
**SIES (Nerul) College of Arts, Science and Commerce**

**NAAC Re-Accredited 'A' Grade**

**Sri Chandrasekarendra Saraswathy Vidyapuram, Plot 1-C,**

**Sector V, Nerul, Navi Mumbai-400 706.**

**PROJECT REPORT ON**

# *Cryptography*

***(hashing,digital signature & checksum)***

**Submitted By:**

**Darren Dsouza**

**(ROLL NO:02)**

**[ MSc Computer Science Part-1 (SEM-1) ]**

**2022-2023**

**Subject: Software Defined Networking**

**Project Guide : Prof. Flosia Simon**

What is Cryptography?

-Cryptography is technique of securing information and communications through use of codes so that only those persons for whom the information is intended can understand it and process it.

-Thus preventing unauthorized access to information. The prefix “crypt” means “hidden” and suffix graphy means “writing”.

What is Hashing?

-Hashing is the process of transforming any given key or a string of characters into another value**.**

What is Digital signature?

- A digital signature is an electronic, encrypted, stamp of authentication on digital information such as email messages, macros, or electronic documents.

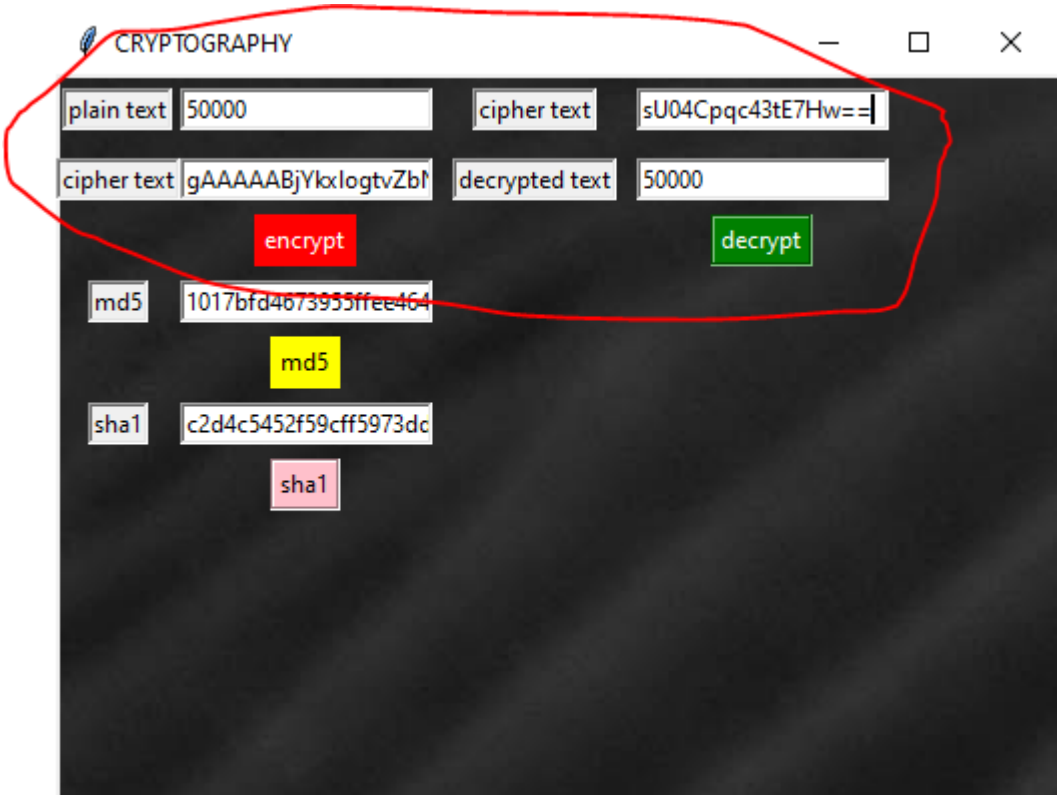
What is checksum ?

-A checksum is a value that represents the number of bits in a transmission message and is used by IT professionals to detect high-level errors within data transmissions.

-It is also used in cryptography check data has not been altered.

Types of Hashing:

Symmetric encryption:

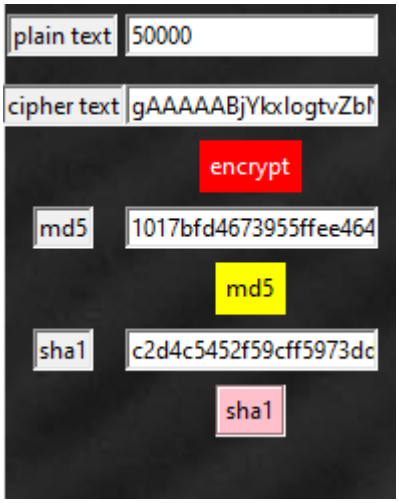



-In this project demonstration,I have first implemented symmetric encryption using fernet encryption which uses a single key for encryption of data.below is an image showing the use of a single key.

```
    pt = e1.get()
    key=Fernet.generate_key()
    global f
    f=Fernet(key)
    ct = f.encrypt(bytes(pt,encoding='ascii'))
    e2.insert(0, ct)

def decryptMessage():
    ct1 = e3.get()
    pt1 = f.decrypt(bytes(ct1,encoding='ascii'))
    e4.insert(0, pt1)
```

