© 2018 IJSRSET | Volume 4 | Issue 4 | Print ISSN: 2395-1990 | Online ISSN : 2394-4099
Themed Section : Engineering and Technology

Network Security using Python

Dr. S. Vidhya

Associate Professor, Department of Information Technology, KG College of Arts and Science, Coimbatore, Tamilnadu, India

ABSTRACT

We uses large amount of information every day. Data is the most valuable factor. Cryptography allows people to keep confidence in the electronic world. Cryptography includes a large number of algorithms which are used in developing secure applications. Many Programming languages are frequently used to implement cryptographic function. Python is the widely used programming language to implement cryptography functions. In this paper I am going to analysis the various functions available in the Python Cryptographic Toolkit (PyCrypto).

Keywords : PyCrypto, Cryptography, DES, AES, API, MD2, MD4, MD5, RIPEMD and SHA

I. INTRODUCTION

Top business organizations spend billions of amount every day to secure their computer networks and to preserve their data safe. As the complication of the systems and the networks are increasing, vulnerabilities are also increasing and the task of securing the networks is becoming more complex. This leads Network Security a vital part of today's businesses. Cryptography is a science that applies complex mathematics and logic to design strong encryption methods. Cryptography is also an art.

Many programming languages are sprinkled all over the Internet, more and make a lot of people confused to select appropriate one. The mostly used programming languages for cryptography are python, Golang, Ruby, C##, Java and Haskell[1].

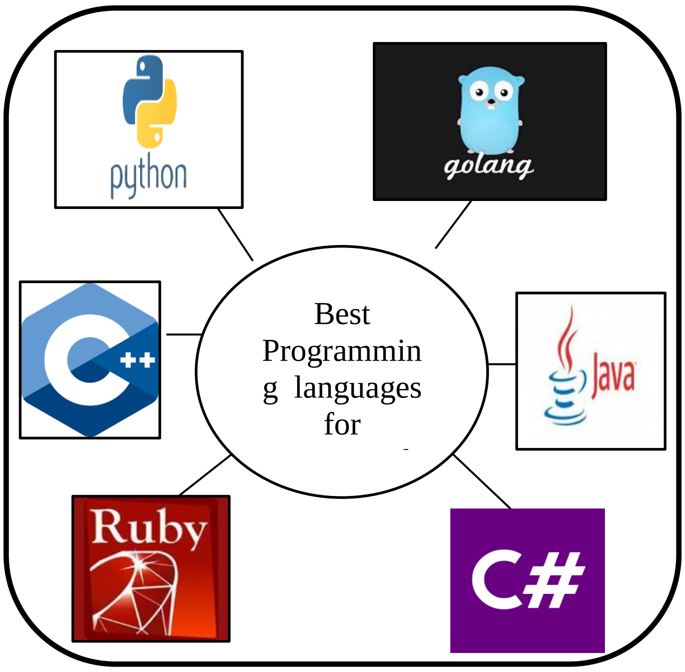

Figure 1. Best programming languages for cryptography

II. BASICS OF CRYPTOGRAPHY

Cryptography is the science of using mathematics to encrypt and decrypt data. Cryptography allows you to store sensitive information or transmit it through insecure networks so that it cannot be read by anyone except the authorized recipient. The conventional set of cryptograph algorithms based on three groups.

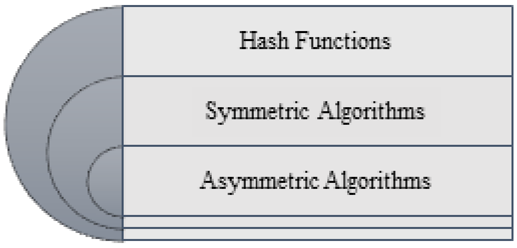

Figure 2. Fundamentals of Cryptography algorithms

A. Hash Functions

A cryptographic hash function is a type of algorithm that can be run on a part of data, like an individual file or a password, to produce a value called a checksum. The main use of a cryptographic hash function is to verify the authenticity of a part of data. Two files can be guaranteed to be identical only if the checksums generated in each file, using the same cryptographic hash function, are equal.

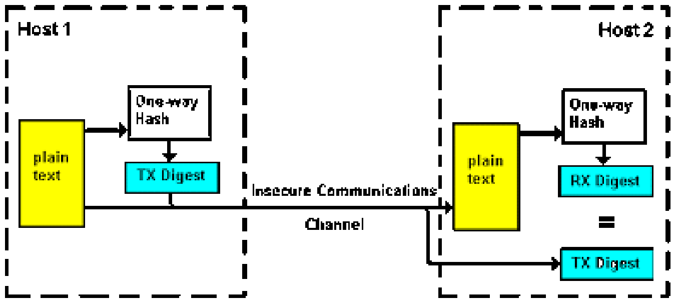

Figure 3. Secure hash algorithm

It is also used as an one way function that is the checksum is created only from the message but not vice versa. Hash functions can be used for integrity check, or, in link with a public-key algorithm, can be used to implement digital signatures[2].

B. Symmetric Algorithms

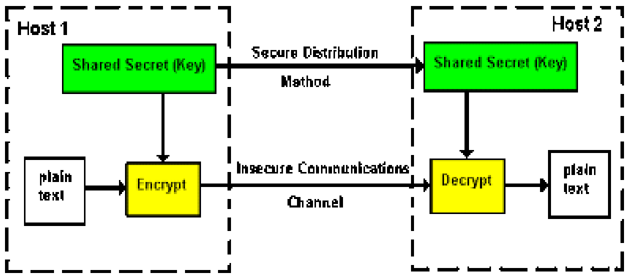

Figure 4. Symmetric Encryption

In secret or symmetric key algorithm, the key is a shared secret between two communicating parties. Encryption and decryption both use the same key.

