Prashant Kumar, Pamela Banerjee

A python package for hybrid file encryption and decryption. securefile is for n-layer file encryption. This package provides a basic two-way encryption algorithm for a file. It supports approximately all kind of file encoding. The package provides RSA, DES, AES and Shift Cipher and base64 algorithm for file encoding and decoding.

Hybrid Cryptography on Cloud

Securefile Python Package

Full Documentation: **https://github.com/Krprashant94/Hybrid-Cryptography-on-Cloud**

Secure File

A python package for hybrid file encryption and decryption. **securefile** is for n-layer file encryption. This package provides a basic two-way encryption algorithm for a file. It supports approximately all kind of file encoding. The package provides RSA, DES, AES and Shift Cipher and base64 algorithm for file encoding and decoding.

Transmitting sensitive data over a public channel is quite insecure. To secure the data over public channel we need to encrypt the data. So that no third party can access that information. It may be possible that encrypting with one algorithm can decoded by reverse engineering. However, using n-layer of the different algorithm makes it more secure and decoding such kind of cipher text with reverse engineering will take approximately unpractical time for a supercomputer.

# ****Dependency****:

* pyserial
  + >>> python -m pip install pyserial

Native :

* base64
* random
* binascii
* re

# Installation:

To install this package with pip command type

>>> pip install securefile

**Module Structure**:

digraph foo {
    "Securefile Module" -> "DES Algorithm"
    "Securefile Module" -> "RSA Algorithm"
    "Securefile Module" -> "AES Algorithm"
    "Securefile Module" -> "    Keyset   "
    "Securefile Module" -> "Secure Serial"

    "Securefile Module"[shape=ractangle]
    "AES Algorithm"[shape=ractangle]
    "DES Algorithm"[shape=ractangle]
    "RSA Algorithm"[shape=ractangle]
    "    Keyset   "[shape=ractangle]
    "Secure Serial"[shape=ractangle]
}

# Uses:

**from** securefile **import** Encrypt

**from** securefile.keyset **import** RSA\_KEY**,** DES\_KEY**,** AES\_KEY

des\_key = DES\_KEY.genrate**(**'12345678123456781234567812345678'**)**

aes\_key = AES\_KEY.genrate**(**'700102030405060708090a0b0c0d0e0f'**)**

rsa\_public\_key = RSA\_KEY.public\_key\_genrate**(**18285**,** 57067**)**

rsa\_private\_key = RSA\_KEY.private\_key\_genrate**(**6861**,** 57067**)**

chiper\_shift = 3

enc = Encrypt**(**'test.txt'**,** delimiter=':'**)**

enc.open**()**

enc.base64\_encrypt**()**

enc.aes\_encrypt**(**aes\_key**,** commit=**True)**

enc.des\_encrypt**(**des\_key**,** commit=**True)**

enc.rsa\_encrypt**(**rsa\_private\_key**,** commit=**True)**

enc.caesar\_cipher**(**key\_shift=chiper\_shift**,** commit=**True)**

enc.caesar\_decipher**(**key\_shift=chiper\_shift**,** commit=**True)**

enc.rsa\_decrypt**(**rsa\_public\_key**,** commit=**True)**

enc.des\_decrypt**(**des\_key**,** commit=**True)**

enc.aes\_decrypt**(**aes\_key**,** commit=**True)**

enc.base64\_decrypt**(**commit=**True)**

Encrypt Class

class securefile.**Encrypt**(file\_name, delimiter=':')

Encrypt is the centre class for the entire module it is a wrapper class for all encryption and decryption algorithm.

Example

**from** securefile **import** Encrypt

enc = Encrypt**(**'test.md'**,** delimiter=':'**)**

|  |  |
| --- | --- |
| **Special Method:** | * [**\_\_init\_\_()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\encrypt.html#securefile.Encrypt.__init__) * **\_\_open\_check()** |
| **Method:** | * [**open()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\open.html#securefile.Encrypt.open) [**close()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\close.html#securefile.Encrypt.close) [**create()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\create.html#securefile.Encrypt.create) [**commit()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\commit.html#securefile.Encrypt.commit) * [**des\_encrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\des_enc.html#securefile.Encrypt.des_encrypt) * [**des\_decrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\des_dec.html#securefile.Encrypt.des_decrypt) * [**caesar\_cipher()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\caesar_cipher.html#securefile.Encrypt.caesar_cipher) * [**caesar\_decipher()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\caesar_decipher.html#securefile.Encrypt.caesar_decipher) * [**rsa\_encrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\rsa_enc.html#securefile.Encrypt.rsa_encrypt) * [**rsa\_decrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\rsa_dec.html#securefile.Encrypt.rsa_decrypt) * [**aes\_encrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\aes_enc.html#securefile.Encrypt.aes_encrypt) * [**aes\_decrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\aes_dec.html#securefile.Encrypt.aes_decrypt) * [**base64\_encrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\base64_enc.html#securefile.Encrypt.base64_encrypt) * [**base64\_decrypt()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\base64_dec.html#securefile.Encrypt.base64_decrypt) * [**get\_text()**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\enc\get_text.html#securefile.Encrypt.get_text) |
| **Attributes:** | * \_\_file\_name * \_\_plain\_text * \_\_open\_flag |

