National College of Ireland

Configuration Manual

MSc Research Project

Master of Science in Cyber Security

Vikas Padmanabha Naidu
Student ID: 23396989

School of Computing
National College of Ireland

Supervisor: Kamil Mahajan

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Vikas Padmanabha Naidu
Student ID:	23396989
Programme:	MSc in Cyber Security **Year:** 2025-2026
Module:	MSc Research Project
Lecturer:	Kamil Mahajan
Submission Due Date:	28/01/2026
Project Title:	Minimal Footprint Visual Data Protection for Secure Telemetry in Resource Constrained Wireless Visual Sensing Platforms.
Word Count:	1429 **Page Count:** 10

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: Vikas Padmanabha Naidu

Date: 28/01/2026

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

Vikas Padmanabha Naidu
23396989

1. Introduction

This document describes the implementation of a hybrid cryptographic cipher that is a combination of Polybius square substitution and Vigenere-like keystream encryption. The implementation was designed for a reversible encryption system for academic research and performance evaluation.

The system is intended to illustrate a hybrid design of encryption that combines classical concepts of substitution with modern deterministic hashing, offer an end to end set of security, quality and performance measures to cryptographic analysis, offer experimental analysis via series of automated testing and benchmarking, and provide a structure and reproducible research through organized experimentation configuration and telemetry.

2. Implementation Overview

The implementation consists of five major components:

2.1 Core Cryptographic Module

The cipher core uses the hybrid Polybius-Vigenere algorithm of encryption using a deterministic 16x16 Polybius substitution table (256 elements) managed by a SHA-256-seeded Fisher-Yates shuffle, Vigenere-style keystream created through iterative SHA-256 hashing, combined encryption step that makes use of CBC-style chaining, substitution and deterministic reversible transformation that is suitable to controlled experimental analysis.

2.2 Streaming Processing Module

Block-based streaming encryption/decryption for memory-efficient processing of large files, block-based encryption, default block size of 512 bytes, block chaining of all blocks continuously, keystream continuation based on the block index through context extension to allow file and in-memory services.

2.3 Metrics and Evaluation Module

In this module, research metrics, Security metrics: Shannon entropy, NPCR, UACI, avalanche effect, Quality metrics: PSNR and SSIM (and reconstruction fidelity), Performance metrics: execution time and peak memory usage are implemented.

