REPORT
ON

‘A Robust Authentication Scheme for Multiple Servers Architecture’

BY

Name of the Student	ID Number
B.SAILENDRA AKASH	***2017AAPS0455H***
K.JYOTHIRMAI	***2017AAPS0290H***

Prepared on completion of the
Cryptography Term Course Project (BITS F463)

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

2019

INTRODUCTION

Data storage has been a very keen interesting issue these days. Many growing industries and enterprises are moving on to using cloud services instead of using local storage servers.All this is a result of development and invasion of cloud computing services into the real world.But the cloud services have issues like privacy issues,security issues etc.To manage the huge amount of data from each of the enterprises,there needs to be a different access policies for each of the users.In the same way,users might also want resources of networks belonging to multiple service providers on the same platform which resulted in having a unique authentication scheme for multiple servers architecture which will be able to make users to utilise network services present on different servers with a unique registration which seems like a plausible solution for solving the problems stated above.

In the initial process of authentication for multiple servers,the RC(registration center) will generate private key for servers and primary secret key for users.Next,the users and the servers can proceed with the authentication and further the key agreement protocol with their corresponding generated private keys.The RC(registration center) will make sure that the user and servers once registered can access and enjoy services with their unique user ID and password in any of the servers.There are two distinct kinds of authentication modes present in multiple server architecture.They are three party authentication mode and two party authentication mode.In the two party authentication mode only the users and the servers take part, there is no third party or registration center where as in three party authentication mode the registration center involves into each authentication that is being made to check the validity of the remaining two parties'. For the actual implementation of the authentication between user and server in the two party authentication mode the users need to keep all secret keys corresponding to each of the server in portable device.The users device will have very less computational power compared to the RC.So the RC in the three party authentication mode handles all the hectic computations that are needed to be performed.The limitation that is common to both the authentication modes is that the RC cannot obtain the shared session key that operates between the other two parties'.

In this project we tried to reproduce the papers H. Wang et al. and related papers Jangirala et al.,Kumari et al.In the Jangirala et al two party authentication mode was followed in which there is no registration center.In this paper the authors implemented the three party authentication scheme to overcome the problems in Jangirala et al.Also using the BAN logic to achieve tripartite authentication.This work shows improved security maintaining the efficiency consistently compared to the previous works.

Related works

We worked on jangirala scheme and a bit on kumari scheme to understand the attacks and how our paper is different and more safe from all the risks which will be discussed. In jangirala scheme there are four phases which will be discussed further in detail. The registration center selects a random number y and primary key x which is a secret key to calculate hash of concatenate of x and y and hash of y and shares these with server through communication channel which is secured . There are registration, login,authentication and key agreement, password change phases. At the end of the first three phases user and server negotiates on a session key. In the last phase the user can change password if he is willing to. As said earlier this scheme is prone to many attacks like impersonation attack, server spoofing, user privacy disclosure.

Our scheme is to create a protocol with a three party authentication and multiple servers architecture. Our scheme consists of five phases same as jangirala's but registration is divided into two namely server registration phase and user registration phase.Finally we compare our work with other schemes. Further it is evident that our scheme can bear maximum range of security threats and have better usage compared to other solutions.

In jangirala scheme the users as well as servers share common secret key hash of y and it is the main reason for many flaws that are mentioned.So in our scheme we use user security key as O(1) and it is widely used in wireless environment.

For encryption and decryption we used AES(Advanced encryption standard) and SHA-256, AESto implement hash function and symmetric encryption and decryption.

IMPLEMENTATION:

We implemented jangirala scheme first in python. First the user has to register in registration center. (The code for the implementation of Jangirala et al. has been mailed)

Jangirala Scheme:

Registration phase: These are the notations,

U_i	denotes user
S_j	denotes the service providing server
RC	denotes the registration center
ID_i	denotes U_i's username
PW_i	denotes U_i's password
SID_j	denotes S_j's identity
x	denotes the primary secret key of the syste
SK	denotes the shared session key
$H(\cdot)$	denotes collision resistant hash function
$E_k(M)$	denotes encrypting a message M with k
$D_k(C)$	denotes decrypting a ciphertext C with k

User chooses a random number b and calculates $A_i = H(ID_i \oplus b \oplus PW_i)$ and registers in RC with {IDi,Ai} it includes hidden credentials of user. Registration centre receives the request and computes After this RC issues smart card which has {Ci,Di,Ei,H(·), H(y)} . User receives this smart card and keys Li.

