Welcome to Cipherion

Secure File Encryption Utility

Cipherion is a Python-based file encryption utility built on established cryptographic standards. It operates entirely offline and supports any file type through a terminal-driven interface.

BlackPyLab · Version 1.0.0

What is Cipherion?

Cipherion is an offline Python-based file encryption utility. It protects files of any type using modern cryptographic primitives — AES-256-CTR for encryption, PBKDF2-HMAC-SHA256 for key derivation, and HMAC-SHA256 for integrity verification. All operations run locally. No network connection is required at any point.

Offline Operation

All cryptographic operations execute locally. No data leaves the system during encryption or decryption.

Any File Type

Cipherion encrypts raw file bytes. File extension, format, or size does not affect compatibility.

Cross-Platform

Runs on Windows, macOS, and Linux. Requires Python 3.8 or later and the `cryptography` package.

Terminal Interface

Interactive command-line workflow with guided prompts for file selection, password entry, and output configuration.

How Encryption Works

Cipherion applies a layered cryptographic pipeline to each file. The process begins with user-provided credentials and produces a self-contained encrypted package with embedded integrity verification. Each stage is deterministic and auditable.

The resulting `.zip` package contains the encrypted payload alongside all parameters required for decryption: salt, nonce, and HMAC tag. No external metadata store is needed.

Key Features

AES-256 Encryption

AES-256 in CTR mode encrypts file contents. CTR mode converts the block cipher into a stream cipher, enabling efficient processing of arbitrary-length data.

PBKDF2 Key Derivation

Passwords are processed through PBKDF2-HMAC-SHA256 with 100,000 iterations and a randomly generated salt, producing a 256-bit encryption key.

HMAC Integrity Check

An HMAC-SHA256 tag is computed over the ciphertext. During decryption, the tag is verified in constant time to detect tampering or corruption.

Streaming Encryption

Files are processed in chunks. Large files do not need to be loaded entirely into memory before encryption begins.

Random Salt & Nonce

A new salt and nonce are generated for every encryption operation using `os.urandom`, ensuring unique ciphertext even for identical inputs.

ZIP Packaging

Encrypted output is bundled into a `.zip` archive containing the ciphertext, salt, nonce, HMAC tag, and optional metadata file.

Multi-Language Interface

The terminal interface supports multiple languages. Language selection is available at startup.

Low Memory Usage

Streaming chunk processing keeps memory footprint bounded regardless of input file size.

Encrypt Any File

Cipherion operates on raw file bytes. It does not inspect file extensions, headers, or format signatures. Any file that can be read as a byte stream is a valid input.

"If a file can be read as bytes, Cipherion can encrypt it."

Documents

PDF, DOCX, XLSX, PPTX, ODT, RTF

Archives

ZIP, TAR, GZ, RAR, 7Z

Media

MP4, AVI, MP3, WAV, PNG, JPEG

Code & Data

PY, JS, SQL, DB, JSON, XML, YAML

Executables

EXE, DLL, SO, BIN, APP

Config Files

CONF, INI, ENV, CFG, PROPERTIES

Unknown or proprietary file types are handled identically. Cipherion makes no assumptions about file structure or content semantics.

Security Principles

Cipherion's security model is based on established, peer-reviewed cryptographic standards. Each design decision prioritizes correctness and auditability over convenience.

Cryptographic Primitives

- **AES-256-CTR** — Counter mode encryption with 256-bit key
- **PBKDF2-HMAC-SHA256** — Key derivation with 100,000 iterations
- **Random Salt** — 16-byte salt generated per operation via `os.urandom`
- **Random Nonce** — 16-byte nonce generated per operation via `os.urandom`
- **HMAC-SHA256** — Authentication tag computed over ciphertext
- **Constant-Time Verification** — HMAC comparison uses `hmac.compare_digest`

Operational Guarantees

- **100% Offline** — No network calls during encryption or decryption
- **No Cloud** — Files are never uploaded or transmitted
- **No Telemetry** — No usage data is collected or reported
- **No Internet Required** — Fully functional in air-gapped environments

Dependencies

Cipherion relies on the `cryptography` package (FIPS-validated backend) and the Python standard library. Source code is available for independent review.

Demo Edition

This demonstration presents the decryption workflow of Cipherion using a pre-generated encrypted package. It is intended to illustrate the operational behavior of the full product in a controlled, reproducible context.

Demo Edition Includes

- Pre-generated encrypted sample package
- Guided decryption walkthrough
- HMAC integrity verification demonstration
- Terminal interface experience
- Output extraction and validation

Full Edition Adds

- Encrypt any user-provided file
- Unlimited encryption and decryption operations
- Metadata management (filename, description, timestamps)
- Advanced output path configuration
- Full interactive encryption workflow
- Multi-language terminal interface

The demo does not impose functional limitations on the cryptographic layer. It restricts the input source to a bundled sample file. All security properties — key derivation, encryption, HMAC verification — operate identically to the full edition.

Thank You

Thank you for exploring Cipherion. This demonstration reflects the design philosophy of the full product: simplicity, security, and reliability. Each component has been implemented with attention to correctness and transparency.

Simplicity

A focused terminal interface with clear prompts. No unnecessary configuration, no hidden behavior.

Security

Established cryptographic standards, applied correctly. Open dependencies, auditable implementation.

Reliability

Consistent behavior across platforms. Deterministic output. Integrity verification on every decryption.

We hope you find the complete version useful for your file protection workflows. For documentation, source code, and updates, refer to the BlackPyLab repository.

BlackPyLab · Version 1.0.0