pynacl-cellar

Justin Quick

Aug 06, 2022

CONTENTS

Salt Cellar is a Python program that protects your files and folders using hard encryption. The `pynacl-cellar` package provides the command line tool `cellar` to encrypt and decrypt your files/folders using a secret key and protect them from prying eyes. Files are quickly encrypted/decrypted fully asynchronously using asyncio/aiofiles.

CHAPTER

ONE

ENCRYPTION

The hard encryption is accomplished using the PyNaCl package and the libsodium library. The underlying encryption algorithm is Salsa20 which is fast and increases the file size very minimally. By default, `cellar` encrypts the files in place, overwriting the original files with the encrypted version.

> :warning: **DO NOT FORGET YOUR KEY!** This program will encrypt your files and make them unusable until you decrypt them. If you lose/forget the secret key then the files will not be recoverable. Use at your own risk

CHAPTER

TWO

KEYS

A secret key for use with the tool should be 32 bytes long. It can be stored as a file, environment variable or entered in the command line. If the key is too short it will be truncated and if it's too long it will be padded with null bytes.

CHAPTER

THREE

INSTALL

- Install libsodium
- Recommend using pipx for installing the CLI tool

  ```
  pipx install pynacl-cellar
  ```

- Then run the command with pipx

  ```
  pipx run cellar ...
  ```

CHAPTER

FOUR

USAGE

The CLI command is `cellar` and you can call `encrypt` or `decrypt` on a set of paths. Paths can be files, folders or `-` for stdin

```
Usage: cellar [OPTIONS] COMMAND [ARGS]...

Options:
  --version                Show the version and exit.
  -v, --verbosity          Output level WARN/INFO/DEBUG
  -l, --log-file FILENAME  File path to write logs to
  -k, --key-file FILENAME  File path to use for secret key or CELLAR_KEYFILE env var
  -p, --key-phrase TEXT    Text to use as secret key. Use "-" to read from stdin. Do NOT␣
↪type your key via command line! It will show in your shell history
  -P, --key-prompt         Prompt for the secret key (default)
  --help                   Show this message and exit.

Commands:
  decrypt  Decrypts given paths.
  encrypt  Encrypts given paths.
```

CHAPTER

FIVE

ENV VARS

5.1 CELLAR_KEYFILE

A file that contains the content of your private key (32 bytes)

5.2 CELLAR_KEYPHRASE

A string that contains the content of your private key (32 bytes)

5.3 CELLAR_LOGFILE

A filename to use for logging

CHAPTER

SIX

EXAMPLE

6.1 Encrypt a given directory

```
$ cellar -vv encrypt test-dir/
Secret key:
WARNING cellar __init__: Key too short, padding to to 32 characters
INFO cellar encrypt_file: Encrypted file test-dir/mypic.jpg
INFO cellar encrypt_dir: Encrypted directory test-dir
```

6.2 Encrypt stdin

```
$ echo foobarbaz | cellar encrypt -
9TBS*S
# Decrypt it using pipes
$ echo foobarbaz | cellar encrypt - | cellar decrypt -
foobarbaz
```

6.3 Encrypt files w/ pipe redirection

```
$ cellar encrypt - < plain.txt > encrypted.txt
```

6.3.1 Running cellar

cellar

```
cellar [OPTIONS] COMMAND [ARGS]...
```

Options

--version
Show the version and exit.

-v, --verbosity
Output level WARN/INFO/DEBUG

-l, --log-file <log_file>
File path to write logs to

-k, --key-file <key_file>
File path to use for secret key or CELLAR_KEYFILE env var

-p, --key-phrase <key_phrase>
Text to use as secret key. Use "-" to read from stdin. Do NOT type your key via command line! It will show in your shell history

-P, --key-prompt
Prompt for the secret key (default)

Environment variables

CELLAR_LOGFILE
Provide a default for *-l*

CELLAR_KEYFILE
Provide a default for *-k*

CELLAR_KEYPHRASE
Provide a default for *-p*

decrypt

Decrypts given paths. Can be either files or directories

```
cellar decrypt [OPTIONS] PATHS...
```

Arguments

PATHS
Required argument(s)

encrypt

Encrypts given paths. Can be either files or directories

```
cellar encrypt [OPTIONS] PATHS...
```

Arguments

PATHS

Required argument(s)

6.3.2 API

cellar.cli

cellar.crypt

class cellar.crypt.**BaseCellar**(*key*, *encoder_class=<class 'nacl.encoding.URLSafeBase64Encoder'>*, *block_size=1048576*, *concurrency=100*)

Bases: `object`

Main encryption class to enc/decrypt streams, files and directories. Manages the PyNaCl SecretBox/nonce/keys

async decrypt(*ciphertext*, *decode=True*)

Encrypts ciphertext to plaintext. By default it decodes using the URLSafeBase64Encoder Catches any errors (like bad dec key) and logs them before exiting

async decrypt_stream(*instream*, *outstream=<_io.BufferedWriter name='<stdout>'>*, *decode=False*)

Decrypts a stream and outputs it to another (default stdout)

async encrypt(*plaintext*, *encode=True*)

Encrypts plaintext to ciphertext. By default it encodes using the URLSafeBase64Encoder

async encrypt_stream(*instream*, *outstream=<_io.BufferedWriter name='<stdout>'>*, *encode=False*)

Encrypts a stream and outputs it to another (default stdout)

async map_crypto(*func*, *iters*)

property nonce

Random nonce to fix box size

async read_write_crypto(*infile*, *outfile*, *encrypt=True*)

exception cellar.crypt.**DecryptionError**

Bases: `Exception`

class cellar.crypt.**EncryptedPathCellar**(*key*, *encoder_class=<class 'nacl.encoding.URLSafeBase64Encoder'>*, *block_size=1048576*, *concurrency=100*)

Bases: *BaseCellar*

Cellar that encrypts the filenames as well as the content

async decrypt_dir(*encdir*, *preserve=False*)

Decrypts entire directory with all file/dir names and file content If preserve is True, encdir is preserved but by default it's deleted

async decrypt_file(*cipherfile*, *plainfile=None*, *preserve=False*)

async encrypt_dir(*plaindir*, *preserve=False*)

Encrypts entire directory with all file/dir names and file content If preserve is True, plaindir is preserved but by default it's deleted

async encrypt_file(*plainfile*, *cipherfile=None*, *preserve=False*)

prefix = '.enc.'

class cellar.crypt.**OverwritePathCellar**(*key*, *encoder_class=<class 'nacl.encoding.URLSafeBase64Encoder'>*, *block_size=1048576*, *concurrency=100*)

Bases: *BaseCellar*

async decrypt_dir(*cipherdir*, *preserve=False*)

async decrypt_file(*cipherfile*, *preserve=None*)

async encrypt_dir(*plaindir*, *preserve=False*)

async encrypt_file(*plainfile*, *preserve=None*)

PYTHON MODULE INDEX

C

INDEX

Symbols

B

C

D

E

M

N

O

P

R