Login phase :

User with the smart card calculates

$$b = L_i \oplus H(ID_i \| PW_i),\ A_i = H(ID_i \oplus b \oplus PW_i),\ C_i^* = H(ID_i \| H(y) \| A_i).$$

and checks if it is equal to Ci.If not it terminates thye login that password and username did not match. User calculates

$$CID_i = A_i \oplus H(D_i \| SID_i \| N_i),\ P_{ii} = E_i \oplus H(H(SID_j \| H(y)) \| N_i),\ M_1 = H(P_{ij} \| CID_i \| A_i \| N_i),\ M_2 = H(SID_j \| H(y)) \oplus N_i.$$

where Ni is random number and submits the login request {CIDi,Pij,M1,M2} to server.

Authentication and Key agreement phase:

Server gets the login request from user and then computes

$$N_i = M_2 \oplus H(SID_j \| H(y)),\ E_i =$$

$P_{ij} \oplus H(H(SID_j \| H(y)) \| N_i)$, $B_i = E_i \oplus H(x \| y)$, $D_i = H(B_i \| H(x \| y))$ and $A_i = CID_i \oplus H(D_i \| SID_j \| N_i)$ .So to validate the user has access authorization, server calculates $M_1^* = H(P_{ij} \| CID_i \| A_i \| N_i)$ and verifies M1*=M1 . If it holds good it further computes $M_3 = H(SK_{ij} \| A_i \| SID_j \| N_j)$, $M_4 = SK_{ij} \oplus N_j$. and the messages are sent to user again.User again needs to confirm these messages; if it hols good it computes $M_5 = H(SK_{ij} \| A_i \| SID_j \| N_i \| N_j)$ and sends it to the server. Then they both agree on the generated session key.

Password change phase:

If user wants to change the password, smart card calculates Ci* and asks the user to enter a new password. and it computes Ai new, Cinew, Li new and finally replaces them and realizes password update.

Jangirala scheme is prone to many attacks.User privacy disclosure is any adversary can get hash of y and can get the login request message and he can access all user privacy. Impersonation attacks are the adversary extracts values hash of y and D i from User ’s smart card by side-channel attack and transmits login request to server. And he successfully impersonates as user to deceive the server.Server spoofing attack is a server who has secret keys hash of y which are identical for all servers administered by registration center.To overcome all these we implemented a new scheme.

Our scheme:

We implemented H. Wang et al. scheme in python. First the user has to register in registration center. (The code for the implementation of our paper has been mailed)

Server Registration phase:

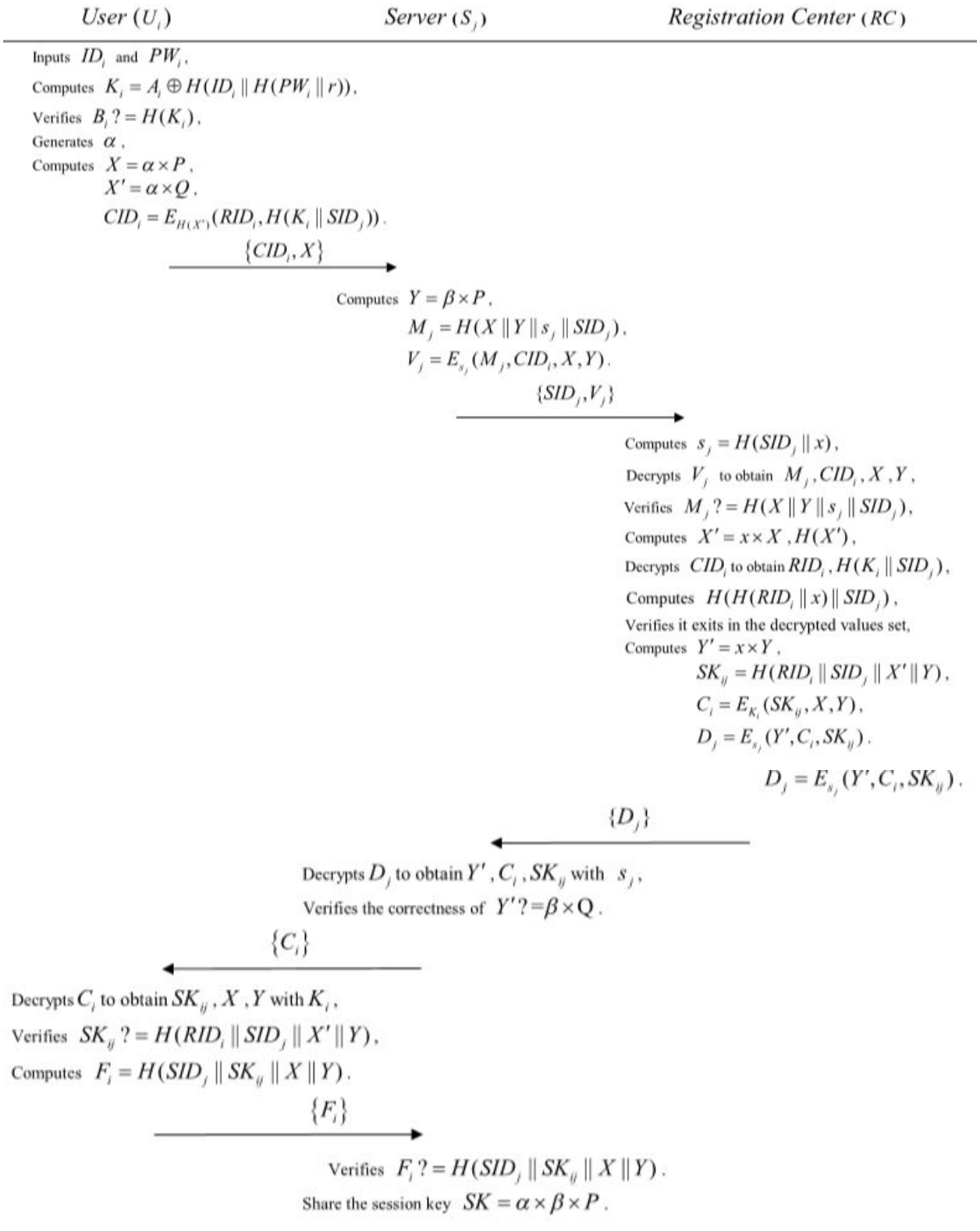
Registration center chooses a large prime p and generates an elliptic curve group with generator P of order p.It even selects random number x belongs to field Zp as the key.

Server registration phase:

