebase

Clone the repository and navigate to the project directory:

```
git clone https://github.com/Kaid00/Blockchain-Tier.git
cd Blockchain-Tier
```

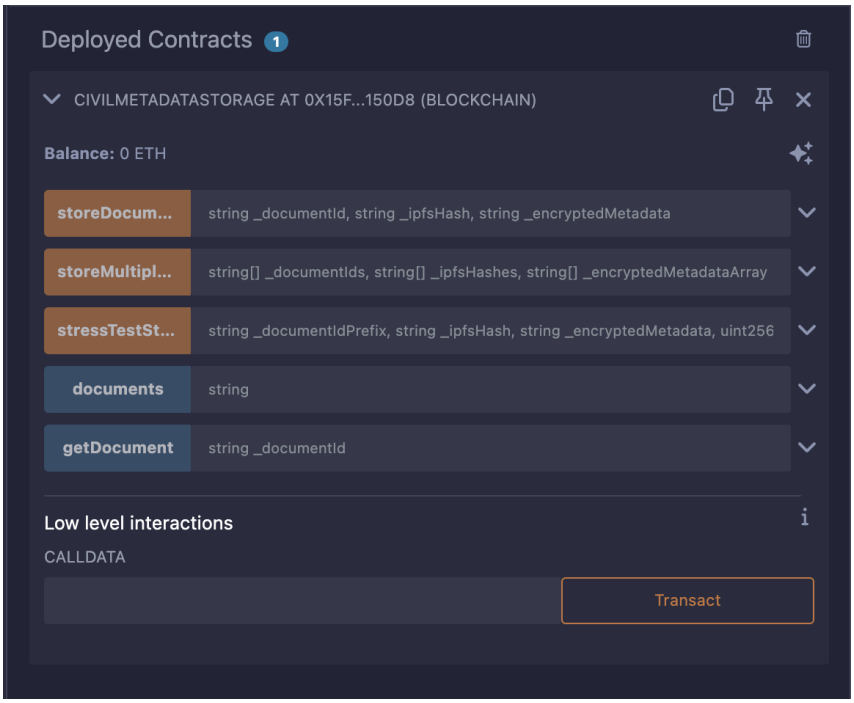
3.2.3 Upload Codebase to Remix

In your Remix IDE, click the **upload folder** button and upload the smart contract code base

3.2.4 Setup Codebase in Remix IDE

1. Click "Upload Folder" button (folder icon)
2. Select the "CivilDataSmartContract" from your computer and upload to Remix
3. Open the CivilMetaDataStressTest.sol file in your IDE, and hit the "Compile" button

[15] Metamask: `https://metamask.io`
[16] Remix: `https://remix.ethereum.org/`
[17] Brave: `https://brave.com/download/`



3.2.5 MetaMask Integration

1. Navigate to "Deploy & Run Transactions" plugin
2. Select "Injected Provider - MetaMask" from Environment dropdown
3. Approve MetaMask connection prompt
4. Verify network selection in MetaMask
5. Connect your Metamask wallet to the desired network (Sepolia, Polygon zkEVM, or Binance Smart Chain).
6. Click the "Deploy" button.

3.2.6 Interact with the Contract

- Once the contract is deployed, in the "Deploy & Run Transactions" plugin, you should see the deployed contracts 3.2.6
- Call the contract's functions to store, retrieve, and stress-test documents.

3.2.7 Data Collection

- Find the Transaction Hash: After deploying or interacting with the contract, note down the transaction hash from Metamask or Remix.
- Search on the Explorer: Paste the transaction hash into the search bar of the relevant block explorer.
- Inspect Transaction Details: The explorer will provide detailed information, including:

- Gas used
- Gas price
- Transaction fee
- Execution time