Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Network Security and Cryptography Fundamentals

Schnorr Digital Signature

Faculty: Renukadevi Saravanan

Slot : B1+TB1

Team Members:

Jaswanth Pulipati – 19MIA1039

Ravipati Sree Karthik – 19MIA1088

Mansoor Khan Lodi – 19MIA1094

Abstract

In this project we are going to see how Schnoor Digital Signature works, firstly we are going to implement the Schnorr Digital signature based on its algorithm and then we will be showing live application/demonstration where files are encrypted using Schnorr digital signature algorithm and sent to receiver and receiver will be verifying the file and accept it.

Introduction

In cryptography, a Schnorr signature is a digital signature produced by the Schnorr signature algorithm that was described by Claus Schnorr. It is a digital signature scheme known for its simplicity, is efficient and generates short signatures. It is one of the protocols used to implement “Proof Of Knowledge”.

In cryptography, a proof of knowledge is an interactive proof in which the prover succeeds in ‘convincing’ a verifier that the prover knows something ‘X’. For a machine to know ‘X’ is defined in terms of computation. A machine knows ‘X’ if this ‘X’ can be computed. The Verifier either accepts or rejects the proof.

The signature proof is supposed to convince the Verifier that they are communicating with a user who knows the private key corresponding to the public key. In other words, the Verifier should be convinced that they are communicating with the Prover without knowing the private key.

Module Description

We have divided the project into three modules

- Creating Public and Private Keys
- Generating the hash value and creating Digital signature to the input file
- Verification of Document with the hash value and publicly available information

Hardware/Software Used

1. Jupyter Notebook (Coding Platform):

Project Jupyter is a project and community whose goal is to "develop open-source software, open-standards, and services for interactive computing across dozens of programming languages".

It was spun off from IPython in 2014 by Fernando Pérez and Brian Granger. Project Jupyter's name is a reference to the three core programming languages supported by Jupyter, which are Julia, Python and R, and also a homage to Galileo's notebooks recording the discovery of the moons of Jupiter.

Project Jupyter has developed and supported the interactive computing products Jupyter Notebook, JupyterHub, and JupyterLab. Jupyter is a NumFOCUS fiscally sponsored project.

2. Python (Programming Language):

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.

Guido van Rossum began working on Python in the late 1980s, as a successor to the ABC programming language, and first released it in 1991 as Python 0.9.0.[33] Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a cycle-detecting garbage collection system (in addition to reference counting). Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.

Python consistently ranks as one of the most popular programming languages.

3. Gmail (For transmission of message files):

Gmail is a free email service provided by Google. As of 2019, it had 1.5 billion active users worldwide. A user typically accesses Gmail in a web browser or the official mobile app. Google also supports the use of email clients via the POP and IMAP protocols.

At its launch in 2004, Gmail provided a storage capacity of one gigabyte per user, which was significantly higher than its competitors offered at the time. Today, the service comes with 15 gigabytes of storage. Users can receive emails up to 50 megabytes in size, including attachments, while they can send emails up to 25 megabytes. In order to send larger files, users can insert files from Google Drive into the message. Gmail has a search-oriented interface and a "conversation view" similar to an Internet forum. The service is notable among website developers for its early adoption of Ajax.

4. SHA-1 (Cryptographic hash function):

In cryptography, SHA-1 (Secure Hash Algorithm 1) is a cryptographic hash function which takes an input and produces a 160-bit (20-byte) hash value known as a message digest – typically rendered as a hexadecimal number, 40 digits long. It was designed by the United States National Security Agency, and is a U.S. Federal Information Processing Standard.

Since 2005, SHA-1 has not been considered secure against well-funded opponents; as of 2010 many organizations have recommended its replacement. NIST formally deprecated use of SHA-1 in 2011 and disallowed its use for digital signatures in 2013. As of 2020, chosen-prefix attacks against SHA-1 are practical. As such, it is recommended to remove SHA-1 from products as soon as possible and instead use SHA-2 or SHA-3. Replacing SHA-1 is urgent where it is used for digital signatures.

All major web browser vendors ceased acceptance of SHA-1 SSL certificates in 2017. In February 2017, CWI Amsterdam and Google announced they had performed a collision attack against SHA-1, publishing two dissimilar PDF files which produced the same SHA-1 hash. However, SHA-1 is still secure for HMAC.

Microsoft has discontinued SHA-1 code signing support for Windows Update on August 7, 2020.

5. Notepad (Text Editor for writing messages):

Windows Notepad is a simple text editor for Windows; it creates and edits plain text documents. First released in 1983 to commercialize the computer mouse in MS-DOS, Notepad has been part of every version of Windows ever since.

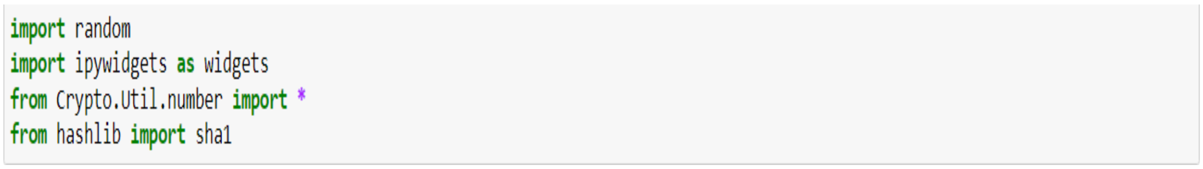
Notepad can edit text files (bearing the ".txt" filename extension) and compatible formats, such as batch files, INI files, and log files. Notepad can read and write plain texts encoded in ASCII, UTF-8, and UTF-16. It supports both left-to-right and right-to-left based languages.

6. Hardware used:

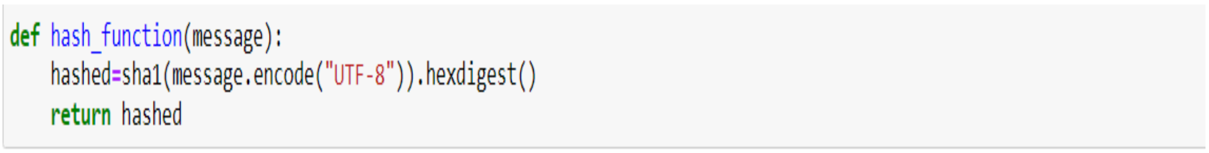
Laptop with intel i5 8th generation processor and 8gb of RAM.

Sample Algorithm and Coding with Screenshots

- We have imported the required libraries for our project, the hash function is used to read and encode the given input file

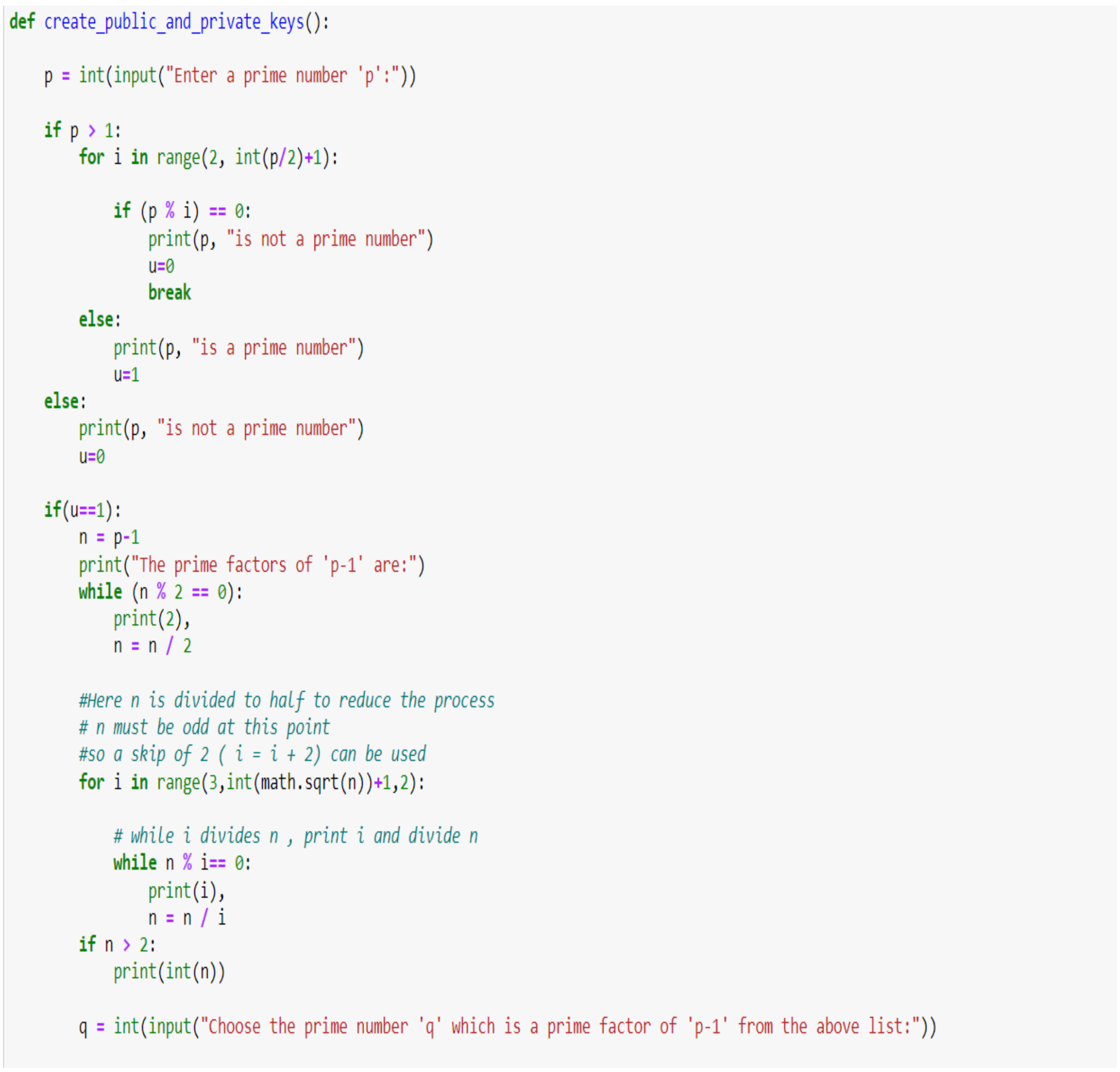


```
import random
import ipywidgets as widgets
from Crypto.Util.number import *
from hashlib import sha1
```