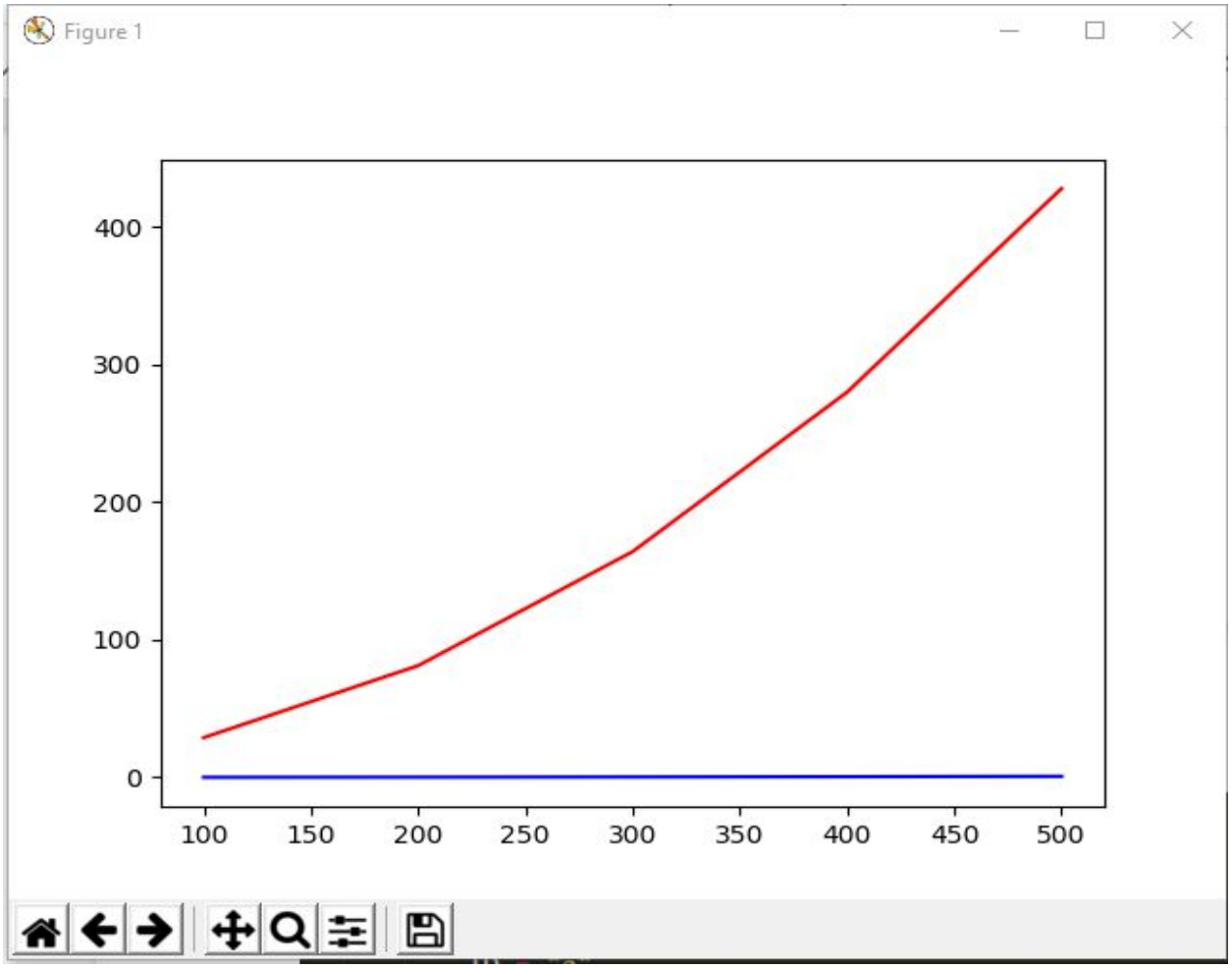
In this servers can register in the registration center for providing services at any time other than initialization of the whole process.First server transmits its ID to RC over secure channel and registration center computes $H(SID_i \| x)$. on receiving this server becomes legitimate server.

User registration phase:

This scheme not only facilitates servers but also users during registration.

