International Journal of Science and Research (IJSR)
ISSN: 2319-7064
ResearchGate Impact Factor (2018): 0.28 | SJIF (2018): 7.426

Implementation of DES Algorithm in Python

Sakshi Agarwal[1], P K Bharti[2], Rajesh Kumar Pathak[3]

[1]Research Scholar, Department of CSE, Maharishi University of Information Technology, Lucknow, India

[2]Professor, Department of CSE, Maharishi University of Information Technology, Lucknow, India

[3]Professor, Department of CSE, VGI, Ghaziabad, India

Abstract: ***This paper overviews the implementation of DES algorithm in python language. It illustrates underlying ideas and common techniques without going into too many details on each topic. Comparative study between implementation of DES algorithm in Python language and Java language is also illustrated.***

Keywords: Encryption, Decryption, Python, Python Cryptography, Java, One-time Padding key

1. Introduction

1.1 Cryptography

Cryptography implies to create written or generated codes to keep our information secure. It converts the information into a format that is unreadable for an unauthorized user .It also allows it to be transmitted without unauthorized entities decoding it back into a readable format. Information security uses cryptography on several levels. The information maintains its integrity during transit and while being stored. Cryptography aids in nonrepudiation, information security, authentication of data. This implies that the sender and the delivery of a message can be verified [1].

1.1.1 Types of Cryptography

1.1.1.1 Symmetric Key Cryptography

Symmetric-key cryptography refers to encryption in which both the sender and receiver shares the same key. Symmetric key ciphers can be implemented in two ways, either block ciphers or stream ciphers. A block cipher enciphers input in blocks of plaintext as opposed to individual characters, the input form used by a stream cipher.

The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are block ciphers that have been designated cryptography standards by the US government. In a stream cipher, the output stream is created based on a hidden internal state that changes as the cipher operates. That internal state is initially set up using the secret key material [3].

- **Data Encryption Standards (DES)**

It is a symmetric-key algorithm for the encryption and decryption of electronic data. Although insecure, it was highly significant in the advancement of current era cryptography [1].

It was developed in the early 1970s at IBM and based on an earlier design by Feistel structure, the algorithm was submitted to the National Bureau of Standards (NBS). DES is based on block cipher technology .It is an algorithm that uses a fixed-length string of plain text bits and transforms it through a series of complex operations into another cipher text string of the same length. DES is having the block size of 64 bits.

1.1.1.2 Asymmetrc Key Cryptography

Public-key cryptography is also known as asymmetric key cryptography .It is a cryptographic system that requires pairs of keys as public keys which can be circulated widely, and private keys which are known only to the user. This system accomplishes two function i.e. authentication and encryption.

Asymmetric Key Cryptography Techniques

- Diffie-Hellman key agreement
- Rivest-Shamir Adleman (RSA)
- Elliptic Curve Cryptography (ECC)
- El Gamal
- Digital Signature Algorithm (DSA).

1.2 History of Python

Python, was developed in an educational environment. Guido van Rossum created Python over the 1989/1990 winter holidays while working as a researcher in Amsterdam, who named it after Monty Python's Flying Circus. The result was Python (named, by the way, a big fan of Monty Python's Flying Circus). He released Python via Internet FTP distribution and carries on developing and improving the language for its increasing amount of programmers and users [9].

Python	
Paradigm-	multi-paradigm: object-oriented, imperative,functional, reflective
Appeared in	1991
Designed by	Guido van Rossum
Developer	Python Software Foundation
Stable release	3.1.2/ March 21, 2010 2.7/ July 3, 2010
Typing discipline	duck, dynamic, strong
Major	C Python, Iron Python, Jython, Python for S60, Py Py, Unladen Swallow Implementations.
Dialects	Stackless Python, R Python
Influenced by	ABC, ALGOL C, Haskell, Icon, Lisp, Modula-3, Perl, Java

Influenced	Boo, Cobra, D, Falcon, Groovy, Ruby, JavaScript
OS	Cross-platform
License	Python Software Foundation License
Usual file extensions	.py, .pyw, .pyc, .pyo, .pyd
Website	www.python.org

Python is a high-level programming language whose design philosophy emphasizes code readability. Python aims to combine "remarkable power with very clear syntax". It structures a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts [7].

2. Comparative Study

2.1 Comparison of Python and Java

I present a list of side-by-side comparisons of features of Java and Python. If you look at these comparisons, they are based on running time required by the respective language [5,6].