The Data Encryption Standard (DES) and the Advanced Encryption Standard (AES) are examples of symmetric key algorithms.

C. Asymmetric Algorithms

Asymmetric encryption uses a pair of keys such as public and private keys. Anything encrypted with one of the keys, can only be decrypted with the other. The public key can also be used for sender authentication in certain scenarios.

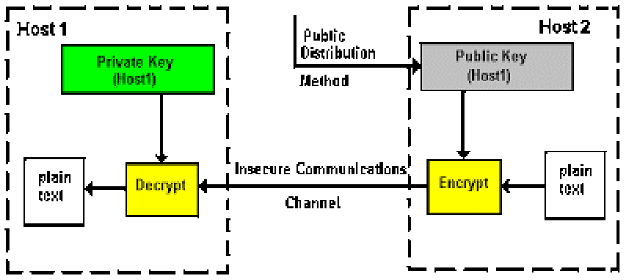

Figure 4. Asymmetric Encryption

The two keys are mathematically related, but it is impossible to find the private key from the public key. The RSA (after the inventors Rivest, Shamir, and Adelman) algorithm is an example of a public key algorithm[4].

III. THE PYTHON CRYPTOGRAPHIC TOOLKIT

The PyCrypto is a Python Cryptographic Toolkit. This is a collection of secure hash functions and different types of encryption algorithms. The Python cryptography toolkit is aimed to provide a reliable and stable base for writing cryptographic functions in python programming language.

There are many Python libraries to implement cryptography functions such as M2Crypto, PyCrypto, pyOpenSSL, python-nss, and Botan's Python bindings. The widely used library is PyCrypto.

A. Hash Functions

Cryptographic hash functions take arbitrary binary strings as input, and produce a fixed-length output called checksum or digest. It is impossible to derive the original input data from the digest. The cryptographic hash function is one-way (pre-image resistance). Given the digest of one message, it is also impossible to find another message (second pre-image) with the same digest (weak collision resistance). Finally, it is infeasible to find two arbitrary messages with the same digest (strong collision resistance)[3].

An API for Cryptographic Hash Functions look like this.

```
>>> from Crypto.Hash import SHA256
>>> hash = SHA256.new()
>>> hash.update('message')
>>> hash.digest()
'\xabS\n\x13\xe4Y\x14\x98+y\xf9\xb7\xe3\xfb\
xa9\x94\xcf\xd1\xf3\xfb"\xf7\x1c\xea\x1a\xfb\
xf0+F\x0cm\x1d'
```

Whenever we want to hash a message, we have to create a new hash object with the new() function in the relevant algorithm module. The hashing algorithms currently implemented are MD2, MD4, MD5, RIPEMD and SHA.

B. Symmetric Algorithms

Encryption algorithms take some text as input and produce ciphertext using a variable key. AES is very fast and secure, and it is the de facto standard for symmetric encryption. An API for Cryptographic Hash Functions look like this.

```
>>> from Crypto.Cipher import AES
>>> from Crypto.Random import
      get_random_bytes
>>> key = b'Sixteen byte key'
>>> iv = get_random_bytes(16)
>>> cipher = AES.new(key, AES.MODE_CFB, iv)
    >>> msg = iv + cipher.encrypt(b'Attack at
dawn')
```

The PyCrypto block-level encryption API is very low-level, it accepts the key to be either 16, 24 or 32 bytes long For AES, the standard key sizes are 128,192 and 256. The longer the key provides the stronger encryption.

C. Asymmetric Algorithms

It is easy to generate a private/public key pair with pycrypto. The key size in terms of bits is one of the factor. In an given example the key size is 1024 bits. Larger is more secure. Many public key algorithms can also be used to sign messages; run the message to be signed through a decryption with private key key[5].

```
>>> from Crypto.PublicKey import RSA
>>> from Crypto import Random
>>> random_generator =
Random.new().read
>>> key = RSA.generate(1024,
random_generator)
>>> key
<_RSAobj @0x7f60cf1b57e8
n(1024),e,d,p,q,u,private>
```

Anyone receiving the message can encrypt it with available public key and read the message. Some algorithms do only encryption, others can both encrypt and authenticate.

IV. CONCLUSION

Python is a high-level, interpreted and general-purpose dynamic programming language that emphases on code readability. The syntax in Python comforts the programmers to do coding in fewer steps as compared to Java or C++. It downloads extensive library. It contains web browsers, threading, cryptography functions CGI, email, image manipulation etcOther subpackages of the Pycrypto includes; Crypto.Cipher, Crypto.Hash, Crypto.Protocol, Crypto.PublicKey, Crypto.Signature, and Crypto.Util

V. REFERENCES

[1]. Siyaram Gupta and Madhu Sharma, "A Python Based Enhanced Secret Sharing Scheme to Secure Information using Cryptography Techniques, International Journal of Advanced Science and Technology, Volume 71, pp. 15-20.

[2]. D. Litzenberger. Official PyCrypto website. https: // ww.dlitz.net / software/pycrypto/.

[3]. Toshima Singh Rajput, Kamini Maheshwar, Dr. Taruna Jain, Algorithmic Approach to Encrypted Modes of Transmission of Real Time Media in a VOIP Architecture, International Journal of Scientific Research in Computer Science, Engineering and Information Technology, Volume 3, Issue 1, pp. 1876 – 1880.

[4]. R. N. Iyare, S. D. Walker, Improved High Definition Multimedia Interface Authentication Mechanism, Journal of Computer and

Communications Vol.02 No.12(2014), Article ID:50705,7 pages.

[5]. Dr. Vidhya, An Efficient Encryption Scheme for Small Arbitrary Length Domains, International Journal for Research in Applied Science and Engineering Technology, Volume 6, Issue 1, pp. 2890 -2893.