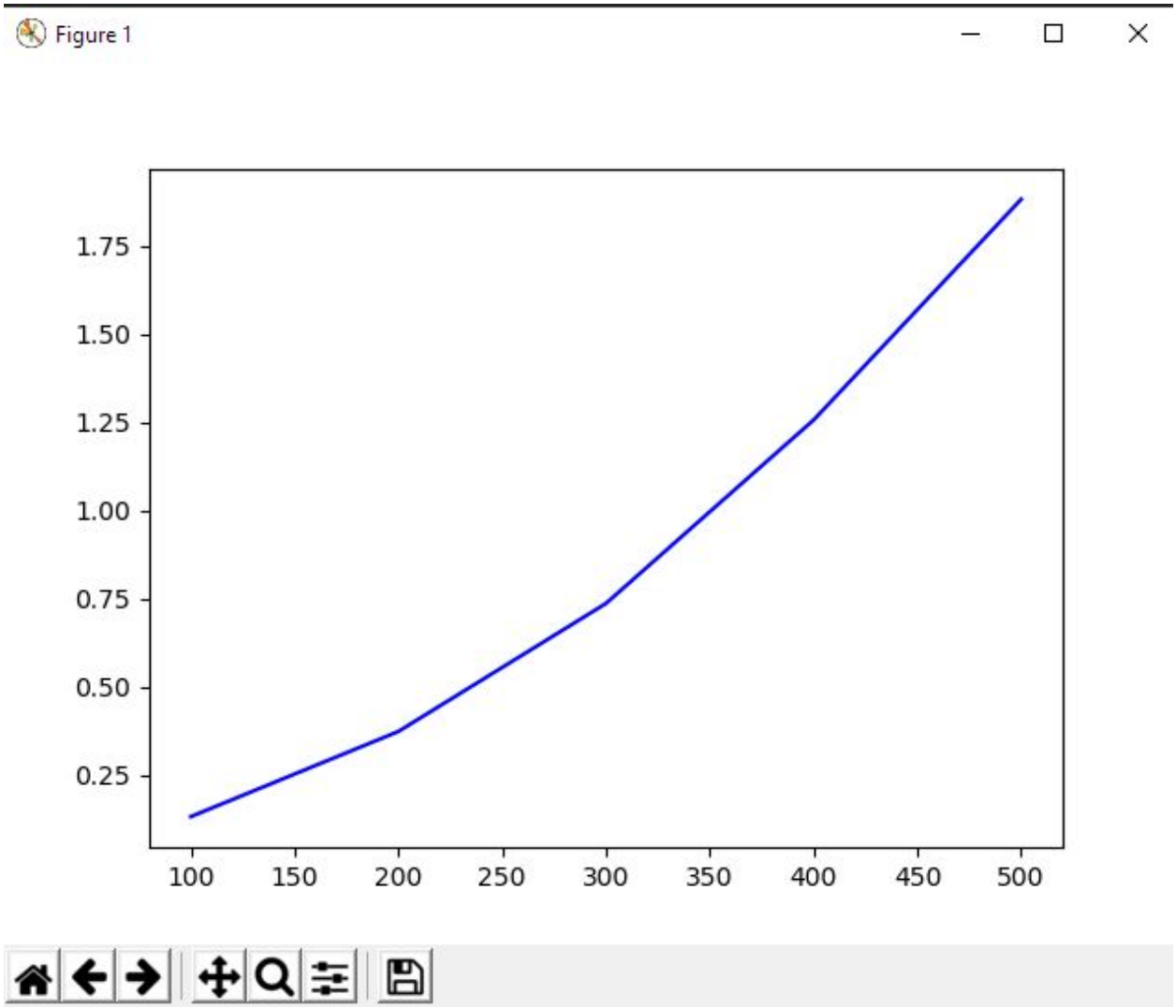


RESULTS AND GRAPH:

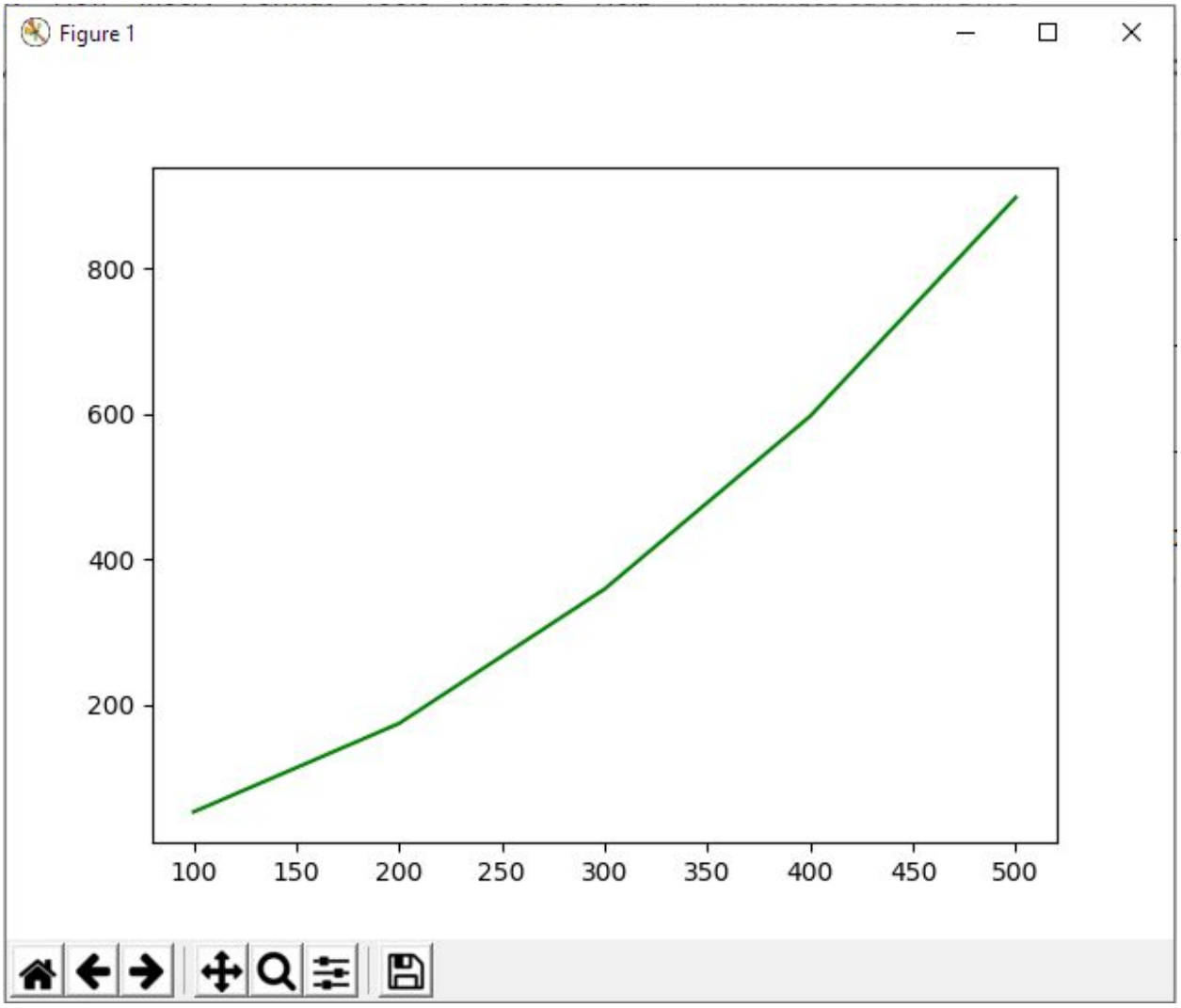


This a programmatically generated graph by using the matplotlib library available in python.

The graph shown above is the comparison of performance analysis of the papers we implemented.The Y-axis on the graph is the seconds scale (it is the time taken by the whole process of authentication right from registration to generating the final session key.), the X-axis is the number of computations in server and registration center.As we can see from the graph the red line is the plot of time taken vs the server and registration center computation of our paper H. Wang et al.The blue line is the same plot that of the Jangirala et al. scheme



As it can be seen that jangirala et al scheme takes very less time(the scale of the y axis in this graph is in milliseconds) compared to H. Wang et al. because there is no registration center in their scheme and authentication is 2 party authentication mode.So we can say that our paper actually takes more computational time because of more computations but provides more security compared to the other papers.



The above graph shows the computational time taken vs the computations of server and registration center.

	Jangirala et al.	Kumari et al.	H. Wang et al.
Resistance of impersonation attack	No	No	Yes
Resistance to off-line password guessing attack	Yes	Yes	Yes
Resistance of server spoofing attack	No	No	Yes
Safe-guarding user anonymity	No	No	Yes
Resistance of replay attack	Yes	Yes	Yes
Providing known key security	Yes	Yes	Yes
Providing perfect forward secrecy	Yes	Yes	Yes

We can see from the graphs and the above table that this paper implementation and scheme can deal well the security threats compared to the others authentication schemes in other papers.There is no doubt it provides high security compared to other authentication schemes.The security check is also proved by using BAN logic and this authentication scheme achieves tripartite authentication certification.The costs of computation of both RC,server and the user side is lower compared to other papers except Jangiral et.al. But that scheme has the three threat attacks as mentioned before and can be seen in the table.

The simulation of server and registration center using python3 in windows i5 7th generation and intel ® processor,We implemented the SHA-256,advanced encryption standard(AES)to implement symmetric encryption and decryption and nacl library to implement hash function

Conclusion:

In this project implementation we discovered many issues with Jangirala et al. authentication scheme for multiple servers architecture authentication where all servers and users will have to use the common secret H(y) which is the reason for the above mentioned security issues.In this paper a three party authentication scheme with tripartite certification strategy.A proof by using BAN logic is also done to evaluate the current scheme of authentication.This scheme also provides resistance from the mentioned security flaws.