Table 1: Time is measured in seconds

S.no	Test	Java	Python	Comparison
1	Standard Output	138.8	30.58	Python 4.5X Faster than Java
2	Hash table	17.0	8.22	Python 2X Faster than Java
3	I/O	56.72	47.36	Python 1.2X Faster than Java
4	List	5.94	14.32	Java 2.4X Faster than Python
5	Native Methods	2.475	0.04	Python 6.3X Faster than Java
6	Object Allocation	23.65	211.11	Java 8X Faster than Python
7	Interpreter Speed	0.43	2.29	Java 5.3X Faster than Python

3. Python Cryptography

Cryptography is a package which delivers cryptographic recipes and primitives to Python developers. Cryptographic standard library supports Python 2.7, Python 3.4+, and PyPy 5.3+.Cryptography includes both high level recipes and low level interfaces to common cryptographic algorithms such as symmetric ciphers, message digests, and key derivation functions. For example, to encrypt something with cryptography's high level symmetric encryption recipe:

```
from cryptography.fernet import Fernet
>>> # Put this somewhere safe!
>>> key = Fernet.generate_key()
>>> f = Fernet(key)
>>> token = f.encrypt(b"A really secret message. Not for prying eyes.")
>>> token
>>> f.decrypt(token)
'A really secret message. Not for prying eyes.'
```

3.1 Installation

We can install cryptography with pip. The wheel package on Windows is a statically linked build (as of 0.5) so all dependencies are included. To install, you will typically just run pip install cryptography [12].

3.3 Layout

Cryptography is broadly divided into two levels.

1) Safe cryptographic recipes required to no configuration choices. These are easy and safe to use and don't require developers to make many decisions.
2) Low-level cryptographic primitives work is referred to as the "hazardous materials" or "hazmat" layer. They exists in the (cryptography .hazmat) package. We recommend using the recipes layer whenever possible, the hazmat layer is used only when necessary [10].

The recipes layer:

(I) Fernet (symmetric encryption)

Fernet is an implementation of symmetric (also known as "secret key") authenticated cryptography. Fernet also has support for implementing key rotation via Multi Fernet.

class cryptography.fernet.Fernet(key)

This class provides both encryption and decryption facilities.

```
>>> from cryptography.fernet import Fernet
>>> key = Fernet.generate_key()
>>> f = Fernet(key)
>>> token = f.encrypt(b"my deep dark secret")
>>> token
b'...'
>>> f.decrypt(token)
b'my deep dark secret'
```

classmethod generate_key() : Generates a fresh fernet key.

Encrypt (data) : Encrypts the data that is passed. The result of this encryption is known as a "Fernet token". It ensures strong privacy and authenticity.

Decrypt (token, ttl=None) : Decrypts a Fernet token. We will receive the original plaintext as a result if it is successfully decrypted, otherwise an exception will be raised.

token (bytes) – The Fernet token.

classcryptography.fernet.MultiFernet(fernets) : This class implies the rotation of key for Fernet. It requires a list of Fernet instances and implements the same API with the exception of one additional method: MultiFernet.rotate().

```
>>> from cryptography.fernet import Fernet, MultiFernet
>>> key1 = Fernet(Fernet.generate_key())
>>> key2 = Fernet(Fernet.generate_key())
>>> f = MultiFernet([key1, key2])
>>> token = f.encrypt(b"Secret message!")
>>> token
>>> f.decrypt(token)
b'Secret message!
```

key (bytes) – A URL-safe base64-encoded 32-byte key. This must be kept secret. Anyone with this key is able to create and read messages.

3.4 Using passwords with Fernet

Passwords with Fernet can be used very easily.To do this, you need to run the password through a key derivation function such as PBKDF2HMAC, bcrypt or Scrypt [11].

```
>>> import base64
>>> import os
>>> from cryptography.fernet import Fernet
>>> from cryptography.hazmat.backends import default_backend
>>> from cryptography.hazmat.primitives import hashes
>>> from cryptography.hazmat.primitives.kdf.pbkdf2
import PBKDF2HMAC
>>> password = b"password"
>>> salt = os.urandom(16)
>>> kdf = PBKDF2HMAC(
...     algorithm=hashes.SHA256(),
...     length=32,
...     salt=salt,
...     iterations=100000,
...     backend=default_backend()
... )
>>> key = base64.urlsafe_b64encode(kdf.derive(password))
>>> f = Fernet(key)
>>> token = f.encrypt(b"Secret message!")
>>> token
b'...'
>>> f.decrypt(token)
b'Secret message!'
```

Implementation

Conceptually, Fernet takes a user-provided message (an arbitrary sequence of bytes), a key (256 bits), and the current time, and produces a token, which contains the message in a form that can't be read or altered without the key. All cryptographic function in this version is done with AES 128 in CBC mode. All base 64 encoding is done with the "URL and Filename Safe" variant, defined in RFC 4648 as "base64url".