2.4 Experiment Management Module

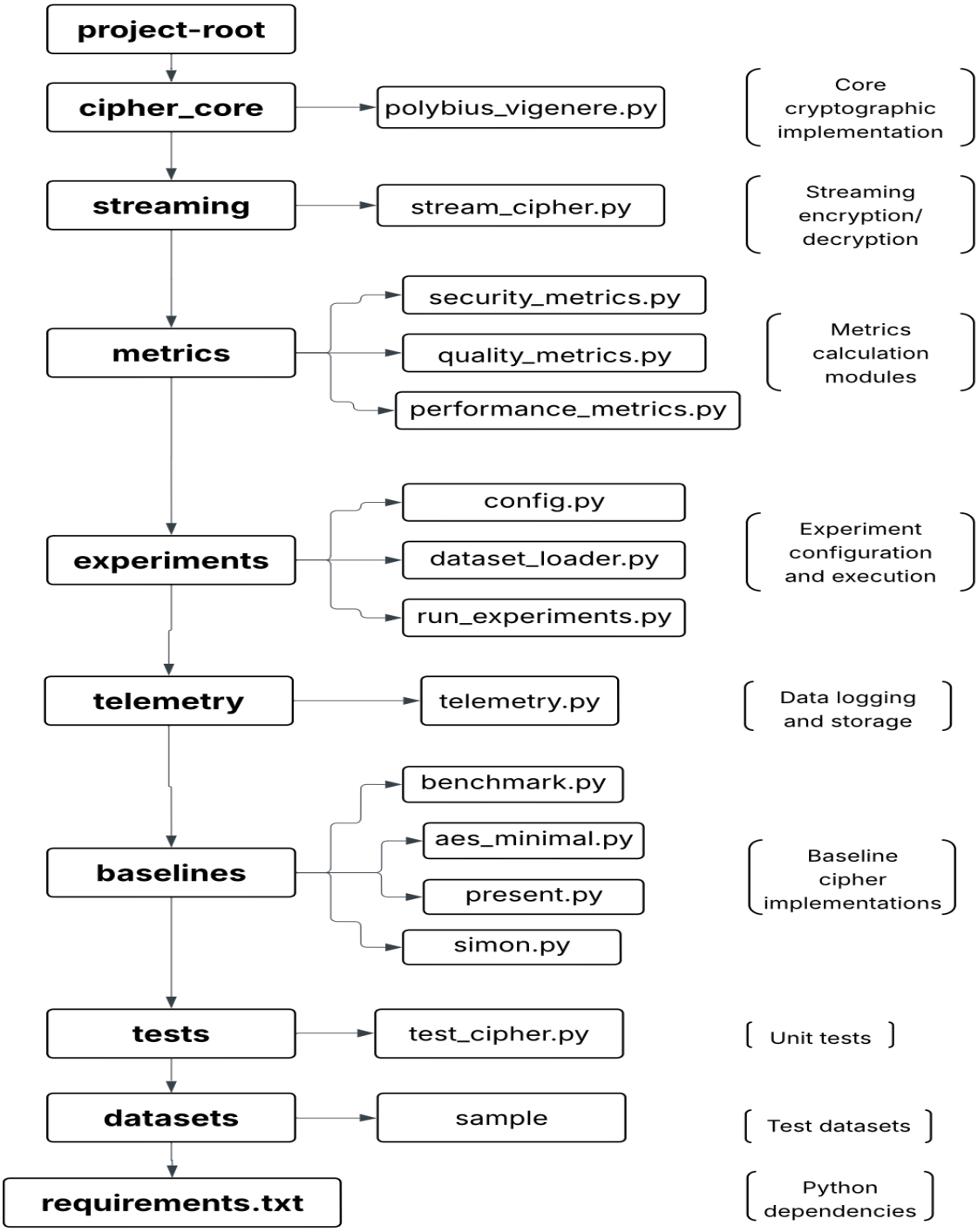
Experiment configuration and execution with data classes configured and executed experiment, was combined with dataset handling and the telemetry subsystem.

2.5 Telemetry and Data Management Module

Storage and export mechanism based on SQLite and metric outputs of every run and CSV export to be analyzed externally.

3. System Architecture

3.1 Directory Structure



3.2 Component Dependencies

main.py

```
├── cipher_core / polybius_vigenere.py
├── streaming / stream_cipher.py.
├── experiments / config.py
├── experiments / run_experiments.py
└── telemetry / telemetry.py
```

run_experiments.py

```
├── experiments / config.py
├── experiments / dataset_loader.py
├── cipher_core / polybius_vigenere.py
├── streaming / stream_cipher.py
├── metrics / security_metrics.py
├── metrics / quality_metrics.py
└── metrics / performance_metrics.py
```

4. Implementation Details

4.1 Cryptographic Algorithm

4.1.1 Polybius Table Generation

The table of Polybius substitution is created in a deterministic manner using SHA-256 as a pseudorandom number of generator seed of the secret key. The algorithm first initializes table as identity mapping [0, 1, 2, …, 255] bytes then perform Fisher-Yates shuffle from end to beginning and uses SHA-256(key || counter) to generate random keys for each swap that ensures deterministic, reproducible table generation.

4.1.2 Keystream Generation

The keystream is generated using iterative SHA-256 chaining:

- Ho = SHA-256(key + context)
- H_{i+1} = SHA-256(H_i)

The output provides a deterministic, repeatable byte stream of arbitrary length.

4.1.3 Encryption Process

For each block of 512 bytes CBC-style XOR with preceding ciphertext byte, Polybius substitution, add keystream byte (not a 256-bit number) then produce ciphertext and returning the final block of the last byte to be chained.

4.1.4 Decryption Process

For each block of 512 bytes first its subtracted with keystream modulo 256 and applied inverse Polybius substitution to reverse CBC chaining using previous ciphertext byte.

