

Configuration Manual

MSc Research Project
Cloud Computing

Amrutha Shivashankaramurthy
Student ID: 23172151

School of Computing
National College of Ireland

Supervisor: Prof. Diego Lugones

National College of Ireland
MSc Project Submission Sheet
School of Computing



Student Name: Amrutha Shivashankaramurthy
Student ID: 23172151
Programme: Cloud Computing **Year:** 2025
Module: MSc Research Project
Lecturer: Prof. Diego Lugones
Submission Due Date: 26/05/2025
Project Title: ASSET: A Parallel Lightweight Cryptographic Framework for IoT and Cloud Security

Word Count: 800 **Page Count:** 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.

Signature: Amrutha Shivashankaramurthy

Date: 26th May 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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

Amrutha Shivashankaramurthy
Student ID: 23172151

1 Introduction

This handbook contains information about the setup and prerequisites for the proposed model, including required libraries and software. The configuration manual also gives instructions on how to use the algorithms required to create the proposed model.

2 System Configuration Requirements:

- Processor Intel i5
- Operating System Windows ≥ 10
- RAM 8 GB
- Hard Disk 512 Gb

3 Prerequisites :

- Programming : python > 3.8: Python is a high level programming language known for its readability and versatility. It is convenient to learn and a popular choice for both beginners and experienced developers.
- Tools : Visual Studio Code (VS Code) is a free, lightweight code editor developed by Microsoft that runs on Windows, macOS and Linux.
- Cloud Services : AWS Cloud9, EC2.

4 Libraries to Install:

- Flask : It is a lightweight web application framework. Due to its flexibility along with the ability to scale up to complex applications it is one of the popular choices.
- simonspeckciphers : The simonspeckciphers Python library provides implementations of the Simon and Speck block ciphers – lightweight cryptographic algorithms designed by the NSA (National Security Agency) in 2013.
- pycryptodome : pycryptodome is a cryptographic library for Python that serves as a modern, secure and actively maintained replacement for the now-obsolete pycrypto. It provides cryptographic primitives and utilities in pure Python and C, suitable for a wide range of applications.
- Psutil : psutil (process and system utilities) is a python library for fetching the information on running processes and utilization of the system(CPU, disks, memory, sensors, network) in Python.
- Threadpoolctl : Python helpers to keep count of the number of threads used in the threadpool-backed of common libraries used for scientific computing and data science (For example : NumPy and OpenMP)

- Cryptography : Cryptography is a python library that includes both low level interfaces and high level recipes to common cryptographic algorithms such as symmetric ciphers, key derivation functions and message digests.

5 Algorithms Used:

- SPECK
- SIMON
- AES
- DES

6 Workflow Overview:

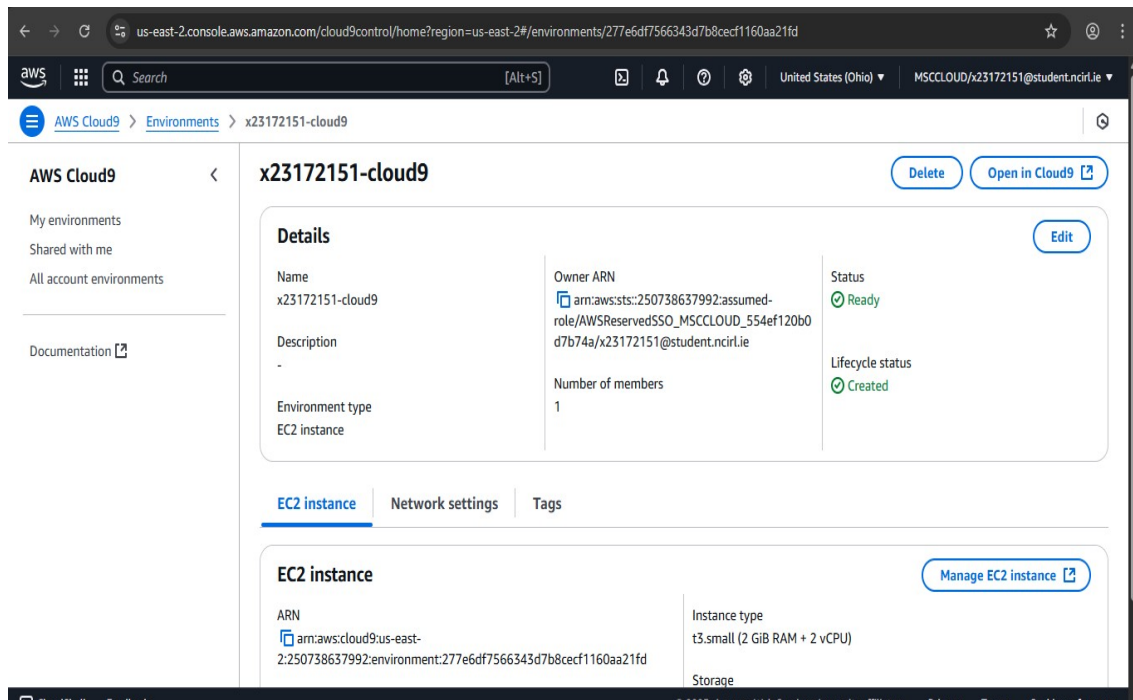
- To run the application we need to install above mentioned tools, libraries and packages. The important libraries are Flask, simonciphers, pycryptodome
- Flask is a python web framework used for web interface in the application.
- The SPECK and SIMON algorithm is used from the simonspeckciphers package. Pycryptodome provides cryptographic services such as encryption, decryption, hashing and digital signatures.
- Parallel encryption and decryption is done using multi-threading.
- This study is focusing on parallel based hybrid approach on the following algorithms. AES + SPECK + SIMON and DES + SPECK + SIMON.
- The input is divided into three nearly equal parts and each part is assigned to algorithms for encryption and further decryption is performed. Finally the decrypted outcome is concatenated as final output.
- User can provide the text input for the selected combination of algorithms which gives the ciphertext decrypted data and the performance measures like Key generation time, Encryption time, Decryption time, Throughput and Total CPU Usage.

