Project: CryptoCore - Technical Requirements Document (Sprint 2)

Sprint Goal: Implement the core confidential modes of operation (CBC, CFB, OFB, CTR).

1. Project Structure & Repository Hygiene

The existing codebase must be extended while maintaining its structure and quality.

ID	Requirement Description	Priority
STR-1	All requirements from Sprint 1 (STR-1 to STR-4) **must** still be met.	Must
STR-2	The `README.md` file **must** be updated to include: - Documentation for the new CLI `--mode` options (`cbc`, `cfb`, `ofb`, `ctr`). - Updated usage examples showing the new modes and IV handling. - Instructions for the new interoperability tests with OpenSSL.	Must
STR-3	New source code for the modes **must** be integrated into the existing logical structure (e.g., within a `src/modes/` directory).	Must

2. Command-Line Interface (CLI) Parser

The CLI must be extended to support the new modes and the IV parameter.

ID	Requirement Description	Priority
CLI-1	The `--mode` argument **must** be extended to accept the new values: `cbc`, `cfb`, `ofb`, and `ctr`.	Must
CLI-2	The tool **must** handle the IV correctly based on the operation: - **For Encryption:** A secure random IV **must** be generated. The `--iv` flag **should not be accepted** during encryption. - **For Decryption:** The `--iv IV` flag **must** be accepted. The IV **must** be provided as a hexadecimal string.	Must
CLI-3	The CLI **must** validate that `--iv` is only provided in decryption mode. If provided during encryption, it should be ignored and a warning may be printed, or it should be treated as an error.	Should

Example Invocations:

```
# Encryption (IV is generated, not provided)
$ cryptocore --algorithm aes --mode cbc --encrypt --key 000102...0f --input plaintext.txt --output ciphertext.bin

# Decryption (IV must be provided by the user)
$ cryptocore --algorithm aes --mode cbc --decrypt --key 000102...0f --iv AABBCCDDEEFF00112233445566778899 --input ciphertext.bin --output decrypted.txt
```

3. Core Cryptographic Implementation

The four new modes must be implemented correctly from scratch, using the existing AES primitive.

ID	Requirement Description	Priority
CRY-1	All requirements from Sprint 1 regarding the use of the AES primitive (CRY-1, CRY-2) **must** still be met.	Must
CRY-2	The logic for **CBC, CFB, OFB, and CTR** modes **must** be implemented from scratch by the student.	Must
CRY-3	**Mode-Specific Requirements:** - **CBC (Cipher Block Chaining):** Must implement the chaining mechanism where each ciphertext block is XORed with the next plaintext block. Requires Padding (PKCS#7). - **CFB (Cipher Feedback):** Must be implemented as a stream cipher. The segment size must be the full block size (128-bit). This mode does not require padding. - **OFB (Output Feedback):** Must be implemented as a stream cipher, generating a keystream that is independent of the plaintext. This mode does not require padding. - **CTR (Counter):** Must be implemented as a stream cipher. A counter must be initialized from the IV and incremented for each block. This mode does not require padding.	Must
CRY-4	**Padding Logic must be updated:** - Modes that require padding (ECB, CBC) **must** continue to use PKCS#7. - Stream modes that do not require padding (CFB, OFB, CTR) **must not** pad the plaintext. The implementation must process partial final blocks correctly without altering their size.	Must

4. Initialization Vector (IV) Handling

Secure and correct handling of the IV is critical for the security of these modes.

ID	Requirement Description	Priority
IV-1	**For Encryption:** - The IV **must** be generated using a cryptographically secure random number generator (CSPRNG). - **For Python:** Use `os.urandom(16)`. - **For C:** Use `RAND_bytes()` from OpenSSL.	Must
IV-2	The generated IV **must** be prepended to the beginning of the ciphertext output file. - **File Format for Encryption:** `<16-byte IV><Ciphertext Bytes>`	Must
IV-3	**For Decryption:** - If the `--iv` argument is provided, the tool **must** use that value for decryption. - If the `--iv` argument is **not** provided, the tool **must** read the first 16 bytes from the input ciphertext file and use them as the IV. The remaining bytes are the actual ciphertext to be decrypted.	Must
IV-4	The IV **must** always be 16 bytes long (the AES block size) for all modes.	Must

5. File I/O

The file handling must be updated to manage the new IV-in-file format.

ID	Requirement Description	Priority
IO-1	**During Encryption:** The tool **must** write the 16-byte generated IV to the output file before writing any ciphertext bytes.	Must
IO-2	**During Decryption (without `--iv`):** The tool **must** read the first 16 bytes of the input file to obtain the IV. The decryption operation must then be performed on the rest of the file.	Must
IO-3	The tool **must** handle the case where a decryption input file is too short to contain a full IV (less than 16 bytes) by throwing a clear error.	Must

6. Testing & Verification

Interoperability with the standard OpenSSL tool is a key requirement for this sprint.

ID	Requirement Description	Priority
TEST-1	**Round-trip test:** Encrypting and then decrypting a file with the same tool **must** always produce a file identical to the original for every new mode.	Must
TEST-2	**Interoperability Test - Your tool to OpenSSL:** 1. Encrypt a file using your tool (e.g., in CBC mode). 2. Decrypt the resulting ciphertext file using the OpenSSL CLI command. 3. The decrypted file from OpenSSL **must** match the original plaintext. `openssl enc -aes-128-<mode> -d -K <key_hex> -iv <iv_hex> -in <ciphertext_file> -out <decrypted_file>`	Must
TEST-3	**Interoperability Test - OpenSSL to Your tool:** 1. Encrypt a file using the OpenSSL CLI command. 2. Decrypt the resulting ciphertext file using your tool. You must provide the correct IV used during encryption. 3. The decrypted file from your tool **must** match the original plaintext. `openssl enc -aes-128-<mode> -K <key_hex> -iv <iv_hex> -in <plaintext_file> -out <ciphertext_file>`	Must
TEST-4	The student **must** provide a test script or detailed commands in the `README.md` demonstrating successful interoperability tests for each of the four new modes.	Should

Example OpenSSL Interoperability Commands:

```
# 1. Encrypt with your tool, Decrypt with OpenSSL
$ cryptocore --algorithm aes --mode cbc --encrypt --key 000102...0f --input plain.txt --output cipher.bin
# Extract the IV from the first 16 bytes of cipher.bin (e.g., using `dd` or `hexdump`)
$ dd if=cipher.bin of=iv.bin bs=16 count=1
$ dd if=cipher.bin of=ciphertext_only.bin bs=16 skip=1
$ openssl enc -aes-128-cbc -d -K 000102030405060708090A0B0C0D0E0F -iv $(xxd -p iv.bin | tr -d '\n') -in ciphertext_only.bin -out decrypted.txt

# 2. Encrypt with OpenSSL, Decrypt with your tool
$ openssl enc -aes-128-cbc -K 000102030405060708090A0B0C0D0E0F -iv AABBCCDDEEFF00112233445566778899 -in plain.txt -out openssl_cipher.bin
$ cryptocore --algorithm aes --mode cbc --decrypt --key 000102...0f --iv AABBCCDDEEFF00112233445566778899 --input openssl_cipher.bin --output decrypted.txt
```