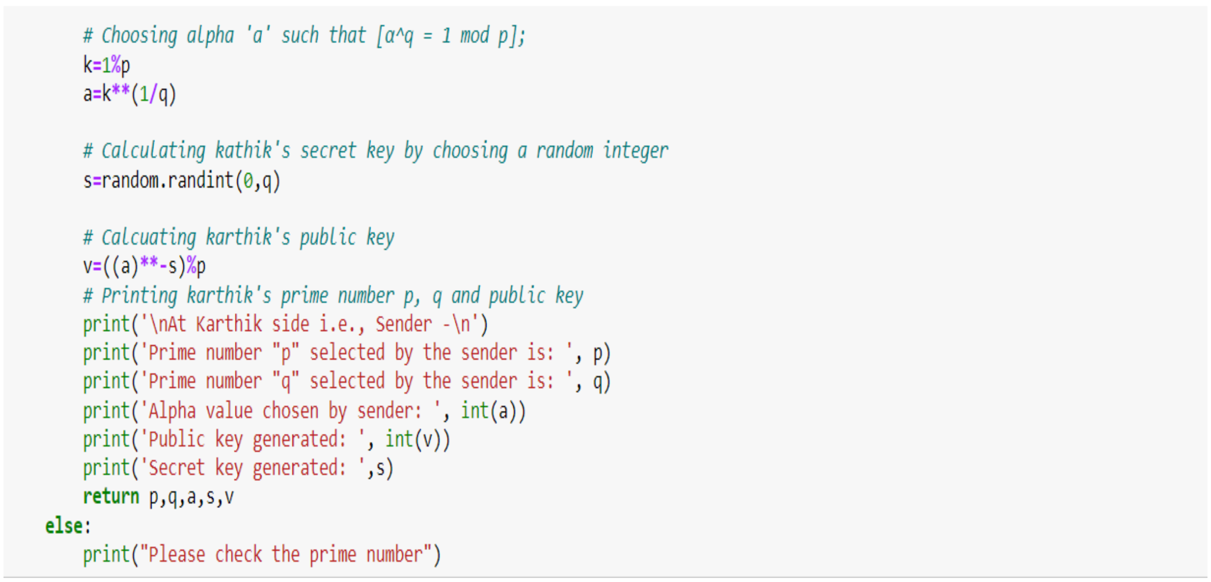
```
def hash_function(message):
    hashed=sha1(message.encode("UTF-8")).hexdigest()
    return hashed
```

- We will be creating public and private keys with the following variables
- P and q are prime numbers where q is the factor of p-1

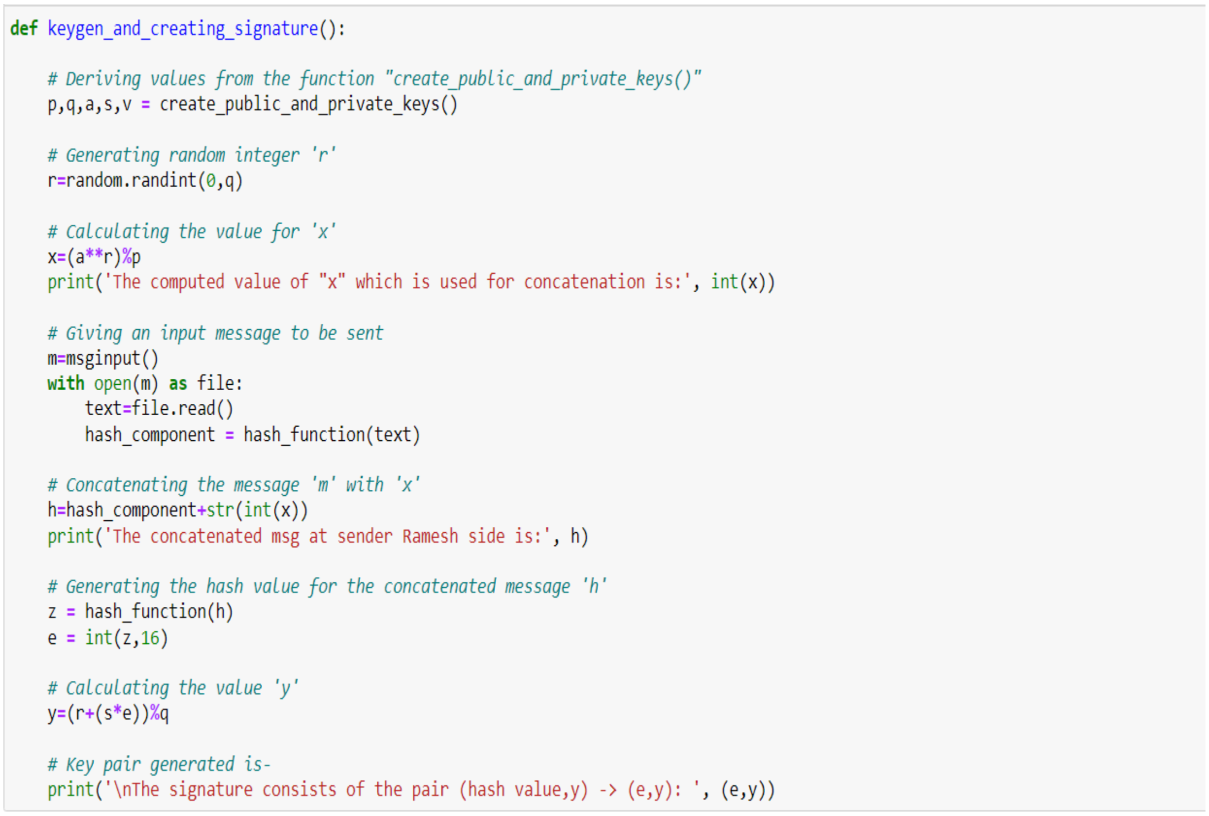


```python
def create_public_and_private_keys():

    p = int(input("Enter a prime number 'p':"))

    if p > 1:
        for i in range(2, int(p/2)+1):

            if (p % i) == 0:
                print(p, "is not a prime number")
                u=0
                break
        else:
            print(p, "is a prime number")
            u=1
    else:
        print(p, "is not a prime number")
        u=0

    if(u==1):
        n = p-1
        print("The prime factors of 'p-1' are:")
        while (n % 2 == 0):
            print(2),
            n = n / 2

        #Here n is divided to half to reduce the process
        # n must be odd at this point
        #so a skip of 2 ( i = i + 2) can be used
        for i in range(3,int(math.sqrt(n))+1,2):

            # while i divides n , print i and divide n
            while n % i== 0:
                print(i),
                n = n / i
        if n > 2:
            print(int(n))

        q = int(input("Choose the prime number 'q' which is a prime factor of 'p-1' from the above list:"))
```