7 Implementation

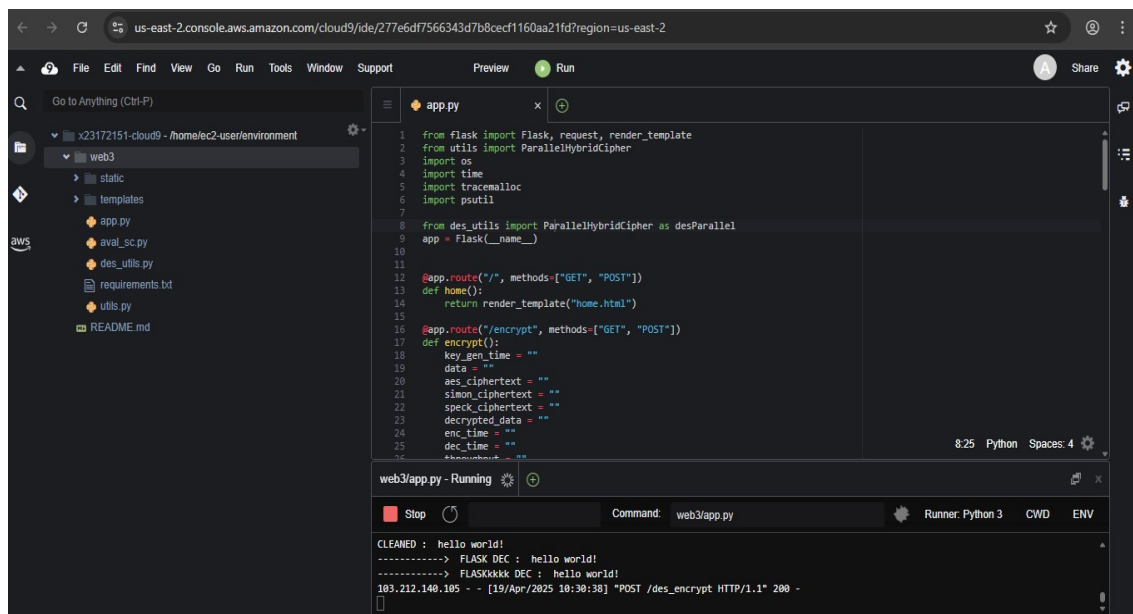
7.1 AWS Deployment

7.1.1: AWS Cloud9:

- Firstly create a cloud 9 environment by filling all required details
- Further click on open which leads to cloud9 IDE.
- Later add the required project files to access from AWS



- Finally run app.py to run the project.