4.2 Streaming Processing

Streaming module works with data in blocks that can be configured (default at 512 bytes) so that it can handle large files using less memory, it features CBC chaining across block boundaries, derives keystream continuation using context tagging, supports byte-stream and encryption/decryption.

4.3 Security Metrics

4.3.1 Shannon Entropy

Measures randomness/unpredictability of data, which is calculated between 0 to 8, higher values indicate more random/encrypted data

```
# Entropy Calculation
entropy_value = -np.sum(p * np.log2(p))
```

Figure 1: Entropy calculation

4.3.2 Number of Pixel Change Rate (NPCR)

Measures percentage of pixels that changed between two images, ranges from 0 to 100%, higher values indicate better sensitivity to plaintext changes.

```
# NPCR Calculation
npcr_value = np.sum(img1 != img2) / img1.size * 100
```

Figure 2: NPCR calculation

4.3.3 Unified Average Changing Intensity (UACI)

Measures average intensity changes between two images, ranges from 0 to 100%, UACI evaluates diffusion properties of encryption.

```
# UACI Calculation
uaci_value = np.mean(np.abs(img1 - img2)) / 255 * 100
```

Figure 3: UACI calculation

4.3.4 Avalanche Effect

Measures bit flip ratio when input changes slightly, ideal value is 50% bit changes.

```
# Avalanche Calculation
avalanche_value = bit_diff / (len(original) * 8)
```

Figure 4: Avalanche calculation

4.4 Quality Metrics

4.4.1 Peak Signal-to-Noise Ratio (PSNR)

Measures image quality/reconstruction accuracy, ranges from 0 to infinity, perfect reconstruction is confirmed by infinity.

```
# PSNR Calculation
psnr_value = 20.0 * np.log10(max_pixel_value / np.sqrt(mse))
```

Figure 5: PSNR calculation

4.4.2 Structural Similarity Index (SSIM)

Measures structural similarity between images, ranges from -1 to 1, identical images show an output of 1.

```
# SSIM Calculation
numerator = (2 * mu1 * mu2 + c1) * (2 * sigma12 + c2)
denominator = (mu1 ** 2 + mu2 ** 2 + c1) * (sigma1_sq + sigma2_sq + c2)

ssim_value = numerator / denominator
```

Figure 6: SSIM calculation

4.5 Performance Metrics

4.5.1 Execution Time

To measure execution time: time.perf_counter(), which returns time taken for execution in seconds for microsecond level measurements.

4.5.2 Memory Usage

Measures peak memory usage using tracemalloc module which returns peak memory usage in bytes and tracks memory allocation during function execution.

4.6 Experiment Configuration

Experiments are configured using data classes.

```python
@dataclass
class ExperimentConfig:

   # Experiments are configured using dataclasses

    dataset_name: str
    image_name: str
    image_path: str
    cipher_name: str
    key: bytes
    key_descriptor: str
    block_size: int = 512
    context: bytes = b""
    config_label: str = "default"
    num_runs: int = 1
```

Figure 7: Configuration based on dataset structure

The system uses ExperimentConfig data classes and automatically generates configuration suites based on dataset structure.

4.7 Telemetry System

The telemetry system relies on SQLite as a structured and deterministic data-storage database so that all experiments which are carried out using the hybrid encryption framework is fully reproducible. Experiment metadata (dataset identifiers, cipher settings, block size, contexts and execution parameters) is stored in a special experiments table and the related metric outputs (entropy, NPCR, UACI, avalanche, PSNR, SSIM, execution time, memory consumption) are stored in a related metrics table. This relational structure allows the detailed reconstruction of any experimental run and does not allow configuration detail loss. The telemetry module enables smooth exporting of all the results stored in it to CSV where it can be analyzed using other tools like the power BI, looker studio or Grafana.