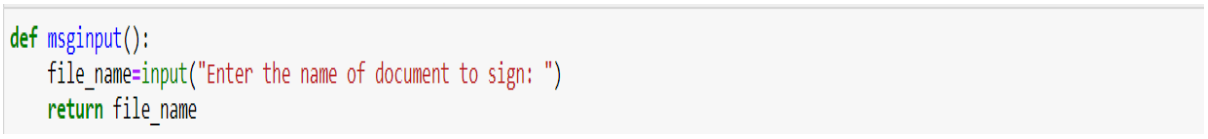
- s is the secret key of our algorithm which will be generated randomly each time.
- a is the alpha value computed by the formula [a^q = 1 mod p]
- v is the public key generated by the formula [v = a^(-s) mod p]


```
        # Choosing alpha 'a' such that [a^q = 1 mod p];
        k=1%p
        a=k**(1/q)

        # Calculating kathik's secret key by choosing a random integer
        s=random.randint(0,q)

        # Calcuating karthik's public key
        v=((a)**-s)%p
        # Printing karthik's prime number p, q and public key
        print('\nAt Karthik side i.e., Sender -\n')
        print('Prime number "p" selected by the sender is: ', p)
        print('Prime number "q" selected by the sender is: ', q)
        print('Alpha value chosen by sender: ', int(a))
        print('Public key generated: ', int(v))
        print('Secret key generated: ',s)
        return p,q,a,s,v
    else:
        print("Please check the prime number")
```

- the variable x is used for concatenation of message
- The keypair is generated hash value and y value are printed

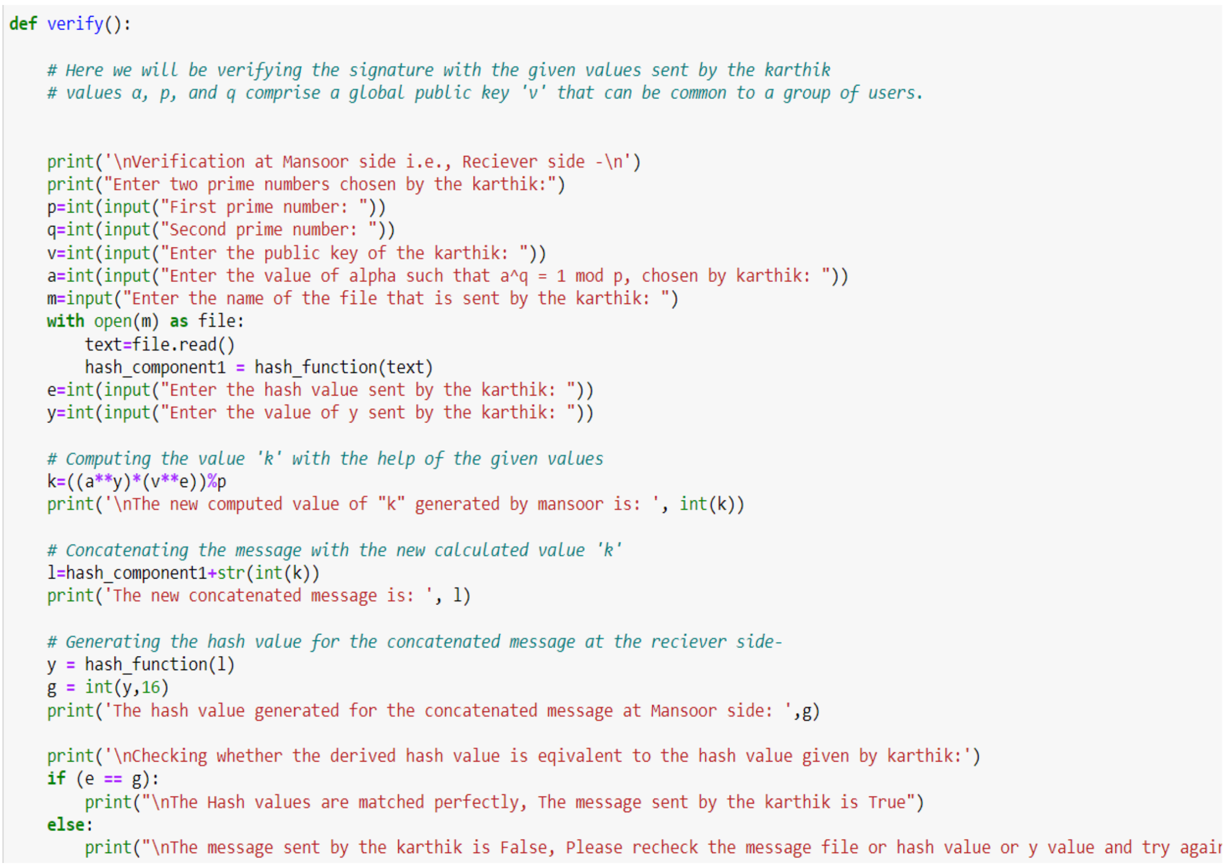

```
def keygen_and_creating_signature():

    # Deriving values from the function "create_public_and_private_keys()"
    p,q,a,s,v = create_public_and_private_keys()

    # Generating random integer 'r'
    r=random.randint(0,q)

    # Calculating the value for 'x'
    x=(a**r)%p
    print('The computed value of "x" which is used for concatenation is:', int(x))

    # Giving an input message to be sent
    m=msginput()
    with open(m) as file:
        text=file.read()
        hash_component = hash_function(text)

    # Concatenating the message 'm' with 'x'
    h=hash_component+str(int(x))
    print('The concatenated msg at sender Ramesh side is:', h)

    # Generating the hash value for the concatenated message 'h'
    z = hash_function(h)
    e = int(z,16)

    # Calculating the value 'y'
    y=(r+(s*e))%q

    # Key pair generated is-
    print('\nThe signature consists of the pair (hash value,y) -> (e,y): ', (e,y))
```

- msginput function is used to read the required document to sign. Then the signature is created.

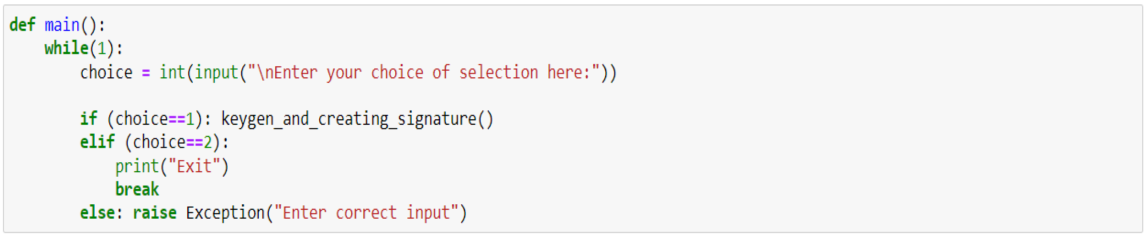


```python
def msginput():
    file_name=input("Enter the name of document to sign: ")
    return file_name
```

