White Paper: SecureCrypt

Version: 2.0

Date: August 20, 2025

Author: SecureCrypt Development Team**:**

Abstract

SecureCrypt is an open-source desktop application that provides military-grade AES-256 encryption for files and folders, with all processing occurring locally on the user's device. This whitepaper details the technical implementation, security architecture, and transparency measures of the application to establish trust with potential users.

1. Introduction

SecureCrypt addresses the growing need for simple yet powerful encryption tools that prioritize user privacy. Unlike cloud-based solutions, SecureCrypt ensures that files never leave the user's computer during encryption or decryption processes, eliminating potential exposure points.

2. Technical Architecture

2.1 Core Encryption Implementation

SecureCrypt employs authenticated AES-256 encryption in CBC mode with the following security features:

- **Key Derivation**: PBKDF2 with 100,000 iterations using SHA-512
- **Key Separation**: Dual-key system (32-byte encryption key + 32-byte authentication key)
- **Authentication**: HMAC-SHA256 verification for integrity checking
- **Salting**: 16-byte random salt for each encryption operation
- **Initialization Vector**: 16-byte random IV for each encryption

2.2 Streaming Architecture

The application processes files using a streaming approach with 64KB chunks, enabling:

- Support for files up to 8GB in size
- Minimal memory footprint regardless of file size
- Continuous processing without loading entire files into memory

2.3 Folder Encryption

SecureCrypt implements intelligent folder encryption by:

- Creating temporary ZIP archives (without compression for speed)
- Preserving directory structure within the encrypted container
- Adding file type metadata to distinguish between single files and folders

3. Security Analysis

3.1 Cryptography

The implementation uses the well-vetted PyCryptodome library with:

- NIST-approved AES-256 algorithm
- Proper cryptographic random number generation (get_random_bytes)
- Secure key stretching through PBKDF2 with high iteration count
- HMAC authentication to prevent tampering

3.2 Privacy Assurance

- Zero data transmission: All operations occur locally
- Temporary file cleanup: Comprehensive removal of intermediate files
- Memory management: Secure handling of sensitive data in memory

3.3 Threat Mitigation

- Executable file blocking: Prevention of potentially malicious file encryption
- Operation cooldown: Rate limiting to prevent resource exhaustion attacks
- Input validation: Comprehensive checks on all user inputs

4. Transparency Measures

4.1 Open Source Availability

The complete source code is available on GitHub, allowing:

- Public code review by security researchers
- Verification of cryptographic implementation
- Community contributions and improvements

4.2 Reproducible Builds

The provided build script enables:

- Transparent compilation process
- Verification that the distributed executable matches the source code
- Custom builds for paranoid users

4.3 No Telemetry or Data Collection

The application contains zero:

- Analytics collection
- Usage tracking
- External communications beyond update checks (if implemented)

5. Usage Guidelines

5.1 Recommended Practices

- Use strong, unique passwords for encryption
- Maintain backups of encrypted files
- Verify the integrity of downloads through checksums
- Keep the application updated

5.2 File Naming Convention

Encrypted files follow the pattern:

- Single files: `encrypted_[originalname].[extension].safe`
- Folders: `encrypted_folder.safe`

6. Technical Specifications

Aspect	Specification
Maximum File Size	8GB
Encryption Algorithm	AES-256-CBC
Key Derivation	PBKDF2-HMAC-SHA512 (100,000 iterations)
Authentication	HMAC-SHA256
Chunk Size	64KB
Supported Platforms	Windows (executable), Cross-platform (source)

7. Frequently Asked Questions

Q: Can SecureCrypt be audited by third parties?

A: Yes, the complete source code is available for review and audit.

Q: What happens if I lose my encryption password?

A: The encryption is secure without backdoors. Password recovery is impossible by design.

Q: How does SecureCrypt compare to commercial alternatives?

A: It provides similar encryption strength without subscription costs or data collection.

Q: Is the executable code signed?

A: Currently, the build process produces unsigned executables. Users concerned about this can compile from source.

8. Conclusion

SecureCrypt represents a transparent, privacy-focused approach to file encryption. By keeping all processing local and providing full source code access, it offers verifiable security without relying on

trust in third parties. The application is suitable for individuals and organizations requiring strong file encryption with complete control over their data.

9. References

- GitHub Repository: https://github.com/Oascent1/seccrypt
- PyCryptodome Documentation: https://www.pycryptodome.org/
- NIST Encryption Standards: https://csrc.nist.gov/projects/cryptographic-standards-and-guidelines