CODE:

```
import tinyec
from tinyec import registry
import nacl.hash
import random
import sys
import math
import binascii
import time
curve = registry.get_curve('secp192r1')
import secrets
import os
import base64
from Crypto.Cipher import AES
P=curve.g
x= secrets.randbelow(curve.field.n)
Q=x*P
from CryptoAPI import *
ar=[]

def strtobin(x):
    e=x.encode('utf-8')
    h = binascii.hexlify(e)
    i = int(h, 16)
    return i

def concatenate(a, b="", c="", d="", e="", f=""):
    temp = str(a) + str(b) + str(c) + str(d) + str(e) + str(f)
    temp = temp.encode()
    return temp

count = 0
array =[100,200,300,400,500]
```

```
for i in array:

    while(i!=0):

        print("Please enter your ID")
        ID = input()

        print("Enter your password")
        PW = input()

        print("server identity")
        SID = input()

        start_time = time.time()

        s = nacl.hash.sha256(concatenate(SID,x))
        R = secrets.randbelow(curve.field.n)

         #////////////////////////////////////////////////USER REGISTRATION////


        r = secrets.randbelow(curve.field.n)
        RPW = nacl.hash.sha256(concatenate(PW,r))
        RID = nacl.hash.sha256(concatenate(ID,R))
        K = int(nacl.hash.sha256(concatenate(RID,x)),16)
        A = K ^ int(nacl.hash.sha256(concatenate(ID,RPW)),16)
        B = nacl.hash.sha256(concatenate(K))
```

```
        K1 = A ^ int(nacl.hash.sha256(concatenate(ID,nacl.hash.sha256(concatenate(PW,r)))),16)
        B1 = nacl.hash.sha256(concatenate(K1))


        temp=0
        while temp==0:
            if B1 == B:
                temp=1
                continue
            elif B1 != B:
                print("login terminated")
                temp=1
                sys.exit()
        alpha = secrets.randbelow(curve.field.n)
        X = alpha * P
        X1 = alpha * Q
        list1 = [RID,nacl.hash.sha256(concatenate(K1,SID)) ]
        CID = encrypt(nacl.hash.sha256(concatenate(X1)),list1)
        zz,z = decrypt(nacl.hash.sha256(concatenate(X1)),CID)
        beta = secrets.randbelow(curve.field.n)
        #////////////////////////////////////////////////////////server/////////////////////////////////////
        Y = beta * P
        M = nacl.hash.sha256(concatenate(X,Y,s,SID))
        list2 = [M,CID,X,Y]
        V = encrypt(s,list2)
        #////////////////////////////////////////////////////Registartion center//////////////////////////
        dec1,dec2,dec3,dec4 = decrypt(s,V)

        # if  (M!=dec1) or (CID!=dec2) or (X!=dec3) or (Y!=dec4):
        #   print("cannot proceed further...error")

        RID_new,dec5 = decrypt(nacl.hash.sha256(concatenate(x*X)),CID)

```

```
            if K == K1 and nacl.hash.sha256(concatenate(K1,SID)) == z :
                print(" ")
            else:
                print("request aborted")

            Y1 = x * Y
            SK = nacl.hash.sha256(concatenate(RID,SID,X1,Y))
            list3 = [SK,X,Y]
            K = nacl.hash.sha256(concatenate(RID,x))
            C2 = encrypt(K,list3)
            list4 =[Y1,C2,SK]
            D = encrypt(s,list4)
            #////////////////////////////////server/////////////////////////////////////////
            dec6,dec7,dec8 = decrypt(s,D)
            temp=0
            while temp==0:
                if Y1 == beta * Q:
                    temp=1
                    continue
                elif Y1 != beta * Q:
                    print("login terminated")
                    temp=1
                    sys.exit()
            dec9,dec10,dec11 = decrypt(K,C2)
            temp=0
            while temp==0:
                if dec9 == nacl.hash.sha256(concatenate(RID,SID,X1,Y)):
                    temp=1
                    continue
                elif dec9 != nacl.hash.sha256(concatenate(RID,SID,X1,Y)):
                    print("terminated")
                    temp=1
                    sys.exit()
```

```
            F =nacl.hash.sha256(concatenate(SID,SK,X,Y))
            F1 =nacl.hash.sha256(concatenate(SID,dec9,X,Y))
            temp=0
            while temp==0:
                if F1 == F:
                    print(" a")
                    temp=1
                    continue
                elif F1 != F:
                    print("terminated")
                    temp=1
                    sys.exit()

            N = random.randrange(50,9999,1)
            N1 = random.randrange(50,9999,1)
            SK1 = alpha * beta * P
            SK2 = nacl.hash.sha256(concatenate(SK,A,SID,N,D,N1))

            # print(SK1)

            # print(SK2)

            # print("--- %s seconds ---" % (time.time() - start_time))

            i=i-1
            count = count + time.time()-start_time
        print(count)
        ar1.append(count)
    import matplotlib.pyplot as plt

    plt.plot(array,ar1,'r')
    plt.plot(array,ar2,'b')
    plt.show()
```