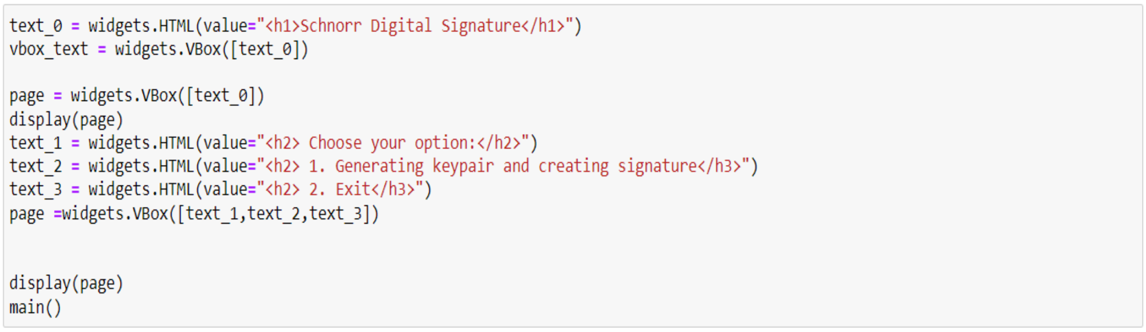
- verification happens at receiver side where inputs are taken and signature is being verified after calculations.



```python
def verify():

    # Here we will be verifying the signature with the given values sent by the karthik
    # values a, p, and q comprise a global public key 'v' that can be common to a group of users.


    print('\nVerification at Mansoor side i.e., Reciever side -\n')
    print("Enter two prime numbers chosen by the karthik:")
    p=int(input("First prime number: "))
    q=int(input("Second prime number: "))
    v=int(input("Enter the public key of the karthik: "))
    a=int(input("Enter the value of alpha such that a^q = 1 mod p, chosen by karthik: "))
    m=input("Enter the name of the file that is sent by the karthik: ")
    with open(m) as file:
        text=file.read()
        hash_component1 = hash_function(text)
    e=int(input("Enter the hash value sent by the karthik: "))
    y=int(input("Enter the value of y sent by the karthik: "))

    # Computing the value 'k' with the help of the given values
    k=((a**y)*(v**e))%p
    print('\nThe new computed value of "k" generated by mansoor is: ', int(k))

    # Concatenating the message with the new calculated value 'k'
    l=hash_component1+str(int(k))
    print('The new concatenated message is: ', l)

    # Generating the hash value for the concatenated message at the reciever side-
    y = hash_function(l)
    g = int(y,16)
    print('The hash value generated for the concatenated message at Mansoor side: ',g)

    print('\nChecking whether the derived hash value is eqivalent to the hash value given by karthik:')
    if (e == g):
        print("\nThe Hash values are matched perfectly, The message sent by the karthik is True")
    else:
        print("\nThe message sent by the karthik is False, Please recheck the message file or hash value or y value and try agai
```



```python
def main():
    while(1):
        choice = int(input("\nEnter your choice of selection here:"))

        if (choice==1): keygen_and_creating_signature()
        elif (choice==2):
            print("Exit")
            break
        else: raise Exception("Enter correct input")
```

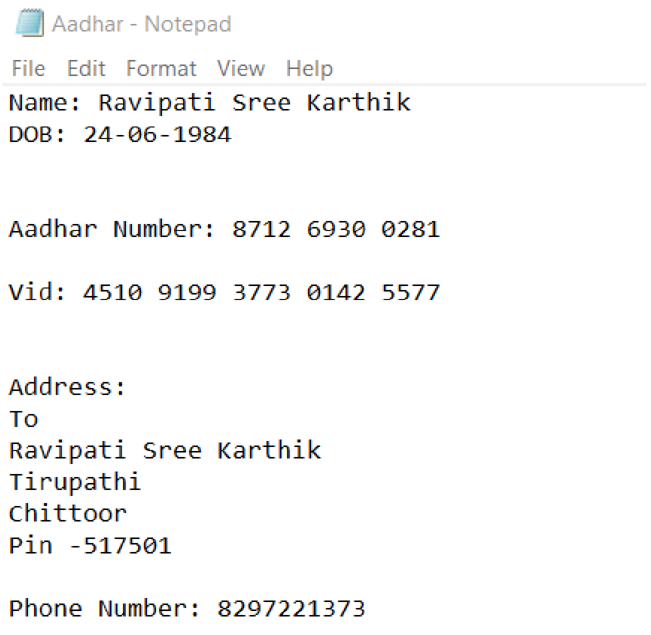


```python
text_0 = widgets.HTML(value="<h1>Schnorr Digital Signature</h1>")
vbox_text = widgets.VBox([text_0])

page = widgets.VBox([text_0])
display(page)
text_1 = widgets.HTML(value="<h2> Choose your option:</h2>")
text_2 = widgets.HTML(value="<h2> 1. Generating keypair and creating signature</h3>")
text_3 = widgets.HTML(value="<h2> 2. Exit</h3>")
page =widgets.VBox([text_1,text_2,text_3])


display(page)
main()
```

Result and Conclusion

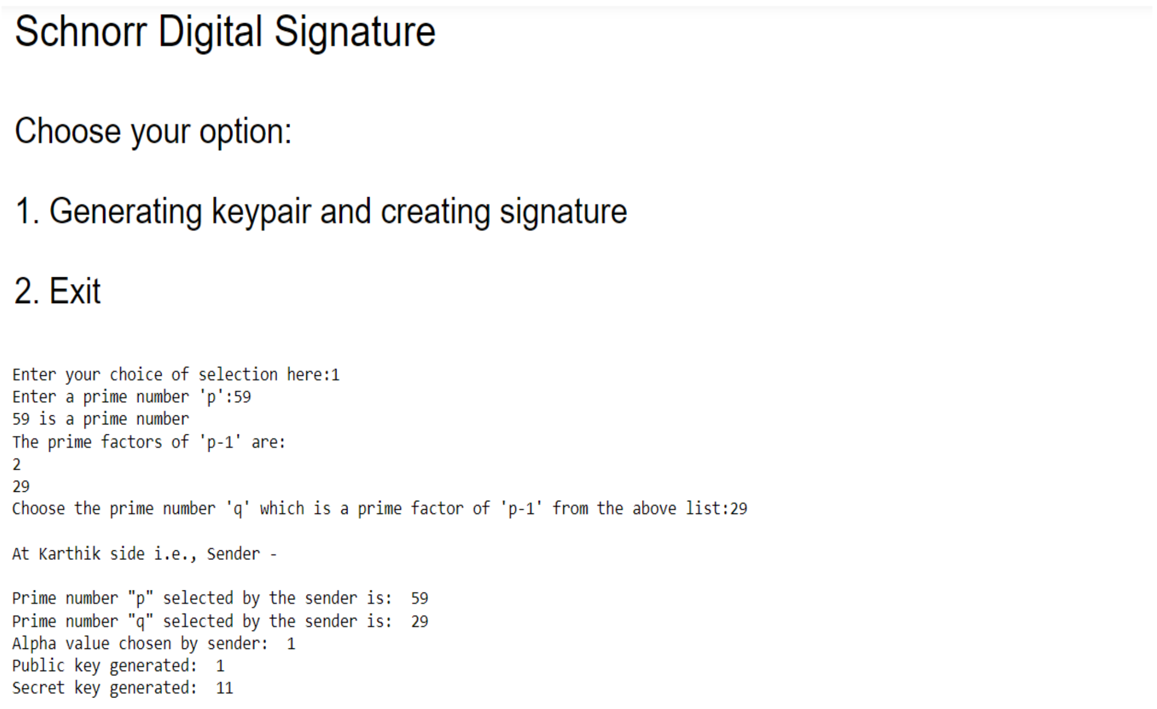
The code is being executed and here are the results.

Original text document that is to be Digitally Signed by Karthik and that is to be sent to Mansoor.

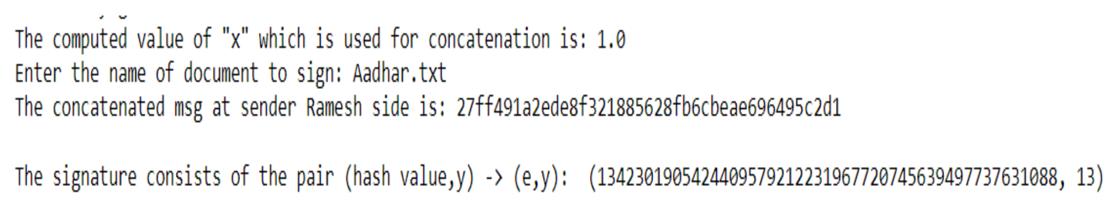


```
Name: Ravipati Sree Karthik
DOB: 24-06-1984


Aadhar Number: 8712 6930 0281

Vid: 4510 9199 3773 0142 5577


Address:
To
Ravipati Sree Karthik
Tirupathi
Chittoor
Pin -517501

Phone Number: 8297221373
```

At the sender side (Karthik) will be generating signature and will be sending them to Mansoor.