5. Configuration and Usage

5.1 Installation and Dependencies

Required Python Version: Python 3.7 or higher

Dependencies (requirements.txt):

pytest>=7.0.0
Pillow>=9.0.0
numpy>=1.21.0
cryptography>=3.4.0

Installation: pip install -r requirements.txt

5.2 Experiment Execution

5.2.1 Command-Line Interface

Run Demo: python main.py demo

```
(base) vikaspadmanabha@Vikas-macbookpro project-root 2 vikas % python main.py demo
============================================================
Hybrid Polybius + Vigenère Cipher Demo
============================================================

Original data: b'Hello, World! This is a test of the hybrid cipher.'
Key length: 24 bytes

Encrypted: b'\x94E\xf3C\x14\x9f\xc7d$\xc5\xbd\xe9v\x0b\xd9\x92B\x87=\x9d\xa6\xa9\x0b\xfa\xcaM\xec\x
a4\x0b\x1a\xcd\xaf\x9a\xc6Je\xd4W\x01\xcc\xe5\xf6\x841\x14?`sC\xe0'... (showing first 50 bytes)
Decrypted: b'Hello, World! This is a test of the hybrid cipher.'

Round-trip successful: True

------------------------------------------------------------
Testing with context (different keystream):
Same data, context1: 68eef3122d602bf4960c558f06e6fc594e523186...
Same data, context2: 09b2c0f57ed6cc375ebba1c5f5bed0f23780d652...
Different ciphertexts: True

============================================================
Demo completed successfully!
(base) vikaspadmanabha@Vikas-macbookpro project-root 2 vikas %
```

Figure 8: Shows the output of running the hybrid Polybius–Vigenere cipher for text encryption.

Run Experiments: python main.py experiments

```
(base) vikaspadmanabha@Vikas-macbookpro project-root 2 vikas % python3 main.py experiments

============================================================
Running Cryptographic Experiments
============================================================

Initializing database: results.db
Loading experiment configurations...
Found 3 experiment configurations

Running experiments...

Logging 3 results to database...
  ✓ Lena-Original-Image-512x512-pixels.png - hybrid_polybius_vigenere
  ✓ image.png - hybrid_polybius_vigenere
  ✓ lena-colour.png - hybrid_polybius_vigenere

Exporting results to CSV: results.csv
Export completed!

============================================================
Experiments completed!
```

Figure 9: Shows the output of running the hybrid Polybius–Vigenere cipher for image encryption

5.2.2 Programmatic Usage

Basic Encryption/Decryption:

```
    from cipher_core.polybius_vigenere import HybridCipher

    key = b"secret_key_12345678"
    cipher = HybridCipher(key)

    data = b"Hello, World!"
    encrypted = cipher.encrypt(data)
    decrypted = cipher.decrypt(encrypted)
```

Figure 10: Basic encryption/decryption

Streaming File Encryption:

```
from streaming.stream_cipher import encrypt_file, decrypt_file

key = b"secret_key_12345678"
encrypt_file("input.txt", "output.enc", key, block_size=512)
decrypt_file("output.enc", "decrypted.txt", key, block_size=512)
```

Figure 11: Streaming file encryption

Running Experiments:

```
from experiments.config import create_default_configs
from experiments.run_experiments import run_experiment_suite
from telemetry.telemetry import init_db, log_experiment, log_metrics, export_to_csv

# Initialize database
init_db("results.db")

# Create experiment configurations
suite = create_default_configs("datasets")

# Run experiments
results = run_experiment_suite(suite.configs)

# Log results
for result in results:
    if result["success"]:
        exp_id = log_experiment("results.db", result["config"])
        log_metrics("results.db", exp_id, result["metrics"])

# Export to CSV
export_to_csv("results.db", "results.csv")
```