## \_\_init\_\_(file\_name, delimiter=':')

Encrypt class object takes two argument

## Encrypt.open

Open current file and read the data contains. Binary file reading using basic python file open()

Uses:

>>> enc.open**()**

## Encrypt.close

close current file and flush the memory

Uses:

>>> enc.open**()**

>>> enc.close**()**

## Encrypt.create

Create an Encrypt object form file

a = Encrypt.create**(**'test.md'**,** delimiter=':'**)**

a.open**()**

print**(**a.get\_text**())**

## Encrypt.commit

Save current progress in the working file. (Commit the changes)

## Encrypt.des\_encrypt

Encrypt a plain text with DES algorithm.

Example:

**from** securefile **import** Encrypt

**from** securefile.keyset **import** DES\_KEY

des\_key = DES\_KEY.genrate**(**'12345678123456781234567812345678'**)**

enc = Encrypt**(**'test.md'**)**

enc.open**()**

enc.des\_encrypt**(**des\_key**,** commit=**True)**

enc.close**()**

## Encrypt.des\_decrypt

Decrypt a cipher text into plain text using DES algorithm.

Example:

**from** securefile **import** Encrypt

**from** securefile.keyset **import** DES\_KEY

des\_key = DES\_KEY.genrate**(**'12345678123456781234567812345678'**)**

enc = Encrypt**(**'test.md'**)**

enc.open**()**

enc.des\_decrypt**(**des\_key**,** commit=**True)**

enc.close**()**

## Encrypt.caesar\_cipher

The action of a Caesar cipher is to replace each plaintext letter with a different one a fixed number of places down the alphabet. The cipher illustrated here uses a left shift of three, so that (for example) each occurrence of E in the plaintext becomes B in the cipher text. Encrypt a plain text into cipher text.

Example:

**from** securefile **import** Encrypt

enc = Encrypt**(**'test.md'**)**

enc.open**()**

enc.caesar\_cipher**(**key\_shift=3**,** commit=**True)**

enc.close**()**

## Encrypt.caesar\_decipher

Decrypt a cipher text into plain text.

See **[caesar\_cipher](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\enc\\caesar_cipher.html" \l "securefile.Encrypt.caesar_cipher" \o "securefile.Encrypt.caesar_cipher)**

Example:

...

enc.caesar\_decipher**(**key\_shift=3**,** commit=**True)**

...

## Encrypt.rsa\_encrypt

Encrypt a plain text with RSA algorithm. ( **[rsa\_algorithm](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\rsa_algorithm.html" \l "module-securefile.rsa_algorithm" \o "securefile.rsa_algorithm)** )

Example:

**from** securefile **import** Encrypt

**from** securefile.keyset **import** RSA\_KEY

rsa\_private\_key = RSA\_KEY.private\_key\_genrate**(**6861**,** 57067**)**

enc = Encrypt**(**'make.txt'**)**

enc.open**()**

enc.rsa\_encrypt**(**rsa\_private\_key**)**

enc.close**()**

## Encrypt.rsa\_decrypt

Decrypt a cipher text into plain text. ( **[rsa\_encrypt](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\enc\\rsa_enc.html" \l "securefile.Encrypt.rsa_encrypt" \o "securefile.Encrypt.rsa_encrypt)** )

Example:

...

enc.open**()**

enc.rsa\_decrypt**(**rsa\_public\_key**,** commit=**True)**

enc.close**()**

## Encrypt.aes\_encrypt

Encrypt a plain text with AES algorithm. ( **[aes\_algorithm](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\aes_algorithm.html" \l "module-securefile.aes_algorithm" \o "securefile.aes_algorithm)** )

Example:

**from** securefile **import** Encrypt

**from** securefile.keyset **import** AES\_KEY

aes\_key = AES\_KEY.genrate**(**'700102030405060708090a0b0c0d0e0f'**)**

enc = Encrypt**(**'test.txt'**)**

enc.open**()**

enc.aes\_encrypt**(**aes\_key**)**

enc.close**()**

## Encrypt.aes\_decrypt

Decrypt a cipher text into plain text. ( **[aes\_encrypt](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\enc\\aes_enc.html" \l "securefile.Encrypt.aes_encrypt" \o "securefile.Encrypt.aes_encrypt)** )

Example:

...

enc.open**()**

enc.aes\_decrypt**(**des\_key**,** commit=**True)**

enc.close**()**

## Encrypt.base64\_encrypt

Base64 encoding schemes are commonly used when there is a need to encode binary data that needs be stored and transferred over media that are designed to deal with textual data. This is to ensure that the data remains intact without modification during transport.

Uses:

Encrypt a plain text into cipher text using b64encode.

