Project: CryptoCore - Technical Requirements Document (Sprint 1)

Sprint Goal: Establish the codebase and implement core block cipher operations in ECB mode.

1. Project Structure & Repository Hygiene

The codebase must be well-organized, documented, and buildable.

ID	Requirement Description	Priority
STR-1	The project **must** be hosted in a Git repository (e.g., on GitHub, GitLab).	Must
STR-2	A `README.md` file **must** be present in the root directory with the following minimal content: - Project Name and One-Sentence Description. - **Build Instructions:** Clear, step-by-step commands to compile and install the tool. - **Usage Instructions:** A clear example of the CLI command based on the deliverable format. - **Dependencies:** A list of all external libraries and tools required (e.g., Python 3.x, `pycryptodome`, OpenSSL).	Must
STR-3	The repository **must** include a build system or setup script: - **For Python:** A `setup.py` or `pyproject.toml` file, **or** a clear `requirements.txt` file. - **For C:** A `Makefile` that correctly compiles the tool into a binary named `cryptocore`.	Must
STR-4	The source code **must** be organized logically. Suggested structure: `project_root/ ├── src/ # Main source code │ ├── cli_parser.c|py │ ├── file_io.c|py │ └── modes/ # e.g., ecb.py or ecb.c ├── include/ # (C-specific) Header files ├── tests/ # Unit or integration tests ├── Makefile # (C-specific) ├── setup.py # (Python-specific) └── README.md`	Should

2. Command-Line Interface (CLI) Parser

The tool must accept arguments in a specific, predictable format.

ID	Requirement Description	Priority
CLI-1	The tool **must** be invokable from the command line as `cryptocore` (after building).	Must
CLI-2	The CLI **must** accept the following arguments: - `--algorithm ALGORITHM` : Specifies the cipher. For this sprint, it **must** accept `aes`. - `--mode MODE` : Specifies the mode of operation. For this sprint, it **must** accept `ecb`. - `--encrypt` or `--decrypt` : **Exactly one** of these flags **must** be provided to specify the operation. - `--key KEY` : **Must** accept a string representing the key. For AES-128, this **must** be a 16-byte value. - `--input INPUT_FILE` : **Must** accept a filesystem path to the input file. - `--output OUTPUT_FILE` : **Must** accept a filesystem path to the output file.	Must
CLI-3	The argument for `--key` **must** be provided as a hexadecimal string (e.g., `00112233445566778899aabbccddeeff`).	Must
CLI-4	The CLI **must** validate all arguments. If any argument is missing, malformed, or conflicting (e.g., both `--encrypt` and `--decrypt` are set), the tool **must** print a clear error message to `stderr` and exit with a non-zero status code.	Must
CLI-5	If the `--output` argument is not provided, the tool **should** derive a default output filename (e.g., `input.txt.enc` for encryption, `ciphertext.enc.dec` for decryption).	Could

Example Invocations:

```
# Encryption
$ cryptocore --algorithm aes --mode ecb --encrypt --key 000102030405060708090a0b0c0d0e0f --input plaintext.txt --output ciphertext.bin

# Decryption
$ cryptocore --algorithm aes --mode ecb --decrypt --key 000102030405060708090a0b0c0d0e0f --input ciphertext.bin --output decrypted.txt
```

3. Core Cryptographic Implementation

The implementation must correctly use the AES primitive to perform ECB mode operations.

ID	Requirement Description	Priority
CRY-1	The implementation **must** be for **AES-128** (a 128-bit block cipher with a 128-bit key).	Must
CRY-2	The core AES encryption and decryption functions (the primitive) **must not** be implemented from scratch. - **For Python:** The `AES` module from `Crypto.Cipher` (from the `pycryptodome` library) **must** be used. - **For C:** The `AES_encrypt` and `AES_decrypt` functions from `openssl/evp.h` or `openssl/aes.h` **must** be used.	Must
CRY-3	The **ECB mode** logic (splitting the input into blocks, padding, and calling the AES primitive per block) **must** be implemented by the student.	Must
CRY-4	The implementation **must** handle **padding**. The PKCS#7 padding standard **must** be used. - **On encryption:** The plaintext **must** be padded to be a multiple of the 16-byte block size. - **On decryption:** The padding **must** be validated and removed after decryption.	Must
CRY-5	The tool **must** handle both text and binary files correctly. The input and output **must** be treated as binary streams (e.g., `rb` and `wb` modes in Python).	Must

4. File I/O

The tool must reliably read from and write to the filesystem.

ID	Requirement Description	Priority
IO-1	The tool **must** read the entire contents of the file specified by `--input`.	Must
IO-2	The tool **must** write the resulting (encrypted or decrypted) data to the file specified by `--output`.	Must
IO-3	The tool **must** handle file errors gracefully (e.g., if the input file does not exist or is not readable). It must print an informative error to `stderr` and exit with a non-zero status code.	Must

5. Testing & Verification

The delivered product must be functionally correct.

ID	Requirement Description	Priority
TEST-1	The primary acceptance test: Encrypting a file and then decrypting it **must** produce a file identical to the original. `$ diff original_file.txt decrypted_file.txt` should show no differences.	Must
TEST-2	The student **must** provide a simple test script or documented example in the `README.md` that demonstrates this round-trip test.	Should
TEST-3	The ciphertext produced by the tool **should** be verified against a known-good implementation (e.g., using `openssl enc -aes-128-ecb` on the command line) to ensure the ECB implementation is correct.	Should

Example OpenSSL Verification Command:

```
# Encrypt with OpenSSL for comparison
$ openssl enc -aes-128-ecb \
    -K 000102030405060708090a0b0c0d0e0f \
    -in plaintext.txt \
    -out ciphertext_openssl.bin \
    -nopad # Warning: Only use -nopad if your file is exactly a multiple of 16 bytes. Otherwise, your implementation must handle padding.
```

Summary of Mandatory Technical Stack Choices

- **Language:** C or Python.
- **Core Crypto Library:**
  - **Python:** `pycryptodome` (`pip install pycryptodome`)
  - **C:** OpenSSL (`libcrypto`, link with `-lcrypto`)
- **Padding Standard:** PKCS#7.
- **Key Format:** Hexadecimal string.
- **Mode of Operation:** Electronic Codebook (ECB).

By adhering to these requirements, the student will successfully deliver a structured, secure, and functional codebase that meets the goal of implementing core block cipher operations.