Figure 12: Programmatic experiment execution

5.3 Experiment Configuration

5.3.1 Default Configuration

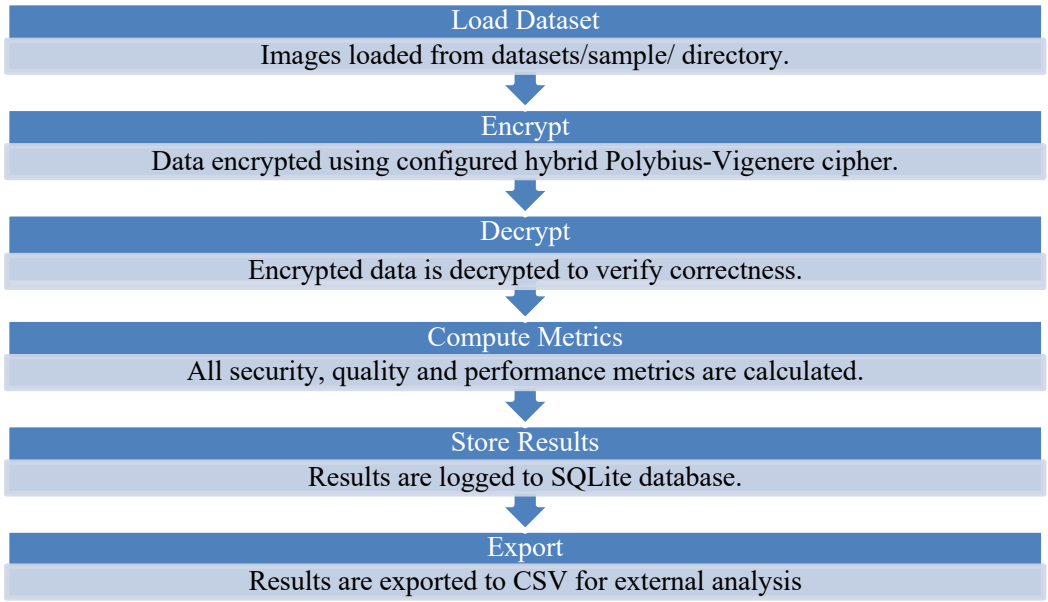
Default configuration generator uses scan of datasets/sample/ directory to generate experiment configurations of each image found (limited to first 3 images) in this directory

5.3.2 Block Size Configuration

The size of the block produces a direct effect on the memory consumption and on the processing efficiency. Smaller blocks (256-512 bytes) use less memory but have to repeat more times when encrypting data and decrypting it, incurring more overhead. Bigger blocks (1024-4096 bytes) do not do so many iterations thus giving better throughput, but they take more memory per operation. The 512 bytes default block size provides a good combination of speed and memory consumption.

6. Evaluation and Metrics

6.1 Experiment Workflow



6.2 Metrics Interpretation

The security metrics used in this framework evaluate how effectively the cipher to confuse plaintext formatting. Encrypted data should have entropy values of about 8 bits per byte with a target of more than 7.5, which is very random. NPCR, which must exceed 99% and UACI, which must be close to 33% when evaluated, are the measures of diffusion behavior that are considered sensitive to changes in the plaintext. The avalanche effect at the ideal value of 0.5, ensures that a one-bit alteration in the input induces extensive changes on the output value. The

quality of reconstruction is evaluated by PSNR and SSIM where PSNR = infinity and SSIM = 1.0 is an ideal decryption that will provide an exact copy of the original image. Execution time measured as an independent parameter through both encryption and decryption and added up as total runtime and maximum memory consumption which represents all the allocations made in the course of the processing including the substitution tables and the keystream buffers.

6.2.1 Baseline Metrics

Baseline performance metrics for AES-minimal, PRESENT, SIMON, and the hybrid cipher are stored in benchmark_results.csv.

Run Benchmark: python -m baselines.benchmark

```
(base) vikaspadmanabha@Vikas-macbookpro project-root 2 vikas % python -m baselines.benchmark

Benchmark results saved → benchmark_results.csv
```

Figure 13: Baseline benchmark execution

7. Results

Results are stored in SQLite database and CSV export enables analysis in external tools.

```
from telemetry.telemetry import export_to_csv

export_to_csv("results.db", "results.csv")
```

Figure 14: CSV results export demo

8. Testing

8.1 Unit Tests

Unit tests are located in tests/test_cipher.py/, tests are conducted for encryption/decryption, correctness, streaming operations, metrics calculations and telemetry.

Run tests: pytest -v

8.2 Correctness Verification

The system verifies correctness by encrypting data and decrypting ciphertext and comparing original and decrypted data (bit-exact match necessary) which will indicate success/failure of results in the experiment.