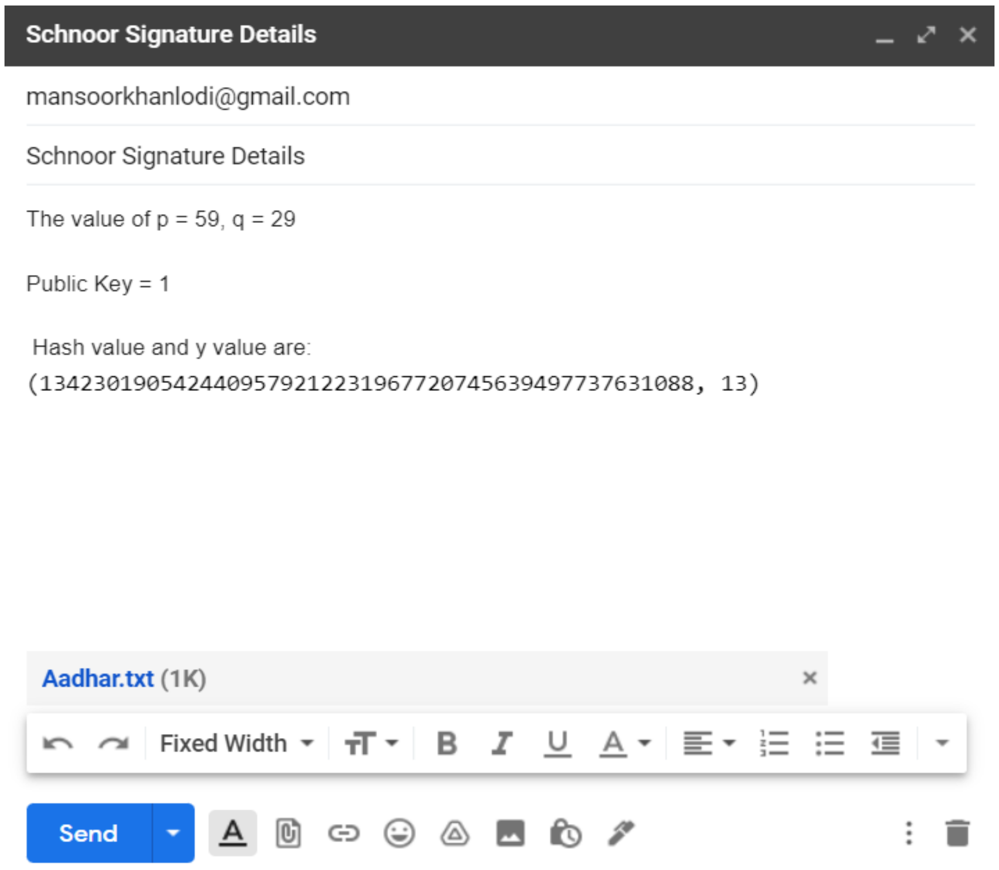
```
Schnorr Digital Signature

Choose your option:

1. Generating keypair and creating signature

2. Exit

Enter your choice of selection here:1
Enter a prime number 'p':59
59 is a prime number
The prime factors of 'p-1' are:
2
29
Choose the prime number 'q' which is a prime factor of 'p-1' from the above list:29

At Karthik side i.e., Sender -

Prime number "p" selected by the sender is:  59
Prime number "q" selected by the sender is:  29
Alpha value chosen by sender:  1
Public key generated:  1
Secret key generated:  11
```



```
The computed value of "x" which is used for concatenation is: 1.0
Enter the name of document to sign: Aadhar.txt
The concatenated msg at sender Ramesh side is: 27ff491a2ede8f321885628fb6cbeae696495c2d1

The signature consists of the pair (hash value,y) -> (e,y):  (1342301905424409579212231967720745639497737631088, 13)
```

The hash value and y value, the prime numbers p and q and the public key are sent to Mansoor using gmail

- Global variables -
- prime numbers,
- p = 59
- q = 29
- Alpha value, a = 1
- Public key, v = 1



Schnoor Signature Details

mansoorkhanlodi@gmail.com

Schnoor Signature Details

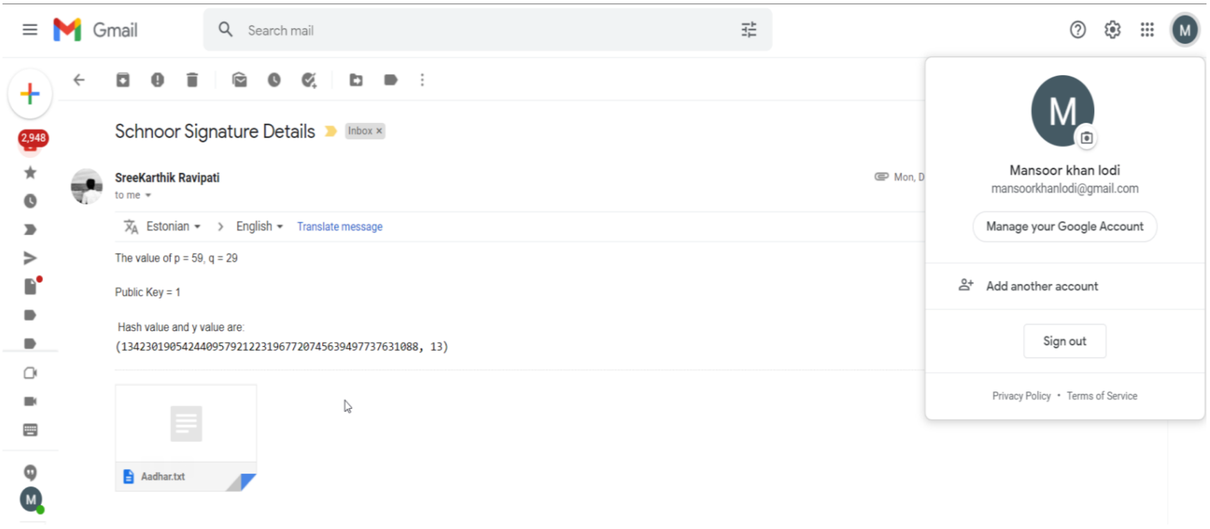
The value of p = 59, q = 29

Public Key = 1

Hash value and y value are:
(1342301905424409579212231967720745639497737631088, 13)

Aadhar.txt (1K)

Mail received at Mansoor side



Schnoor Signature Details

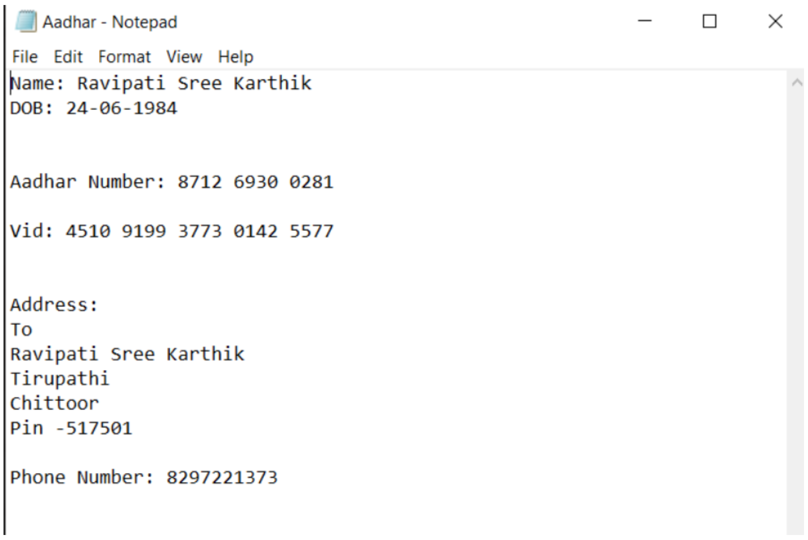
SreeKarthik Ravipati

The value of p = 59, q = 29

Public Key = 1

Hash value and y value are:
(1342301905424409579212231967720745639497737631088, 13)