MD5 & SHA1:




One of the prominently used encryption methods:MD5 & SHA1

The MD5 message-digest algorithm is a cryptographically broken but still widely used hash function producing a 128-bit hash value.The above image shows the output in hexadecimal notation.

SHA1 takes an input and produces a 160-bit (20-byte) hash value known as a message digest

Below is the code for the function:

```
def encryptMd5():
    message=e1.get()
    encode_msg=hashlib.md5(bytes(message,encoding='ascii'))
    encode_msg=encode_msg.hexdigest()
    e5.insert(0,encode_msg)

def SHA1():
    message=e1.get()
    encode_msg=hashlib.sha1(bytes(message,encoding='ascii'))
    encode_msg=encode_msg.hexdigest()
    e6.insert(0,encode_msg)
```

Digital Signature using rsa algorithm:

```
from Crypto.PublicKey import RSA

keypair=RSA.generate(bits=1024)
print(f"Publickey: (n={hex(keypair.n)},e={hex(keypair.e)}")
print(f"Pvtkey: (n={hex(keypair.n)},d={hex(keypair.d)}")
```

```
Publickey: (n=0xdf3f71e82b34fd7ea199ee39f5e82791574c7d1e24321
Pvtkey: (n=0xdf3f71e82b34fd7ea199ee39f5e82791574c7d1e24321171
```

The Rsa algorithm is used to demonstrate asymmetric key encryption. We use crypto.publickey module to have a pair of public and private keys.

below are the images to show how the messages are encrypted and decrypted:

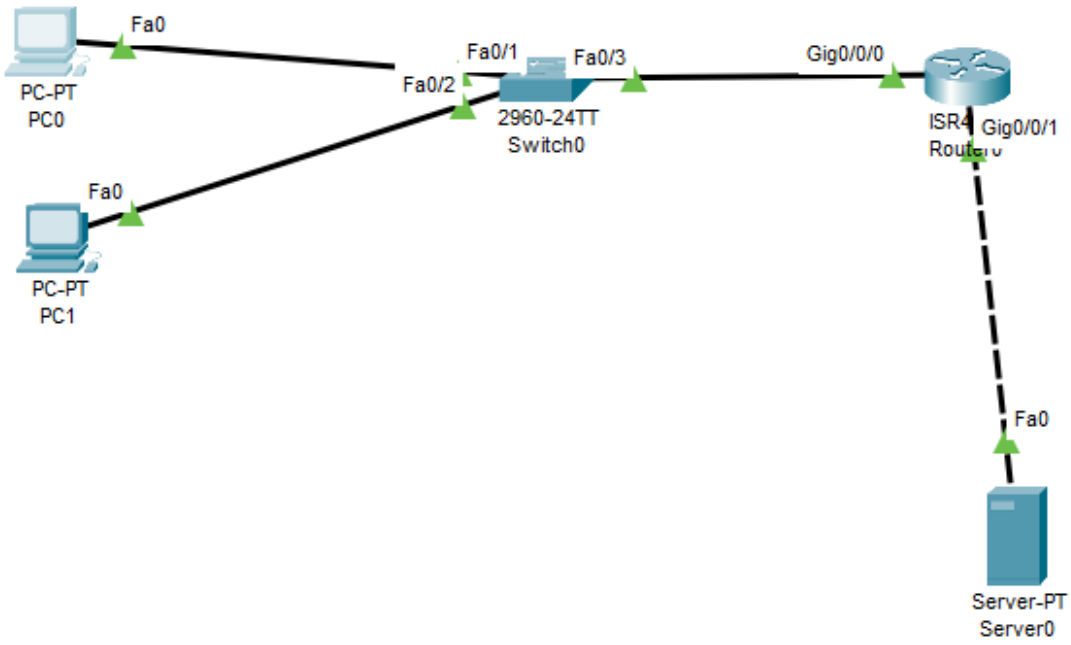
```
msg=b'4655758ebacb8cebecbfd44adfffbe39d4ed8e4b'
hash = int.from_bytes(msg,byteorder='big')
signature=pow(hash,keypair.d,keypair.n) #pvt key
print("Signature:",signature)
print("hash:",hash)
```

```
Signature: 12967738752524227337904443500202856571943028 03
hash: 435639139902041648145277274855620606150737377981295
```

```
#enter signature below to verify
signature=12967738752524227337904443500202856571943028034219810424
hashfromsignature=pow(signature,keypair.e,keypair.n) #public key
print("Signature valid:",hash==hashfromsignature)
print("hashfromsignature:",hashfromsignature)
```