Key Format

A fernet key is the base64url encoding of the following fields:
Signing-key||Encryption-key
Signing-key, 128 bits
Encryption-key, 128 bits

Token Format

A fernet token is the base64 URL encoding of the concatenation of the following fields:
Version||Timestamp||IV||Cipher text
Version, 8 bits
Timestamp, 64 bits
IV, 128 bits
Cipher text, variable length, multiple of 128 bits
HMAC, 256 bits

Token Fields

Version: This field symbolizes the version of the format that is used by the token. Presently, there is one version well defined, with the value of 128 (0x80).

Timestamp: The field is a 64-bit unsigned big integer. It records the number of seconds elapsed between January 1, 1970 UTC and the time, the token was generated.

Initialization Vector (IV): The 128-bit Initialization Vector used in AES encryption and decryption of the Ciphertext. While generating new fernet tokens, the Initialization vector must be chosen uniquely for each token.

Ciphertext: This field has variable size, but is always a multiple of 128 bits, the AES block size. It contains the original input message, padded and encrypted.

HMAC: This field is the 256-bit SHA256 HMAC, under signing-key, of the concatenation of the following fields:

Version||Timestamp||IV||Cipher text

Generating

If a key and a message is given , we can create a Fernet token with the following steps :

1) Record the current time for timestamp field.
2) Choose a unique Initialization vector.
3) Construct the ciphertext:

Verifying

If a key and token is given , verification can be done in the following steps :

1) Base 64 URL decrypt the token.
2) Confirm that the initial byte of token is 0x80.
3) If the user has indicated the maximum age for the token, ensure the recorded timestamp is not too far in the past.
4) Re-Compute the HMAC from the other fields and the user-supplied signing-key.
5) Ensure the recomputed HMAC matches the HMAC field stored in the token, using a constant-time comparison function.
6) Decrypt the cipher text field using AES 128 in CBC mode with the recorded IV and user-supplied encryption-key.
7) Unpad the decrypted plaintext, yielding the original message.

4. DES Implementation in Python

```
In [1]: from cryptography.fernet import Fernet
In [2]:cipher_key = Fernet.generate_key()
In [3]:cipher_key
Out[3]:b'g6EWtxBVirwbIPwjsD44CAiSEXgrbYDOm9P9Ii
ybXxM='
In [4]:cipher = Fernet(cipher_key)
In [5]:text = b'My super secret message'
In [6]:encrypted_text = cipher.encrypt(text)
In [7]:encrypted_text
Out[7]:b'gAAAAABbYA2vPRk4nw_QtZDZlbuBX6av2Tt1
_Aheu-GvIPpuMeqg3lRpwb1nIxzYaP6NTzaZDf-
QeD1i2qYc29Wgmh4sCPg5dImIkPykc3oD_f9RjBW4TAg
='
In [8]:decrypted_text = cipher.decrypt(encrypted_text)
In [9]:decrypted_text = cipher.decrypt(encrypted_text)
In [10]:decrypted_text
Out[10]:b'My super secret message'
```

5. Conclusion

Python is used to develop the source code for DES algorithm. The implementation of Data Encryption Standards (DES) have some advantages like

1) It is a robust programming language and gives an easy usage of the code lines.
2) The debugging and maintenance of DES algorithm is done easily with this language.
3) The code length is minimized as compared to Java.
4) The syntax and semantics are very easy as compared to Java.
5) Byte code of DES is generated by the virtual machine using shell in Python.

References

[1] Data Encryption Standard, Federal Information Processing Standards Publication (FIPS PUB) 46, National Bureau of Standards, Washington, DC (1977).
[2] Electronic Frontier Foundation, Cracking DES, Secrets of Encryption Research, Wiretap Politics & Chip Design, O'reilly, 1998.
[3] Eli Biham, Adi Shamir, Differential Cryptanalysis of DES-like Cryptosystems, Journal of Cryptology, Vol. 4 No. 1, Springer, pp. 3– 72, 1991.
[4] "Exhaustive Cryptanalysis of the NBS Data Encryption Standard" by Diffie, Whitfield and Martin Hellman.
[5] "Java language "[online].Available: https://docs.oracle.com.
[6] Java SE Specifications - Oracle Docs[online].https://docs.oracle.com.
[7] "Keyczar." [Online]. Available: https://github.com/google/keyczar
[8] Lutz Prechelt and Barbara Unger,Technical Report 1/1999, Fakult¨at f¨ur Informatik, Universit¨at Karlsruhe, Germany, March 1999. ftp.ira.uka.de.
[9] P. Gorski and L. L. Iacono, "Towards the usability evaluation of security apis," .
[10] S. Willden, "Keyczar Design Philosophy," 2015. [Online]. Available: https://github.com/google/keyczar/wiki/KeyczarPhilosophy
[11] "The Sodium crypto library (libsodium)." [Online]. Available: https://libsodium.org.
[12] Watts S. Humphrey. A Discipline for Software Engineering. SEI series in Software Engineering. Addison Wesley, Reading, MA, 1995.