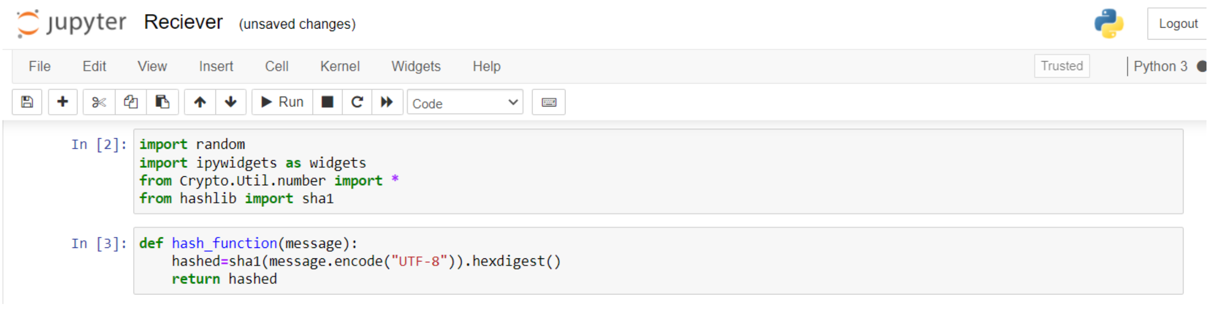
Downloaded Text Document by Mansoor



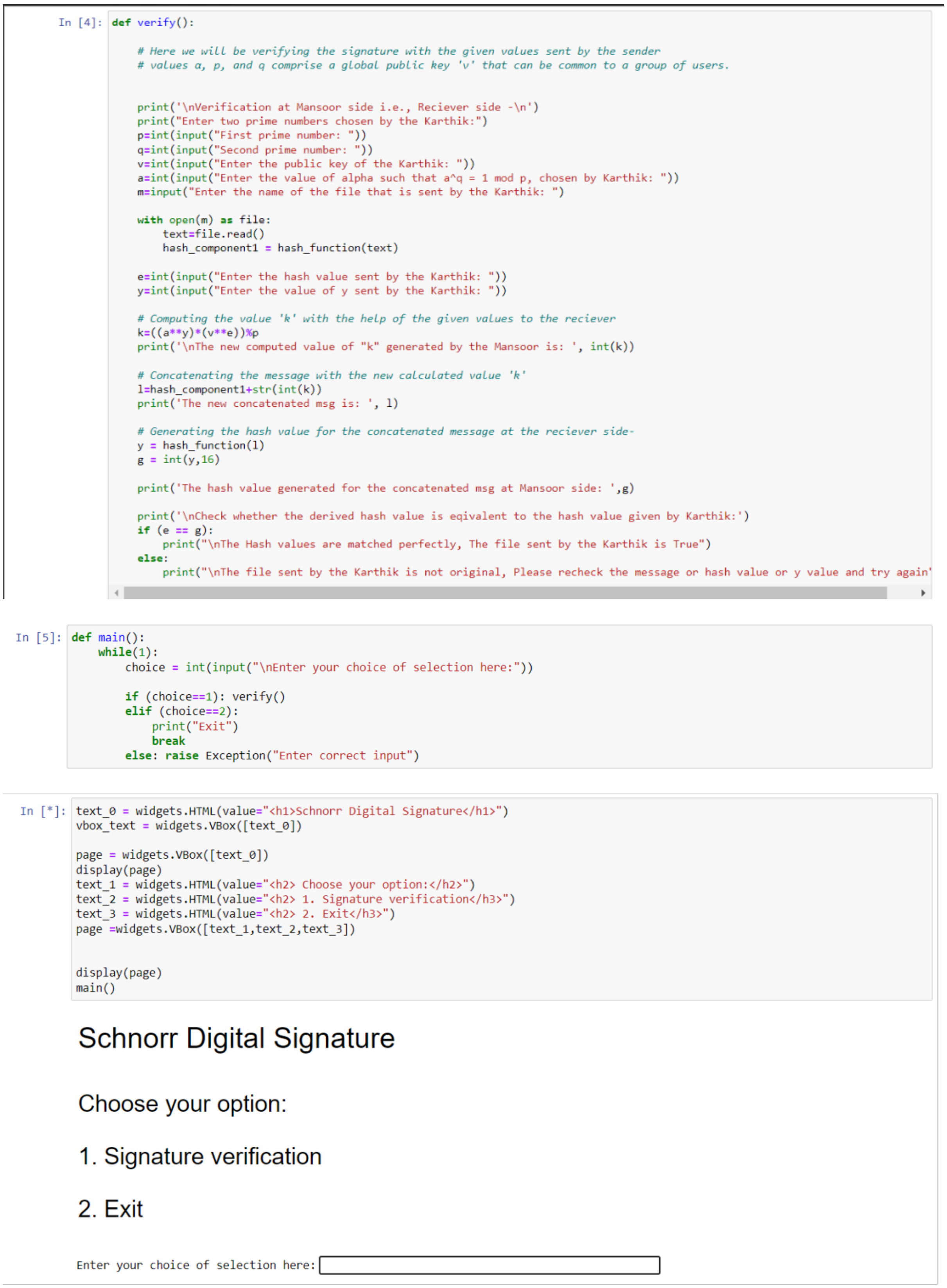
```
Name: Ravipati Sree Karthik
DOB: 24-06-1984


Aadhar Number: 8712 6930 0281

Vid: 4510 9199 3773 0142 5577


Address:
To
Ravipati Sree Karthik
Tirupathi
Chittoor
Pin -517501

Phone Number: 8297221373
```

Code used for verification at Mansoor side:

The hash function reads the input file


jupyter Reciever (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [2]: import random
        import ipywidgets as widgets
        from Crypto.Util.number import *
        from hashlib import sha1

In [3]: def hash_function(message):
            hashed=sha1(message.encode("UTF-8")).hexdigest()
            return hashed
```



```
In [4]: def verify():

            # Here we will be verifying the signature with the given values sent by the sender
            # values a, p, and q comprise a global public key 'v' that can be common to a group of users.


            print('\nVerification at Mansoor side i.e., Reciever side -\n')
            print("Enter two prime numbers chosen by the Karthik:")
            p=int(input("First prime number: "))
            q=int(input("Second prime number: "))
            v=int(input("Enter the public key of the Karthik: "))
            a=int(input("Enter the value of alpha such that a^q = 1 mod p, chosen by Karthik: "))
            m=input("Enter the name of the file that is sent by the Karthik: ")

            with open(m) as file:
                text=file.read()
                hash_component1 = hash_function(text)

            e=int(input("Enter the hash value sent by the Karthik: "))
            y=int(input("Enter the value of y sent by the Karthik: "))

            # Computing the value 'k' with the help of the given values to the reciever
            k=((a**y)*(v**e))%p
            print('\nThe new computed value of "k" generated by the Mansoor is: ', int(k))

            # Concatenating the message with the new calculated value 'k'
            l=hash_component1+str(int(k))
            print('The new concatenated msg is: ', l)

            # Generating the hash value for the concatenated message at the reciever side-
            y = hash_function(l)
            g = int(y,16)

            print('The hash value generated for the concatenated msg at Mansoor side: ',g)

            print('\nCheck whether the derived hash value is eqivalent to the hash value given by Karthik:')
            if (e == g):
                print("\nThe Hash values are matched perfectly, The file sent by the Karthik is True")
            else:
                print("\nThe file sent by the Karthik is not original, Please recheck the message or hash value or y value and try again'
```

```
In [5]: def main():
            while(1):
                choice = int(input("\nEnter your choice of selection here:"))

                if (choice==1): verify()
                elif (choice==2):
                    print("Exit")
                    break
                else: raise Exception("Enter correct input")
```

```
In [*]: text_0 = widgets.HTML(value="<h1>Schnorr Digital Signature</h1>")
        vbox_text = widgets.VBox([text_0])

        page = widgets.VBox([text_0])
        display(page)
        text_1 = widgets.HTML(value="<h2> Choose your option:</h2>")
        text_2 = widgets.HTML(value="<h2> 1. Signature verification</h3>")
        text_3 = widgets.HTML(value="<h2> 2. Exit</h3>")
        page =widgets.VBox([text_1,text_2,text_3])


