

# Project: CryptoCore - Technical Requirements Document (Sprint 8)

**Sprint Goal:** Polish the library, ensure robustness, and create comprehensive documentation. Prepare for demonstration.

## 1. Project Structure & Repository Hygiene

The codebase must be refined to professional standards with comprehensive documentation and testing infrastructure.

ID	Requirement Description	Priority
STR-1	All requirements from previous Sprints (STR-1 to STR-4) <b>must</b> still be met.	Must
STR-2	The repository <b>must</b> include a comprehensive documentation directory structure. <ul style="list-style-type: none"><li>- <b>Required Files:</b> API.md, USERGUIDE.md, DEVELOPMENT.md</li><li>- <b>Suggested Structure:</b><pre>docs/ └── API.md └── USERGUIDE.md └── DEVELOPMENT.md └── examples       STR-3   A comprehensive test suite **must** be organized in a dedicated directory.   Must       - **Suggested Structure:**      </pre></li></ul>	Must
STR-3		
STR-4	All source files <b>must</b> include consistent code documentation (docstrings/comments following language standards).   Must	
tests/		
└── unit/	# Unit tests for individual functions	
└── integration/	# Integration tests for combined features	
└── vectors/	# Known-answer test vectors	
└── run_tests.py	# Main test runner	

STR-4 | All source files **must** include consistent code documentation (docstrings/comments following language standards). | Must |

## 2. Comprehensive Test Suite

A rigorous, automated test suite must validate every component against authoritative test vectors.

ID	Requirement Description	Priority
TEST-1	<b>Unit Test Coverage:</b> Every public function in the library <b>must</b> have corresponding unit tests. <ul style="list-style-type: none"><li>- Coverage <b>should</b> exceed 90% of code paths.</li><li>- Tests <b>must</b> be organized by module (e.g., test_aes.py, test_hmac.py, test_gcm.py).</li></ul>	Must
TEST-2	<b>Known-Answer Tests (KATs):</b> The test suite <b>must</b> include NIST-provided test vectors for every algorithm: <ul style="list-style-type: none"><li>- AES: ECB, CBC, CFB, OFB, CTR modes (NIST SP 800-38A)</li><li>- GCM: Test vectors from NIST SP 800-38D</li><li>- SHA-256, SHA3-256: Test vectors from NIST FIPS 180-4 and FIPS 202</li><li>- HMAC: Test vectors from RFC 4231</li></ul>	Must

ID	Requirement Description	Priority
TEST-3	<ul style="list-style-type: none"> <li>- PBKDF2: Test vectors from RFC 6070</li> </ul> <p><b>Integration Tests:</b> The test suite <b>must</b> include end-to-end tests for the CLI tool:</p> <ul style="list-style-type: none"> <li>- Encryption/decryption round-trip for all modes</li> <li>- Hash verification</li> <li>- HMAC generation and verification</li> <li>- Key derivation</li> </ul>	Must
TEST-4	<p><b>Negative Tests:</b> The test suite <b>must</b> include tests for error conditions:</p> <ul style="list-style-type: none"> <li>- Invalid inputs (wrong key lengths, malformed IVs, etc.)</li> <li>- Authentication failures (tampered ciphertext, wrong AAD, wrong HMAC keys)</li> <li>- File system errors (missing files, permission issues)</li> </ul>	Must
TEST-5	<p><b>Performance Tests:</b> The test suite <b>should</b> include benchmarks for critical operations:</p> <ul style="list-style-type: none"> <li>- AES encryption/decryption throughput</li> <li>- Hash function performance</li> <li>- Key derivation with various iteration counts</li> </ul>	Should
TEST-6	<p><b>Interoperability Tests:</b> The test suite <b>must</b> verify compatibility with external tools:</p> <ul style="list-style-type: none"> <li>- OpenSSL for all symmetric modes</li> <li>- Standard system tools (sha256sum, openssl kdf, etc.)</li> </ul>	Must
TEST-7	<p><b>Memory Safety Tests:</b> The test suite <b>should</b> include tests for memory handling:</p> <ul style="list-style-type: none"> <li>- Large file processing (files &gt; 1GB)</li> <li>- Proper cleanup of sensitive data (keys, passwords)</li> </ul>	Should
TEST-8	<p><b>Test Automation:</b> A single command <b>must</b> run the entire test suite.</p> <ul style="list-style-type: none"> <li>- <b>Python:</b> pytest or custom run_tests.py</li> <li>- <b>C:</b> make test or similar</li> </ul>	Must
<b>Example Test Runner Structure:</b>		
<pre># tests/run_tests.py import unittest import sys  def run_all_tests():     # Discover and run all tests     loader = unittest.TestLoader()     suite = loader.discover('tests/unit', pattern='test_*.py')     suite.addTests(loader.discover('tests/integration', pattern='test_*.py'))      runner = unittest.TextTestRunner(verbosity=2)     result = runner.run(suite)      return result.wasSuccessful()</pre>		

```

if __name__ == '__main__':
    success = run_all_tests()
    sys.exit(0 if success else 1)

```

### 3. API Documentation

Comprehensive API documentation must be created for library users and developers.