Deployed Application Link: <http://18.119.96.204:8080/>

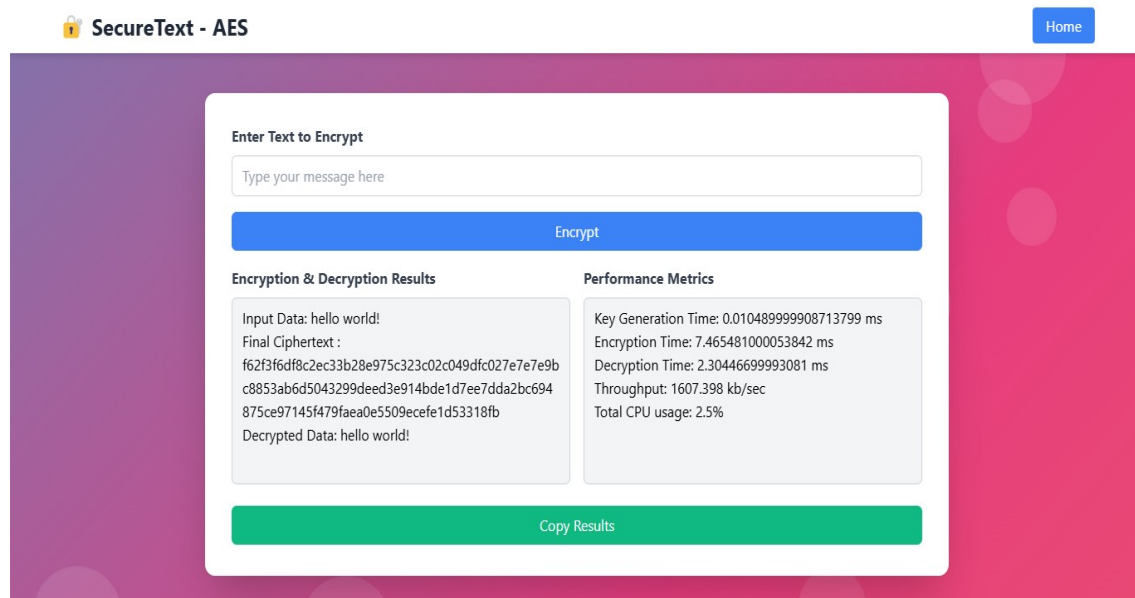
7.2 HomePage

Home page



7.3 ASSET

- Click on AES button, to perform encryption using AES, SIMON and SPECK
- Provide text input and encrypt.
- This provides the necessary results which helps in evaluating the performance.



7.4 DSSET

- Click on DES button, to perform encryption using DES, SIMON and SPECK
- Provide text input and encrypt.

This provides the necessary results which helps in evaluating the performance

The screenshot shows a web application titled "SecureText - DES" with a "Home" button in the top right. The main interface is a white card on a purple and pink background. It contains a text input field labeled "Enter Text to Encrypt" with the placeholder "Type your message here". Below the input is a blue "Encrypt" button. Underneath, there are two columns: "Encryption & Decryption Results" and "Performance Metrics". The results column shows "Input Data: hello world!", "Final Ciphertext: 7a35864ecb0c956c9581e551c3184272ef041a8684cf897a177f62fe29999550d8ed7ac8a02452dd6c70a67e0a4f77ab", and "Decrypted Data: hello world!". The metrics column shows "Key Generation Time: 0.009956999974747305 ms", "Encryption Time: 2.230849999932616 ms", "Decryption Time: 1.921933999938119 ms", "Throughput: 5379.116 kb/sec", and "Total CPU usage: 5.6%". At the bottom of the card is a green "Copy Results" button.

7.5: VS Code (Using VS code as local environment)

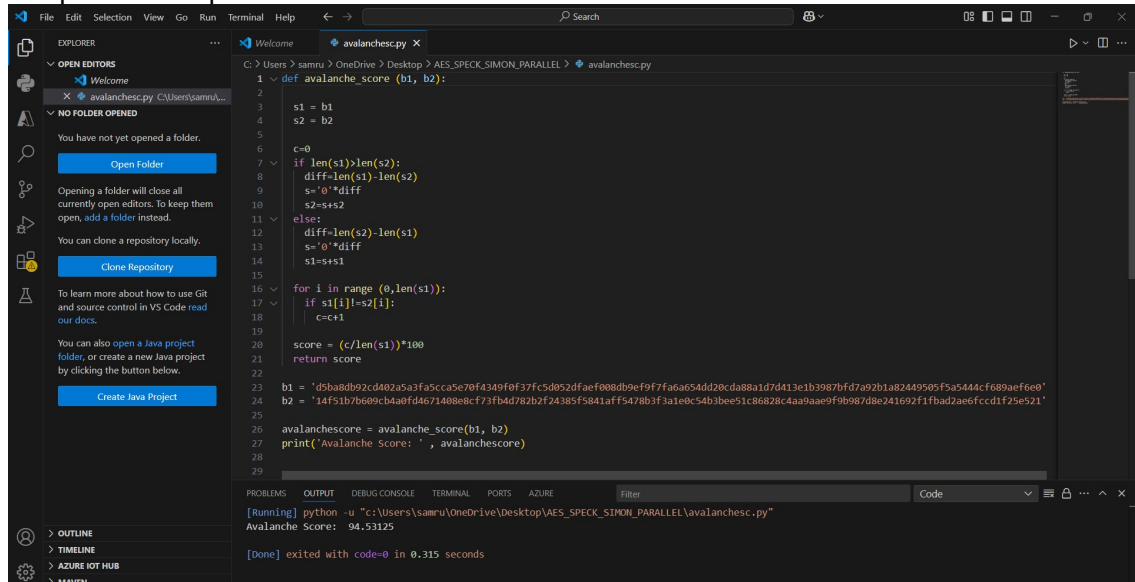
- Install the necessary libraries and run the code and make sure the application is working in local environment.

