Software Requirements Specification (SRS)

for

ByteSpector Toolkit

Yukti Bhatia
230953552
Priyansh Nandan
230953450
Manipal Institute of Technology, Manipal

Version 1.0 – October 2025

Supervisor: Dr. Raviraj Holla
Associate Professor
School of Computer Engineering
Manipal Institute of Technology
MAHE, Manipal, Karnataka, India

Co-Supervisor: Mr. Yogesh Ganapati Chandavakar
School of Computer Engineering
Manipal Institute of Technology
MAHE, Manipal, Karnataka, India

Contents

Abstract

ByteSpector is a modular and extensible cybersecurity toolkit designed to unify data manipulation, cryptography, steganography, and forensic analysis into a single accessible web-based platform. It integrates classical and modern encryption schemes, multiple encoding mechanisms, and digital forensic tools for file integrity and metadata analysis. The purpose of this document is to define the functional and non-functional requirements for ByteSpector, ensuring clarity in development, testing, and deployment.

Keywords: Cybersecurity, Cryptography, Steganography, Forensics, Flask, Python

1 Introduction

1.1 Purpose

The Software Requirements Specification (SRS) document defines the purpose, scope, and detailed requirements of the ByteSpector Toolkit. It serves as a blueprint for the development team to implement all modules and interfaces consistently. It is intended for developers, testers, and cybersecurity researchers who will use or extend the system.

1.2 Scope

ByteSpector aims to consolidate essential tools for data encoding, encryption, and forensic investigation in a single web-based interface. It provides users with capabilities to:

- Perform classical and modern cryptographic operations.
- Conduct steganographic embedding and extraction within images.
- Visualize hidden LSB data and extract file metadata.
- Detect file types using magic byte signatures.
- Transform and hash textual data using standard cryptographic algorithms.

The system promotes educational and professional use in data security, penetration testing, and digital forensics.

1.3 Definitions, Acronyms, and Abbreviations

- API – Application Programming Interface
- AES – Advanced Encryption Standard
- DES – Data Encryption Standard
- LSB – Least Significant Bit
- JSON – JavaScript Object Notation
- SRS – Software Requirements Specification

1.4 References

- Flask Documentation: `https://flask.palletsprojects.com/`
- PyCryptodome Library: `https://pycryptodome.readthedocs.io/`
- Pillow (PIL) Documentation: `https://pillow.readthedocs.io/`
- IEEE Std 830-1998 – IEEE Recommended Practice for Software Requirements Specifications

2 Overall Description

2.1 Product Perspective

ByteSpector follows a modular, client-server architecture. The frontend interacts with the Flask backend through RESTful APIs, allowing modular expansion and testing. Each module (Cipher, Transform, Stego, and Forensic) is encapsulated and can operate independently, ensuring maintainability.

2.2 Product Functions

- Execute encryption/decryption with classical and modern symmetric ciphers.
- Encode or decode text using Base64, Hex, XOR, and hashing techniques.
- Embed or extract hidden information from images using steganography.
- Analyze files through magic byte detection, metadata extraction, and LSB visualization.

2.3 User Characteristics

- **Cybersecurity professionals:** Require integrated analysis and cryptography features.
- **Forensic experts:** Use file-level visualization and signature detection.
- **Students and researchers:** Learn encryption, steganography, and forensic fundamentals interactively.

2.4 Constraints

- Flask and Python (v3.8+) are required at the backend.
- Certain operations are dependent on valid key sizes and formats.
- Image operations require adequate resolution and PNG format for accuracy.
- The frontend must support CORS-enabled requests.

2.5 Assumptions and Dependencies

- Input data is provided in valid JSON or multipart form.
- Users supply valid cryptographic keys.
- The environment has access to dependencies such as Pillow, NumPy, and PyCryptodome.

3 System Features

3.1 Module 1: Symmetric and Classical Ciphers

Description: Provides various symmetric and substitution-based encryption methods.

Endpoints:

- /api/symmetric/Caesar/encrypt, /decrypt
- /api/symmetric/Multiplicative/encrypt, /decrypt
- /api/symmetric/Affine/encrypt, /decrypt
- /api/symmetric/Autokey/encrypt, /decrypt
- /api/symmetric/Vigenere/encrypt, /decrypt
- /api/symmetric/Playfair/encrypt, /decrypt, /keygen
- /api/symmetric/Hill/encrypt, /decrypt
- /api/symmetric/DES/encrypt, /decrypt
- /api/symmetric/AES/encrypt, /decrypt

3.2 Module 2: Data Transformation

Description: Offers text encoding, decoding, and hashing tools for data transformation and validation.

Endpoints:

- /api/transform/Base64/encrypt, /decrypt
- /api/transform/Hex/encrypt, /decrypt
- /api/transform/HashSHA256/encrypt
- /api/transform/HashMD5/encrypt
- /api/transform/XOR

3.3 Module 3: Steganography

Description: Allows embedding and extraction of text messages from images using LSB-based steganography.

Endpoints:

- /api/stego/embed
- /api/stego/extract

3.4 Module 4: Forensic Analysis

Description: Provides utilities for digital file inspection and validation.

Endpoints:

- /api/forensic/lsb_visualize
- /api/forensic/magic_analyze
- /api/forensic/metadata_extract

4 Non-Functional Requirements

- **Performance:** Each request should complete in under 3 seconds for standard file sizes.
- **Security:** Strong key validation and sanitized inputs.
- **Scalability:** Modular architecture supporting addition of new cryptographic or forensic tools.
- **Reliability:** Robust exception handling to prevent crashes on malformed inputs.
- **Portability:** Should run on any OS supporting Python 3.8+.
- **Usability:** Minimal setup, accessible web interface, and consistent API documentation.

5 System Architecture

Architecture Overview: ByteSpector uses a three-tier architecture:

1. **Frontend:** Built with HTML, CSS, and JavaScript; communicates through REST APIs.
2. **Backend:** Python Flask server managing cryptographic, forensic, and steganographic logic.
3. **Temporary Storage:** Used for image upload and metadata extraction.

Data Flow: Users interact via a web interface → API call → backend computation → JSON response or image output.

6 Appendix A: API Summary

Endpoint	**Description**
/api/symmetric/Caesar/encrypt	Encrypt plaintext using Caesar cipher
/api/symmetric/Caesar/decrypt	Decrypt ciphertext using Caesar cipher
/api/symmetric/AES/encrypt	AES encryption (ECB mode)
/api/symmetric/AES/decrypt	AES decryption (ECB mode)
/api/transform/Base64/encrypt	Encode text using Base64
/api/transform/Base64/decrypt	Decode Base64 text
/api/stego/embed	Embed secret message in image
/api/stego/extract	Extract hidden message from image
/api/forensic/magic_analyze	Analyze file headers via magic bytes
/api/forensic/metadata_extract	Extract EXIF metadata from images
/api/forensic/lsb_visualize	Visualize least significant bit layer of image

7 Conclusion

ByteSpector provides a versatile and educational platform for secure data transformation, encryption, and forensic study. Its modular design ensures that future algorithms, steganographic techniques, or forensic methods can be seamlessly integrated. This SRS serves as the foundation for its continued development and enhancement.

— End of Document —