Example:

**from** securefile **import** Encrypt

enc = Encrypt**(**'list\_count.txt'**,** delimiter=','**)**

enc.open**()**

enc.base64\_encrypt**()**

enc.close**()**

## Encrypt.base64\_decrypt

Decrypt a cipher text into plain text using b64decode.

Example:

...

enc.open**()**

enc.base64\_decrypt**(**commit=**True)**

enc.close**()**

## Encrypt.get\_text

Get the char-byte array of current opened file

RSA Algorithm

RSA algorithm is a public key encryption technique and is considered as the most secure way of encryption.

RSA algorithm is asymmetric cryptography algorithm. Asymmetric actually means that it works on two different keys i.e. Public Key and Private Key. As the name describes that, the Public Key is given to everyone and Private Key is kept private.

The idea of RSA is because it is difficult to factorize a large integer. The public key consists of two numbers where one number is multiplication of two large prime numbers. And private key is derived from the same two prime numbers. Therefore, if somebody can factorize the large number, the private key is compromised. Therefore, encryption strength totally lies on the key size and if we double or triple the key size, the strength of encryption increases exponentially. RSA keys can be typically 1024 or 2048 bits long, but experts believe that 1024-bit keys could be broken in the near future. However, till now it seems to be an infeasible task.

It was invented by Rivest, Shamir and Adleman in year 1978 and hence name RSA algorithm.

Advanced Encryption Standard (AES) Algorithm

The more popular and widely adopted symmetric encrypt algorithm likely to be encountered nowadays is the Advanced Encryption Standard (AES). It is found at least six time faster than triple DES.

A replacement for DES was needed as its key size was too small. With increasing computing power, it was considered vulnerable against exhaustive key search attack. Triple DES was designed to overcome this drawback but it was found slow.

The Advanced Encryption Standard (AES), also known by its original name Rijndael, is a specification for the encrypt of electronic data established by the U.S. National Institute of Standards and Technology (NIST) in 2001. AES is a subset of the Rijndael cipher developed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen, who submitted a proposal to NIST during the AES selection process. Rijndael is a family of ciphers with different key and block sizes.

The features of AES are as follows:

* Symmetric key symmetric block cipher
* 128-bit data, 128/192/256-bit keys
* Stronger and faster than Triple-DES
* Provide full specification and design details
* Software implementable in C and Java

# DES Algorithm Module

## Data Encryption Standard (DES) Algorithm

The Data Encryption Standard (DES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST). DES is an implementation of a Feistel Cipher. It uses 16 round Feistel structure. The block size is 64-bit. However, key length is 64-bit; DES has an effective key length of 56 bits, since 8 of the 64 bits of the key are not used by the encryption algorithm (function as check bits only).

[Thanks to pydes project](https://github.com/RobinDavid/pydes)

Keyset Module

A module for all key object. it provide following key object at one:

* DES key
* AES Key
* RSA Key

## Uses

**import** securefile.keyset

## Module Structure:

digraph foo {
    "KeySet Module" -> "RSA_KEY"
    "KeySet Module" -> "AES_KEY"
    "KeySet Module" -> "DES_KEY"

    "KeySet Module"[shape=ractangle]
    "RSA_KEY"[shape=ractangle]
    "AES_KEY"[shape=ractangle]
    "DES_KEY"[shape=ractangle]
}

## Module Classes:

* [RSA Key Class](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\keyset\rsa.html)
* [DES Key Class](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\keyset\des.html)
* [AES Key Class](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\keyset\aes.html)

Secure Serial Module

This module encapsulates the access for the serial port. It support Python running on Windows, OSX, Linux, BSD. This module can only use for reading RSA, AES, DES and Shift Cipher Key from Serial port.

## Dependencies:

* pyserial
  + >>> python -m pip install pyserial

## Uses:

>>> **import** securefile.secureserial

>>> **from** securefile.keyset **import** RSA\_KEY

## Module Classes:

* [Serial Class](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\serial\serial.html)
* [Scanresult](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\serial\scansesult.html)

Python Module Index

|  |  |  |
| --- | --- | --- |
|  |  |  |
| - | [**securefile**](file:///C:\Users\Prashant\Documents\GitHub\Hybrid-Cryptography-on-Cloud\docs\html\index.html#module-securefile) |  |
|  | **[securefile.aes\_algorithm](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\aes_algorithm.html" \l "module-securefile.aes_algorithm)** |  |
|  | **[securefile.des\_algorithm](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\des_algorithm.html" \l "module-securefile.des_algorithm)** |  |
|  | **[securefile.keyset](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\keyset.html" \l "module-securefile.keyset)** |  |
|  | **[securefile.rsa\_algorithm](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\rsa_algorithm.html" \l "module-securefile.rsa_algorithm)** |  |
|  | **[securefile.secureserial](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\secureserial.html" \l "module-securefile.secureserial)** |  |
|  | **[securefile.secureserial.ScanResult](file:///C:\\Users\\Prashant\\Documents\\GitHub\\Hybrid-Cryptography-on-Cloud\\docs\\html\\serial\\scansesult.html" \l "module-securefile.secureserial.ScanResult)** |  |