AES Encryption Project Task Breakdown

Project Overview

This document outlines the task breakdown for the AES Encryption Project, divided among five team members. The project involves implementing the AES (Advanced Encryption Standard) algorithm in C, following a modular design to ensure maintainability, scalability, and clarity.

Task Breakdown for 5 Team Members

Task 1: AES Core Implementation

Responsibilities:

- **Files:**
  - `include/aes.h`
  - `src/aes.c`
- **Tasks:**
  - Implement the main AES encryption and decryption functions.
    - `void aes_encrypt_block(const uint8_t *plaintext, uint8_t *ciphertext, const uint8_t *key);`
    - `void aes_decrypt_block(const uint8_t *ciphertext, uint8_t *plaintext, const uint8_t *key);`
  - Integrate all components (key schedule, utilities, S-boxes) to perform encryption and decryption.
  - Ensure correct implementation of AES rounds:
    - **Encryption Rounds:**
      - Initial AddRoundKey.
      - 9 main rounds (SubBytes, ShiftRows, MixColumns, AddRoundKey).
      - Final round (SubBytes, ShiftRows, AddRoundKey).
    - **Decryption Rounds:**
      - Initial AddRoundKey.
      - 9 main rounds (InvShiftRows, InvSubBytes, AddRoundKey, InvMixColumns).
      - Final round (InvShiftRows, InvSubBytes, AddRoundKey).
  - Write unit tests for AES core functions.
  - Coordinate with other team members to ensure seamless integration.

Deliverables:

- `include/aes.h`
- `src/aes.c`
- Unit tests in `tests/test_aes.c`

Task 2: Key Schedule Implementation

Responsibilities:

- **Files:**
  - `include/key_schedule.h`
  - `src/key_schedule.c`
- **Tasks:**
  - Implement the key expansion algorithm (key schedule) for AES-128.
    - `void key_expansion(const uint8_t *key, uint8_t *expanded_key);`
  - Ensure the expanded key is correctly generated for all rounds.
  - Use the S-box provided by Task 4 for key transformation.
  - Write unit tests for key schedule functions.
  - Document the key schedule process in the code.

Deliverables:

- `include/key_schedule.h`
- `src/key_schedule.c`
- Unit tests in `tests/test_key_schedule.c`

Task 3: File Input/Output Handling

Responsibilities:

- **Files:**
  - `include/file_io.h`
  - `src/file_io.c`
- **Tasks:**
  - Implement functions for reading and writing files.
    - `int read_file(const char *filename, uint8_t **data, size_t *length);`
    - `int write_file(const char *filename, const uint8_t *data, size_t length);`
    - `int read_key(const char *filename, uint8_t *key, size_t key_size);`
  - Handle file errors and edge cases gracefully.
  - Ensure that the file I/O functions are efficient and secure.
  - Write unit tests for file I/O functions.
  - Provide documentation for how to use the file I/O functions.

Deliverables:

- `include/file_io.h`
- `src/file_io.c`
- Unit tests in `tests/test_file_io.c`

Task 4: AES Tables (S-boxes) Implementation

Responsibilities:

- **Files:**
  - `include/aes_tables.h`
  - `src/aes_tables.c`
- **Tasks:**
  - Define the AES S-box (`aes_sbox[256]`) and Inverse S-box (`aes_inv_sbox[256]`) arrays.
  - Ensure the correctness of the S-box values as per the AES standard.
  - Provide these tables for use in encryption, decryption, and key schedule.
  - Write unit tests to verify the S-box and Inverse S-box.
  - Document the purpose and usage of the S-boxes in the code.

Deliverables:

- `include/aes_tables.h`
- `src/aes_tables.c`
- Unit tests in `tests/test_aes_tables.c`

Task 5: Utility Functions Implementation

Responsibilities:

- **Files:**
  - `include/utils.h`
  - `src/utils.c`
- **Tasks:**
  - Implement all utility functions used across the AES implementation.
    - `void rotate_word(uint8_t *word);`
    - `void sub_word(uint8_t *word);`
    - `void add_round_key(uint8_t state[4][4], const uint8_t *round_key);`
    - `void bytes_to_state(const uint8_t *input, uint8_t state[4][4]);`
    - `void state_to_bytes(const uint8_t state[4][4], uint8_t *output);`
    - `uint8_t gf_mul(uint8_t a, uint8_t b);`
    - `void sub_bytes(uint8_t state[4][4]);`
    - `void shift_rows(uint8_t state[4][4]);`
    - `void mix_columns(uint8_t state[4][4]);`
    - `void inv_sub_bytes(uint8_t state[4][4]);`
    - `void inv_shift_rows(uint8_t state[4][4]);`
    - `void inv_mix_columns(uint8_t state[4][4]);`
  - Ensure that all utility functions are optimized and tested.
  - Write unit tests for each utility function.

    - Document each function with Doxygen-style comments.

Deliverables:

- `include/utils.h`
- `src/utils.c`
- Unit tests in `tests/test_utils.c`

Additional Responsibilities for All Team Members

- **Documentation:**
    - Use Doxygen-style comments in all header and source files.
    - Include `@param[in]`, `@param[out]`, and `@return` annotations.
    - Ensure that the documentation is clear and comprehensive.
- **Testing:**
    - Write unit tests for their respective modules.
    - Participate in integration testing.
    - Use known test vectors to validate the implementation.
- **Collaboration:**
    - Regularly communicate progress and challenges.
    - Review code from other team members when requested.
    - Ensure compatibility between modules.

Interdependencies and Collaboration

- **Task 1 (AES Core):**
    - Depends on:
        * Task 2 for the key expansion (`key_schedule.c`).
        * Task 4 for the S-boxes (`aes_tables.c`).
        * Task 5 for utility functions (`utils.c`).
    - Should coordinate with Task 3 for file I/O integration.
- **Task 2 (Key Schedule):**
    - Depends on:
        * Task 4 for the S-box used in key expansion.
        * Task 5 for utility functions like `rotate_word` and `sub_word`.
- **Task 3 (File I/O):**
    - Works independently but needs to ensure the file formats are compatible with the AES core functions.
    - Should coordinate with Task 1 to ensure seamless data flow.
- **Task 4 (AES Tables):**

- Provides the S-boxes used by Tasks 1, 2, and 5.
- Must ensure the tables are correct and accessible.

- **Task 5 (Utilities):**
    - Provides utility functions used by Tasks 1 and 2.
    - Needs to ensure functions are efficient and error-free.

Timeline and Milestones

Milestone	**Responsible Task(s)**
Complete S-boxes (`aes_tables.c`)	Task 4
Complete Utility Functions (`utils.c`)	Task 5
Complete Key Schedule (`key_schedule.c`)	Task 2
Complete File I/O (`file_io.c`)	Task 3
Complete AES Core (`aes.c`)	Task 1
Unit Testing of Individual Modules	All
Integration Testing	All
Final Documentation	All
Project Submission	All

Deliverables Summary

- **Task 1:**
    - `aes.h`, `aes.c`
    - Unit tests: `tests/test_aes.c`
- **Task 2:**
    - `key_schedule.h`, `key_schedule.c`
    - Unit tests: `tests/test_key_schedule.c`
- **Task 3:**
    - `file_io.h`, `file_io.c`
    - Unit tests: `tests/test_file_io.c`
- **Task 4:**
    - `aes_tables.h`, `aes_tables.c`
    - Unit tests: `tests/test_aes_tables.c`
- **Task 5:**
    - `utils.h`, `utils.c`
    - Unit tests: `tests/test_utils.c`