        display(page)
        main()
```

Schnorr Digital Signature

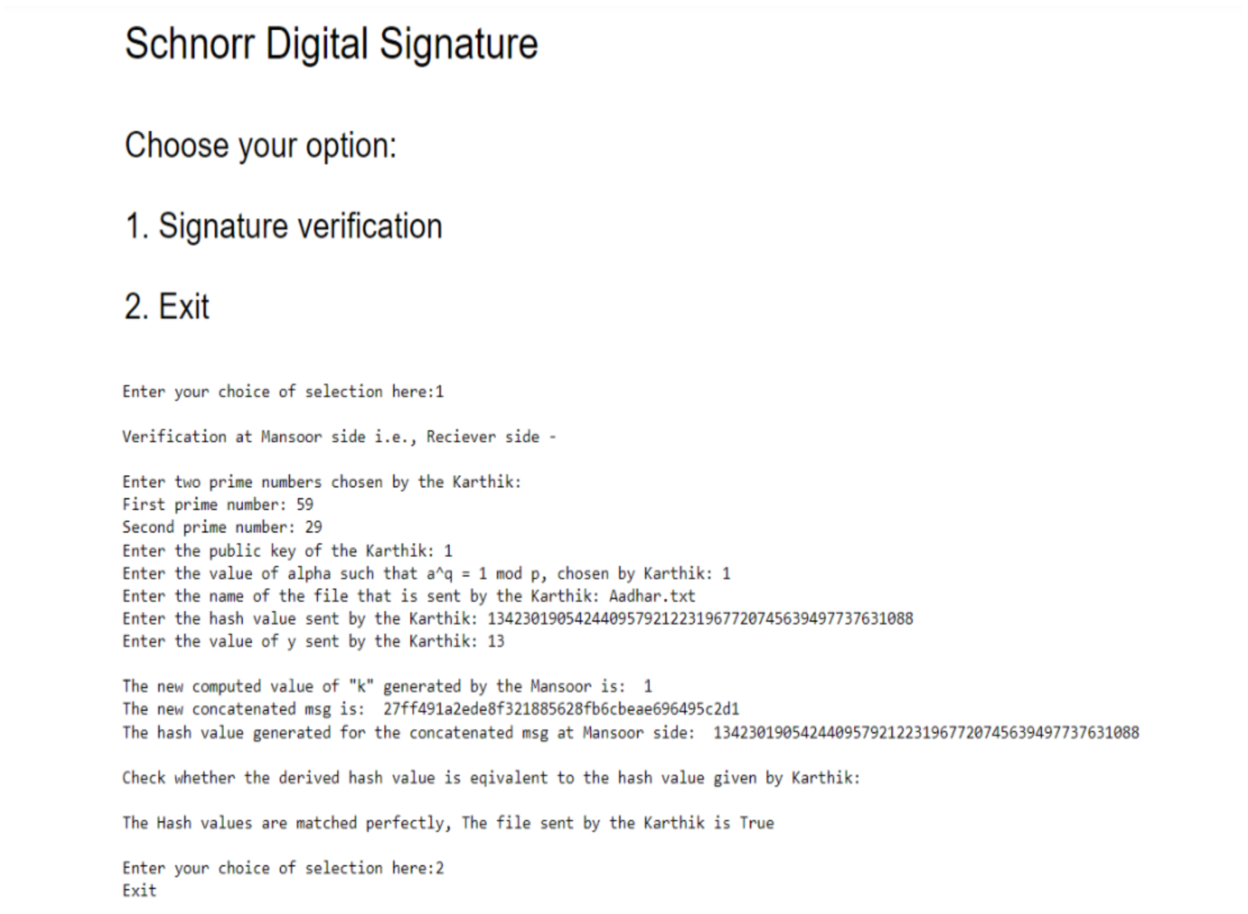
Choose your option:

1. Signature verification

2. Exit

```
Enter your choice of selection here:
```

Using the publicly available details and the hash function the document is verified



Schnorr Digital Signature

Choose your option:

1. Signature verification

2. Exit

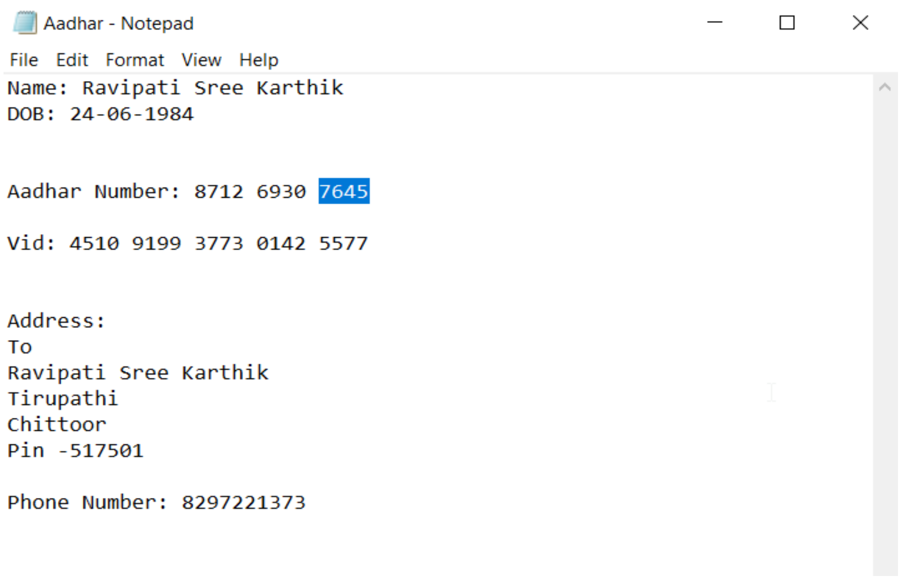
```
Enter your choice of selection here:1

Verification at Mansoor side i.e., Reciever side -

Enter two prime numbers chosen by the Karthik:
First prime number: 59
Second prime number: 29
Enter the public key of the Karthik: 1
Enter the value of alpha such that a^q = 1 mod p, chosen by Karthik: 1
Enter the name of the file that is sent by the Karthik: Aadhar.txt
Enter the hash value sent by the Karthik: 1342301905424409579212231967720745639497737631088
Enter the value of y sent by the Karthik: 13

The new computed value of "k" generated by the Mansoor is:  1
The new concatenated msg is:  27ff491a2ede8f321885628fb6cbeae696495c2d1
The hash value generated for the concatenated msg at Mansoor side:  1342301905424409579212231967720745639497737631088

Check whether the derived hash value is eqivalent to the hash value given by Karthik:

The Hash values are matched perfectly, The file sent by the Karthik is True

Enter your choice of selection here:2
Exit
```

Here we can clearly see in the screenshot, that the hash values are perfectly matched then we can say that the file sent by Karthik is original and not altered.

Consider a scenario where Karthik’s email credentials are compromised by Jaswanth, and file details have been modified using Man in the middle attack and sent to Mansoor.

Now Mansoor cannot identify whether the file sent is been manipulated or not as it is sent by Karthik’s email account

Altered file:

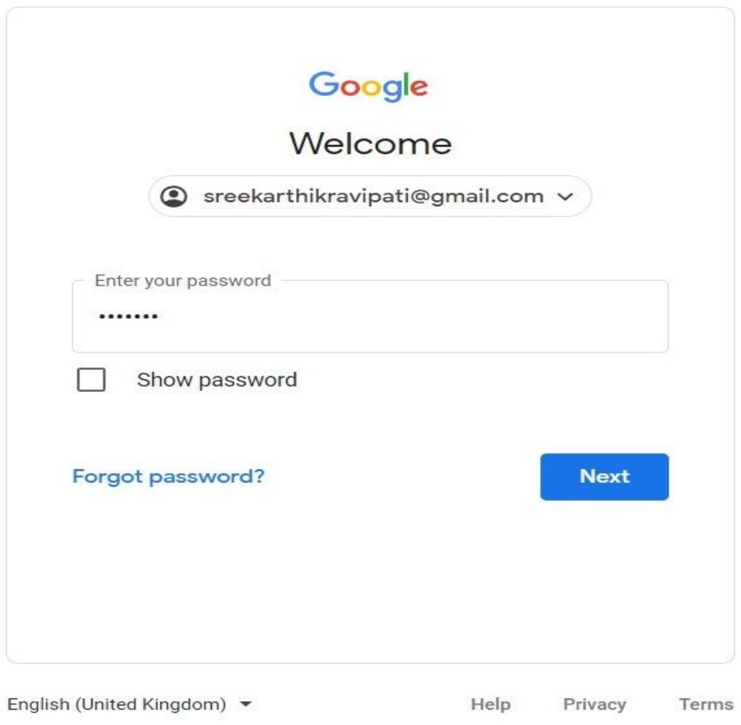


Aadhar - Notepad

File Edit Format View Help