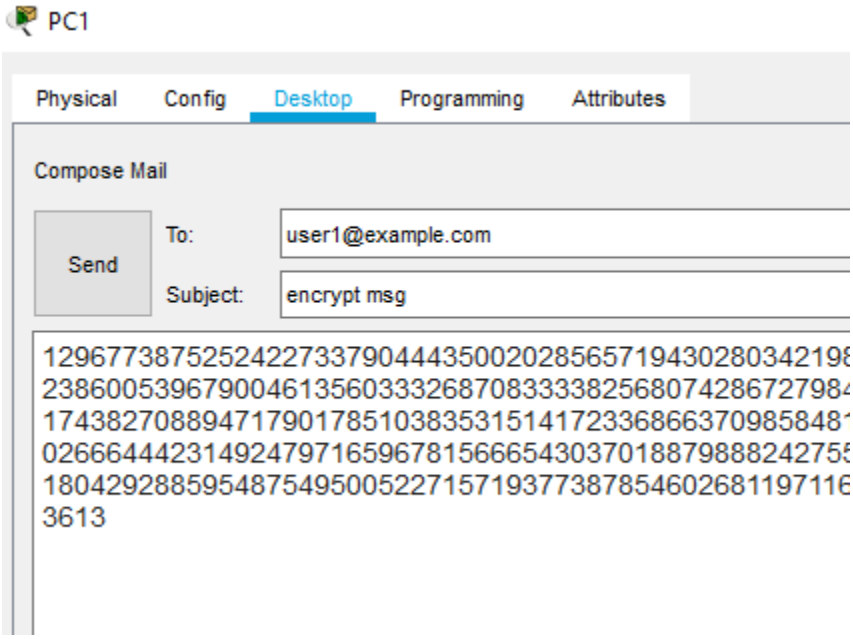
```
Signature valid: True
hashfromsignature: 435639139902041648145277274855620606150737377981
```

SMTP for message passing:

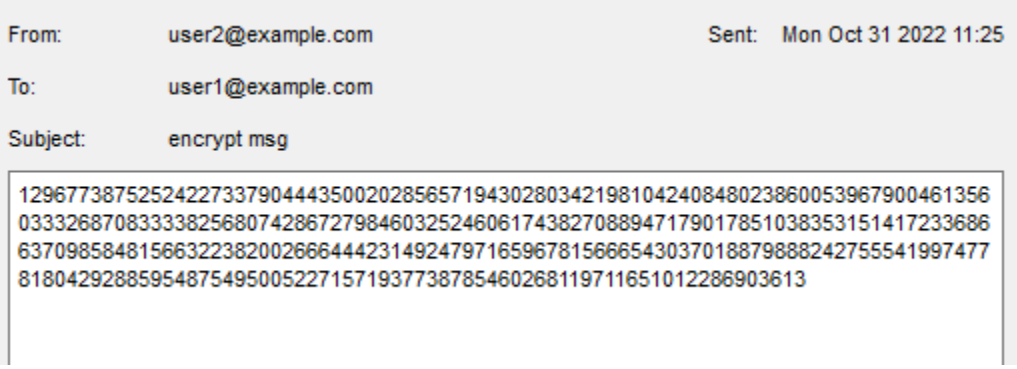



The above network is created in cisco where we use an smtp server to exchange the digital signature between two hosts and show the encryption and decryption using the keys.

sending message from user2:



Message received to user1:




The signature received is decrypted using the public key and the message is correct:

```
#enter signature below to verify
signature=1296773875252422733790444350020285657194302803421981042408
hashfromsignature=pow(signature,keypair.e,keypair.n) #public key
print("Signature valid:",hash==hashfromsignature)
print("hashfromsignature:",hashfromsignature)
```

**hashing,digital signature together helps in maintaining the CIA(confidentiality,Integrity and authenticity of the Messages being shared**

Checksum:

```
---------Sender side----------
Enter the Sum(16bit):
1111111111111111
The checksum is 0

---------Receiver side-----------
Enter the receiver side Checksum 24 bit(16+8 bit checksum)
1111111111111110
The checksum at receiver side 00000000
There is no error in transmission
>>>
```

The checksum is implemented in python which verifies the correctness of packet being exchanged between sender and receiver.the message received is correct only if the checksum at receiver is 0.

below image shows an example of error in transmission

```
---------Sender side----------
Enter the Sum(16bit):
1111111111111111
The checksum is 0

---------Receiver side-----------
Enter the receiver side Checksum 24 bit(16+8 bit checksum)
1111111101111110
The checksum at receiver side: 10000000
There is Error in transmission
>>>
```