The screenshot shows a VS Code editor with a project named "web3". The Explorer panel on the left shows a file structure with folders "__pycache__", "static", and "templates", and files "des.html", "home.html", "index.html", "app.py", "avalanchesc.py", "des_utils.py", "locustfile.py", "requirements.txt", and "utils.py". The main editor window shows the code for "app.py". The code imports Flask, request, render_template, ParallelHybridCipher, os, time, tracemalloc, and psutil. It defines a Flask app and two routes: a home page and an encrypt endpoint. The encrypt endpoint uses ParallelHybridCipher for encryption and decryption, and it tracks performance metrics like key generation time, encryption time, decryption time, and throughput.

7.5.1 : To Calculate avalanche score:

- By using the following file avalanche.py we calculate the avalanche score or each input.
- Initially give text input and collect the generated cipher text from the web application and paste as the input for b1.

- Further modify the given input and collect the generated cipher for the modified input and paste it as input for b2.



The screenshot shows the Visual Studio Code interface. The Explorer pane on the left shows the file 'avalanchesc.py' in the directory 'C:\Users\samru\OneDrive\Desktop\AES_SPECK_SIMON_PARALLEL'. The main editor area displays the following Python code:

```

1 def avalanche_score (b1, b2):
2
3     s1 = b1
4     s2 = b2
5
6     c=0
7     if len(s1)>len(s2):
8         diff=len(s1)-len(s2)
9         s='0'*diff
10        s2=s+s2
11    else:
12        diff=len(s2)-len(s1)
13        s='0'*diff
14        s1=s+s1
15
16    for i in range (0,len(s1)):
17        if s1[i]!=s2[i]:
18            c=c+1
19
20    score = (c/len(s1))*100
21    return score
22
23 b1 = 'dsba8db92cd482a5a3fa5cca5e70f43d9f0f37fcd052dfaef008db9ef9f7fa0a654dd20cda88a1d7d413e1b3987bfd7a92b1a82449505f5a5444cf689aef6e0'
24 b2 = '14f51b7b609cb4a0fd4671408e8cf73fbad782b2f2438f5841aff5478b3f3a1e0c54b3bee51c86828c4aa9aaef9f9b987d8e241692f1fbad2ae6fccdf25e521'
25
26 avalanche_score = avalanche_score(b1, b2)
27 print('Avalanche Score: ', avalanche_score)
28
29

```

The Output pane at the bottom shows the execution results:

```

[Running] python -u "c:\Users\samru\OneDrive\Desktop\AES_SPECK_SIMON_PARALLEL\avalanchesc.py"
Avalanche Score: 94.53125
[Done] exited with code=0 in 0.315 seconds

```

- Run the code and it will provide the avalanche score. This helps in identifying the encryption strength and diffusion.

References

1. Using Python in Visual Studio Code:
<https://code.visualstudio.com/docs/python/python-tutorial>
2. How to use Cloud9 IDE:
<https://docs.aws.amazon.com/cloud9/latest/user-guide/ide.html>
3. Connecting to EC2 from Cloud9:
<https://docs.aws.amazon.com/cloud9/latest/user-guide/ec2-ssh.html>
4. Accessing pycryptodome libraries:
<https://pycryptodome.readthedocs.io/>
5. Using psutil libraries:
<https://psutil.readthedocs.io/en/latest/>
6. Benefits of threadpoolctl libraries:
<https://psutil.readthedocs.io/en/latest/>