```
Name: Ravipati Sree Karthik
DOB: 24-06-1984


Aadhar Number: 8712 6930 7645

Vid: 4510 9199 3773 0142 5577


Address:
To
Ravipati Sree Karthik
Tirupathi
Chittoor
Pin -517501

Phone Number: 8297221373
```

Now Jaswanth will be using Karthik Gmail credentials so that he can send the altered file without knowing that he has sent it.



Google

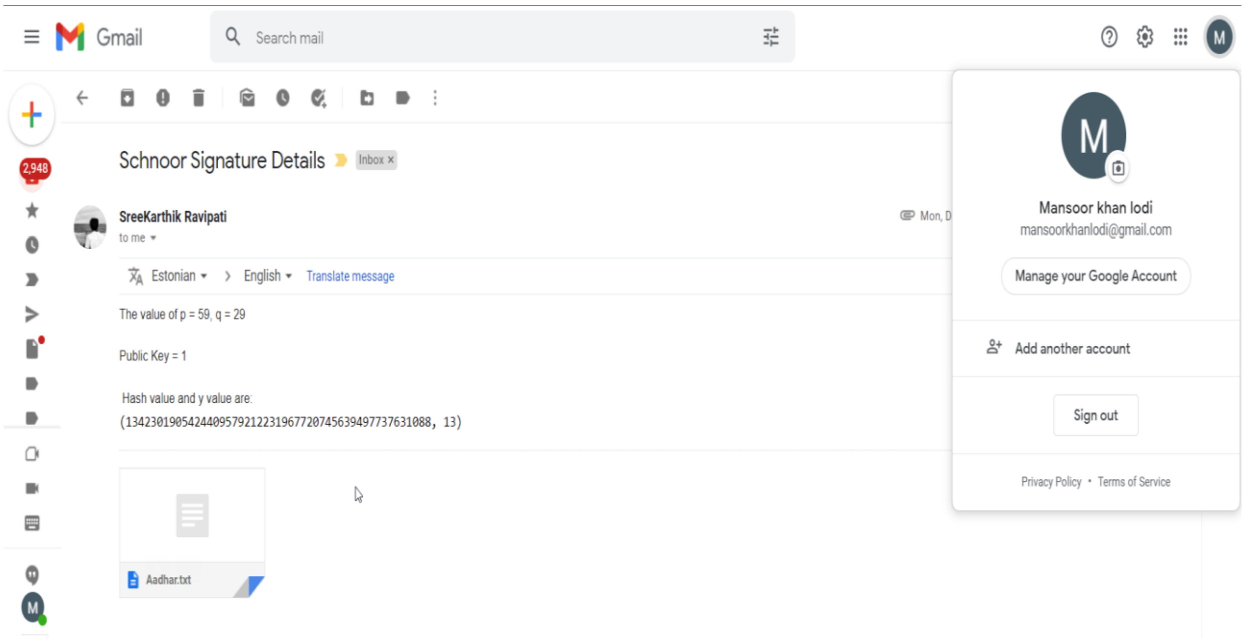
Welcome

sreekarthikravipati@gmail.com

Enter your password: •••••••

Show password

Mail received at Mansoor side:



Schnoor Signature Details

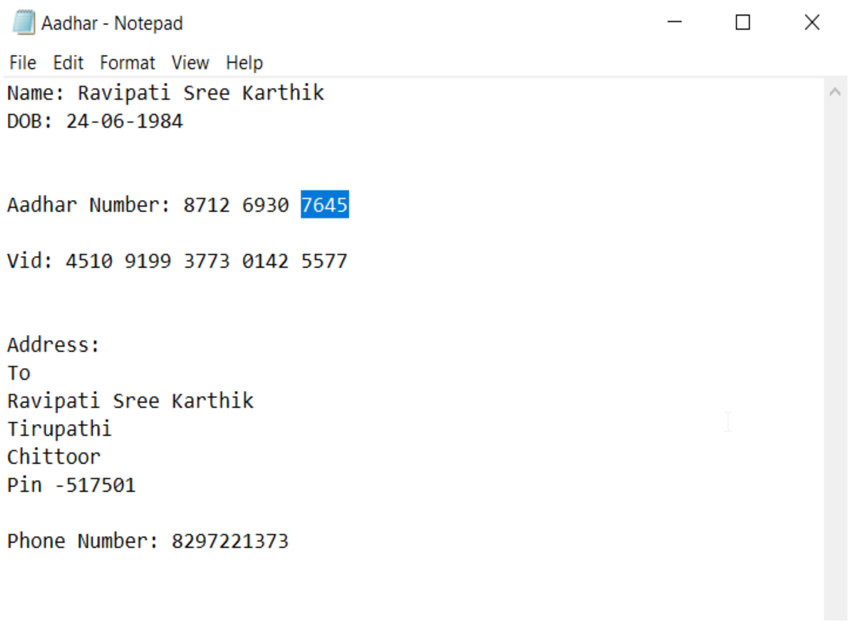
SreeKarthik Ravipati
to me

The value of p = 59, q = 29

Public Key = 1

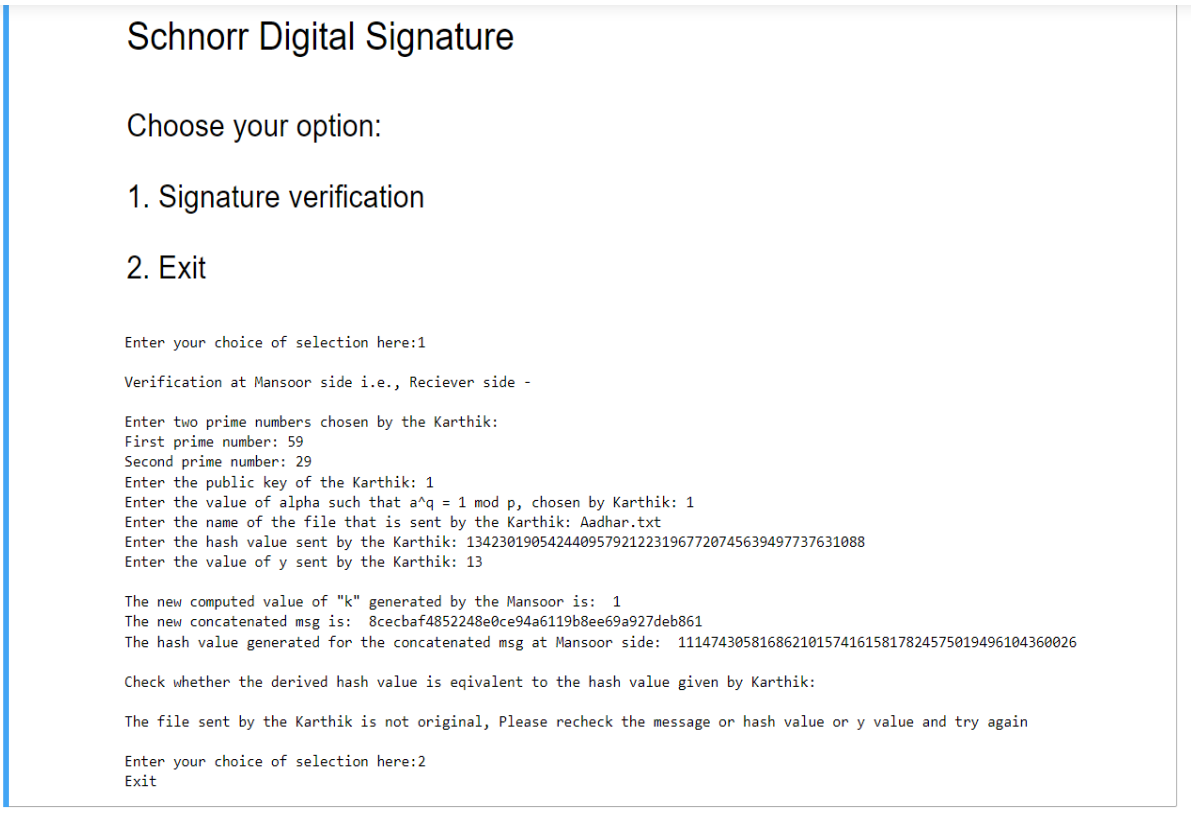
Hash value and y value are:
(1342301905424409579212231967720745639497737631088, 13)

Downloaded text file:



```
Name: Ravipati Sree Karthik
DOB: 24-06-1984


Aadhar Number: 8712 6930 7645

Vid: 4510 9199 3773 0142 5577


Address:
To
Ravipati Sree Karthik
Tirupathi
Chittoor
Pin -517501

Phone Number: 8297221373
```

At Mansoor side we will be checking if the file is altered or not



Schnorr Digital Signature

Choose your option:

1. Signature verification

2. Exit

```
Enter your choice of selection here:1

Verification at Mansoor side i.e., Reciever side -

Enter two prime numbers chosen by the Karthik:
First prime number: 59
Second prime number: 29
Enter the public key of the Karthik: 1
Enter the value of alpha such that a^q = 1 mod p, chosen by Karthik: 1
Enter the name of the file that is sent by the Karthik: Aadhar.txt
Enter the hash value sent by the Karthik: 1342301905424409579212231967720745639497737631088
Enter the value of y sent by the Karthik: 13

The new computed value of "k" generated by the Mansoor is:  1
The new concatenated msg is:  8cecbaf4852248e0ce94a6119b8ee69a927deb861
The hash value generated for the concatenated msg at Mansoor side:  1114743058168621015741615817824575019496104360026

Check whether the derived hash value is eqivalent to the hash value given by Karthik:

The file sent by the Karthik is not original, Please recheck the message or hash value or y value and try again

Enter your choice of selection here:2
Exit
```

Here we can clearly see in the screenshot, that the hash values are not matched then we can say that the file sent by Karthik is altered.

Concluding we can say that the Schnorr signature can detect the alteration if we modify the message file modification here can be little or huge Schnorr signature can detect it easily.

References

https://www.geeksforgeeks.org/schnorr-digital-signature/

https://en.wikipedia.org/wiki/Schnorr_signature