ID	Requirement Description	Priority
DOC-1	An API.md file <b>must</b> document every public function, class, and module in the library.	Must
DOC-2	For each function/class, the documentation <b>must</b> include: <ul style="list-style-type: none"> <li>- Purpose and functionality description</li> <li>- Complete signature with parameter types and return type</li> <li>- Parameter descriptions (including units, formats, constraints)</li> <li>- Return value description</li> <li>- Error conditions and exceptions raised</li> <li>- Examples of usage</li> <li>- Security considerations (if applicable)</li> </ul>	Must
DOC-3	The API documentation <b>must</b> be organized by module: <ul style="list-style-type: none"> <li>- cryptocore.aes - Block cipher functions</li> <li>- cryptocore.modes - Encryption modes (ECB, CBC, CFB, OFB, CTR, GCM)</li> <li>- cryptocore.hash - Hash functions (SHA-256, SHA3-256)</li> <li>- cryptocore.mac - MAC functions (HMAC, CMAC)</li> <li>- cryptocore.kdf - Key derivation functions (PBKDF2, key hierarchy)</li> <li>- cryptocore.csprng - Cryptographically secure random number generation</li> </ul>	Must
DOC-4	The documentation <b>must</b> include a dependency graph or module relationship diagram.	Should
DOC-5	The documentation <b>must</b> include version information and compatibility notes.	Must

#### Example API Documentation Format:

```

# CryptoCore API Documentation

## Module: cryptocore.aes

### `encrypt_block(key, plaintext)`
Encrypts a single 16-byte block using AES-128.

**Parameters:**
- `key` (bytes): 16-byte encryption key
- `plaintext` (bytes): 16-byte block to encrypt

**Returns:**
- `bytes`: 16-byte encrypted block

```

**\*\*Raises:\*\***

- `ValueError`: If key or plaintext length is incorrect

```
**Example:**  
```python  
from cryptocore.aes import encrypt_block  
  
key = b'0' * 16  
plaintext = b'hello world!!!!'  
ciphertext = encrypt_block(key, plaintext)
```

**Security Considerations:** - Never use ECB mode for multiple blocks. Use authenticated encryption modes like GCM instead.

## 4. User Guide

A comprehensive user guide must be created for end users of the CLI tool.

ID	Requirement Description	Priority
UG-1	A <code>USERGUIDE.md</code> file <b>must</b> provide complete instructions for using the CLI tool.	Must
UG-2	The user guide <b>must</b> include installation instructions for various platforms.	Must
UG-3	The user guide <b>must</b> include examples for every major use case:	Must
	<ul style="list-style-type: none"> <li>- File encryption/decryption with all modes</li> </ul>	
	<ul style="list-style-type: none"> <li>- File hashing and verification</li> </ul>	
	<ul style="list-style-type: none"> <li>- HMAC generation and verification</li> </ul>	
	<ul style="list-style-type: none"> <li>- Key derivation from passwords</li> </ul>	
	<ul style="list-style-type: none"> <li>- Working with associated data (GCM)</li> </ul>	
UG-4	The user guide <b>must</b> include troubleshooting section with common errors and	
	solutions.	Must
UG-5	The user guide <b>must</b> include security best practices:	Must
	<ul style="list-style-type: none"> <li>- Key management recommendations</li> </ul>	
	<ul style="list-style-type: none"> <li>- Mode selection guidance</li> </ul>	
	<ul style="list-style-type: none"> <li>- Password security guidelines</li> </ul>	
UG-6	The user guide <b>must</b> include a quick reference cheat sheet.	Must
UG-7	The user guide <b>should</b> include comparison with other tools (OpenSSL, GPG)	
	for users familiar with those tools.	Should

## **Example User Guide Section:**

## ## File Encryption and Decryption

### ### Basic AES Encryption

To encrypt a file with AES-256 in CBC mode:

```hash

```
cryptocore --algorithm aes --mode cbc --encrypt \
--key 00112233445566778899aabhbccddeeff00112233445566778899aabhbccddeeff \
```

```
--input secret.txt \
--output secret.enc
```

## Working with GCM and Associated Data

GCM provides both confidentiality and integrity protection. You can also include additional authenticated data (AAD):

```
# Encrypt with AAD
cryptocore --algorithm aes --mode gcm --encrypt \
--key 00112233445566778899aabbccddeeff \
--input database.sql \
--output database.enc \
--aad "database_version_3.2"

# Decrypt with the same AAD (verification happens automatically)
cryptocore --algorithm aes --mode gcm --decrypt \
--key 00112233445566778899aabbccddeeff \
--input database.enc \
--output database_decrypted.sql \
--aad "database_version_3.2"
```

**Important:** If the AAD doesn't match or the ciphertext was tampered with, the tool will fail without outputting any data.

## 5. Quality Assurance

Final quality checks to ensure production-ready code.

| ID   | Requirement Description  | Priority |
|------|--|----------|
| QA-1 | All code <b>must</b> be reviewed for security vulnerabilities: <ul style="list-style-type: none"><li>- No hardcoded keys or passwords</li><li>- Proper memory management (for C)</li><li>- Constant-time operations where required</li><li>- Secure random number generation</li></ul> | Must     |
| QA-2 | The code <b>must</b> be checked for common issues: <ul style="list-style-type: none"><li>- Buffer overflows (C)</li><li>- Integer overflows</li><li>- Memory leaks</li></ul>   | Must     |
| QA-3 | All external dependencies <b>must</b> be documented and pinned to specific versions.   | Must     |
| QA-4 | The project <b>must</b> include a CHANGELOG.md documenting changes from previous sprints.  | Must     |
| QA-5 | The project <b>must</b> include a CONTRIBUTING.md with guidelines for future contributors.   | Should   |
| QA-6 | The project <b>should</b> include a SECURITY.md file with security disclosure guidelines.  | Should   |

## Example Security Checklist:

```
# Security Checklist

## Critical Security Requirements
- [ ] No secret keys logged or printed (except when explicitly requested)
- [ ] All random values generated using cryptographically secure RNG
- [ ] Sensitive memory cleared after use
- [ ] Authentication performed before decryption (GCM, HMAC)
- [ ] No use of deprecated or insecure algorithms (ECB mode only for testing)
- [ ] Input validation on all user-provided data
- [ ] Proper error handling without leaking sensitive information
```